Une solution opérationnelle de localisation pour des véhicules autonomes basée sur le SLAM

Cyril Roussillon

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par

l'Institut National des Sciences Appliquées (INSA) de Toulouse

---

Présentée et soutenue le 21 octobre 2013 *par*

Cyril Roussillon

Une solution opérationnelle de localisation pour des véhicules autonomes basée sur le SLAM

Directeur de thèse	M. Simon Lacroix
Rapporteurs	M. Éric Marchand
	M. Guy Le Besnerais
Examinateurs	M. Rachid Alami
	Mme Eva Crück
	M. François Defaÿ

École Doctorale	Systèmes (EDSYS)
Spécialité	Systèmes embarqués
Unité de Recherche	LAAS - CNRS

Résumé

Les applications de la robotique mobile autonome en environnements extérieurs sont nombreuses : surveillance de site à la recherche d'anomalies, campagne d'acquisition de données, exploration, recherche de victimes sur des lieux de catastrophes, etc, et l'intérêt de la robotique pour ces applications est d'autant plus grand que les environnements peuvent être dangereux ou risqués pour l'homme. La localisation des robots est une fonction clé dans ces contextes car elle est indispensable à de nombreuses autres fonctions, particulièrement la construction de modèles d'environnement, l'exécution des trajectoires, ou la supervision des missions.

Ces travaux présentent la construction d'une solution de localisation pour des robots autonomes, conçue pour être à la fois un outil générique de recherche et un outil opérationnel pour localiser nos robots lors de leurs missions de navigation autonome, capable de gérer de fortes dynamiques de mouvement.

En partant d'une solution de localisation et cartographie simultanées (SLAM) basée sur l'utilisation d'une simple caméra, différentes solutions sont successivement construites en ajoutant progressivement des capteurs afin de pallier les difficultés rencontrées lors des évaluations, et ce jusqu'à obtenir un système robuste et précis combinant plusieurs caméras, une centrale inertielle et l'odométrie, et ayant en outre la possibilité d'intégrer des estimations de positions absolues quand elles peuvent être produites (par un récepteur GPS ou un algorithme exploitant une carte initiale). Une analyse profonde des capacités et limitations des différents systèmes est systématiquement effectuée, en considérant notamment l'intérêt d'estimer en ligne les calibrages extrinsèques et biais des capteurs. Un accent particulier est mis sur l'exécution temps réel des algorithmes à bord du robot et sur leur robustesse : cela implique la résolution de nombreux problèmes, portant notamment sur les aspects temporels de la gestion des données.

Une large évaluation sur différents jeux de données réalistes permet d'évaluer et de valider les différents développements proposés tout au long du manuscrit.

Mots clés : Robotique, Mobile, Autonome, Localisation, SLAM, EKF, Temps réel

Abstract

An operational localization solution based on SLAM for autonomous vehicles.

There are numerous applications of outdoor autonomous mobile robots : surveillance of areas to detect anomalies, data acquisition campaigns, exploration, search of victims in disaster areas, etc. Robot localization is a key function in these contexts because it is necessary for a lot of essential robotics tasks, in particular to build environment models, follow paths or supervise the execution of the missions.

This work presents the development of a localization solution for autonomous robots, designed to be both a generic research tool and an effective tool to localize robots navigating with highly dynamic movements.

Starting with a simultaneous localization and mapping (SLAM) solution using a single camera, several solutions are successively built by gradually adding sensors, until obtaining a robust and precise system combining several cameras, an inertial sensor and odometry, which is in addition able to integrate absolute position measures when available (*e.g.* as provided by GPS or a map-based localization scheme). A in-depth analysis of the abilities and limitations of the different systems is systematically made, in particular considering the advantages of estimating online extrinsic calibration parameters and sensor biases. A particular emphasis is set on real time execution of the algorithms on board the robots and on their robustness, requiring to address various problems related to temporal aspects of data management.

Thorough evaluations using different realistic datasets allows to evaluate and validate the proposed work throughout the manuscript.

Keywords : Robotics, Mobile, Autonomous, Localization, SLAM, EKF, Real time

Table des matières

Notations

Notations mathématiques

a : un scalaire ou une fonction.

$\mathbf{a}$: un vecteur.

$\mathbf{A}$: une matrice, ou plus spécifiquement la covariance de $\mathbf{a}$.

σ_a : l'écart-type de a.

$\mathbf{A_e}$: la jacobienne de la variable $\mathbf{a}$ par rapport à la variable $\mathbf{e}$.

$\tilde{a}$: une mesure de la variable a.

$\hat{a}$: une estimation de la variable a.

$\otimes$: le produit de quaternions.

$\wedge$: le produit vectoriel.

v2q : l'opération de conversion d'un vecteur rotation en quaternion.

q2R : l'opération de conversion d'un quaternion en matrice de rotation.

Vocabulaire

cap (*yaw* en anglais) : angle dans le plan horizontal entre l'axe longitudinal et un axe de référence (souvent le nord géographique – sa variation produit un mouvement de **lacet**).

assiette (*pitch* en anglais) : angle dans le plan vertical entre l'axe longitudinal et l'horizontale (sa variation produit un mouvement de **tangage**).

gîte (*roll* en anglais) : angle dans le plan vertical entre l'axe transversal et l'horizontale (sa variation produit un mouvement de **roulis**).

attitude (ou **angles d'attitude**) : orientation par rapport à la verticale (angles d'assiette et de gîte)

incidence (ou **angle d'incidence**) : angle entre le vecteur vitesse et l'axe longitudinal du robot

erreur de biais : erreur additive systématique

erreur de gain : erreur multiplicative systématique

bruit : erreur aléatoire haute fréquence qui n'est pas modélisable par des erreurs de biais et de gain.

capteur **proprioceptif** : nous désignerons ainsi dans ce document un capteur qui mesure directement une partie de l'état interne du robot.

capteur **extéroceptif** : nous désignerons ainsi dans ce document un capteur qui mesure des éléments de l'environnement.

disparité : distance entre les images d'un même point dans deux capteurs différents.

un estimateur sera dit **consistant** [1] (ou dans un état consistant) s'il fournit une estimation de moyenne et d'incertitude statistiquement compatible avec la réalité (en particulier on dira qu'il est inconsistant s'il est surconfiant).

une mesure est **compatible** avec un estimateur si elle est statistiquement plausible selon son estimation.

Abréviations

SLAM : *Simulatenous Localization and Mapping*, ou cartographie et localisation simultanées

EKF : *Extended Kalman Filter*, ou filtre de Kalman étendu.

IMU : *Inertial Measurement Unit*, ou centrale inertielle, qui fournit les accélérations linéaires et vitesses angulaires chacun selon les trois axes.

INS : *Inertial Navigation System*, qui fournit une estimation de la position et de l'orientation sur la base de données inertielles

FOG : *Fiber Optic Gyrometer*, ou gyromètre à fibre optique.

MoCap : *Motion Capture* (capture de mouvement), système qui permet de mesurer précisément la position et l'orientation d'un objet.

AHP : *Anchored Homogeneous Point*, une paramétrisation des amers de type points. On écrira **point AHP** pour désigner un point de paramétrisation AHP malgré le pléonasme apparent, par souci de clarté.

AHPL : *Anchored Homogeneous Points Line*, une paramétrisation des amers de type ligne par deux points AHP.

RMS : *Root Mean Square*, ou moyenne quadratique, utilisée pour évaluer l'erreur absolue d'une trajectoire.

1. Voir section 3.1 page 28 pour plus d'explications sur les raisons d'utiliser ce terme.

Notre problématique et son contexte, importance de la localisation. Nos objectifs, notre approche et présentation du manuscrit.

Chapitre 1

Introduction

Sommaire

1.1 Contexte : robotique en environnements extérieurs

Les applications de la robotique mobile autonome en environnements extérieurs sont nombreuses : surveillance de site à la recherche d'anomalies, campagne d'acquisition de données, exploration, recherche de victimes sur des lieux de catastrophes, etc. L'intérêt de la robotique autonome pour de telles applications est d'autant plus grand que les environnements peuvent être dangereux ou risqués pour l'homme, limitant les possibilités d'intervention directe, ou que les missions couvrent de tels espaces que l'efficacité (voire la possibilité) de solutions téléopérées devient très faible.

Du point de vue de la robotique, ce qui caractérise principalement ces applications est la grande taille des espaces dans lesquels évoluent les robots, et par conséquent la durée d'exécution des missions. Les systèmes déployés doivent posséder une *grande mobilité*, la taille des espaces imposant des vitesses de locomotion suffisamment rapides, bien supérieures à celles des robots d'exploration planétaire par exemple. La capacité de *navigation autonome* est donc naturellement une fonction essentielle, sur la base de laquelle se bâtissent l'ensemble des autres fonctions nécessaires à la réalisation de la mission : planification de tâches pour la surveillance, le suivi d'événement ou l'exploration ; analyse de situation ; construction de modèles d'environnement (parfois pour la finalité de la mission, et pratiquement toujours pour les besons de la navigation autonome) ; etc.

Notons que la grande taille des environnements considérés et les durées des missions rendent également particulièrement utiles l'exploitation simultanée de plusieurs robots. Cela permet d'une part de développer des synergies opérationnelles, principalement en augmentant l'élongation spatiale des systèmes déployés et en apportant de la robustesse aux pannes des robots, et d'autre part de développer des synergies pour les fonctions robotiques impliquées. On peut en effet définir des schémas où les robots s'échangent des informations sur l'environnemt dans lequel il évoluent (par exemple, un robot aérien supporte un robot terrestre en lui fournissant des informations sur l'environnement qui l'entoure), où ils s'assistent pour la localisation, voire même où un robot transporte un autre (ainsi un AUV à décollage vertical peut-être apporté sur site par un robot terrestre).

1.2 De l'importance de la localisation

1.2.1 Définition

Le problème de la localisation consiste à fournir des informations sur la *position* d'un objet exprimées par référence à un repère connu (dans le sens général du terme), et cherchant à approcher au mieux la réalité.

Le terme *position* peut-être défini de manière qualitative, symbolique (« le robot A est dans la pièce X »), auquel cas on parle de localisation qualitative ou topologique, ou bien de manière quantitative, par un ensemble de valeurs numériques associées à un repère, auquel cas on parle de localisation métrique. L'ensemble de nos travaux traitent de localisation métrique dans l'espace, où la position est définie par 6 paramètres.

L'information de position, qu'elle soit qualitative ou quantitative, peut être exprimée de différentes manières, dépendant d'une part de l'approche utilisée pour résoudre le problème de la localisation, et d'autre part de la fonctionnalité qui exploite la localisation. Mais cette expression doit toujours contenir à la fois une information de position et une information de confiance exprimant sa proximité avec la réalité. Cette information de confiance est en effet indispensable pour représenter les incertitudes, erreurs ou imprécisions des informations qui ont mené à une estimation de la position : elle permet d'intégrer d'autres informations, et est aussi utile aux processus qui exploitent l'estimée de la position (pour garantir qu'une trajectoire ne générera pas de collisions par exemple, ou bien pour pour qualifier la qualité d'un modèle de l'environnement construit). Pour la localisation quantitative, on pourra par exemple exploiter un jeu d'intervalles dans lesquels on garantit que la position réelle se trouve, ou bien une distribution de probabilités, paramétrique (gaussiennes par exemple) ou non (population de particules par exemple).

1.2.2 Localisation en robotique mobile

La localisation est une fonction clé dans le contexte de la robotique mobile autonome car elle est indispensable à de nombreuses autres fonctions, particulièrement :

— *construction de modèles d'environnement.* Une mauvaise localisation pourra créer des discontinuités et plus généralement des incohérences qui rendront difficile voire impossible l'exploitation des modèles de l'environnement construits sur la base des données acquises par le robot (notamment pour la planification des déplacements, mais aussi pour la reconnaissance de lieux, l'analyse de situation, ...).

— *exécution des trajectoires.* L'échec de suivi ou le suivi d'une mauvaise trajectoire provoquée par une localisation trop peu performante peut avoir de graves conséquences, par exemple pour l'évitement des obstacles.

— *supervision de la locomotion.* Les organes de mobilité du robot peuvent être mis en échec face à des situations particulières (glissements, embourbements, ...), qui sont fréquents en environnements extérieurs. La détection de telles situations est un problème de diagnostic, pour lequel l'évolution de l'information de localisation est une donnée d'entrée importante.

— *supervision des missions.* Les missions sont le plus souvent définies en terme de positions d'objectifs à rallier ou de zones d'intérêts, décrites de façon métrique (« goto (x, y) », « explore *area* », ...), et le suivi de la localisation du robot lors de ces évolutions est une information de base qui est exploitée pour superviser ses activités dans le but de réaliser les missions.

— *localisation d'événements.* La localisation d'événements ou de cibles est la principale information opérationnelle dans la plupart des missions de surveillance ou d'observation. Réalisée par les capteurs dédiés embarqués à bord des robots, cette localisation se base nécessairement sur la localisation du robot. De manière plus générale, *l'analyse de situation* requiert aussi la localisation de différents éléments.

On voit donc que la localisation joue un rôle dans de nombreuses fonctionnalités robotiques, sur un large spectre de « niveaux », du diagnostic de la locomotion à la supervision globale

du déroulement de la mission.

1.2.3 Quelle qualité de localisation ?

Indépendamment des informations et des processus exploités pour produire une information de localisation, toute solution de localisation peut être qualifiée par les critères indépendants suivants :

— *précision* de la localisation. Pour des applications robotique, cette précision va de quelques centimètres à plusieurs mètres.

— *fréquence* de mise à jour de l'information : de plusieurs centaines de Hertz à uniquement « de temps en temps » (« 1 fois par minute » par exemple).

— *repère de référence* dans lequel est exprimée la position : local au robot, ou global à l'environnement

Notons que ces critères s'appliquent aussi pour qualifier une solution de localisation qualitative non métrique – la précision et la notion de repère de référence étant définies différemment.

Bien entendu, l'expression de ces critères dépend de la fonctionnalité qui exploite l'information de position. Ainsi pour des tâches de construction de modèles d'environnement pour la navigation, ou d'exécution de trajectoire, on cherchera à avoir une localisation très précise, et à haute fréquence, mais on se contentera d'avoir ces qualités dans un repère local. Au contraire pour des tâches de supervision et de planification de mission, on se contentera d'une faible précision et fréquence, mais on exigera que la précision soit bornée dans un repère global. Enfin d'autres tâches comme la construction de modèles globaux précis nécessiteront une grande précision, et ce dans un repère global.

1.3 Objectif de la thèse et approche

La localisation joue un rôle central pour l'autonomie des robots car elle est nécessaire à la réalisation d'un large spectre de fonctionnalités. De plus, le nombre et la variété des informations qui permettent de bâtir une solution de localisation à bord d'un robot sont grands : capteurs de différentes natures, modèles d'évolution du robot, informations relatives à l'environnement, ... L'état de l'art dans le domaine est donc naturellement extrêmement conséquent, comme nous allons le voir dans le chapitre 2.

L'objectif de maximiser l'ensemble des critères qui définissent une solution de localisation (précision centimétrique à plusieurs centaines de Hertz dans un repère absolu, sur une longue durée et dans toute la zone d'évolution du robot) n'est aujourd'hui pas atteint – et il n'est sans doute pas nécessaire de l'atteindre. Cependant, nos efforts visent à l'approcher, et notre principal objectif est de définir un *framework* permettant de façon *souple, rigoureuse et générique* d'inclure toutes les informations de localisation présentes à bord du robot lorsqu'elles sont disponibles.

Les approches dites de « cartographie et localisation simultanée » (SLAM) introduites en robotique permettent notamment de s'affranchir d'infrastructure coûteuse de localisation, et de la connaissance précise a priori de l'environnement d'opération – deux éléments qui ne sont pas disponibles dans la majorité des applications robotiques que nous considérons. Nous avons donc choisi d'exploiter une approche SLAM comme base dans l'architecture de notre *framework*, l'intégration de l'ensemble des données fournies par les capteurs se faisant dans le cadre d'estimation ainsi défini.

L'ensemble de nos travaux ont été menés avec le souci permanent de définir des *solutions opérationnelles*, pour lesquelles le seul moyen de validation est de les intégrer et de les évaluer à bord de robots dans des contextes réalistes. Le développement de telles solutions opérationnelles nécessite d'une part une base théorique éprouvée, et d'autre part la résolution d'un ensemble de difficultés inhérentes à leur mise en oeuvre, lesquelles difficultés nécessitent aussi des investigations théoriques.

1.4 Contributions et organisation du manuscrit

Chapitre 2, nous commençons ce manuscrit par un état de l'art général mais synthétique des solutions de localisation pour la navigation, en recensant les principaux capteurs et solutions de localisation existants, et en nous orientant vers les techniques qui nous semblent les plus à même de répondre à nos besoins, à savoir les techniques de SLAM.

Chapitre 3, nous introduisons le framework « RT-SLAM », développé pour être à la fois un outil de recherche générique et un outil opérationnel pour localiser nos robots. Cette présentation est précédée de quelques bases théoriques sur l'estimation par filtrage de Kalman, et suivie par une présentation des moyens expérimentaux et des séquences de données qui seront utilisées dans la suite du document pour évaluer et valider les résultats obtenus.

Chapitre 4, nous présentons une solution fonctionnelle de localisation entièrement basée sur le SLAM, en décrivant son fonctionnement et les différentes variantes. Nous en profitons pour commencer à évoquer des problèmes de robustesse et d'exécution en ligne des algorithmes et proposer des solutions pour les résoudre.

Chapitre 5, nous étudions les possibilités d'intégration d'autres capteurs en observation ou prédiction du filtre afin de lever les limitations du système vues précédemment, et proposons d'intégrer une centrale inertielle. Nous expliquons en le justifiant le modèle utilisé, et analysons les nouvelles capacités et limitations du système. Nous continuons également à résoudre de nouveaux problèmes posés par l'exécution en ligne de la solution proposée, qui intègre désormais différents capteurs.

Chapitre 6, nous passons en revue l'ensemble des autres capteurs qu'il serait intéressant d'intégrer afin d'atténuer les derniers défauts du SLAM visuel-inertiel. Nous analysons les moyens de réaliser ces intégrations, avec des approches parfois originales : odométrie, GPS, gyromètre à fibre optique, caméras supplémentaires, ... qui nous amènent à nouveau à considérer des problèmes posé par leur intégration en ligne.

Chapitre 7 enfin, nous présentons un bilan de nos résultats, avant de proposer quelques pers-

pectives, notamment une présentation détaillée et étayée d'une architecture générale de localisation multirobots et large échelle, qui permet d'intégrer toutes les informations de localisation disponibles.

Le manuscrit est complété par quelques annexes sur différents développements accessoires qui ont été réalisés, avec notamment le développement d'un détecteur visuel d'objets afin de mettre en pratique des observations mutuelles de robot de façon automatique.

Un état de l'art synthétique des solutions au problème de la localisation pour un mobile. Aperçu des techniques de navigation à l'estime, et de la localisation exploitant une cartographie. Intérêt des techniques de SLAM en robotique, et synthèse.

Chapitre 2

Etat de l'art

Sommaire

Le problème de la localisation se pose naturellement pour tout système en mouvement, et a donc fait l'objet de développements scientifiques et techniques dès l'époque des premiers navigateurs de la Méditerranée [1]. De tout temps, les techniques les plus récentes ont été exploitées, voire développées pour le résoudre – ainsi le chronomètre de Harrisson. Et de nos jours, la disponibilité de récepteurs GPS couplée à des systèmes d'information géographiques a permi de développer le marché dit des services géolocalisés (*location based services*), dorénavant accessibles à tous.

La variété des besoins de localisation et des solutions développées est telle qu'il est illusoire de prétendre la résumer en quelques pages, de nombreux ouvrages étant déjà consacrés à chacune des solutions proposées. Nous allons cependant donner ici un aperçu de l'ensemble des techniques de localisation applicables à la robotique mobile, qui exploitent principalement des technologies à bas coût. Le problème de la localisation est un problème de traitement de l'information, et ce chapitre est structuré en conséquence : la section 2.1 recense l'ensemble des informations disponibles pour le résoudre, et les sections 2.2 à 2.4 présentent les trois principales approches de traitement de ces informations. Le chapitre se conclut par une synthèse, qui justifie et présente notre approche du problème de la localisation.

2.1 Moyens de localisation

Les informations qui permettent d'estimer la position d'un mobile sont de différentes natures : elles peuvent être directement liées au mouvement où à la position du mobile, ou bien être relatives à l'environnement dans lequel il évolue. Nous les distinguons ici selon leur origine : acquises par des capteurs embarqués, ou bien initialement connues.

2.1.1 Capteurs embarqués

a Capteurs de mouvement

Encodeurs Les encodeurs de position ou de vitesse au niveau des roues d'un robot permettent d'estimer sa position et son orientation dans le temps en intégrant des déplacements élémentaires (odométrie). L'odométrie est pratiquement systématiquement exploitée pour les robots à roues, ne serait-ce que parce que la présence de codeurs ou tachymètres est nécessaire à leur contrôle. Elle est cependant sujette à des erreurs difficilement modélisables liées à des problèmes d'adhérence, eux-mêmes difficiles à détecter – et ce même avec des roues odométriques indépendantes des roues motrices.

Capteurs inertiels Les gyromètres mesurent les vitesses angulaires, et les accéléromètres mesurent les accélérations, qui peuvent être intégrées pour estimer les 6 paramètres de position : on parle de *navigation inertielle*. Cette technique de localisation a été initialement développée pour le contrôle de missiles, puis d'avions, de navires et de sous-marins. L'évolution des techniques des capteurs inertiels (gyromètres laser, à fibre optique, gyromètres et

1. L'ouvrage de vulgarisation [Williams, 1992] donne un bon aperçu historique de ces développements.

accéléromètres à *état solide* basés sur des microsystèmes electromécanique, etc.) fait qu'ils sont maintenant largement disponibles, et présents dans de nombreux appareils électroniques.

b Capteurs de « constantes physiques »

Capteurs inertiels Les accéléromètres mesurent également l'accélération de la pesanteur $\mathbf{g}$, ce qui permet de stabiliser les angles d'attitude (assiette et gîte) en observant la verticale (deux accéléromètres utilisés uniquement à cette fin et alors que le robot est à l'arrêt définissent un *inclinomètre*). Les gyromètres mesurent également la rotation de la Terre, mais il s'agit d'une mesure qui doit être compensée pour les capteurs les plus sensibles, et qui n'est exploitable pour estimer aucun des paramètres d'orientation.

Magnétomètres Héritiers des compas magnétiques, un ensemble de 3 magnétomètres montés orthogonalement permet de s'orienter par rapport au nord magnétique terrestre, et ce quelle que soit l'attitude du capteur. Mais les magnétomètres sont sensibles aux distorsions du champ magnétique terrestre causées par la présence de masses ferro-magnétiques, et aux autres sources de champs magnétique. Si il est aisé de calibrer le champ magnétique statique induit par le robot, il est plus délicat d'éliminer la contribution des champs dynamiques dus par exemple aux moteurs. Les perturbations du champ causées par des masses ferro-magnétiques dans l'environnement sont elles par contre quasiment impossibles à prévoir, et donc à corriger [2].

c Capteurs d'environnement

Télémètres Les télémètres mesurent des distances aux objets de l'environnement, selon différentes technologies : ultrasons (sonar), infrarouge, laser, ondes radio (radar), etc. Ils peuvent être utilisés directement pour estimer des distances à des balises ou éléments connus (ainsi pour un engin volant ils donnent une mesure de la distance au sol), mais sont surtout exploités en robotique pour la localisation par rapport à un modèle connu de l'environnement et pour la cartographie.

Caméras De manière analogue aux télémètres, les différents types de caméras (couleur ou noir et blanc, perspectives ou panoramiques, sensibles dans le spectre visible ou infrarouge, etc.) peuvent permettre d'obtenir des mesures de localisation en détectant des éléments identifiables dans l'environnement, en estimant des déplacements élémentaires, ou en construisant une carte de l'environnement.

d Capteurs exploitant une infrastructure

Il s'agit principalement de techniques de radio-navigation, héritières du système LORAN.

2. Notons à ce sujet une récente approche qui exploite les variations locales de champ magnétique [Dorveaux, 2011], commercialisée par la société Sysnav.

GNSS Les systèmes de géolocalisation comme le GPS utilisent une constellation de satellites pour se localiser sur la surface de la Terre par trilatération. L'avantage principal de cette solution est une localisation absolue qui ne dérive pas, mais de précison limitée en l'absence de correction différentielle [3]. Le GPS est aussi sujet à des aléas de réception (trajets multiples en environnements urbains) et inopérant en l'absence de signaux (environnements intérieurs, forêts, etc.).

Réseaux de communication Les réseaux de communication tels que GSM permettent également de se localiser en estimant la distance aux antennes grâce à la qualité de signal reçu. On a à nouveau une localisation absolue, très peu précise en revanche (au mieux de l'ordre de quelque dizaines de mètres). En environnements intérieurs, les réseaux Wifi peuvent permettre d'estimer une position avec une précision environ métrique, mais en exploitant des techniques d'indexation qui requièrent une exploration préalable pour « cartographier » les caractéristiques des signaux reçus.

Enfin, les sytèmes de radiolocalisation par balises UWB permettent désormais d'estimer une position avec une précision décimétrique à l'intérieur des bâtiments.

e Bilan sur les capteurs

Ce rapide aperçu des capteurs permettant la localisation d'un mobile est loin d'être complet : de nombreux autres capteurs plus rarement exploités en robotique existent (vélocimètres de diverses natures, baromètres pour la mesure de l'altitude, radars, dont l'utilisation en robotique tarde à percer, etc) – on trouvera une liste plus exhaustive dans [Borenstein *et al.*, 1995] [4]. Notons simplement qu'ils sont de natures variées, tant dans la technologie qui les compose que dans la nature des informations qu'ils fournissent.

Enfin mentionnons pour mémoire l'existence de capteurs déportés (non embarqués), dont l'analyse des données permet d'estimer des informations de localisation : ainsi le système de localisation par satellites Argos où les systèmes dits de *capture de mouvement* par vision.

2.1.2 Informations a priori

Ces informations peuvent porter sur le mobile lui-même, ou bien sur l'environnement dans lequel il évolue.

a Modèles de mouvement

Le modèle dynamique d'un système décrit l'évolution de ses paramètres d'état en fonction d'entrées, et notamment les paramètres de position dans le cas où le système considéré est mobile. Il constitue donc une information exploitable pour estimer la position d'un mobile,

3. Même si la plupart de GPS bas-coût intègrent désormais les signaux de correction EGNOS et fournissent une position de précision métrique

4. dont une version légèrement mise à jour est disponible sur `http://www-personal.umich.edu/~johannb/Papers/pos96rep.pdf`

qui est d'autant plus précise que le système n'est pas perturbé par des éléments non modélisés – ainsi un satellite, et dans une moindre mesure tout aéronef. Mais face à des pertubations non modélisées et/ou non mesurables, un modèle dynamique peut s'avéréer peu utile à la localisation : c'est le cas pour un robot terrestre sur un terrain extrêmement accidenté par exemple.

Notons qu'en l'absence d'un tel modèle qui explicite les relations entre l'état d'un système et ses entrées, on peut faire des hypothèses générales sur le mouvement du mobile considéré : vitesse constante, accélarations bornées, roulement sans glissements latéraux, etc.

b Informations sur l'environnement

Modèles initiaux Une carte de l'environnement peut naturellement aider à la localisation [5], grâce à des processus de mise correspondance d'éléments représentés dans la carte avec les données perçues sur l'environnement. Différents types de cartes peuvent être exploités, selon le type de données perçues sur l'environnement que permettent les capteurs embarqués : une coupe plane horizontale exhibant les éléments d'un bâtiment (« carte 2D ») permet à un robot muni d'une nappe télémétrique plane de se localiser, une carte représentant simplement les positions de balises données permet à un robot capable de détecter ces balises de se localiser, etc.

Il est important de noter que l'évolution des moyens de cartographier l'environnement a provoqué ces dernières décennies une véritable révolution, récemment amplifiée par les nouveaux moyens de communications qui ont permis l'apparition de nouveaux usages de ces informations (on parle de *système d'information géographique*). La quasi totalité de la surface de la terre est maintenant précisément modélisée par un ensemble de cartes intégrant différentes informations, et il en va de même pour les fonds marins, les principaux bâtiments de nos villes, voire pour d'autres planètes (ainsi certaines zones de Mars sont représentées par un modèle numérique de terrain de résolution métrique).

Autres informations D'autres connaissances initiales relatives à l'environnement (elles aussi souvent représentées par des cartes) peuvent aider à la localisation d'un robot sans exploiter des capteurs d'environnement. C'est par exemple le cas du champ magnétique et du champ gravitationnel terrestre, qui peut permettre l'estimation complète des 3 orientations d'un mobile. La connaissance de courants aériens ou marins peut aussi être explicitée dans le modèle du mouvement, et la connaissance de la nature du sol peut aider à traiter les informations odométriques ou mêmes inertielles.

2.1.3 Synthèse

Variété des informations Un constat évident de ce bref aperçu de l'ensemble des informations qui peuvent être exploitées pour la localisation est qu'elles sont de natures extrêmement variées : certaines mesurent directement un ou plusieurs paramètres de localisation, d'autres

5. La première raison d'être historique des cartes est justement la localisation, la seconde étant la détermination de déplacements.

nécessitent d'être intégrées dans le temps, d'autres encore nécessitent d'être « convoluées » à des informations initiales, etc. Cette richesse définit des complémentarités et/ou des redondances, qui permettent de bâtir différents processus de *fusion des données* pour définir une solution de localisation.

Les possibilités de combinaison des informations sont donc nombreuses, et le choix d'une solution de localisation est guidé par différents critères :

— Les besoins en localisation, exprimés selon les trois qualificatifs généraux introduits précédemment (précision, fréquence et repère de référence),

— Des critères de coûts sont naturellement à considérer : ainsi il peut être exclu d'embarquer une centrale inertielle de grade avionique à bord d'un robot terrestre,

— Des critères de *disponibilité* impactent le choix : le GPS sera bien sûr exclu pour des robots évoluant principalement en environnements intérieurs, les techniques de localisation exploitant une carte ne sont pas exploitables en environnements totalement inconnus, etc.

— Des critères de *dépendance* par rapport à une infrastructure ou une source d'information externe, ainsi que des critères de *fiabilité* – qui excluent par exemple l'utilisation du GPS pour des applications militaires, car il est aisément brouillable.

Typologie des capteurs En ce qui concerne l'ensemble des capteurs embarqués, nous les distinguons selon deux grandes catégories :

— les capteurs *proprioceptifs*, qui mesurent directement une partie de l'état interne du robot. Pour obtenir une localisation complète il est en général nécessaire d'intégrer dans le temps leurs informations, et on parle alors de *navigation à l'estime* (section 2.2).

— les capteurs *extéroceptifs*, qui mesurent des éléments de l'environnement. Pour obtenir une localisation il est en général nécessaire de connaître une carte de cet environnement comme référence, et on parle alors de *localisation par rapport à un modèle initial* (section 2.3 page 17).

Notons que la définition proposée ici de capteurs *proprioceptifs* et *extéroceptifs* se base sur une notion étendue d'état interne du robot, comprenant notamment positions et vitesses dans l'espace, ce qui n'est pas le cas dans la définition originelle qui vient de la physiologie. Cette définition étendue permet cependant une classification pertinente des capteurs pour le problème de la localisation du point de vue de l'architecture des processus de fusion de données, et sera utilisée dans la suite de ce document.

Les deux sections suivantes vont donner un aperçu des techniques d'exploitation des données proprioceptives et extéroceptives. Enfin, la section 2.4 introduit les processus de cartographie et localisation simultanées (SLAM), qui permettent de profiter des qualités de stabilité de la localisation par rapport à un modèle en construisant ce modèle à partir de données extéroceptives.

2.2 Navigation à l'estime

La navigation à l'estime consiste à estimer la localisation d'un mobile par cumul d'estimées de déplacements élémentaires. Il s'agit d'un moyen de localisation simple, notamment exploité par les premiers navigateurs.

2.2.1 Odométrie

L'odométrie basée sur l'intégration des informations renvoyées par des encodeurs placés sur les roues est sans doute la solution la plus simple pour localiser les robots mobiles, dont la mise en équations est triviale. Le principe est d'intégrer les estimations de déplacement élémentaires, et par conséquence l'erreur de l'estimée de localisation par rapport à un référentiel initial croît de manière non bornée.

Sur un sol parfaitement plan et avec de faibles dynamiques de mouvement, les codeurs des roues motrices permettent cependant d'obtenir une précision de localisation suffisante pour assurer le suivi de trajectoires par exemple. L'exploitation de roues odométriques non motrices améliore la précision, mais l'impact des erreurs sur l'estimée des rotations reste grand : l'exploitation d'un gyromètre à fibre optique précis (dérives de l'ordre de quelques degrés par heure, disponible pour quelques milliers d'euros), combiné à l'odométrie pour l'estimation des translations dans un système de *gyrodométrie* permet de fortement améliorer les performances [Borenstein et Feng, 1996].

De tels résultats ne sont cependant applicables que pour la localisation à trois degrés de liberté, dans le plan. Si l'on souhaite une estimation complète de la position et de l'attitude du véhicule, à 6 degrés de liberté, on a alors besoin d'inclinomètres, qui permettent d'estimer l'élévation en intégrant l'angle d'assiette avec les déplacements [Mallet, 2001]. Les inclinomètres sont cependant très sensibles à la dynamique des déplacements car ils assimilent la direction de l'accélération subie à celle de la pesanteur verticale. En combinant les informations de gyromètres et des accéléromètres pour l'estimation des angles d'attitude, les centrales inertielles fournissent une estimation des angles d'attitude.

Mais des difficultés subsistent sur des terrains accidentés, à cause de glissements et dérapages, et aussi parce que les véhicules sont généralement équipés de suspensions ou de pneus présentant une certaine flexibilité. La direction de déplacement du véhicule n'est alors pas toujours alignée avec l'assiette du véhicule, à cause du phénomène de transfert de masse : ainsi le véhicule lève le nez lors des accélérations et pique du nez lors des décélérations, ce qui provoque des erreurs significatives dans l'estimation de l'élévation.

2.2.2 Navigation inertielle

Trois accéléromètres orthogonaux associés à trois gyromètres orthogonaux permettent de déterminer complètement les changements de position et d'orientation d'un mobile. Les équations sont cependant autrement complexes que pour l'odométrie : d'une part les capteurs inertiels fonctionnent dans un référentiel galiléen, et mesurent donc la rotation de la Terre sur elle-même ainsi que la gravité. Il convient donc de les éliminer pour estimer la position et

l'orientation du mobile, mais la direction de la gravité ainsi que du vecteur rotation de la Terre dans les mesures dépendent elles-mêmes de l'orientation absolue du mobile. D'autre part les capteurs ne sont pas parfaits, et leurs mesures présentent des biais et des erreurs de facteur d'échelle, qu'il est également nécessaire de déterminer. La navigation inertielle a largement été étudiée, et les systèmes les plus sophistiqués nécessitent des techniques d'estimation poussées.

Contrairement à l'odométrie, la navigation inertielle est immune aux fautes d'adhérence sur le sol et à l'inconnue de la direction de la vitesse. Cependant cette intégration dérive également, en particulier celle de l'accélération qu'il est nécessaire d'intégrer doublement pour obtenir une position. La sensibilité aux bruits et biais est ainsi multipliée, rendant nécessaire l'utilisation de matériels extrêmement onéreux pour obtenir des performances raisonnables, réservés à des applications avioniques ou militaires. Les centrales inertielles bas coût sont en général utilisées en simples centrales d'attitude, en remplacement d'inclinomètres, car les angles d'attitude peuvent bénéficier de la mesure de la gravité pour être mesurés en de manière absolue et stabilisés.

Les centrales inertielles bas coût peuvent également être utilisées dans des techniques de navigation hybrides, où d'autres capteurs les stabilisent. Elles apportent alors au système une localisation complète, une grande sensibilité à la dynamique des mouvement, et présentent l'avantage d'être très peu fautives, hormis les saturations qui sont aisées à détecter.

Notons qu'une solution de localisation exploitant la navigation inertielle intègre pratiquement toujours d'aures informations, relatives à la dynamique des mouvements du mobile notamment, ou relatives à l'environnement : savoir que le robot évolue sur le sol, connaître son axe de déplacement, savoir que l'on évolue sur Terre et que les accéléromètres mesurent l'accélération de la pesanteur, etc. sont autant d'éléments qui permettent de bâtir des schémas de fusion de données.

2.2.3 Odométrie optique

Les premiers algorithmes qui exploitent la vision pour la localisation métrique d'un robot sont apparus à la fin des années 90 [Mallet *et al.*, 2000]. La technique de base repose sur la mise en correspondance de pixels entre les images de deux paires stéréoscopiques acquises lors du mouvement (cette mise en correspondance est une instance particulière du problème dit *d'association des données*). Les coordonnées des pixels ainsi appariés étant connues grâce à la stéréovision dans le repère associé à chacune des deux positions, il est aisé de restituer les 6 paramètres de déplacement entre les deux paires stéréoscopiques – par exemple par application d'une technique de minimisation par moindres carrés. Le terme « odométrie optique » vient du fait que de manière tout à fait analogue à l'odométrie, la position globale est estimée par composition d'estimations de déplacements élémentaires, dont les erreurs se cumulent au cours du temps.

Une telle technique a notamment été exploitée avec succès sur les robots martiens du JPL [Maimone *et al.*, 2007], et de nombreuses variations et améliorations on depuis été apportées – on trouvera une récente synthèse de l'état de l'art dans [Scaramuzza et Fraundorfer, 2011; Fraundorfer et Scaramuzza, 2012].

Des approches analogues peuvent être exploitées directement avec des données de profondeur (techniques dites de *scan-matching* pour les données acquises par une nappe laser plane [Lu et Milios, 1997], et d'*Iterative Closest Point*) pour des données acquises dans un angle solide – initialement décrite dans [Besl et McKay, 1992]), et nous allons voir qu'elles ont rapidement été étendues pour la résolution du problème SLAM.

2.3 Localisation par rapport à un modèle initial

Les solutions de localisation par rapport à un modèle initial ne souffrent pas de l'inconvénient principal des techniques de navigation à l'estime, à savoir la croissance non bornée des erreurs. Elles se divisent en deux grandes catégories, selon qu'elles exploitent un environnement spécialement équipé ou non.

2.3.1 Exploitation d'une infratructure dédiée

La première classe de systèmes fonctionne avec un environnement équipé, le modèle initial décrivant la géométrie de cet équipement. On place dans l'environnement des balises actives (ultrasons, infrarouges, UWB, ...) ou bien des amers passifs identifiables à des positions connues, et des capteurs extéroceptifs les reconnaissent et les localisent au moins partiellement (distance, angle, ...) ce qui permet au robot de se localiser complètement. Différentes systèmes d'équations permettent d'estimer la position, par exemple selon que la mesure correspond à une distance estimée par temps de transmission (trilatération), à des angles relatifs des balises (triangulation) ou à des différences de temps de transmission entre les différentes balises (système hyperbolique, comme le LORAN). Ces équations sont exploitées par des *processus d'estimation*, qui exploitent les grandeurs mesurées et les incertitudes associées afin de déterminer une estimation de la position du robot et les incertitudes associées.

On inclut dans cette catégorie les systèmes de géolocalisation par satellites tel que le GPS, bien que ces capteurs viennent habituellement complètement intégrés en fournissant directement une partie de l'état interne du robot (sa position), et peuvent donc être considérés fonctionnellement comme des capteurs proprioceptifs selon notre typologie. On a cependant intérêt lorsque l'on souhaite intégrer ces informations avec d'autres capteurs à considérer les données brutes (pseudo-distances aux satellites) pour une précision optimale : on parle alors d'*intégration fortement couplée*, à l'opposé de l'*intégration faiblement couplée*.

2.3.2 Exploitation de cartes de l'environnement

La seconde classe de systèmes fonctionne avec un environnement non spécialement équipé dans un but de localisation. Il est alors nécessaire de disposer d'un modèle initial de l'environnement compatible avec le capteur extéroceptif qui va chercher à se localiser dedans. Il peut s'agir d'un modèle 3D complet, d'un modèle numérique de terrain, d'un nuage d'amers avec des descripteurs, etc. La résolution de la localisation repose alors sur deux fonctions essentielles :

— La mise en correspondance entre des données acquises et la carte de l'environnement

(association de données),

— Et l'estimation de la position et de l'erreur associée sur la base de la mise en correspondance (processus d'estimation).

Différentes solutions ont été proposées pour le processus d'estimation, principalement pour pouvoir gérer les ambiguïtés de mise en correspondance, toujours nombreuses entre données et carte initiale : approches markoviennes [Fox *et al.*, 1998] ou filtrage particulaire par exemple [Levinson *et al.*, 2007] (ces derniers travaux sont à la base de la localisation des « Google autonomous cars »). D'une manière générale l'estimation bayésienne est particulièrement bien adaptée à ce type d'ambiguïtés de mise en correspondance.

2.4 SLAM

Nous avons vu qu'il est difficile de définir des systèmes de navigation à l'estime dont la précision est suffisante pour un coût raisonnable, et surtout que la dérive de ces systèmes ne peut pas être bornée, ce qui est un inconvénient critique pour des missions de longue durée, et ce même dans le cas de missions de navigation dans un espace restreint.

Par contre, l'exploitation d'une infrastructure dédiée et de cartes de l'environnement permet de développer des solutions de localisation absolues, parfois très précises. Mais le principal problème de ces solutions est leur disponibilité : en l'absence d'infrastructure ou si ses signaux sont impossibles à percevoir, et en l'absence d'une carte précise de nature compatible avec les capteurs extéroceptifs embarqués, elle ne sont tout simplement pas applicables.

La disponibilité de capteurs extéroceptifs embarqués à bord de robots permet pourtant de construire un modèle de l'environnement (une carte). La difficulté est que le processus de construction d'une carte *nécessite* de connaître la position des capteurs, justement. La résolution simultanée des deux problèmes de localisation et de cartographie constitue le problème du SLAM (*Simultaneous Localisation And Mapping*), qui consiste d'une part à construire un modèle de l'environnement à partir des données acquises, et d'autre part à exploiter cette carte pour la localisation du robot. Les techniques de SLAM permettent d'une part de faire la synthèse entre techniques de navigation à l'estime et techniques de localisation par rapport à un modèle initial, et surtout elles considèrent le problème de la localisation et de la cartographie comme *un seul et unique problème*, qu'il s'agit donc de résoudre en tant que tel. La figure 2.1 illustre le problème du SLAM.

Le SLAM a fait l'objet de *très nombreux travaux* depuis près de trois décennies dans la communauté des roboticiens, dont nous présentons ici un rapide aperçu.

2.4.1 Aperçu historique

La prise de conscience du caractère unifié des problèmes de la cartographie et de la localisation s'est faite dès le développement de recherches sur le robots mobiles autonomes, au milieu des années 80 [Chatila et Laumond, 1985; Smith *et al.*, 1987; Durrant-Whyte, 1988], et les premières solutions ont été ensuite rapidement proposées à l'INRIA [Ayache et Faugeras,

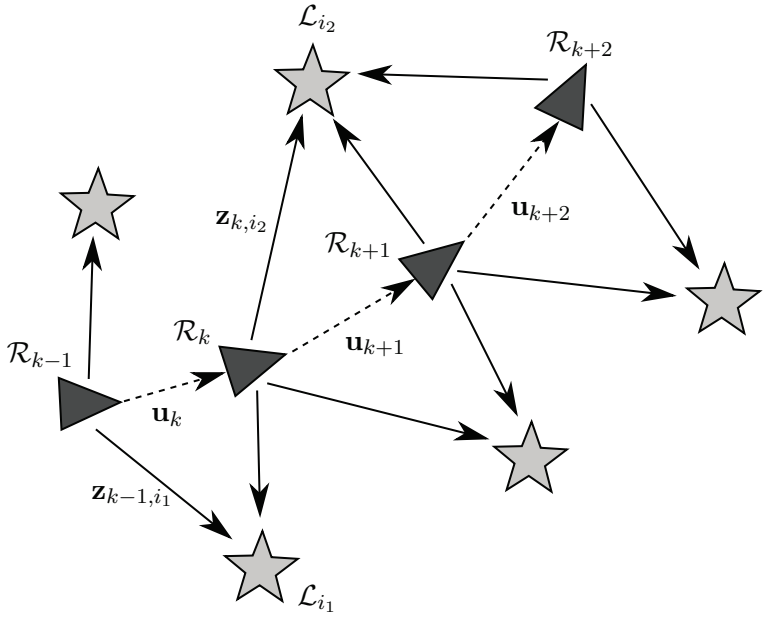


FIGURE 2.1 – [a] Illustration du problème de SLAM. Le robot qui se déplace dans l'espace est à la position $\mathcal{R}_k$ à l'instant k. Il dispose d'observations $\mathbf{z}_{k,i}$ des amers $\mathcal{L}_i$ situés dans l'environnement (provenant de capteurs extéroceptifs), et optionnellement d'observations $\mathbf{u}_k$ de son déplacement entre $k-1$ et k (provenant d'une commande, d'une mesure ou d'un modèle). Il doit alors simultanément estimer sa position $\mathcal{R}$ et la position des amers $\mathcal{L}$.

a. Illustration inspirée de [Bailey et Durrant-Whyte, 2006]

1988], à l'université d'Oxford [Leonard et Durrant-Whyte, 1991] et au LAAS [Moutarlier et Chatila, 1991]. Le problème a ensuite suscité peu d'intérêt jusqu'à la fin des années 90, où sa formalisation probabiliste a été énoncée par [Thrun *et al.*, 1998] : le SLAM a alors suscité un véritable engouement dans la communauté, justifié par son caractère central pour l'autonomie des robots d'une part, et par sa complexité intrinsèque, face à laquelle un très large spectre de travaux ont été proposés. Initialement développé sur la base de données télémétriques ultrason puis laser, la vision a été exploitée pour le SLAM au début des années 2000, d'abord en stéréovision [Jung et Lacroix, 2003; Se *et al.*, 2005], puis en vision monoculaire [Davison, 2003; Solà, 2007] et panoramique [Lemaire et Lacroix, 2007].

Parallèlement à ces travaux menés dans la communauté robotique, les chercheurs en vision proposaient des solutions au problème dit de *Structure From Motion* (SFM), qui consiste à reconstruire la géométrie d'une scène sur la base d'une séquence d'images – en passant par la détermination des positions de la caméra. Le problème est aussi connu sous le nom « d'ajustement de faisceaux » [Triggs *et al.*, 2000], et a permi de développer des techniques d'optimisation applicables à des problèmes de très grandes dimensions.

C'est lorsque les roboticiens ont commencé à exploiter la vision pour le SLAM que les deux communautés se sont rapprochées : les approches SLAM et SFM résolvent en fait le même problème, de manière incrémentale par les roboticiens (et alors le plus souvent avec des techniques d'estimations stochastiques : filtrage de Kalman étendu ou sans parfum, filtre d'information [Thrun *et al.*, 2004], filtrage particulaire [Montemerlo et Thrun, 2003]), tandis le problème

était résolu hors ligne dans la communauté de vision (grâce à des techniques d'optimisation non linéaires). Ce rapprochement a contribué à populariser les techniques d'optimisation auprès des roboticiens, alors confrontés aux problèmes de consistance des approches stochastiques (et notamment du filtrage de Kalman étendu). Mais la formalisation du problème par des réseaux bayésiens [6] a aussi permis de développer un ensemble de nouvelles solutions très performantes [Kaess *et al.*, 2008].

Mentionnons aussi les techniques de SLAM qualitatives qui exploitent la vision : les premières contributions sont apparues à la fin des années 90 (« view-based navigation » [Matsumoto *et al.*, 1998]), et des techniques d'indexation d'images par « sac de mots » basées sur des fondements théoriques solides ont été proposées à la fin des années 2000 [Cummins et Newman, 2011; Angeli *et al.*, 2008; Botterill *et al.*, 2010].

Le problème du SLAM est donc désormais très bien compris, et il existe un véritable corpus de solutions dans la Littérature (on trouvera plus de détails sur l'histoire du SLAM dans [Bailey et Durrant-Whyte, 2006], et un état de l'art synthétique en 2006 dans [Durrant-Whyte et Bailey, 2006]). Mais les approches opérationnelles restent très rares, et exploitent à notre connaissance soit des nappes laser planes pour résoudre la localisation dans un plan (on peut par exemple mentionner les robots de la société Marathon Target – `http://www.marathon-targets.com/`), ou bien de radars dans des environnements où sont disposés des réflecteurs.

2.4.2 Caractérisation d'une approche de SLAM

Toute approche de SLAM métrique basée sur des amers implique l'implémentation des grandes fonctions suivantes :

— Détection d'amers sur la base des données extéroceptives acquises par le robot, et estimation de leur position relative au robot.
— Estimation des déplacements du robot.
— Association de données : mise en correspondance des amers détectés depuis des positions différentes.
— Estimation : mise à jour de la position des amers et de la localisation du robot.

La variabilité de réalisation de ces grandes fonctions est très grande et a été largement explorée par la communauté.

— Un amer est un élément de l'environnement auquel une position géométrique peut être associée, et la plupart des combinaisons types d'amers / capteur extéroceptif ont été explorées (points, segments de droite, plans, objets quelconques, extrait et localisées par tout type de système de vision ou de télémétrie). Notons qu'outre la position géométrique, pas toujours complètement observable, la considération de *descripteurs* d'amers dans l'espace des données du capteur peut apporter grandement pour les processus d'association de données.
— Tous les moyens qui permettent d'estimer les déplacements du robot sont a priori exploitables pour une solution de SLAM.

6. ou *Probabilistic Graphical Models*

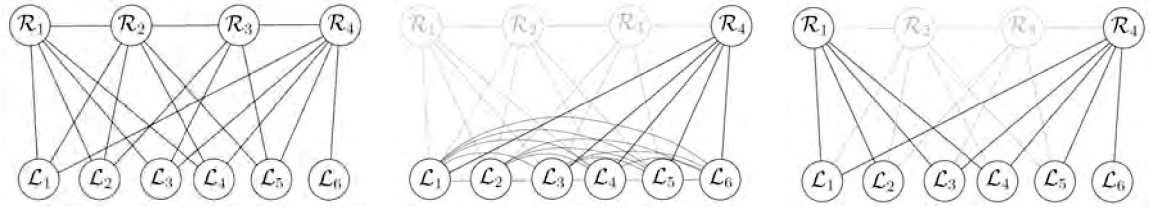
— L'association des données est un processus qui est au cœur du SLAM (et ce pour *toutes* les approches du SLAM, pas uniquement pour les approches basées amers). Il peut être résolu conjointement au processus d'estimation global, les amers étant associés sur la seule base de leurs positions et incertitudes estimées, ou bien de manière indépendante, en exploitant les descripteurs associés aux amers – des approches hybrides ont naturellement été proposées.

— Les processus d'estimation sont ceux qui ont le plus mobilisé l'attention des chercheurs, ils constituent le « moteur » de toute approche de SLAM. On peut distinguer deux grandes familles : les processus d'estimation stochastiques, et les processus d'optimisation non linéaire. Les contributions portent sur l'analyse de leur complexité algorithmique, sur leurs propriétés intrinsèques de convergence, sur leur robustesse aux erreurs d'association des données, ... Même pour une solution d'estimation donnée, de nombreuses implémentations peuvent être distinguées, par exemple selon la paramétrisation des amers ou le choix de repère de référence.

La variété des solutions qui ont été proposées pour le problème du SLAM vient aussi naturellement de la variété des environnements et des types de robots considérés, qui peuvent être aériens, terrestres (en environnements intérieurs, urbains ou naturels), sous-marins.

2.4.3 Présentation des principales méthodes d'estimation probabiliste

Le problème du SLAM *complet*, tel que présenté figure 2.1 page 19, se traduit simplement en un champ aléatoire de Markov, dont un exemple est donné figure 2.2(a), qui met en évidence les liens entre les variables à estimer, à savoir les positions du robot et les amers.



(a) Problème du SLAM complet. (b) Approche filtrage. (c) Approche lissage.

FIGURE 2.2 – [a] Présentation sous forme de champ aléatoire de Markov du problème du SLAM et des simplifications de différentes approches.

a. Illustration issue de [Strasdat *et al.*, 2010]

Ce problème est cependant calculatoirement trop lourd pour pouvoir être résolu en temps réel en ligne, et doit être simplifié. Deux grandes approches ont pour cela été proposées.

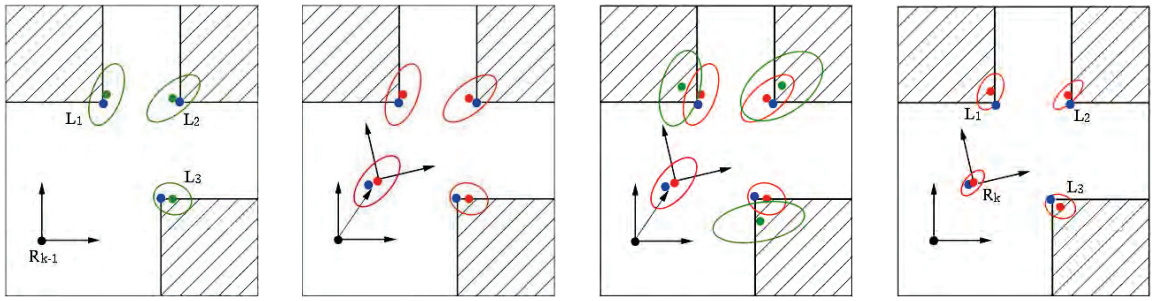
La première, la plus ancienne, consiste à marginaliser (éliminer en transférant l'information dans les autres états) les positions passées pour ne garder que la position actuelle du robot. L'information se retrouve transférée dans des liens entre les différants amers, comme illustré figure 2.2(b). Des filtres bayésiens récursifs sont ensuite utilisés pour réaliser l'inférence.

La seconde approche, plus récente, consiste à ne conserver qu'un sous échantillon régulier de

positions (figure 2.2(c)), appelées images clés (*keyframes*), en éliminant simplement les autres de façon à ne pas ajouter de nouveaux liens entre les variables et conserver une représentation *creuse* du problème, qui facilite sa résolution. Un lissage probabiliste est ensuite effectué pour réaliser l'inférence, par maximisation de la fonction de vraisemblance.

a Filtrage récursif

Les filtres récursifs, ou filtres bayésiens, sont des approximations du filtre de Bayes théorique, visant à estimer les valeurs les plus vraisemblables des variables. Les différentes étapes de résolution du SLAM dans ce cadre sont illustrées figure 2.3.



(a) Le robot mesure des amers dans l'environnement avec ses capteurs, avec une certaine incertitude, dont il se va se servir ensuite pour initialiser leur état.

(b) Il se déplace alors et calcule une estimée de sa nouvelle position, entachée d'incertitude.

(c) Il réalise ensuite de nouvelles mesures des amers, toujours entâchées d'erreurs.

(d) Il fusionne alors les nouvelles informations, faisant diminuer à la fois son incertitude de localisation et de positionnement des amers.

FIGURE 2.3 – [a] Présentation des étapes de résolution du SLAM dans le cadre d'un filtre récursif (mesures en vert, état estimé en rouge, vérité terrain en bleu).

a. Illustration issue de [Lemaire, 2006]

— L'**EKF** (*Extended Kalman Filter*, ou filtre de Kalman étendu) est la solution la plus ancienne et la plus classique de résolution du SLAM. Elle consiste à linéariser les fonctions d'évolution et d'observation autour de l'estimation courante pour appliquer un filtre de Kalman linéaire. La méthode a fait ses preuves, mais a l'inconvénient d'avoir une complexité calculatoire et mémoire quadratique avec le nombre d'amers dans la carte, ainsi que de mauvaises propriétés de consistance (surconfiance) à cause de l'approximation de la linéarisation utilisée. Il est également sensible aux erreurs d'appariemments des amers, qui ne peuvent pas être remis en cause et peuvent corrompre définitivement le filtre, et nécessite donc en général l'emploi de mécanismes de détection de ces erreurs.

Des variantes ont été proposées pour obtenir de meilleures propriétés de consistance :

— L'**UKF** (*Unscented Kalman Filter*, ou filtre de Kalman sans parfum) ne recourt pas à un développement de Taylor pour linéariser, mais au lieu de cela propage les incertitudes *via* un échantillonnage puis une régression. Les avantages sont que les non-linéarités sont approximées au second ordre au lieu du premier ordre, ce qui rend le filtre plus

précis et plus consistant, et qu'il ne nécessite pas le calcul de jacobiennes. Il présente cependant l'inconvénient d'être plus sensible à l'hypothèse gaussienne des incertitudes, car il utilise un échantillonnage déterministe optimisé pour des gaussiennes. Dans la plupart des applications pratiques les différences de précision avec un EKF sont cependant négligeables [Thrun *et al.*, 2005].

— L'**IEKF** (*Iterative Extended Kalman Filter* est une autre variante consistant à itérer les corrections pour améliorer la précision des points de linéarisation utilisés. Cela est bien sûr plus lent, mais comme l'UKF n'améliore en réalité que peu la consistance ou la précision [Bailey *et al.*, 2006a].

— Le **FEJ-EKF** (*First-Estimate-Jacobian EKF*) [Huang *et al.*, 2008] part du constat qu'il n'est pas nécessaire d'utiliser des jacobiennes idéales pour avoir un comportement consistant, mais qu'il suffit qu'elles ne changent pas pour éviter la réduction injustifiée de l'incertitude, et propose de n'utiliser que la jacobienne basée sur la première estimation disponible de l'état des amers. Cela n'est cependant pas réalisable avec une observabilité partielle des amers, et nos tests sur des données réelles consistant à fixer les jacobiennes quand l'estimation des amers est devenue suffisamment précise n'ont montré qu'une faible amélioration de la consistance (environ 15% de réduction, voir tableau 7.4 page 181), et parfois une légère dégradation de la précision.

D'autres méthodes ont également été proposées pour obtenir une meilleure complexité calculatoire :

— Le **filtre d'information** étendu est le dual de l'EKF dans l'espace d'information : là où l'étape de prédiction est simple pour le filtre de Kalman, elle est compliquée pour le filtre d'information, et là où l'étape de correction est compliquée pour le fitre de Kalman, elle est simple pour le filtre d'information. Les avantages du filtre d'information sont qu'il tend à être plus stable numériquement, et qu'il permet plus facilement de travailler avec des grandes incertitudes (tant que les fonctions restent linéaires). Un autre avantage dans le cas du problème du SLAM est que la matrice d'information est presque creuse (seuls les amers qui ont été observés au même moment ont des liens forts), et que la forcer à être creuse en enlevant un peu d'information n'est pas aussi néfaste que pour la matrice de covariance du filtre de Kalman, ce qui permet d'optimiser les calculs pour réaliser des mises à jour en temps constant. Il reste qu'obtenir la moyenne de l'état, qui est la sortie voulue de l'algorithme, et qui est nécessaire pour linéariser les fonctions d'évolution et d'observation, nécessite d'inverser la matrice d'information, ce qui est très coûteux. **SEIF** [Thrun *et al.*, 2004] résout le problème en le propageant numériquement, ce qui est à nouveau une approximation. Au final l'algorithme tourne en temps constant, mais sa précision et sa consistance sont inférieurs à l'EKF. D'autres algorithmes ont été proposés pour réaliser moins d'approximations, aux dépens du gain en rapidité.

— Du côté des filtres non paramétriques, le **filtre particulaire** a également été utilisé avec un certain succès, notamment avec FastSlam [Montemerlo et Thrun, 2003]. Le principe du filtre particulaire est de représenter les probabilités par un ensemble d'hypothèses discrètes (les particules). L'avantage majeur est qu'il peut ainsi représenter n'importe quelle distribution, pas seulement gaussienne, ce qui lui permet notamment de très bien gérer de grandes incertitudes ou des distributions multimodales. FastSlam brise la malédiction de la dimension pour le SLAM en observant que les amers sont condition-

nellement indépendants étant donnée la trajectoire du robot (figure 2.2(a) page 21), ce qui lui permet de les décorréler en utilisant des hypothèses de trajectoire comme particules, et de fonctionner en temps linéaire avec le nombre d'amers, et même logarithmique après ajout d'optimisations sur la gestion des amers. Le nécessaire rééchantillonnage des particules finit cependant par supprimer les corrélations lointaines, ce qui fait perdre de l'information et de la précision, et complique les fermetures de boucle. La précision est au final encore inférieure à l'EKF, et la consistance n'est pas meilleure [Thrun *et al.*, 2005] [Bailey *et al.*, 2006b].

— Les techniques de **sous-cartes** visent selon le paradigme diviser pour régner à décomposer le problème en cartes locales de taille réduite sur lesquelles un EKF tourne. Cela permet de réaliser les traitements à la vitesse des capteurs en temps constant, tout en améliorant la consistance en travaillant avec des incertitudes plus faibles. Une procédure d'optimisation de la position des cartes peut alors tourner en arrière-plan à une fréquence beaucoup plus faible. Les difficultés sont de limiter la perte d'information pendant la transition entre cartes, de les fusionner ou de les utiliser telles quelles par la suite.

b Lissage probabiliste

Le lissage probabiliste, encore appelé ajustement de faisceaux (*bundle adjustment*) ou optimisation globle, a prouvé ses excellentes capacités de précision dans les domaines de la photogrammétrie puis de la reconstruction SFM (*Structure From Motion*) en vision par ordinateur. Il consiste simplement en la minimisation au d'une fonction de coût non linéaire, par exemple avec la technique d'optimisation de Levenberg-Marquardt. Son avantage par rapport au filtre et qu'il réalise plusieurs étapes de linéarisation par raffinement, en étant capable de remettre en cause ses choix passés de linéarisation, ainsi que d'appariemments d'amers. Son inconvénient est que les méthodes de minimisation utilisées jusque là n'étaient pas incrémentales et longues à calculer, les rendant inadaptées à la robotique. L'essentiel des travaux de recherche dans le domaine consiste donc à les rendre plus rapides et incrémentales.

— Les méthodes d'**ajustement de faisceau global**, bien que sélectionnant des images clé, optimisent en permanence toute la trajectoire. Elles se basent sur la structure creuse de la jacobienne des fonctions d'observation pour la décomposer plus rapidement afin de résoudre le système linéaire associé. Avec **Square Root SAM**, [Dellaert et Kaess, 2006] montrent l'importance de réordonner les variables à optimiser dans la matrice décrivant le système à l'aide d'une heuristique pour qu'elle reste la plus creuse possible lors de la factorisation nécessaire à la résolution. Puis avec **iSAM** [Kaess *et al.*, 2008] propose une mise à jour incrémentale de la factorisation lorsque qu'il n'y a pas ajout de fermeture de boucle. [Konolige, 2010] propose également d'exploiter la structure creuse de second niveau, entre les positions, afin d'améliorer la vitesse des calculs. Ces techniques restent cependant toujours trop lentes lorsque la trajectoire devient longue.

— Le problème peut également être transformé en **pose SLAM** [Ila *et al.*, 2010] ou **frame SLAM** [Konolige et Agrawal, 2008], qui consiste à ne plus estimer la position des amers, mais à simplement les utiliser pour générer des contraintes entre les positions du robot,

qui sont ensuite optimisées indépendamment. Le problème de l'augmentation constante du nombre de positions se pose cependant toujours.

— Afin d'être plus rapides, les techniques d'**ajustement de faisceau local** ne retiennent que les dernières images clés dans l'optimisaton, s'approchant ainsi de l'odométrie visuelle [Mouragnon *et al.*, 2009]. Elles sont souvent couplées à un ajustement global qui se déroule plus rarement et en arrière-plan. **PTAM** [Klein et Murray, 2007] va encore plus loin en n'effectuant que le tracking de points à la fréquence des images – mais de façon dense – et en effectuant l'ajustement local en arrière-plan, un ajustement global n'étant réalisé que lors des fermetures de boucle. Dès que l'environnement devient trop grand l'ajustement global devient cependant beaucoup trop long. Ces techniques d'ajustement local se montrent cependant plus rapides et précises que les techniques locales équivalentes en filtrage, principalement dû au fait qu'elles présentent une faible complexité vis à vis du nombre d'amers, et qu'augmenter la quantité d'amers est plus précis qu'augmenter le nombre d'images [Strasdat *et al.*, 2010].

2.5 Synthèse

Les techniques basées SLAM présentent l'avantage fondamental d'être la seule approche permettant d'espérer pouvoir limiter la dérive de la localisation d'un robot sans utilisation d'une infrastructure ou d'informations a priori sur l'environnement, et son choix comme fondation d'une solution de localisation qui se veut générique est donc naturel.

Ce problème peut être résolu par différentes techniques d'estimation, et un choix a dû être fait, qui s'est porté sur le filtrage de Kalman étendu pour les raisons suivantes :

— Il s'agit d'une solution bien comprise, dont les défauts peuvent être maîtrisés avec des techniques de sous-cartes [Estrada *et al.*, 2005], qui facilitent en outre la gestion des modèles d'environnement, la recherche des fermetures de boucle, et la localisation collaborative multirobots (section 7.4 page 184).

— Nous avons un objectif d'intégration des algorithmes à bord des robots pour qu'ils fonctionnent en temps réel, et [Strasdat *et al.*, 2010] ont montré que les méthodes de filtrage étaient supérieures aux techniques d'optimisation globale pour des faibles ressources calculatoires, ce qui est encore le cas sur les drones légers de type quadrirotor.

— Il fournit directement la covariance de l'état estimé, ce qui permet de façon relativement simple d'intégrer autant de capteurs que l'on souhaite, ce qui est indispensable pour obtenir une solution opérationnelle. Les techniques d'optimisation globale sont plus balbutiantes à ce niveau (on pourra citer les travaux de [Eudes, 2011]), et présentent un surcoût important. L'intégration serrée de capteurs est également très délicate.

— Ce choix n'est pas définitif, et il est simple de changer de type de filtre, et tout à fait faisable de le remplacer par une solution d'optimisation globale en conservant l'architecture du *framework* et un réutilisant une grande partie des fonctionnalités développées.

— Enfin on ne peut complètement nier l'influence d'une histoire du filtrage au sein du laboratoire, qui a contribué à pousser l'équipe de développement initiale de RT-SLAM à se tourner vers cette solution.

Présentation des bases théoriques du filtrage de Kalman, de son application au problème du SLAM, puis de l'architecture, du fonctionnement et des propriétés générales de RT-SLAM. Enfin introduction des séquences de données et des métriques d'évaluation utilisées tout au long du document.

Chapitre 3

Présentation générale du framework RT-SLAM

Sommaire

Nous commençons par présenter dans ce chapitre ce qu'est le filtre de Kalman étendu, et son application au problème du SLAM. Nous décrirons ensuite l'architecture générique du framework RT-SLAM et ses différentes propriétés générales, en soulignant les différentes possibilités offertes. Enfin nous introduirons les différentes séquences de données utilisées dans la suite de ce document avec les plateformes ayant permis leur acquisition, ainsi que la méthodologie d'évaluation utilisée.

3.1 Les bases du filtrage de Kalman

Le but d'un estimateur tel que le filtre de Kalman est d'estimer l'état d'un système dynamique à partir de mesures. Les notions suivantes sur les estimateurs, les systèmes, et les mesures, sont nécessaires à la compréhension de la suite :

- Un système est *observable* si son état peut être déterminé à partir de ses mesures.
- Un estimateur est *optimal* si il fournit la meilleure estimation de l'état qu'il est possible d'obtenir théoriquement avec l'information contenue dans les mesures utilisées.
- Un estimateur est *convergent* si son estimée converge toujours vers la vraie valeur en augmentant à l'infini le nombre d'échantillons.
- Un estimateur sera dit *consistant*[1], s'il est dans un état où il fournit une estimation correcte de l'incertitude de son estimation de l'état, sans être surconfiant (*ie* sans sous-évaluer l'incertitude), mais sans être excessivement sous-confiant (*ie* la sur-évaluer) non plus.
- Un estimateur est *intègre* s'il est capable de savoir lorsqu'il n'est pas consistant.
- Un estimateur est *divergent* si l'incertitude de son estimation de l'état n'est pas bornée. On réservera cependant ce terme dans la suite de ce document à une divergence très rapide de l'incertitude et de l'erreur rendant l'estimateur inexploitable.
- Un bruit est *gaussien centré* si sa distribution suit une loi normale de moyenne nulle. Si la moyenne est non nulle, on dira que le bruit est entâché d'un *biais*, et on parlera alors de préférence d'*erreur* afin de réserver le terme bruit à une erreur non biaisée.

3.1.1 Définition du système

Un système dynamique est défini par trois éléments : son état, son équation d'évolution, et ses équations d'observation.

1. **L'état**. Noté $\mathbf{x_k}$, c'est un vecteur stockant les différents paramètres qui représent le système, à l'instant k.
2. **L'équation d'évolution**. Notée f, elle décrit l'évolution de l'état au cours du temps, en fonction de la commande réelle $\mathbf{u_k}$ appliquée au système, et d'un bruit additif gaussien centré $\mathbf{n_k}$:

$$\mathbf{x_k} = f\left(\mathbf{x_{k-1}}, \mathbf{u_k}\right) + \mathbf{n_k} \tag{3.1}$$

1. Le terme *consistant* est un anglicisme provenant de l'anglais consistent, synonyme de convergent. On distingue donc la propriété, en utilisant le terme français, et l'état, en utilisant l'anglicisme. Les deux notions ne sont donc pas équivalentes, mais liées : un estimateur convergent ne peut pas être inconsistant.

En pratique, on dispose seulement d'une valeur mesurée $\tilde{\mathbf{u}}_\mathbf{k}$ de la commande, qui est entâchée d'un bruit additif gaussien centré $\mathbf{m_k}$:

$$\tilde{\mathbf{u}}_\mathbf{k} = \mathbf{u_k} + \mathbf{m_k} \tag{3.2}$$

3. **L'équation d'observation**. Notée h, elle établit le lien entre entre l'état et la mesure réelle $\mathbf{z_k}$ de l'état du système :

$$\mathbf{z_k} = h\left(\mathbf{x_k}\right) \tag{3.3}$$

En pratique, encore une fois la mesure obtenue $\tilde{\mathbf{z}}_\mathbf{k}$ est entâchée d'un bruit additif gaussien centré $\mathbf{v_k}$:

$$\tilde{\mathbf{z}}_\mathbf{k} = h\left(\mathbf{x_k}\right) + \mathbf{v_k} \tag{3.4}$$

3.1.2 Le filtre de Kalman

Le filtre de Kalman est un estimateur de l'état d'un système dynamique linéaire à partir de mesures, qui peuvent être incomplètes ou bruitées. C'est un estimateur récursif, basé sur l'hypothèse markovienne que l'état courant ne dépend que de l'état précédent et d'une commande appliquée à cet instant, et que les mesures ne dépendent que de l'état courant.

Dans le cas particulier d'un système linéaire, l'équation d'évolution s'écrit sous la forme :

$$\mathbf{x_k} = \mathbf{F} \cdot \mathbf{x_{k-1}} + \mathbf{G} \cdot \mathbf{u_k} + \mathbf{n_k}$$

où $\mathbf{F}$ et $\mathbf{G}$ sont les matrices reliant l'état actuel respectivement à l'état à l'instant précédent et à l'entrée de commande.

Pour un observateur également linéaire, l'équation d'observation s'écrit sous la forme :

$$\tilde{\mathbf{z}}_\mathbf{k} = \mathbf{H} \cdot \mathbf{x_k} + \mathbf{v_k}$$

où $\mathbf{H}$ est la matrice reliant la mesure à l'état.

L'état du filtre est quant à lui constitué de deux variables :

— $\hat{\mathbf{x}}_\mathbf{k}$: l'estimation (a posteriori) de l'état $\mathbf{x_k}$ (on notera $\hat{\mathbf{x}}_\mathbf{k}^-$ l'estimation a priori de l'état),

— $\hat{\mathbf{P}}_\mathbf{k}$: l'estimation de l'incertitude de $\hat{\mathbf{x}}_\mathbf{k}$ (*i.e.* l'estimation de cov $\left(\hat{\mathbf{x}}_\mathbf{k} - \mathbf{x_k}\right)$).

L'intégration d'une mesure dans le filtre se fait en deux étapes.

L'étape de prédiction prédit l'état et sa covariance à l'instant de la mesure avec l'équation d'évolution :

$$\hat{\mathbf{x}}_\mathbf{k}^- = \mathbf{F} \cdot \hat{\mathbf{x}}_\mathbf{k-1} + \mathbf{G} \cdot \tilde{\mathbf{u}}_\mathbf{k} \tag{3.5}$$

$$\hat{\mathbf{P}}_\mathbf{k}^- = \mathbf{F} \cdot \hat{\mathbf{P}}_\mathbf{k-1} \cdot \mathbf{F}^T + \mathbf{G} \cdot \mathbf{U_k} \cdot \mathbf{G}^T + \mathbf{N_k} \tag{3.6}$$

où $\mathbf{U_k}$ et $\mathbf{N_k}$ sont les matrices de covariance de $\tilde{\mathbf{u}}_\mathbf{k}$ et $\mathbf{n_k}$.

L'étape de correction corrige l'estimation de l'état avec les équations de Kalman. On commence par définir plusieurs grandeurs :

$$\begin{array}{rcll} \hat{\mathbf{z}}_\mathbf{k} & = & \mathbf{H} \cdot \hat{\mathbf{x}}_\mathbf{k}^- & \text{mesure attendue (\textit{expectation})} \\ \mathbf{y}_\mathbf{k} & = & \tilde{\mathbf{z}}_\mathbf{k} - \hat{\mathbf{z}}_\mathbf{k} & \text{innovation} \\ \mathbf{Y}_\mathbf{k} & = & \mathbf{V}_\mathbf{k} + \mathbf{H} \cdot \hat{\mathbf{P}}_\mathbf{k}^- \cdot \mathbf{H}^T & \text{covariance de l'innovation} \\ \mathbf{K}_\mathbf{k} & = & \hat{\mathbf{P}}_\mathbf{k}^- \cdot \mathbf{H}^T \cdot {\mathbf{Y}_\mathbf{k}}^{-1} & \text{gain de Kalman} \end{array}$$

Puis on montre que la correction optimale de l'état du filtre se fait avec les équations suivantes :

$$\hat{\mathbf{x}}_\mathbf{k} = \hat{\mathbf{x}}_\mathbf{k}^- + \mathbf{K}_\mathbf{k} \cdot \mathbf{y}_\mathbf{k} \tag{3.7}$$

$$\hat{\mathbf{P}}_\mathbf{k} = (\mathbf{I} - \mathbf{K}_\mathbf{k} \cdot \mathbf{H}) \cdot \hat{\mathbf{P}}_\mathbf{k}^- \tag{3.8}$$

3.1.3 Le filtre de Kalman étendu

Le filtre de Kalman étendu (ou EKF pour *Extended Kalman Filter*) est une généralisation du filtre de Kalman pour les systèmes aux fonctions d'évolution et d'observation non linéaires.

La non-linéarité ne pose pas de problème pour le calcul de la moyenne, et f et h peuvent donc être utilisés telles quelles :

$$\hat{\mathbf{x}}_\mathbf{k}^- = f\left(\hat{\mathbf{x}}_{\mathbf{k-1}}, \tilde{\mathbf{u}}_\mathbf{k}\right) \tag{3.9}$$

$$\hat{\mathbf{z}}_\mathbf{k} = h\left(\hat{\mathbf{x}}_\mathbf{k}^-\right) \tag{3.10}$$

En revanche pour les calculs des matrices de covariance, il est nécessaire de linéariser les fonctions au point de l'état estimé. Les mêmes équations que pour le filtre de Kalman sont utilisées, mais en remplaçant les matrices $\mathbf{F}$ et $\mathbf{H}$ par les jacobiennes des fonctions d'évolution et d'observation, qui dépendent alors du temps :

$$\mathbf{F}_\mathbf{k} = \frac{\partial f}{\partial \mathbf{x}}\Big|_{\hat{\mathbf{x}}_{\mathbf{k-1}}, \tilde{\mathbf{u}}_\mathbf{k}} \tag{3.11}$$

$$\mathbf{H}_\mathbf{k} = \frac{\partial h}{\partial \mathbf{x}}\Big|_{\hat{\mathbf{x}}_\mathbf{k}^-} \tag{3.12}$$

Contrairement au filtre de Kalman qui fournit une estimation optimale sur un système linéaire, le filtre de Kalman étendu est pas optimal : il ne garantit pas la convergence sur un système observable, à cause de l'approximation de l'opération de linéarisation et des erreurs sur les points de linéarisation.

Aussi « l'ampleur » de la non linéarité des fonctions d'évolution et d'observation est un critère important dans le choix des paramétrisations des différentes grandeurs à estimer.

De même on aura intérêt à appliquer des corrections à une fréquence suffisamment élevée par rapport à la dynamique du système, afin d'appliquer la linéarisation de l'équation de prédiction sur un domaine où elle reste une bonne approximation.

Remarque 1 Le fait de lever la contrainte de constance de $\mathbf{F}$ permet également de traiter le système avec une fréquence d'échantillonnage variable, $\mathbf{F}$ pouvant alors dépendre de l'intervalle

de temps $\mathrm{d}t_k$ réel entre les instants $k-1$ et k. Cependant il est tout à fait possible de faire de même pour un filtre de Kalman classique appliqué à un système linéaire, sans perdre les propriétés d'optimalité du filtre ($\mathbf{F}$ n'est constante que si le système est linéaire et $\mathrm{d}t_k$ est constant, mais le filtre de Kalman est optimal seulement si le système est linéaire).

Remarque 2 Les hypothèses d'erreurs gaussiennes et centrées sont très importantes. Dans le cas contraire le filtre ne pourra extraire le signal du signal bruité. En particulier il considèrera que les biais, issus d'erreurs non centrées, font partie du signal, ce qui faussera l'estimation. Il est donc important lorsque l'on connait l'existance de ces biais de modéliser les erreurs jusqu'à ce que l'erreur résiduelle ne soit plus constituée que d'un bruit gaussien et centré.

3.1.4 Application au problème de SLAM

Dans un algorithme de SLAM-EKF, le vecteur d'état est constitué de la concaténation de l'état du robot $\mathcal{R}_k$ à l'instant courant k et des états des différents amers $\mathcal{L}_i$:

$$\mathbf{x_k} = (\mathcal{R}_k \ \mathcal{L}_1 \ \cdots \ \mathcal{L}_n)^T \tag{3.13}$$

Une représentation graphique de cet état est donné figure 3.1.

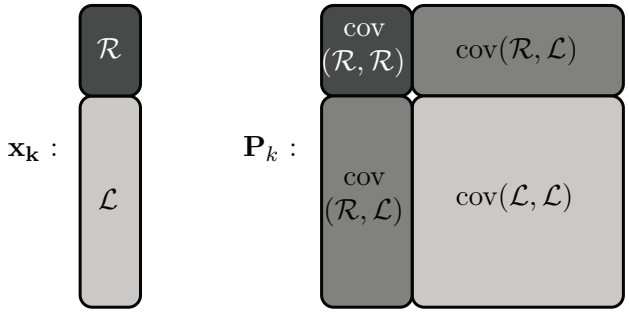


FIGURE 3.1 – Représentation graphique de l'état du filtre EKF pour le problème de SLAM.

Dans l'hypothèse d'un environnement statique, l'équation d'état ne concerne que le robot :

$$\mathcal{R}_k^- = f\left(\mathcal{R}_{k-1}, \mathbf{u_k}\right) + \mathbf{n_k} \tag{3.14}$$

où $\mathbf{u_k}$ est la commande et $\mathbf{n_k}$ un bruit additif gaussien centré.

L'équation est alors linéarisée pour pouvoir appliquer les équations du filtre de Kalman :

$$\mathcal{R}_k^- = \mathbf{F_k} \cdot \mathcal{R}_{k-1} + \mathbf{G_k} \cdot \mathbf{u_k} + \mathbf{n_k} \tag{3.15}$$

où $\mathbf{F_k}$ et $\mathbf{G_k}$ sont les jacobiennes respectivement en $\mathcal{R}_{k-1}$ et $\mathbf{u_k}$ de f.

De même l'équation d'observation de l'amer i ne dépend que de l'état du robot et de l'amer concerné :

$$\mathbf{z_{k,i}} = h\left(\mathcal{R}_k^-, \mathcal{L}_i\right) + \mathbf{v_k} \tag{3.16}$$

où $\mathbf{v_k}$ est un bruit additif gaussien centré, et qui est linéarisée en :

$$\mathbf{z_{k,i}} = \mathbf{H_k} \cdot \left(\mathcal{R}_k^- \ \mathcal{L}_i\right)^T + \mathbf{v_k} \tag{3.17}$$

où $\mathbf{H_k}$ est la jacobienne de h en $\mathcal{R}_k^-, \mathcal{L}_i$.

Le filtre de Kalman étendu fournit alors les équations permettant d'estimer $\mathbf{x_k}$ et sa matrice de covariance à travers les étapes de prédiction et de correction.

En pratique la commande $\mathbf{u_k}$ pourra être fournie soit par une commande, soit par un modèle, soit par une mesure.

3.1.5 Propagation des incertitudes

a Utilisation des jacobiennes

Comme on l'a vu, dans un EKF les jacobiennes sont utilisées pour convertir les incertitudes. Plus généralement et même à l'extérieur du filtre elles peuvent être utilisées pour convertir l'incertitude des variables transformées par une fonction.

Soit la fonction f :

$$\mathbf{x} = (x_1, x_2, \cdots, x_n)^T \qquad \mathbf{y} = (y_1, y_2, \cdots, y_m)^T \qquad \mathbf{y} = f(\mathbf{x}) \tag{3.18}$$

La jacobienne de f se définit de la façon suivante :

$$\mathbf{F} = \begin{pmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_2}{\partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial y_n}{\partial x_1} & \frac{\partial y_n}{\partial x_2} & \cdots & \frac{\partial y_n}{\partial x_n} \end{pmatrix} \tag{3.19}$$

Pour la valeur particulière $\mathbf{x_0}$ de covariance $\mathbf{X_0}$, on évalue la jacobienne en $\mathbf{x_0}$ (on linéarise). On utilise la notation commode $\mathbf{Y_x}$ indiquant clairement que l'on considère la jacobienne de la variable $\mathbf{y}$ par rapport à la variable $\mathbf{x}$:

$$\mathbf{Y_x} = \mathbf{F}|_{\mathbf{x_0}} \tag{3.20}$$

La covariance $\mathbf{Y_0}$ de $\mathbf{y_0} = f(\mathbf{x_0})$ s'obtient alors ainsi :

$$\mathbf{Y_0} = \mathbf{Y_x} \cdot \mathbf{X_0} \cdot \mathbf{Y_x}^T \tag{3.21}$$

Exemple Considérons par exemple la fonction f :

$$\mathbf{p} = (x, y, z)^T \qquad \mathbf{s} = (u, v)^T \qquad \mathbf{s} = f(\mathbf{p}) = \begin{pmatrix} x/z \\ y/z \\ 1 \end{pmatrix} \tag{3.22}$$

La jacobienne de f est alors :

$$\mathbf{F} = \begin{matrix} \\ \frac{\partial u}{\partial} \\ \frac{\partial v}{\partial} \end{matrix} \begin{matrix} \begin{matrix} \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \end{matrix} \\ \begin{pmatrix} 1 & 0 & -x/z^2 \\ 0 & 1 & -y/z^2 \end{pmatrix} \end{matrix} \tag{3.23}$$

Évaluée en $\mathbf{p_0} = (2, 3, 2)^T$, on obtient :

$$\mathbf{S_p} = \mathbf{F}|_{\mathbf{p_0}} = \begin{pmatrix} 1 & 0 & -1/2 \\ 0 & 1 & -3/4 \end{pmatrix} \tag{3.24}$$

Calcul symbolique Ces calculs pouvant être rapidement complexes et enclins à erreurs, en pratique on peut également faire appel à des outils de calcul symbolique comme Maple, Mathematica ou la toolbox de calcul symbolique de Matlab. Certains comme Maple permettent également de générer automatiquement du code C ce qui simplifie le travail et évite toute erreur de conversion.

b Construction de la jacobienne

On vient de voir comment construire la jacobienne directement par sa définition. En pratique, seules les fonctions élémentaires seront définies ainsi. Pour des fonctions plus complexes, on cherchera à les décomposer et à calculer leur jacobienne totale à partir des jacobiennes des opérations plus simples.

Deux principales règles sont utilisées pour cela.

Composition Pour des fonctions plus complexes, on décompose la fonction en opérations de base, et on utilise la règle de composition des jacobiennes. Soient les fonctions f et g :

$$\mathbf{y} = f(\mathbf{x}) \qquad \mathbf{z} = g(\mathbf{y}) \tag{3.25}$$

Alors la jacobienne de $f \circ g$ est le produit des jacobiennes de f et g :

$$\mathbf{Z_x} = \mathbf{Z_y} \cdot \mathbf{Y_x} \tag{3.26}$$

Variables indépendantes Soit la fonction f des deux variables aux erreurs indépendantes $\mathbf{x}$ et $\mathbf{y}$:

$$\mathbf{z} = f(\mathbf{x}, \mathbf{y}) \tag{3.27}$$

On pourrait concaténer les deux variables $\mathbf{x}$ et $\mathbf{y}$:

$$\mathbf{m} = \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} \qquad \mathbf{M} = \begin{pmatrix} \mathbf{X} & \mathbf{0} \\ \mathbf{0} & \mathbf{Y} \end{pmatrix} \qquad \mathbf{Z_m} = \begin{pmatrix} \mathbf{Z_x} & \mathbf{Z_y} \end{pmatrix} \tag{3.28}$$

En appliquant l'équation 3.21 et en simplifiant on obtient directement :

$$\mathbf{Z} = \mathbf{Z_x} \cdot \mathbf{X} \cdot \mathbf{Z_x}^T + \mathbf{Z_y} \cdot \mathbf{Y} \cdot \mathbf{Z_y}^T \tag{3.29}$$

Toutes les jacobiennes nécessaires sont donc calculées manuellement ou avec un logiciel de calcul symbolique pour toutes les opérations de base, puis par application de ces règles pour la construction des jacobiennes plus complexes. [2].

2. Les nombreuses jacobiennes utilisées ne sont pas présentées dans ce document, leur détermination résulte en effet d'une tâche automatique et déterministe, leur présentation serait d'un niveau de détail proche du code.

c Intégration des incertitudes

Une précaution particulière est à prendre lors des opérations d'intégration temporelle des grandeurs. En effet, en général, lorsque l'on modélise un système, il y a deux sources différentes d'erreur : l'erreur causée par une mauvaise paramétrisation du modèle, et l'erreur causée par l'inadéquation du type de modèle. Ces deux erreurs sont indépendantes, et leurs variances respectives s'ajoutent donc.

Erreur de paramétrisation Le modèle choisi pour l'intégration d'une grandeur est en général la méthode des rectangles. La première source d'erreur – les paramètres du modèle – vient donc de l'erreur sur la grandeur à intégrer. L'intégration consistant à multiplier la grandeur par $\mathrm{d}t$, l'erreur est donc également multipliée par $\mathrm{d}t$, et sa variance par $(\mathrm{d}t)^2$. Ce mécanisme est en réalité géré naturellement par la conversion par linéarisation avec la jacobienne comme vu plus haut au paragraphe 3.1.5.a page 32 Par exemple avec l'intégration de la vitesse en position :

$$\mathbf{p}^+ = \mathbf{p} + \mathbf{v}\cdot\mathrm{d}t \quad \Rightarrow \quad \mathbf{P_p} = \mathbf{I} \quad \text{et} \quad \mathbf{P_v} = \mathbf{I}\cdot\mathrm{d}t \tag{3.30}$$

$$\mathbf{P}^+ = \mathbf{P_p}\cdot\mathbf{P}\cdot{\mathbf{P_p}}^T + \mathbf{P_v}\cdot\mathbf{V}\cdot{\mathbf{P_v}}^T = \mathbf{P} + \mathbf{V}\cdot(\mathrm{d}t)^2 \tag{3.31}$$

Erreur de modélisation Mais le type de modèle n'est pas non plus parfaitement approprié, et même s'il est correctement paramétrisé, il n'est pas parfaitement conforme à la réalité de l'intégration continue dans le système physique. Deux cas distincts se présentent alors :

— L'erreur de modélisation est faible, en l'occurrence la grandeur à intégrer est effectivement proche d'être constante sur la période d'intégration $\mathrm{d}t$, dans le sens plutôt monotone. Dans ce cas on modélise cette erreur comme ayant une distribution gaussienne d'écart-type σ *à l'échelle de l'expérimentation*, et σ caractérise l'étendue des valeurs *possibles* de l'erreur pendant $\mathrm{d}t$. Cette erreur déterministe, bien qu'inconnue, est donc multipliée par $\mathrm{d}t$.

— L'erreur de modélisation est grande, en l'occurrence la grandeur à intégrer change souvent pendant la période d'intégration $\mathrm{d}t$, dans le sens où elle n'est pas monotone et se comporte comme un bruit. Dans ce cas on modélise cette erreur comme ayant une distribution gaussienne d'écart-type σ *à l'échelle de la période d'intégration*, et σ caractérise l'étendue des valeurs *prises* par l'erreur pendant $\mathrm{d}t$. Cette erreur aléatoire s'intègre donc en $\sqrt{\mathrm{d}t}$.

Selon les cas on pourra négliger certaines erreurs par rapport à d'autres afin de simplifier les calculs.

Exemples

— Intégration de la vitesse sur une période courte. En général sur une période courte, le modèle de vitesse constante est une bonne approximation, et on a donc une variance de la position somme de la variance de l'erreur de paramétrisation en $(\mathrm{d}t)^2$ et de la variance de l'erreur de modélisation en $(\mathrm{d}t)^2$ également. L'erreur de modélisation est cependant

souvent négligée. Cet exemple illustre ce qu'il se passe dans le cas de l'intégration d'une mesure d'un capteur ou d'une grandeur estimée.

— Intégration de la vitesse sur une période très longue. Sur une période très longue sans observation de la vitesse, c'est-à-dire permettant au système de nombreux changements de vitesse et de direction, on peut considérer la vitesse comme un bruit. Le meilleur modèle est alors de considérer un vecteur vitesse moyen nul, et on aura seulement une erreur de modélisation de variance en $\mathrm{d}t$. Ce cas est cependant rarement applicable car les incertitudes deviennent extrêmement grandes, mais illustre le raisonnement.

— Intégration de la vitesse inconnue sur une période courte. Supposons maintenant qu'aucune mesure ou estimation de la vitesse n'est disponible. On ne peut alors que la supposer nulle. Il n'y a donc pas d'erreur de paramétrisation puisque le modèle vitesse nulle n'a pas de paramètre (le calcul d'intégration est d'ailleurs à proprement parler inexistant). Il reste cependant l'erreur de modélisation avec une variance évoluant en $(\mathrm{d}t)^2$ comme dans le premier exemple. Cet exemple illustre ce qu'il se passe dans le cas d'un modèle de mouvement fixé a priori.

— Biais résultant de l'intégration d'un bruit centré. Comme précédemment, le modèle considère une moyenne nulle du bruit et n'a donc pas de paramètre. L'estimation du biais n'est pas modifiée par l'opération d'intégration, mais son incertitude augmente car il y a une erreur de modélisation, aléatoire cette fois, et dont la variance s'intègre donc en $\mathrm{d}t$. Cet exemple illustre ce qu'il se passe dans le cas de l'estimation des biais d'un capteur.

Ces considérations sont importantes et seront appliquées quand on considèrera l'utilisation de modèles et de capteurs.

3.2 RT-SLAM

La volonté au sein de l'équipe d'utiliser le SLAM comme solution de localisation pratique par défaut sur nos robots a poussé au développement de RT-SLAM, en remplacement de l'implémentation utilisée jusqu'alors, avec deux principaux objectifs :

— Avoir un système suffisamment rapide et souple pour être utilisé en pratique comme outil pour mener à bien des missions plus complexes.

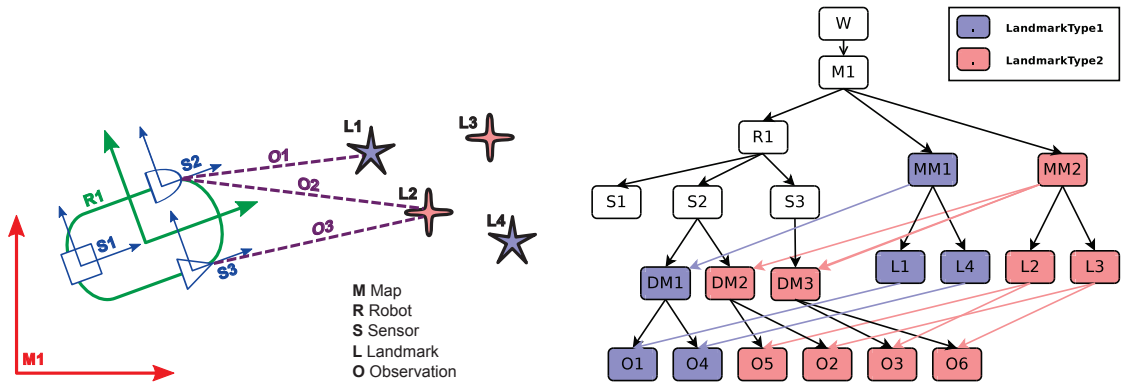
— Être suffisamment générique, modulaire et souple pour rester un outil de recherche et permettre l'implémentation de nouvelles fonctionnalités que ne permettait pas l'ancien système.

En particulier on cherchera à être générique vis à vis des types et paramétrisations d'amers, des fonctions d'obervation des différents capteurs, du nombre de capteurs, des sources de prédiction du filtre, etc.

3.2.1 Architecture

La figure 3.2 page suivante présente les principaux objets définis dans RT-SLAM. Ils englobent les concepts de base d'un système de SLAM : le monde ou environnement (*world*) contient

des cartes (*maps*) ; les cartes contiennent des *robots* et des amers (*landmarks*) ; les robots portent des capteurs (*sensors*) ; et les capteurs font des *observations* des amers. Chacun de ces objets est abstrait, au sens des langages de programmation objet, et peut avoir différentes implémentations selon le besoin. Ils peuvent également contenir d'autres objets, qui peuvent eux-même être génériques.



(a) Les principaux objets dans un contexte de SLAM. Différents robots `Rob` portent plusieurs capteurs `Sen`, fournissant des observations `Obs` des amers `Lmk`. L'état des robots, capteurs et amers sont stockés dans la carte stochastique `Map`. On peut noter qu'il y a une observation par couple capteur-amer.

(b) L'organisation hiérarchique des objets dans RT-SLAM. Chaque carte `M` dans le monde `W` contient des robots `R` and des amers `L`. Un robot possède des capteurs `S`, et une observation `O` est créee pour chaque couple de capteur et amer. Afin de permettre une généricité complète, les gestionnaires de carte `MM` (*map managers*) et de données `DM` (*data manager*) sont introduits, pour permettre d'implémenter différentes stratégies.

FIGURE 3.2 – Les principaux objets dans RT-SLAM

Carte (*Map*) Les cartes contiennent un moteur d'estimation ou d'optimisation : pour le moment RT-SLAM utilise utilise une formulation standard de SLAM-EKF, comme introduite section 3.1.4 page 31. La carte contient donc l'état du filtre, soit l'estimation de l'état du système $\hat{\mathbf{x}}_\mathbf{k}$ et l'estimation de son incertitude $\hat{\mathbf{P}}_\mathbf{k}$.

Robot Les robots peuvent être de différents types selon la façon dont leur état $\mathcal{R}$ est représenté et leur modèle de prédiction. Celui-ci peut-être un simple modèle cinématique (vitesse constante, accélération constante, ...), ou alors un capteur proprioceptif suffisamment rapide, peu bruité, non fautif, et complet (voir détails section 5.1.3 page 87). Un capteur proprioceptif est un exemple d'objet générique contenu dans un objet robot, puisque différents matériels peuvent fournir la même fonctionnalité.

Capteur (*Sensor*) De façon similaire aux robots, les capteurs peuvent également avoir différents modèles : caméra perspective, caméra catadioptrique panoramique, télémètre laser, ... Ils contiennent également un objet pilote générique pour gérer le capteur matériel : caméra firewire, caméra USB, ...

Qui plus est, les capteurs ayant la possibilité d'appartenir à la carte, leur état peut être estimé. Cela ouvre la voie à l'estimation d'autres paramètres tels que le calibrage extrinsèque, erreurs de biais et de gains, et autres.

Amer (*Landmark*) Les amers peuvent être de différents types (points, lignes, plans, ...), et chaque type peut avoir différentes paramétrisations de son état (point euclidien, point *inverse depth*, ...). De plus la paramétrisation d'un amer peut changer au cours du temps, comme expliqué dans la section 4.5.1 page 73. Un objet amer contient également un descripteur qui est utilisé pour l'appariement des données, et qui constitue une description dans l'espace de l'apparence duale à la représentation dans l'espace des états.

Comme le montre la figure 3.2(a) , il est intéressant de remarquer que les objets amers sont communs aux différents capteurs, et tous les capteurs sont donc capables d'observer le même amer (sous réserve qu'ils aient des descripteurs de cet amer compatibles bien sûr). Cela améliore grandement l'observabilité des amers par rapport à un système où les capteurs sont strictement indépendants. Par exemple dans le cas particulier de deux caméras, on reproduit automatiquement l'effet de la stéréovision avec les avantages supplémentaires que les amers peuvent être utilisés même s'ils ne sont visibles que d'une seule caméra ou sont trop lointains pour qu'un système de stéréovision puisse observer leur distance assez précisément et l'exploiter. Ce processus a été introduit dans [Solà *et al.*, 2007] sous la dénomination SLAM *BiCam*.

Observation Dans RT-SLAM la notion d'observation joue un rôle prépondérant. Une observation est un objet réel contenant à la fois des fonctions et des données. Une observation est instanciée pour chaque paire capteur-amer, indépendamment du fait que le capteur ait ou non déjà observé l'amer, puis mise à jour à chaque observation, et sa durée de vie est la même que celle de l'amer associé. Les fonctions contenues sont les modèles classiques de projection et de rétroprojection (qui dépendent des modèles du capteur et de l'amer), tandis que les données stockées consistent en des résultats et variables intermédiaires telles que des matrices Jacobiennes, apparences prédites et observées, innovations, compteurs d'évènements, etc, qui permettent de simplifier et optimiser les calculs.

3.2.2 État du robot

L'état du robot sera toujours au minimum constitué des grandeurs qu'il nous intéresse de connaître, à savoir sa position $\mathbf{p}$ en coordonnées euclidiennes (de dimension 3) et son orientation $\mathbf{q}$ sous forme de quaternion :

$$\mathcal{R} = (\mathbf{p}\ \mathbf{q})^T \tag{3.32}$$

Le choix de la paramétrisation quaternion a l'avantage de ne pas présenter de singularité contrairement aux angles d'euler, ce qui permet de traiter sans problème des angles d'assiette approchant 90°, ce qui peut arriver dans certains cas d'application. L'inconvénient est en revanche la présence d'un degré de liberté supplémentaire, qui impose une contrainte sur le quaternion. Ce problème est géré dans RT-SLAM en normalisant le quaternion dans l'état au

cours de l'étape de prédiction, en tenant compte de la jacobienne de l'opération de normalisation dans les mises à jour de l'incertitude (ainsi l'opération ne coûte rien en temps de calcul puisqu'elle se retrouve « inclue » dans l'opération de prédiction [3]).

Selon les informations utilisées en prédiction ou correction, comme on le verra dans les chapitres suivants, on pourra avoir à ajouter d'autres grandeurs de la dynamique du robot comme des vitesses ou accélérations.

3.2.3 Aspects d'implémentation

a Implémentation de l'EKF

Nous avons utilisé la bibliothèque d'algèbre linéaire Boost uBLAS pour implémenter le filtre de Kalman étendu et toutes les opérations vectorielles et matricielles nécessaires. Il s'agit essentiellement d'un choix historique puisqu'elle était déjà utilisée par l'environnement de développement Jafar dans lequel RT-SLAM a été développé, et qu'aucun besoin ne justifiait de changer pour une autre bibliothèque telle que Eigen par exemple.

Nous avons en particulier fait un usage intensif des structures creuses d'indexation indirecte `indirect_vector` et `indirect_matrix`, qui permettent :

— De gérer dynamiquement l'attribution des élements de l'état du filtre (pour l'état du robot, des capteurs, et des amers).

— De manipuler en toute transparence des sous-vecteurs pour plus de clarté : l'état du robot est un sous-vecteur de l'état du filtre, la position du robot est un sous-vecteur de l'état du robot, les données n'étant stockées qu'à un seul endroit, l'état du filtre.

— D'optimiser de façon simple les calculs pour tenir compte du caractère creux des matrices d'évolution et d'observation, en ne les faisant que sur les blocs non nuls.

L'utilisation de la structure de matrice symétrique `symmetric_matrix` permet également de ne stocker que la moitié de la matrice, de garantir une stabilité numérique accrue, et d'optimiser les calculs en conséquence.

b Gestion des grandeurs estimées

Puisque RT-SLAM a vocation à être générique vis à vis des capteurs à intégrer, il est nécessaire de définir dynamiquement les grandeurs à estimer en fonction des besoins de tous les capteurs ou modèles de mouvement utilisés. Cette tâche doit être centralisée, et ne peut pas être distribuée aux différents capteurs, afin de ne pas dupliquer des grandeurs dans le filtre. Par exemple si deux capteurs ont besoin que la vitesse angulaire soit estimée, cette dernière ne doit se trouver qu'à un endroit dans le filtre, et le modèle ou le capteur qui effectue la prédiction doit savoir qu'elle est estimée pour la mettre à jour.

3. Notons que cela implique cependant de normaliser le quaternion lorsque l'on souhaite l'utiliser en dehors du filtre après la correction, mais seule sa covariance propre doit être mise à jour donc le coût est négligeable par rapport à mettre à jour la covariance de tout l'état.

Nous avons donc implémenté une bibliothèque de grandeurs à estimer concernant le robot dans sa globalité : position, orientation, vitesse, vitesse angulaire, accélération, accélération angulaire, gravité, norme de la gravité, ... Chaque capteur ou modèle doit ensuite déclarer desquels il a besoin, en gardant la possibilité de réserver de l'espace dans le filtre pour son usage privé, par exemple les biais de la centrale inertielle.

c Outils d'analyse

La facilité de débogage est un aspect primordial dans un projet complexe, qui dans notre cas tourne principalement autour de la notion de répétabilité de l'exécution. En effet on le verra beaucoup de mécanismes utilisés impliquent l'utilisation de nombres aléatoires. Ils sont indispensables à certains algorithmes, mais ont également été volontairement utilisés dans beaucoup de mécanismes où il n'étaient pas nécessaires : partout où aucun critère valide ne pouvait motiver une décision particulière, la décision est prise de façon aléatoire. Un exemple typique est dans quelle zone de l'image commencer à chercher des amers : rien ne permet de dire qu'il vaut mieux commencer en bas à droite ou en haut à gauche, donc on le fait aléatoirement. Cela présente un gros avantage en terme d'évaluation de la robustesse du logiciel, puisque l'éxécuter plusieurs fois sur les mêmes données permet de vérifier la sensibilité à ces choix, ce qui revient à simuler des données légèrement différentes. L'inconvénient est que lorsqu'un problème est remarqué dans l'exécution du logiciel, on souhaite pouvoir le répéter en rejouant l'exécution à l'identique. Pour cela il suffit d'initialiser la graine du générateur aléatoire à la même valeur, ce que notre interface permet. Afin d'avoir la garantie que le générateur de nombre aléatoires n'est pas influencé par des bibliothèques extérieures, nous utilisons un générateur séparé pour tous les appels de RT-SLAM dont nous stockons l'état (fonction `rand_r`). En fonctionnement standard, si aucune graine aléatoire n'a été spécifiée, on en génère une au début de l'exécution du programme à partir de la date et de l'identifiant du processus (afin que deux instances démarrées en même temps utilisent une graine différente), puis elle est affichée et enregistrée avec le résultat pour permettre sa réutilisation.

Si cela est suffisant pour faire du débogage en ligne, cela ne l'est pas pour pour identifier un problème qui a eu lieu pendant le fonctionnement en ligne. Pour cela il faut pouvoir rejouer hors ligne exactement la même chose qu'en ligne, ce qui pose plusieurs difficultés :

— pouvoir enregistrer sur le disque l'ensemble des données utilisées sans perturber le fonctionnement en ligne (essentiellement sans le ralentir).

— pouvoir exécuter à l'identique le programme sur ces données, en utilisant les mêmes et en faisant exactement les mêmes calculs avec.

Ces difficultés sont résolues avec les techniques suivantes :

— Afin de ne pas augmenter la charge de calcul du processeur et de garantir la répétabilité, les données sont enregistrées sans compression et sans pertes (format PGM pour les images, format binaire pour les nombres flottants).

— Si enregistrer les données est assez simple, il faut cependant savoir que les entrées-sorties sur le disque sont génératrices de blocages plus ou moins longs (le disque étant partagé par tous les processus du système), et on doit donc utiliser un thread différent

pour réaliser ces écritures si on ne veut pas bloquer l'exécution du programme principal. Nous avons donc implémenté un gestionnaire d'enregistrement qui centralise dans une file d'attente toutes les données à enregistrer et réalise les enregistrements dans un thread séparé.

— Certains problèmes d'exécution que l'on souhaite déboguer sont des plantages provocant l'arrêt brutal du programme, y compris du thread d'enregistrement même s'il n'a pas terminé d'enregistrer toutes les données, notamment celles qui ont causé le problème. Il faut donc intercepter les signaux du système d'exploitation pour laisser les différentes tâches, notamment celle d'enregistrement, terminer leur travail proprement lorsqu'un plantage est signalé.

— Si toutes les données sont enregistrées, toutes ne sont pas nécessairement utilisées lors d'une exécution en ligne, pour cause de retard des données ou de manque de temps de calcul. Il est donc nécessaire d'enregistrer également un journal spécifiant quelles données ont été utilisées, ainsi que les traitements réalisés pour les capteurs pour lesquels on adapte le traitement des données au temps disponible.

— Il est également nécessaire de désactiver dans le compilateur les optimisations des calculs en virgule flottante réalisées par le processeur qui ne garantissent pas la répétabilité.

— Enfin bien sûr comme expliqué auparavant on doit enregistrer la graine aléatoire utilisée.

Une fois cette capacité de répétabilité atteinte, d'autres éléments comme l'écriture de journaux de l'exécution, la possibilité dans l'interface de réaliser le traitement image par image, et d'interagir avec les éléments d'affichage pour obtenir des informations plus détaillées, ont également une grande importance pour permettre de comprendre ce qu'il se passe, détecter et identifier les éventuels problèmes. La figure 3.3 illustre l'affichage de RT-SLAM.

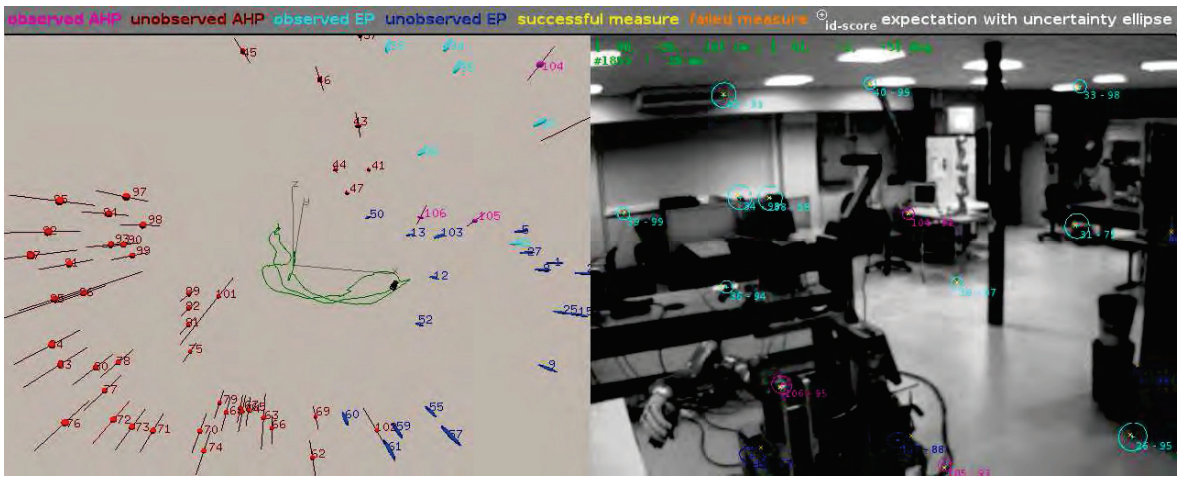


FIGURE 3.3 – Affichage 2D et 3D de RT-SLAM. Les couleurs dans les tons rouges et bleus représentent différentes paramétrisations d'amers, tandis que les couleurs brillantes contre celles plus foncées signalent que l'amer est couramment observé. Les segments illustrent l'incertitude de profondeur quand elle est grande.

3.3 Méthodologie d'évaluation

Les différents résultats obtenus seront présentés tout au long du manuscrit dans des sections dédiées pour illustrer les effets de l'introduction de chaque nouveau capteur, mais également parfois pour justifier des choix de conception.

Nous utiliserons pour cela différentes séquences de données, acquises par différentes plateformes, en suivant une méthodologie précise. Nous allons préciser ces éléments dans cette section.

3.3.1 Les plateformes

a La caméra portée à la main

La première plateforme, utilisée pour faire les premiers tests, est un système de caméra portée à la main et est présentée figure 3.4. Le système est constitué des éléments suivants :

— Une caméra PointGrey Flea2 avec interface Firewire B et capteur 1/3", capable d'acquérir des images de taille VGA (640×480) en niveau de gris à 60 Hz.

— Une centrale inertielle six axes bas coût XSens MTi, fonctionnant à 100 Hz, et dont la sortie de synchronisation est connectée sur l'entrée de déclenchement de la caméra.

— Des marqueurs pour un système de capture de mouvement afin de pouvoir enregistrer une vérité terrain en position et orientation.

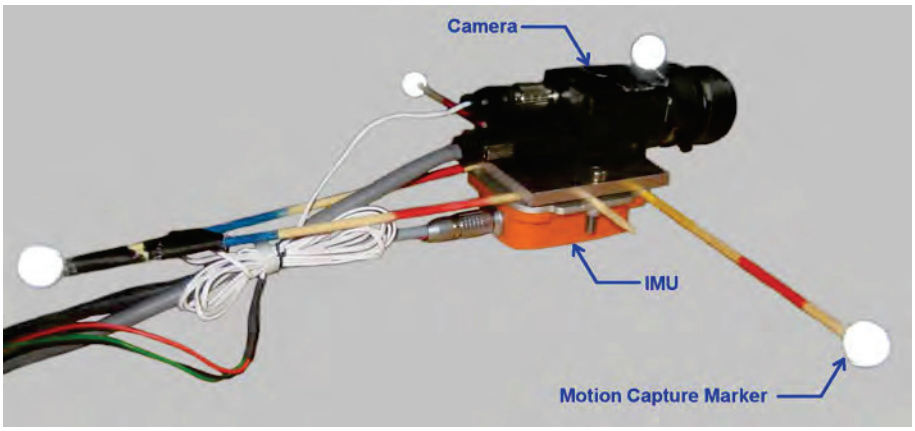


FIGURE 3.4 – La plateforme utilisée *portée à la main*, avec caméra, IMU et marqueurs de capture de mouvement.

b Le robot Mana

La seconde plateforme est le robot Mana du pôle robotique du laboratoire, basé sur une plateforme Segway RMP400 (figure 3.5 page suivante). Il est muni de quatre roues fixes commandables indépendamment (l'interface de commande est cependant en vitesses linéaire et angulaire du robot), et tourne en appliquant un différentiel de vitesse entre ses roues droites et gauches, ce qui implique un glissement des roues sur le sol.

Il est équipé de nombreux capteurs utilisables pour la localisation :

— Une centrale inertielle XSens MTi, identique à celle utilisée pour la plateforme précédente.

— Un GPS-RTK Novatel, capable de fournir une position avec une précision de l'ordre de quelques centimètres quand il reçoit les corrections d'une station de base à proximité, communiquées en par WiFi ou modem série.

— Un gyromètre à fibre optique KVH 5000, fournissant la vitesse angulaire de lacet à une fréquence de 100 Hz, et avec une dérive de l'ordre de quelques degrés par heure.

— Deux caméras AVT Marlin avec interface Firewire A et capteur 2/3", capable d'acquérir des images VGA niveau de gris à 60 Hz, munies d'objectifs Schneider Cinegon de focale 4.8 mm, également reliées à la sortie de synchronisation de la centrale inertielle.

— La plateforme retourne également des données odométriques des quatres roues à la fréquence de 100 Hz (mais les logiciels bas niveau réduisent cette fréquence à 25 Hz).

FIGURE 3.5 – Le robot Mana.

3.3.2 Les séquences

a MOCAP_LOOP

Cette séquence acquise avec le système de caméra tenue à la main propose des mouvements de dynamique modérée pour explorer une grande pièce, y compris une fermeture de boucle. Elle dure 25 secondes (1300 images), et est illustrée figure 3.6 , et dispose des enregistrements de la caméra en VGA à 60 Hz, des données inertielles, et des données de capture de mouvement pour la vérité terrain.

Vidéo : `http://pro.cyril-roussillon.fr/material/phdthesis/mocap_loop.avi`

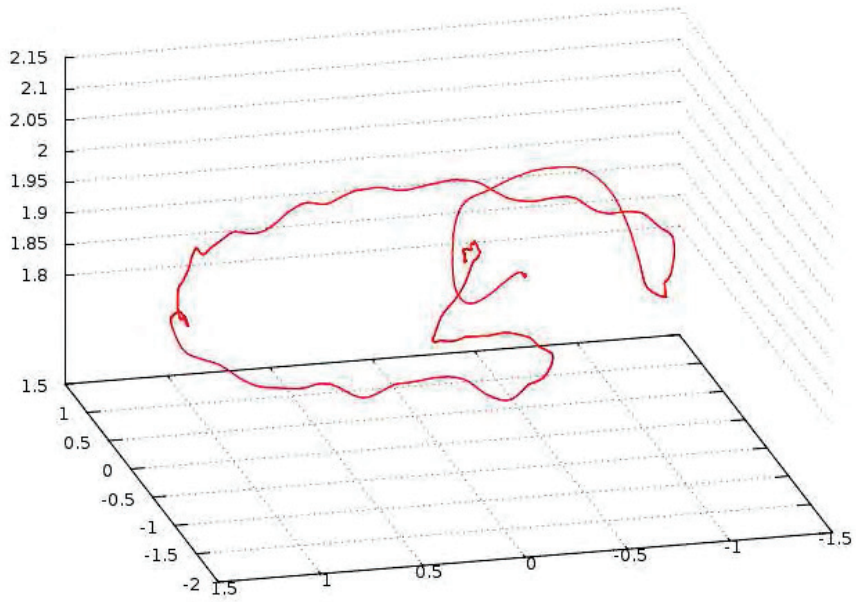


FIGURE 3.6 – Illustration de la séquence MOCAP_LOOP : quelques images et trajectoire estimée par le système de capture de mouvement (vérité terrain).

b MOCAP_FAST

Cette séquence est similaire à la précédente, à part qu'elle est extrêmement dynamique. Les six degrés de liberté sont d'abord successivement excités, puis une oscillation en translation de plus en plus rapide, et enfin une oscillation en rotation de plus en plus rapide, jusqu'à saturer les gyromètres de l'IMU (300 deg/s). Elle dure 65 secondes (3300 images), et est illustrée figure 3.7.

Vidéo : `http://pro.cyril-roussillon.fr/material/phdthesis/mocap_fast.avi`

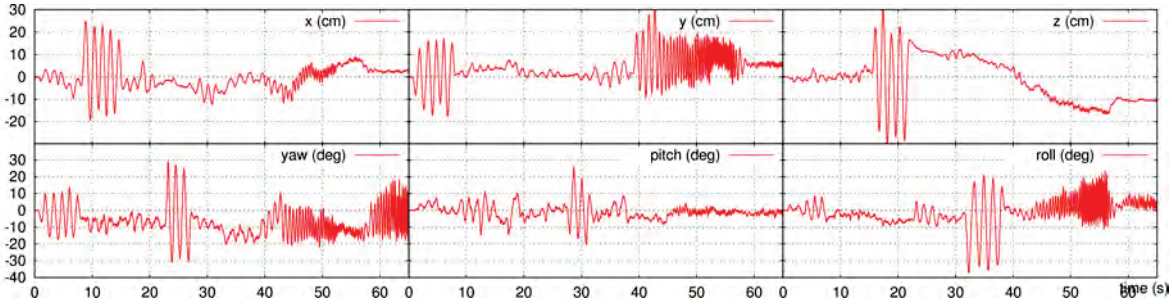


FIGURE 3.7 – Illustration de la séquence MOCAP_FAST. Notons que les dernières images sont floues à cause de la grande vitesse angulaire.

c ESPERCE

Il s'agit d'une séquence d'environ 220 m de longueur (durant 200 secondes pour deux fois 10500 images), acquise par le robot Mana sur un terrain loué par l'Onera sur la commune d'Esperce, en Haute-Garonne. Elle est illustrée figure 3.8 .

Cette séquence a un rôle particulier car elle permettra la comparaison de toutes les solutions utilisables en pratique sur un robot (elle dispose des enregistrements des deux caméras VGA

à 50 Hz, de l'IMU à 100 Hz, de l'odométrie et du GPS-RTK en mode centimétrique).

Elle est également complète dans le sens où elle est très variée, le robot navigant sur différents types de sols (bitume, chemin, herbe rase, herbe un peu haute), traversant différents décors (bâtiments, arbres, plaine) et différentes conditions (soleil, ombre), ainsi que différents reliefs (plat, pentes douces, pentes fortes, franchissement d'obstacles). Cette trajectoire a une longueur limitée à un peu plus de 200 m car l'environnement ne permet pas d'en réaliser une plus longue en conservant une vérité terrain précise. Un autre inconvénient de cette séquence est que le bus Firewire utilisé par les caméras arrivant proche de la saturation, il manque des images et d'autres sont corrompues, comme illustré sur la troisième image de la figure 3.8 (la moitié inférieure de l'image est décalée d'une vingtaine de pixels vers la gauche), mais notre système est robuste par rapport à ces aspects.

Vidéo : `http://pro.cyril-roussillon.fr/material/phdthesis/esperce.avi`

FIGURE 3.8 – Illustration de la séquence ESPERCE. Noter la grande variété de la qualité des images, dans lesquels le soleil est parfois perçu directement : de nombreuses zones sont alors saturées ou sous-exposéees.

d CAYLUS

Il s'agit d'une séquence d'environ 480 m de longueur (qui dure 350 secondes pour 15000 images), toujours acquise par le robot Mana, mais celui-ci navigant cette fois-ci de manière autonome, en construisant un modèle de terrain grâce à son Lidar Velodyne, et en utilisant RT-SLAM pour construire le modèle de terrain et pour rallier les points de passage demandés. Cette acquisition a eu lieu sur le terrain militaire de Caylus, dans le Tarn-et-Garonne.

Cette séquence dispose également des données inertielles, odométriques, et de vérité terrain GPS-RTK centimétrique, mais ne dispose de l'enregistrement que d'une seule caméra, toujours en VGA à 50 Hz.

Le robot navigue essentiellement sur une route, mais coupe régulièrement à travers champ puisque cela est navigable selon ses critères, subissant alors des secousses notables. Il se retrouve également face au soleil lors de la dernière partie.

Vidéo : `http://pro.cyril-roussillon.fr/material/phdthesis/caylus.avi`

FIGURE 3.9 – Illustration de la séquence CAYLUS. Noter à nouveau la grande variété de qualité des images.

3.3.3 La méthodologie

Lorsque l'on souhaite évaluer un système de localisation, on cherche à rendre compte de plusieurs aspects :

— La *précision* d'une part, car on cherche à avoir une erreur la plus faible possible.

— L'*intégrité* d'autre part, car on souhaite connaître la précision de la localisation retournée, afin de prendre en compte toutes les possibilités et de prendre des décisions.

— La *robustesse* enfin, car on souhaite avoir un système qui devient le moins souvent possible inutilisable.

L'étude de ces différentes propriétés passe par une évaluation la plus exhaustive possible :

— différents environnements (types de décor, conditions),

— différents comportements (vitesses, arrêts),

— différentes données (bruit, objets mobiles de dynamiques variées),

— différents capteurs (biais de calibrage, propriétés),

— différentes exécutions (robustesse aux traitements aléatoires, comme vu section 3.2.3.c page 39).

Explorer tous ces critères est une tâche très longue et difficile, mais nous essayons de nous en rapprocher. Pour cela nous utiliserons principalement la séquence de données ESPERCE, car elle est la plus complète en terme de capteurs enregistrés.

L'évaluation d'une solution de SLAM sur une séquence se déroulera ainsi :

— Pour chaque jeux de données, on exécute la solution de SLAM 192 fois avec une graine aléatoire différente (on utilise pour plus de rapidité la ferme de calcul du laboratoire disposant de 192 processeurs). Les réglages utilisés pour ces exécutions sont identiques à ceux utilisés en ligne, qui tournent donc en temps réel.

— On commence par faire une analyse globale de la trajectoire, qui peut facilement être interprétée visuellement [4] : graphique des trajectoires, de l'erreur absolue, et de l'erreur normalisée NEES pour la consistance [Bailey *et al.*, 2006a].

— On découpe ensuite aléatoirement chaque trajectoire en portions représentant une longueur de 40 m, afin de calculer des statistiques chiffrées de précision, notamment des mesures de dérive en $\mathrm{cm}/\sqrt{\mathrm{m}}$ et $\mathrm{mdeg}/\sqrt{\mathrm{m}}$.

— Le but de la mesure de dérive étant d'évaluer les erreurs aléatoires (qui s'intègrent avec la racine carrée de la distance parcourue), elle tend à masquer les effets des biais (qui s'accumulent eux linéairement avec la distance parcourue), à cause de la période d'évaluation relativement courte. On calcule donc en complément la moyenne quadratique de l'erreur de l'ensemble des trajectoires pour rendre compte de l'erreur absolue sur la séquence.

4. Notons que les graphiques présentés pour les 192 exécutions ne seront pas inclus de façon vectorielle, ce qui causerait un fichier beaucoup trop gros et un temps d'affichage beaucoup trop long, mais comme des images matricielles (*raster/bitmap*), avec toutefois une résolution importante de 300 points par pouce

— On tiendra également des statistiques à part des cas de décrochage et divergence le cas échéant.

— Enfin sur la séquence bicam, les données des caméras étant deux fois trop nombreuses (deux caméras à 50 Hz, alors qu'on ne souhaite en traiter qu'une à 50 Hz ou deux à 25 Hz), on dispose en réalité de deux jeux de données correspondant à la même trajectoire. On exécutera donc la solution de SLAM 96 fois sur chaque jeu, en affichant les résultat de deux couleurs différentes afin de mettre en évidence d'éventuels biais liés aux jeux de données. On procédera de même à chaque fois que l'on n'utilisera pas l'ensemble des images.

La vérité terrain est un aspect important dans cette évaluation. On dispose de mesures GPS-RTK de précision centimétrique et à la fréquence de 20 Hz, à part lors de courts masquages (passages en sous-bois principalement) où la précision est dégradée. Nous reconstruisons donc une référence complète de l'origine du robot (au lieu de la position de l'antenne GPS) à partir de la sortie de RT-SLAM intégrant toutes les données disponibles dont le GPS centimétrique, en utilisant plus d'amers que le temps réel ne le permettrait, et en moyennant les résultats de plusieurs exécutions. On pourrais arguer que la référence ainsi obtenue est victime des mêmes biais des capteurs et de RT-SLAM que ce que l'on cherche à évaluer, mais l'intégration du GPS centimétrique à 20 Hz ou de la capture de mouvement millimétrique à 100 Hz contraint très fortement l'estimation, et la trajectoire obtenue est entièrement compatible avec la vérité terrain GPS lorsqu'elle est centimétrique, et la lisse correctement lorsqu'elle est trop imprécise, tout en fournissant une référence angulaire et de vitesse absente de la vérité terrain brute.

3.4 Bilan

Nous avons donc spécifié et développé pas seulement une bibliothèque, mais surtout un véritable *framework* de SLAM, avec une architecture générique, bien définie et ouverte, qui est une base solide pour mener des recherches sur de multiples instances de SLAM. Il est de plus disponible en open-source en ligne : `http://rtslam.openrobots.org`.

Tous les travaux développés dans la suite du manuscrit sont basés sur RT-SLAM, tout comme d'autres thèses au laboratoire se sont appuyées en partie dessus ([Gonzalez, 2013], [Codol, 2013], [Marquez, 2012]) , et il fait l'objet d'une opération de transfert technologique avec la société Nooméo, sous l'égide de la société d'accélération de transfert technologique toulousaine TTT.

Une première instanciation la plus simple possible de RT-SLAM avec seulement une caméra. Présentation des différents types de caméras, types d'amers, et modèles de mouvement utilisables, et introduction des différents traitements propres à la vision.

Chapitre 4

SLAM visuel monoculaire pur

Sommaire

Nous allons voir dans ce chapitre tous les éléments nécessaires à la construction d'un système de SLAM visuel le plus simple qui soit, c'est-à-dire constitué d'une seule caméra. Nous verrons d'abord comment réaliser l'étape de prédiction du filtre de Kalman avec des modèles de mouvement et comment choisir le bon modèle (section 4.1). Puis nous présenterons différents types de caméras utilisables, avec leurs modèles respectifs (section 4.2), avant d'introduire différents types d'amers et de paramétrisations pour ces amers, en expliquant leurs cas d'utilisation, leurs avantages et leurs défauts (section 4.3). Enfin nous verrons comment résoudre tous les problèmes qui se posent en pratique lorsque l'on essaie de faire tourner en ligne un tel algorithme (section 4.4), ainsi que d'autres problèmes moins critiques mais tout de même importants pour obtenir un système robuste (section 4.5).

4.1 Modèles de mouvement

Puisque l'on ne dispose pas d'entrée de commande $\mathbf{u}$, la fonction d'évolution ne dépend que de l'état du robot, et est alors une approximation basée sur la dynamique limitée du système. À complexité croissante on pourra donc avoir un modèle de mouvement à position constante, vitesse constante, accélération constante, jerk constant, etc.

La première question à se poser est quel modèle de mouvement adopter.

Le meilleur choix dépend de plusieurs paramètres :

1. Le type des mesures disponibles. Si la dérivée d'ordre n de la position est mesurée, alors il faut au moins choisir un modèle avec la dérivée d'ordre au moins n constante. En l'occurrence puisque l'on n'observe que la position avec la caméra, on peut démarrer avec un modèle à position constante. Si on avait également un capteur qui mesurait l'accélération, on devrait choisir au moins un modèle à accélération constante.
2. Les rapports entre la durée type d'adéquation du modèle à la dynamique du système en nombre de mesures, l'éloignement en nombre d'ordres de dérivations entre la grandeur mesurée et la grandeur constante du modèle, et l'amplitude du bruit de la mesure. En effet plus ces deux derniers paramètres sont élevés, plus il faut un nombre important de mesures pour correctement l'estimer. Si pendant ce temps la dynamique du système s'est trop éloignée du modèle celui-ci sera toujours mal estimé, avec du retard, et donc peu précis.
3. Plus l'ordre n de la dérivée de la position qui est choisi constant dans le modèle est élevé, plus la divergence sera rapide en l'absence d'observations pour le stabiliser, et difficile à rattraper. À précision équivalente un modèle d'ordre inférieur sera donc plus robuste.

On a donc un compromis entre précision et observabilité du modèle : plus l'ordre n de dérivation est élevé, plus le modèle est précis en théorie, mais plus son observabilité est faible en pratique, le point optimal dépendant des caractéristiques du système (dynamique, fréquence d'échantillonnage, grandeurs mesurées, bruit des mesures).

Nous avons implémenté les modèles à position, vitesse, accélération et jerk constant, dont une évaluation est présentée table 4.1 sur une séquence typique de caméra portée à la main. On constate qu'à 50 Hz le modèle à vitesse constante est le plus précis sur ce type de trajectoire,

suivi par le modèle à accélération constante (les écarts sont statistiquement significatifs). À 25 Hz ou avec une dynamique plus forte, les modèles à position et jerk constant ne sont pas assez stables et ne fonctionnent plus du tout : les erreurs et zones de recherche trop grandes causent des difficultés d'appariement des amers qui font décrocher le SLAM. À cette fréquence le modèle à accélération constante commence également à devenir instable. Les résultats en utilisant un capteur inertiel, introduit chapitre 5 page 85, sont également présentés pour comparaison. On peut d'ores et déjà constater que la précision est sans commune mesure avec celle des modèles de mouvement, et que la sensibilité à la fréquence et à la dynamique est bien moindre.

Les modèles à vitesse et accélération constante sont donc les seuls modèles viables en pratique pour nos applications, et sont maintenant présentés plus en détail.

Prédiction	**Innovation moy. (pix)**	**Innovation max. (pix)**	**Correction position moy. (mm)**	**Correction orientation moy. (mdeg)**
50 Hz				
Position constante	7.09	32.4	8.7	336
Vitesse constante	2.00	10.4	5.8	87
Accélération constante	2.23	14.2	11.0	92
Jerk constant	3.06	19.6	18.8	128
Capteur inertiel	0.60	6.2	0.6	12
25 Hz				
Position constante	–	–	–	–
Vitesse constante	5.89	35.9	12.0	250
Accélération constante	8.03	44.9	20.1	303
Jerk constant	–	–	–	–
Capteur inertiel	0.83	5.7	1.3	22

TABLE 4.1 – Comparaison de différents modèles de prédiction de mouvement, effectués sur une séquence de 20 s caméra portée à la main, de dynamique raisonnable pour que tous les modèles fonctionnent (accélérations maximales d'environ $10\,m/s^2$ et vitesses angulaires maximales d'environ $150\,\mathrm{deg}/s$). L'innovation est la distance dans l'image entre la position prédite d'un amer et sa position mesurée, et la correction est la distance entre la position prédite du robot et sa position après correction par les amers, facteur d'échelle normalisé. On notera que les innovations, et en particulier les valeurs maximales, ne sont qu'indicatives de la précision du modèle car elles intègrent l'incertitude de localisation des amers. Les valeurs sont issues d'un moyennage sur 12 exécutions.

4.1.1 Modèle à vitesse constante

L'état $\mathcal{R}$ du robot est donc étendu avec les paramètres du modèle que l'on souhaite estimer pour pouvoir l'appliquer :

— $\mathbf{v} = (v_x\ v_y\ v_z)$ la vitesse linéaire du robot dans le repère global.

— $\mathbf{w} = (w_x\ w_y\ w_z)$ la vitesse angulaire dans le repère local du robot.

Le choix de repère d'expression de ces grandeurs est motivé par la meilleure linéarité de l'équation d'évolution, comme on va le voir.

L'état complet du robot est donc :

$$\mathcal{R} = (\mathbf{p}\ \mathbf{q}\ \mathbf{v}\ \mathbf{w})^T \tag{4.1}$$

L'équation d'évolution du système est alors :

$$\begin{aligned} \mathbf{p}^+ &= \mathbf{p} + \mathbf{v} \cdot \mathrm{d}t & (4.2)\\ \mathbf{q}^+ &= \mathbf{q} \otimes \mathrm{v2q}\,(\mathbf{w} \cdot \mathrm{d}t) & (4.3)\\ \mathbf{v}^+ &= \mathbf{v} & (4.4)\\ \mathbf{w}^+ &= \mathbf{w} & (4.5) \end{aligned}$$

où $\otimes$ représente le produit de quaternion et v2q l'opération de conversion d'un vecteur rotation en un quaternion.

Il est également nécessaire de mettre à jour les covariances :

$$\begin{aligned} \mathbf{V}^+ &= \mathbf{V} + \mathbf{V_p} & (4.6)\\ \mathbf{W}^+ &= \mathbf{W} + \mathbf{W_p} & (4.7) \end{aligned}$$

où $\mathbf{V_p}$ et $\mathbf{W_p}$ sont les perturbations de la vitesse linéaire et de la vitesse angulaire, qui sont un réglage qui doit refléter l'écart de la dynamique réelle du système à celle du modèle, en l'occurrence l'accélération maximale du système. En général on considère de plus que les perturbations sur les différents axes sont similaires et indépendantes, et la covariance de l'intégration d'une variable aléatoire gaussienne variant proportionnellement à $\mathrm{d}t$, on a :

$$\begin{aligned} \mathbf{V_p} &= \sigma_{V_p}^2 \cdot \mathrm{d}t \cdot \mathbf{I}_3 & (4.8)\\ \mathbf{W_p} &= \sigma_{W_p}^2 \cdot \mathrm{d}t \cdot \mathbf{I}_3 & (4.9) \end{aligned}$$

La covariance du reste de l'état est mise à jour avec la jacobienne de l'équation d'évolution.

Remarque 1 Avec un modèle de mouvement, les perturbations ne sont en réalité pas gaussiennes centrées mais très biaisées (la grandeur considérée constante changeant lentement à l'échelle de $\mathrm{d}t$, elle restera le plus souvent du même côté de son estimée). Les covariances dans les équations 4.8 et 4.9 peuvent donc aussi être considérées proportionnelles à $(\mathrm{d}t)^2$ au lieu de $(\mathrm{d}t)$. Une preuve expérimentale se trouve dans la table 4.1 page précédente puisque les erreurs sont multipliées par 2 quand la période est multipliée par 2.

Remarque 2 Si les paramètres de perturbation doivent effectivement en théorie être basés sur l'accélération maximale du système, ils dépendent en pratique du facteur d'échelle qui n'est pas déterminé dans notre cas de SLAM avec une seule caméra, mais qui est heureusement peu variable une fois l'hypothèse de profondeur des amers fixée. Il est donc nécessaire d'ajuster ces paramètres expérimentalement.

4.1.2 Modèle à accélération constante

De façon similaire, on étend l'état du robot avec les accélérations linéaires $\mathbf{v_a}$ et angulaires $\mathbf{w_a}$, définies dans les mêmes repères que les vitesses correspondantes :

$$\mathcal{R} = (\mathbf{p}\ \mathbf{q}\ \mathbf{v}\ \mathbf{w}\ \mathbf{v_a}\ \mathbf{w_a})^T \tag{4.10}$$

L'équation d'évolution est également complétée :

$$\mathbf{v}^+ = \mathbf{v} + \mathbf{v_a} \cdot \mathrm{d}t \tag{4.11}$$
$$\mathbf{w}^+ = \mathbf{w} + \mathbf{w_a} \cdot \mathrm{d}t \tag{4.12}$$
$$\mathbf{v_a}^+ = \mathbf{v_a} \tag{4.13}$$
$$\mathbf{w_a}^+ = \mathbf{w_a} \tag{4.14}$$

Et les matrices de covariance de $\mathbf{v_a}$ et $\mathbf{w_a}$ sont mises à jour avec les paramètres de perturbation comme le sont celles de $\mathbf{v}$ et $\mathbf{w}$ dans le modèle à vitesse constante, la covariance du reste de l'état étant toujours mise à jour avec la jacobienne de l'équation d'évolution.

4.2 Modèles de caméra

4.2.1 Caméra perspective

Les modèles *pinhole* avec distortion radiale décrivent de manière approchée mais assez fidèlement les caméras classiques dites perspectives.

Le repère d'une caméra selon la convention habituelle et défini avec l'axe **x** vers la droite, **y** vers le bas, et **z** vers l'avant, habituellement noté RDF pour *Right,Down,Front* (figure 4.1).

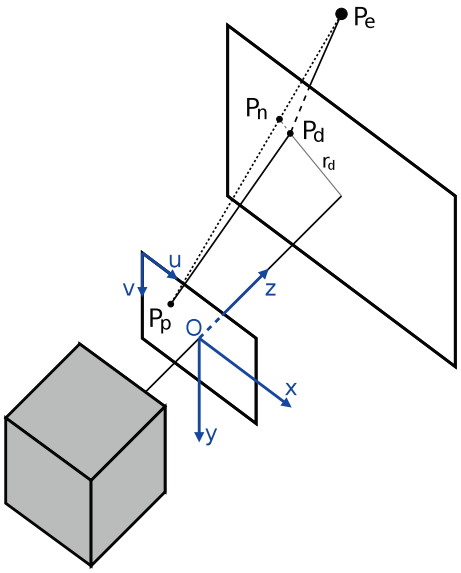


FIGURE 4.1 – Le modèle de caméra perspective

Soit un point $\mathbf{p_e}$ dans l'espace, dont les coordonnées sont exprimées dans le repère de la caméra :

$$\mathbf{p_e} = \begin{pmatrix} x_e \\ y_e \\ z_e \end{pmatrix} \tag{4.15}$$

On définit la projection normalisée de ce point :

$$\mathbf{p_n} = \begin{pmatrix} x_n \\ y_n \end{pmatrix} = \begin{pmatrix} x_e/z_e \\ y_e/z_e \end{pmatrix} \tag{4.16}$$

Un modèle de distortion est alors appliqué, par exemple le modèle polynomial de distortion radiale de paramètres k_1, k_2, k_3 :

$$r = \sqrt{x_n^2 + y_n^2} \tag{4.17}$$

$$r_d = \left(1 + k_1 \cdot r^2 + k_2 \cdot r^4 + k_3 \cdot r^6\right) \cdot r \tag{4.18}$$

$$\mathbf{p_d} = \frac{r_d}{r} \cdot \mathbf{p_n} \tag{4.19}$$

Il ne reste alors plus qu'à appliquer le modèle *pinhole* constitué des focales horizontale f_1 et verticale f_2, et des centres optiques horizontal c_1 et vertical c_2, pour obtenir les coordonnées en pixel dans le plan image :

$$\mathbf{p_p} = \begin{pmatrix} x_d \cdot f_1 + c_1 \\ y_d \cdot f_2 + c_2 \end{pmatrix} \tag{4.20}$$

Cette dernière équation peut encore s'écrire sous forme matricielle en coordonnées homogènes :

$$\mathbf{p_p} = \begin{pmatrix} x_p \\ y_p \\ 1 \end{pmatrix} = \begin{pmatrix} f_1 & 0 & c_1 \\ 0 & f_2 & c_2 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_d \\ y_d \\ 1 \end{pmatrix} \tag{4.21}$$

Certains modèles incluent des termes de distortion tangentielle, ou un paramètre affine pour tenir compte de la non rectangularité des pixels (dit paramètre de *skew*), mais ils sont négligeables sur les caméras modernes et nous ne les avons pas pris en compte pour simplifier le modèle.

4.2.2 Caméra omnidirectionnelle

Nous nous intéressons au cas des caméras dites à projection centrale, c'est-à-dire dont tous les rayons mesurés s'intersectent virtuellement en un unique point, qui sont plus facilement modélisables et utilisables car elles conservent des propriétés habituelles telles que la géométrie épipolaire, et permettent la génération d'images perspectives classiques [Scaramuzza, 2008].

C'est le cas de certains systèmes dioptriques, contenant seulement des lentilles (lentille *fisheye*), et de certains systèmes catadioptriques, contenant des lentilles et des miroirs (caméra perspective avec miroir hyperbolique, ou caméra orthographique avec miroir parabolique).

[Barreto, 2003] propose un modèle couvrant tous les systèmes à projection centrale, constitué d'une caméra perspective et d'une sphère unité se situant à la distance ξ de la caméra, sur

laquelle le point dans l'espace est projeté, puis transformé en fonction de la forme du miroir avec un paramètre ϕ, avant d'être à nouveau projeté dans la caméra avec un modèle *pinhole*. Les paramètres ξ et ϕ sont calculés selon la table 4.2, puis la projection d'un point $\mathbf{p_e} = (x_e\ y_e\ z_e)^T$ dans l'espace en les coordonnées en pixels $\mathbf{p_p}$ dans l'image se fait alors comme suit :

$$\mathbf{p_p} = \begin{pmatrix} x_p \\ y_p \\ 1 \end{pmatrix} = \begin{pmatrix} f_1 & 0 & c_1 \\ 0 & f_2 & c_2 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \xi - \phi & 0 & 0 \\ 0 & \xi - \phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \frac{x_e}{\sqrt{x_e^2 + y_e^2 + z_e^2}} \\ \frac{y_e}{\sqrt{x_e^2 + y_e^2 + z_e^2}} \\ \frac{z_e}{\sqrt{x_e^2 + y_e^2 + z_e^2}} + \xi \end{pmatrix} \tag{4.22}$$

	Equation du miroir	ξ	ϕ
Parabolique	$\sqrt{x^2 + y^2 + z^2} = 2p - z$	1	$1 + 2p$
Hyperbolique	$\frac{\left(z - \frac{d}{2}\right)^2}{\frac{1}{4}\left(\sqrt{d^2 + 4p^2} - 2p\right)^2} - \frac{x^2 + y^2}{p \cdot \left(\sqrt{d^2 + 4p^2} - 2p\right)} = 1$	$\frac{d}{\sqrt{d^2 + 4p^2}}$	$\frac{d + 2p}{\sqrt{d^2 + 4p^2}}$

TABLE 4.2 – Calcul des paramètres ξ et ϕ et fonction du type de miroir et de son équation en fonction de la distance d entre la caméra et le foyer de la cônique, et du paramètre p de la cônique.

Une correction de la distortion comme celle présentée pour les caméras perspective peut également être introduite juste avant l'application du modèle *pinhole*.

4.2.3 Considérations pratiques

a Calibrage d'une caméra

Lorsque l'on souhaite utiliser une caméra réelle, il est nécessaire de déterminer les paramètres du modèle associés à cette caméra. Cette étape d'étalonnage est couramment appelée calibrage. Elle est effectuée par une optimisation pour minimiser une erreur de reprojection sur des points extraits d'une mire de calibrage, souvent les coins d'un échiquier.

Différents outils sont disponibles pour réaliser cette tâche, plus ou moins automatisés. Nous utilisons pour les caméras perspective un outil basé sur la bibliothèque OpenCV qui permet d'extraire automatiquement les coins d'un échiquier et de réaliser l'optimisation. D'autres outils sont disponibles sur Internet, comme la *Camera Calibration Toolbox for Matlab* [Bouguet, 2013] de Jean-Yves Bouguet pour les caméras perspectives. Pour les caméras panoramiques nous utilisons la *Omnidirectional Calibration Toolbox for Matlab* [Mei, 2013] de Christopher Mei.

Une précaution est à prendre lors de l'utilisation de ces outils, en plus des recommandations générales indiquées dans leur documentation. En effet puisque nous ne tenons pas compte des paramètres de distortion tangentiels et du paramètre de *skew*, il faut les forcer à 0 durant le calibrage.

Il est par ailleurs nécessaire de déterminer la position précise du point focal de la caméra, qui doit être utilisé comme origine de la caméra. Il est situé à une distance égale à la longueur focale de l'objectif devant le centre du capteur physique. Il est donc nécessaire de se référer à la documentation constructeur de la caméra pour identifier la position du capteur physique (plan image). Par exemple sur une caméra AVT Marlin, il est aligné avec la face avant du boîtier rectangulaire de la caméra, tandis que sur une caméra PointGrey Flea2 il est situé 5.64 mm en retrait.

b Domaine de validité de la distortion

On a vu section 4.2.1 page 55 (équation 4.19) que l'on utilise la fonction de distortion suivante :

$$d(r) = \left(1 + k_1 \cdot r^2 + k_2 \cdot r^4 + k_3 \cdot r^6\right) \cdot r = r_d \tag{4.23}$$

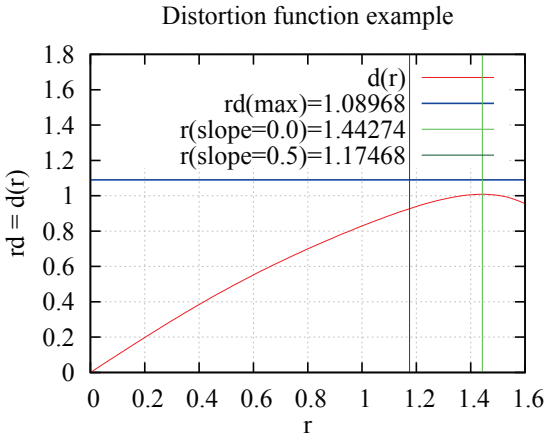
C'est une approximation polynomiale de la distortion réelle de la caméra dans le domaine que constitue l'image. Afin de savoir si un point dans l'espace est visible dans l'image, nous devons le projeter et donc appliquer ce modèle de distortion. Si le point n'est pas visible, nous avons en réalité appliqué le modèle de distortion en dehors du domaine pour lequel il a été optimisé. Or un polynôme de degré élevé comme ce modèle n'est pas nécessairement une fonction monotone, et on constate en pratique que les polynômes utilisés avec trois coefficients de distortion ne le sont pas, et sont même capables de ramener des points qui devraient être très largement en dehors de l'image à l'intérieur de l'image. Si on applique sans vérification le modèle on obtient ainsi des projections dans l'image d'amers qui ne devraient pas l'être, et qui se manifestent sous forme d'amers traversant l'image à grande vitesse sans jamais être appariés (et pour cause !).

Il est donc nécessaire d'éviter d'appliquer le modèle là où il n'est plus valable, le problème étant qu'on a besoin de l'appliquer pour savoir s'il est valable. On peut cependant vérifier s'il y a un risque que le modèle ramène un point hors de l'image dans l'image en étudiant la monotonicité de la fonction de distortion $d(r)$. En effet on sait que cette fonction est monotone croissante dans son domaine de validité, celui de l'image, et qu'elle risque de poser problème si elle devient décroissante.

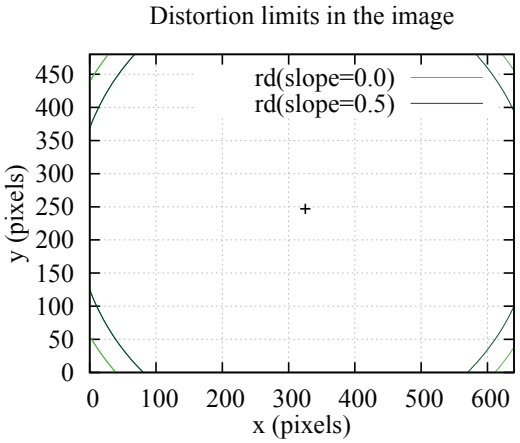
Mais parfois cette inversion de la monotonie de la fonction distortion peut arriver à l'intérieur de l'image. Ce phénomène apparait avec des distortions assez fortes, lorsque les coins de l'image n'ont pas été explorés lors du calibrage. Dans ce cas la fonction de distortion est invalide bien avant cette inversion, puisque cela signifie qu'il est impossible d'atteindre les coins, et qu'au moment de l'annulation de la pente de la fonction de distortion une plage d'angles infinie se projette ponctuellement.

On a adopte donc la convention de limiter la validité de la fonction de distortion lorsque la pente de celle-ci passe en dessous de 0.5, ce qui correspond à une distortion très forte qui rend de toute manière la zone correspondante de l'image très difficile à explorer, ainsi que par sécurité lorsque ceci arrive à l'extérieur de l'image d'appliquer un coefficient de sécurité de 0.95 sur le rayon maxi des points dans l'image afin d'éviter d'utiliser la fonction de distortion dans une zone où elle n'a certainement pas été optimisée. La figure 4.2 illustre ce phénomène par un exemple réel, et montre que ces critères n'enlèvent au final qu'une zone très réduite de

l'image.



(a) On voit ici que $d(r)$ n'atteint jamais la valeur r_d maximale, correspondant aux coins de l'image (demi diagonale de l'image divisée par la focale).



(b) Les zones exclues pour cause de distortion trop forte, dans les coins à l'extérieur des cercles, sont cependant limitées dans les deux cas.

FIGURE 4.2 – Exemple de limitation de la validité de la fonction de distortion pour une caméra AVT Marlin avec objectif Schneider Cinnegon 4.8 mm, d'image de taille 640×480, dont un calibrage donne le centre optique en $(325.29, 246.89)$, les focales $(374.80, 374.69)$, et les coefficients de distortion $(-0.2640, 0.1259, -0.0329)$.

Ces calculs ne dépendant que du calibrage de la caméra, ils n'interviennent qu'une seule fois à l'initialisation du programme. Par simplicité on se permet donc une résolution numérique par dichotomie plutôt que de rechercher une solution analytique dont l'existence n'est pas garantie.

c Inversion de la distortion

Si la projection d'un point dans le plan image ne pose pas de problème et utilise directement les paramètres de distortion fournis par l'étape de calibrage, la rétroprojection d'un point de l'image dans l'espace nécessite d'inverser les équations du modèle présentées section 4.2.1 page 55, et notamment l'équation de distortion radiale 4.19 page 56 qui n'est pas inversible de façon exacte.

La solution adoptée, proposée dans [Solà, 2007] est d'approcher la fonction inverse avec un polynôme du même type :

$$r = \left(1 + c_1 \cdot r_d^2 + c_2 \cdot r_d^4 + c_3 \cdot r_d^6 + \ldots\right) \cdot r_d \tag{4.24}$$

$$\mathbf{P_n} = \frac{r}{r_d} \cdot \mathbf{P_d} \tag{4.25}$$

Pour cela on calcule un échantillon de couples r, r_d avec $r_d = d(r)$ et on optimise les coefficients par une minimisation aux moindres carrés.

Le domaine d'échantillonnage de ces points est très importants, puisqu'il faut garantir qu'il couvre entièrement la partie de l'image utilisée (afin d'y limiter l'erreur), mais pas plus, car ces

contraintes supplémentaires vont dégrader l'approximation dans le domaine utile. En pratique on réutilise donc les calculs effectués dans le paragraphe précédent pour déterminer ce domaine d'optimisation de la fonction inverse.

Pour des caméras avec une forte distortion il est nécessaire d'utiliser plus de coefficients c_i que de coefficients k_i pour être suffisamment précis, par exemple un de plus. On obtient alors des erreurs de l'ordre du dixième de pixels sur le domaine utile.

d Contrôle de l'exposition des caméras

Dans certaines conditions difficiles, il peut être avantageux de contrôler précisément la durée d'exposition de la caméra. En effet lorsque la dynamique du mouvement est trop importante par rapport à la luminosité de la scène, les réglages automatiques de la caméra peuvent décider d'une durée d'exposition trop longue, dans le but de minimiser le bruit présent dans l'image, mais avec pour conséquence de provoquer du flou de bougé.

Or il est préférable dans une certaine mesure pour une application de localisation d'avoir une image bruitée, ce qui est assez bien accepté par les algorithmes de traitement d'image, plutôt qu'une image avec du flou de bougé, ce qui a pour effet de modifier les formes et perturbe plus les algorithmes de traitement d'image, et ne permet pas de dater précisément les images.

Toutes les caméras ne permettent pas d'accéder à des réglages si fins, mais les caméras haut de gamme le permettent en général, notamment les caméras avec interface Firewire que nous utilisons. Différents moyens peuvent être disponibles pour contrôler la durée d'exposition :

- réglage fixe logiciel,
- réglage automatique avec borne maximale fixée logiciellement,
- utilisation d'un signal déclencheur (*trigger*) externe sur une broche de la caméra, et configuration de la caméra pour interpréter la largeur de l'impulsion comme durée d'exposition.

Un réglage manuel, par voie logiciel ou de déclencheur externe présente l'inconvénient de devoir être ajusté selon la luminosité générale de la scène, et idéalement adaptée dynamiquement si celle-ci change trop. Le mode automatique avec borne maximale proposé par les caméras AVT Marlin permet donc de se décharger de cette tâche sur la caméra, et est donc le plus intéressant, mais malheureusement pas disponible sur toutes les caméras.

HDR Certaines caméras, comme les caméras AVT Marlin, proposent également un mode HDR (*High Dynamic Range*), permettant de mieux voir à la fois les zones très sombres et les zones très claires, ce qui peut être utile dans un environnement très ensoleillé avec des zones d'ombre par exemple. Ce système, dit *multiple slope*, qui consiste à réinitialiser les pixels saturés à certains instants pendant la durée d'exposition, permet de passer d'une dynamique de 60 dB pour le capteur simple à près de 100 dB. Un exemple d'une telle image est donné figure 4.3(a) . Son utilisation pose cependant plusieurs difficultés :

- il n'y a plus de relation linéaire entre la luminosité physique de l'objet et la luminosité des pixels dans l'image, ce qui pénalise les algorithmes de traitement d'image qui normalisent

la luminosité pour rechercher une invariance. Il est cependant possible de calibrer cet effet à partir des équations d'intégration du mode HDR et de le corriger.

— la durée d'exposition nécessaire pour exposer les zones les plus sombres peut être assez longue et à l'origine de flou de bougé.

— l'exposition différentes des zones de différentes luminosité sur un même objet ou entre un objet mobile et ce qui se trouve derrière lui provoque des effets visuels non linéaires et non corrigeables. Ceci est illustré figure 4.3(b).

(a) Une image classique en intérieur devant une fenêtre avec un exemple de flou de bougé sur un objet mobile (une prise multiples blanche).

(b) Une image en mode HDR : on constate que les zones très lumineuses à travers la vitre ne sont plus surexposées, tandis que les zones sombres dans les recoins à l'intérieur ne sont pas non plus particulièrement sous-exposées, le contraste étant lui logiquement diminué. Le flou de bougé est différent : l'objet mobile laisse une grande trace blanche en surimpression et est plus déformé.

FIGURE 4.3 – Exemple d'image HDR avec la caméra AVT Marlin

Obturateur On notera enfin qu'il est indispensable d'utiliser une caméra à obturateur global (*global shutter*), qui fait l'acquisition de l'image en une seule fois, et qui permet donc d'affecter une seule date à l'ensemble de l'image. Les caméras bas de gamme comme les webcams sont en général à obturateur roulant (*rolling shutter*), et capturent les lignes une par une, qui sont donc datées chacune différemment (on reconnait assez facilement ces caméras à la déformation de leurs images lorsque la caméra est secouée). Utiliser de telles caméras

nécessiterait un travail supplémentaire délicat afin de tenir compte de la contrainte d'observer les amers dans l'ordre chronologique, l'interdépendance entre date de l'observation et position dans l'image rendant plus difficile l'utilisation d'une prédiction, et les déformations géométriques dues aux mouvements de la caméra pouvant compromettre les descripteurs et nécessiter une correction.

4.3 Modèles d'amers

Différents types d'amers peuvent être utilisés, et chacun peut être paramétrisé de différentes manières. La paramétrisation des amers et le modèle de la caméra sont fortement liés, puisque c'est leur combinaison qui constitue la fonction d'observation h de l'amer utilisée dans le filtre de Kalman, qui relie le modèle de l'amer physique dans l'espace à sa projection dans l'image, qui est mesurée.

4.3.1 Utilisation de points

Les points sont les types d'amers les plus utilisés car ils sont simples et faciles à détecter dans une image.

Si la paramétrisation de la projection d'un point dans l'image est toujours sa position cartésienne (u, v) dans l'image, selon la situation différentes paramétrisations du point dans l'espace sont plus ou moins bien adaptées.

a Paramétrisation euclidienne

Il s'agit de la paramétrisation la plus intuitive et la plus simple des points, à savoir ses coordonnées euclidiennes dans le repère global :

$$\mathbf{p_E} = (x\ y\ z)^T \tag{4.26}$$

Cette paramétrisation est celle utilisée dans la définition des modèles de caméra, et l'équation d'observation d'un point euclidien est donc directement l'équation de projection d'un point dans l'espace dans l'image du modèle de caméra, telle que présentée sections 4.2.1 page 55 et 4.2.2 page 56.

Si cette paramétrisation peut être suffisante avec un système stéréoscopique, elle présente un sérieux problème avec un capteur qui n'observe que les angles et pas la distance (appelé *bearing-only*), comme c'est le cas par exemple avec une caméra seule.

En effet le filtre de Kalman impose d'une part que les incertitudes soient représentables par des fonctions gaussiennes, et d'autre part son extension impose que les équations d'observation et d'état soient peu non linéaires sur la plage d'incertitude [1]. Or pour un amer observé une seule

1. Il faut bien retenir que la non linéarité de la fonction est toujours à mettre en regard avec la plage d'incertitude sur laquelle on souhaite exploiter l'approximation linéaire. Si la fonction est globalement linéaire, elle l'est à toute échelle, pour n'importe quelle incertitude, aussi grande soit-elle. Mais même si au contraire la

fois par une caméra, aucune information n'est disponible sur la distance à laquelle il se trouve, ce qui correspond à une incertitude et un écart-type infinis. La fonction d'observation n'étant pas linéaire, elle ne l'est pas du tout sur cette plage d'incertitude, ce qui est en contradiction avec les hypothèses de l'EKF, et empêche son intégration immédiate dans le filtre sous cette forme.

C'est pourquoi les premières solutions au SLAM monoculaire présentées [Davison, 2003] [Lemaire *et al.*, 2005] ont été dites à *initialisation retardée*, et consistaient à estimer à l'extérieur du filtre la distance de l'amer, et à ne l'intégrer dans le filtre qu'une fois l'incertitude sur cette distance suffisamment réduite pour satisfaire les conditions de normalité et de linéarité. L'inconvénient est que les amers ne sont pas exploités pour la localisation du robot pendant les premières images, alors qu'ils apporteraient déjà de l'information sur l'orientation du robot. D'autre part si l'amer est très éloigné, son incertitude en coordonnées euclidiennes ne se réduira jamais suffisamment et l'amer ne sera jamais exploité pour la localisation du robot, alors qu'un amer à l'infini représente une ancre remarquable pour l'orientation du robot.

La solution utilisée actuellement est venue de plusieurs modifications de la paramétrisation.

b Paramétrisation AHP

Les premières paramétrisations pour résoudre ce problème ont été proposées par [Montiel, 2006], et consistent à utiliser une paramétrisation de type polaire, associée à l'ajout d'une *ancre*, qui est la position en coordonnées euclidiennes du robot au moment de la première observation de l'amer. Ceci permet de limiter le nombre de paramètres d'état impactés par l'inconnue de distance à un seul, ce qui est nécessaire car les relations non linéaires entre les paramètres ne peuvent être représentées correctement par les corrélations croisées de la matrice de covariance. Cette représentation polaire est alors modifiée afin de ne plus utiliser directement la distance d du point, inconnue, mais son inverse $\rho = 1/d$. Ainsi ce paramètre appelé *inverse depth* ne varie que dans la plage $[0, 1]$ lorsque l'amer se trouve dans une plage de distance entre 1 m et l'infini, plage que l'on peut couvrir avec une gaussienne d'écart-type fini, et sur laquelle la fonction d'observation présente des conditions de linéarité acceptables.

Une autre amélioration a été proposée dans [Solà, 2005], et consiste à utiliser une représentation homogène avec vecteur direction pointant sur l'amer au lieu des deux angles de la représentation polaire, l'objectif étant d'améliorer la linéarité des équations d'observation et donc le comportement du filtre de Kalman. Cette paramétrisation que nous avons utilisée, présentée figure 4.4 page suivante est appelée AHP pour *Anchored Homogeneous Point* :

$$\mathbf{p_{AH}} = \left(\mathbf{p_0}^T\ \mathbf{u}^T\ \rho\right)^T \tag{4.27}$$

$$= \left(x_0\ y_0\ z_0\ u_x\ u_y\ u_z\ \rho\right)^T \tag{4.28}$$

où $\mathbf{p_0}$ est l'ancre, $\mathbf{u}$ le vecteur direction et ρ la distance inverse entre l'ancre et l'amer.

La relation entre le point euclidien et le point AHP est immédiate :

$$\mathbf{p_E} = \mathbf{p_0} + \frac{\mathbf{u}}{\rho} \tag{4.29}$$

courbure de la fonction est très grande en un point, si les incertitudes en question sont très faibles, elle peut être suffisamment linéaire sur cette plage d'incertitude très réduite.

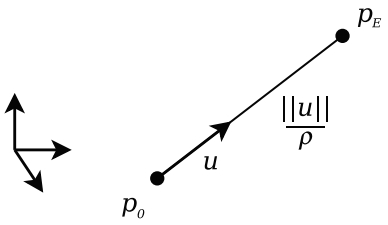


FIGURE 4.4 – La paramétrisation de points AHP

La fonction d'observation est alors la composition de l'équation de projection d'un point euclidien du modèle de caméra avec la conversion d'un point AHP en un point euclidien (équation 4.29 page précédente).

Initialisation Une certaine liberté est présente dans l'initialisation du paramètre non observable de la paramétrisation, à savoir la distance inverse. La correspondance entre la moyenne utilisée et la distance moyenne réelle des amers lors de leur initialisation va définir approximativement le facteur d'échelle de la trajectoire obtenue si rien d'autre ne le fixe. L'incertitude sur la distance inverse doit contenir 0 (distance infinie), et l'inverse de la distance minimale $d_{\min}$ à laquelle on peut avoir des amers, mais on peut la définir de plusieurs façons selon la probabilité que l'on veut donner à ces extrêmes (même si l'on couvre des distances inverses négatives et donc impossibles, cela permet d'être plus consistant si on a souvent des amers à l'infini) :

— $\mathcal{N}\left(\frac{1}{3\cdot d_{\min}}, \frac{1}{3\cdot d_{\min}}\right)$ donne $d_{\min}$ à 2 sigmas (5% de probabilité d'être au plus à $d_{\min}$), et l'infini à 1 sigma (30% de probabilité d'être à l'infini). Ce sont les valeurs que nous utilisons en pratique.

— $\mathcal{N}\left(\frac{1}{2\cdot d_{\min}}, \frac{1}{4\cdot d_{\min}}\right)$ donne $d_{\min}$ à 2 sigmas (5%), et l'infini à 2 sigmas (5%). Peut mieux convenir en environnement intérieur.

— $\mathcal{N}\left(\frac{1}{2\cdot d_{\min}}, \frac{1}{5\cdot d_{\min}}\right)$ donne $d_{\min}$ à 2.5 sigmas (1%), et l'infini à 2.5 sigmas (1%).

4.3.2 Utilisation de lignes

Les segments ou lignes présentent plusieurs avantages sur les points :

— ils représentent la géométrie de l'environnement, qui est une propriété intrinsèque et donc invariante aux changements de point de vue, de capteur, et de conditions d'illumination.

— ils fournissent un squelette de modèle 3D qui est exploitable pour d'autres tâches que la localisation.

La paramétrisation de la ligne dans l'image est définie par un point porteur (u, v) et une orientation α, employée par le détecteur et suiveur de segments DSEG [Berger et Lacroix, 2010] que nous utilisons.

Plusieurs paramétrisations de la ligne dans l'espace sont possibles et ont été proposées. Les premières étaient basées sur les coordonnées de Plücker [Solà *et al.*, 2009], mais nous nous

sommes orientés vers les paramétrisations basées sur deux points, qui ont été démontrées plus linéaires [Solà *et al.*, 2012] et ne nécessitant pas d'imposition de contraintes.

a Paramétrisation double point

Cette paramétrisation consiste à modéliser la ligne par deux points dans l'espace, qui ne font que définir une droite porteuse. En particulier ces points ne sont pas les extrémités d'un segment, puisque l'on estime dans le filtre des lignes, justement car les extrémités des segments peuvent être très instables en fonction du point de vue. Comme on souhaite tout de même avoir une idée des portions observées des lignes pour savoir où s'attendre à les retrouver, on considère à côté de cela que les amers sont des segments en maintenant leurs extrémités par un processus externe au filtre, en définissant les segments comme la concaténation des portions observées.

Ces points peuvent être euclidiens ou AHP selon le besoin, de façon similaire au choix pour les points seuls, en notant que dans le cas AHP l'ancre est commune aux deux points :

$$\mathbf{l_E} = \left(\mathbf{p_1}^T\ \mathbf{p_2}^T\right) \tag{4.30}$$

$$\mathbf{l_{AH}} = \left(\mathbf{p_0}^T\ \mathbf{u_1}^T\ \rho_1\ \mathbf{u_2}^T\ \rho_2\right) \tag{4.31}$$

La paramétrisation de la ligne dans l'image utilisée par le détecteur comprenant un angle, elle introduirait des non linéarités supplémentaires dans la fonction d'observation. Nous voulons donc la transformer en une paramétrisation avec deux points cartésiens dans le plan, qui permet à la fonction d'observation de consister simplement en la projection des deux points dans l'image. Cependant cette représentation possède deux degrés de liberté supplémentaires non contraints (une ligne ne possède que deux degrés de liberté), qui participent artificiellement à l'innovation. Il est donc indispensable d'éliminer leur participation, en ne conservant que la composante orthogonale à la ligne estimée. Les deux points de la mesure ($\mathbf{m_1}\ \mathbf{m_2}$) sont pour cela choisis comme la projection orthogonale des deux points de la mesure attendue ($\mathbf{e_1}\ \mathbf{e_2}$) (*expectation*) sur la droite de la mesure, comme illustré figure 4.5.

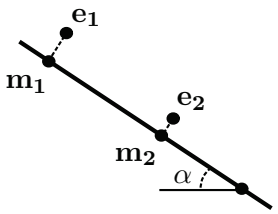


FIGURE 4.5 – Calcul de l'innovation des lignes double point, en utilisant pour mesure les deux points constitués de la projection orthogonale de la mesure attendue sur la droite mesurée, afin d'éliminer l'influence des degrés de liberté superflus de la représentation par double point sur l'innovation.

4.4 Aspects d'implémentation

La mise en pratique concrète dans un système de tous ces éléments théoriques d'estimation, et en particulier un système qui fonctionne en ligne, nécessite beaucoup de considérations

supplémentaires, qui sont présentées dans cette section.

4.4.1 Recherche active

Dans l'objectif de réduire le plus possible le temps consacré à l'appariement d'amers, et d'exploiter au maximum les informations d'incertitude fournies par le filtre de Kalman, nous avons implémenté une stratégie de recherche active des amers, présentée sous la dénomination *active search* dans [Davison *et al.*, 2007].

Le principe est de ne rechercher les amers à mettre en correspondance entre les images que dans la zone où le filtre s'attend avec une probabilité non négligeable de le trouver. Après l'étape de prédiction du filtre de Kalman, comme on l'a vu section 3.1.2 page 29 la fonction d'observation de l'amer permet de prédire sa position attendue $\hat{\mathbf{z}}$ dans l'image. Par ailleurs il est possible de calculer son incertitude, qui n'est autre que la matrice de covariance Y de l'innovation, à ceci près qu'on doit utiliser une estimation majorante du bruit de mesure puisque la mesure n'a pas encore eu lieu. Cette incertitude prendra en dimension 2 une forme d'ellipse d'incertitude, du fait de l'hypothèse gaussienne des incertitudes, calculable directement à partir de Y, et formant ainsi une région d'intérêt pour l'algorithme d'appariement, comme illustré figure 4.6.

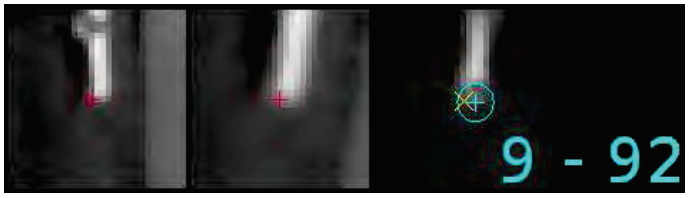


FIGURE 4.6 – La construction de l'ellipse de recherche des amers par prédiction. La première imagette est issue du descripteur : c'est l'amer tel qu'il était vu lors de sa détection initiale. La seconde imagette est une prédiction de son apparence actuelle basée sur le changement de point de vue depuis sa détection. La troisième imagette est l'amer retrouvé dans l'image courante, la croix jaune indiquant sa position mesurée, et l'ellipse bleue sa zone de recherche attendue.

Les amers sont alors séquentiellement projetés dans le plan d'observation, appariés et corrigés les uns après les autres, voyant leur zone de recherche diminuer progressivement au fur et à mesure que l'incertitude de localisation du robot diminue. Il est également possible de choisir de corriger en priorité les amers qui ont les ellipses d'incertitude d'observation les plus grosses, car ce sont ceux dont l'observation apporte le plus d'information au système.

Cette technique présente ainsi deux avantages :

— Elle limite énormément les traitements à effectuer sur les données brutes du capteur en ne considérant que la sous-partie intéressante de ces données, permettant d'atteindre des fréquences de traitement très importantes (par exemple 60 Hz avec des images de taille VGA).

— Elle permet de pouvoir décider à n'importe quel moment d'interrompre les corrections, même si tous les amers n'ont pas été traités, car ils auront une ellipse d'incertitude plus grosse pour la prochaine image et seront donc traités en priorité. Ceci apporte un aspect *temps réel dur* à l'algorithme qui est très utile lors d'une exécution en ligne sur le robot, permettant d'éviter de devoir jeter des images sans les traiter par manque de temps.

Un inconvénient de cette méthode est cependant que lorsque le système n'est plus consistant on ne retrouvera plus les amers, et le SLAM casse complètement à ce moment. Cela est donc d'un côté un manque de robustesse, mais de toute façon si le filtre n'est plus consistant il ne serait pas correct de lui imposer des mesures incompatibles, et il vaut mieux détecter cette situation et se réinitialiser.

4.4.2 Contrôle de compatibilité

La *détection des éléments aberrants* est importante en SLAM-EKF, car l'information qu'apporte leur observation dans le filtre ne peut plus être remise en cause, ce qui peut perturber fortement le filtre. Ils ont deux principales origines : les erreurs d'appariement au niveau traitement des données brutes, et les objets mobiles dans l'environnement.

Le contrôle de compatibilité, aussi appelé contrôle des résidus ou *gating* en anglais, est un procédé classique pour détecter les éléments aberrants en filtrage, qui consiste à considérer comme erronée toute observation sortant de l'ellipse de prédiction à $3\,\sigma$ (figure 4.6). Ceci revient à dire que la distance de Mahalanobis entre la mesure $\tilde{\mathbf{z}}_\mathbf{k}$ et la mesure attendue $\hat{\mathbf{z}}_\mathbf{k}$ est supérieure à 3, ce qui s'exprime directement à partir de l'innovation $\mathbf{y} = \tilde{\mathbf{z}}_\mathbf{k} - \hat{\mathbf{z}}_\mathbf{k}$ et sa covariance $\mathbf{Y}$:

$$\sqrt{\mathbf{y}^T \cdot \mathbf{Y}^{-1} \cdot \mathbf{y}} > 3 \tag{4.32}$$

Un rejet signifie alors que l'on a un niveau de confiance de 99% que l'erreur ne provient pas uniquement du bruit. L'observation est alors dite *incompatible* avec le filtre.

Il faut cependant bien noter que ce coefficient 3 ne correspond au niveau de confiance de 99% que pour une innovation de dimension 2, comme l'observation d'un point dans une image. En règle générale et en toute rigueur, pour une innovation de dimension n, et un niveau de confiance désiré α, il faut lire la table de distribution de la loi du χ^2 et choisir comme coefficient la racine carrée de la valeur trouvée pour la dimension n et la probabilité $1 - \alpha$. Il est également possible d'adopter une politique plus conservatrice en choisissant un seuil à 95% par exemple.

Remarque L'élimination des amers sur les objets mobiles avec le contrôle de compatibilité présente un inconvénient. Si la dynamique de l'objet est faible, elle peut provoquer un biais dans l'erreur d'appariement, et donc un bruit non gaussien. En effet, si l'objet se déplace suffisamment lentement par rapport à la fréquence des observations et l'incertitude accordée aux mesures, l'estimée peut être corrompue plus rapidement que le déplacement de l'objet et suivre l'objet sans jamais être détectée comme incompatible par le contrôle de compatibilité. C'est un phénomène que l'on peut facilement observer en déplaçant lentement son doigt tendu devant la caméra par exemple, et qui peut se manifester en conditions réelles dans des feuillages agités par le vent, dans les nuages, ou sur des personnes essayant de rester immobiles. On pourrait imaginer diverses solutions pour détecter ce phénomène, comme vérifier que l'ensemble des innovations associées à chaque amer suit une distribution gaussienne, mais cela reste délicat à réaliser (notamment car il demeurerait des ambiguités avec une prédiction possiblement biaisée lors des accélérations, décélérations, ou un léger décalage de synchronisation

d'un capteur, et car on ne pourrait de toute façon pas supprimer l'information déjà intégrée lors de la suppression de l'amer). En pratique cela ne pose pas réellement de problème pour l'estimation de la position car ces amers « pathologiques » sont en général minoritaires et de mouvement de moyenne nulle. Le seul cas qui peut être réellement gênant est celui des nuages au démarrage lorsqu'aucun amer n'a commencé à converger, le système ne pouvant exclure que c'est lui qui est en train de bouger. Il est donc nécessaire de limiter en extérieur le temps à l'arrêt au démarrage du système (puisque c'est un phénomène qui reste lent), ou encore utiliser d'autres capteurs en complément comme on le verra dans les chapitres suivants.

4.4.3 One-point RANSAC

Le contrôle de compatibilité n'est cependant pas toujours suffisant pour garantir une robustesse suffisante et l'élimination de tous les éléments aberrants.

En particulier immédiatement après l'étape de prédiction, les ellipses d'incertitude peuvent être grandes, rendant le contrôle de compatibilité peu efficace. De façon plus grave, si une correction est acceptée sur un élément aberrant lors des premières corrections d'amer de l'image, il y a de forts risques pour qu'elle rende le filtre inconsistant et incompatible avec les observations des amers suivants de l'image. Le SLAM n'aura alors utilisé que des éléments aberrants, ce qui aboutira à sa divergence.

Une solution à ce problème a été proposée dans [Civera *et al.*, 2009], appelée *one-point RANSAC*.

Pour rappel, l'algorithme classique RANSAC (RANdom SAmple Consensus) fonctionne comme suit :

1. Si un modèle nécessite n éléments pour être estimé (par exemple deux points sont nécessaires pour modéliser une droite), on choisit aléatoirement n éléments de l'ensemble et on estime le modèle correspondant.
2. On teste ensuite la compatibilité de tous les autres éléments avec le modèle produit, et on compte combien d'entre eux sont compatibles.
3. On réitère m fois l'opération, et on choisit l'ensemble d'éléments compatibles le plus grand.

Le nombre m peut être calculé statistiquement de façon simple à partir de la proportion maximale β d'éléments aberrants qui peuvent être présents dans l'ensemble et la probabilité maximale α du risque que l'on souhaite prendre de ne pas parvenir à sélectionner au moins une fois n éléments non aberrants :

$$\alpha \approx (1 - (1 - \beta)^n)^m$$

Ainsi plus n est grand, plus m devra également être grand pour conserver le même risque α. Ceci peut amener à tester beaucoup de modèles et prendre beaucoup de temps à calculer. Par exemple si on veut être robuste à $\beta = 50\%$ d'outliers avec un risque $\alpha \leq 5\%$, on aura $m = 5$ pour $n = 1$, $m = 11$ pour $n = 2$, et $m = 23$ pour $n = 3$.

En l'occurrence dans notre cas il y a besoin d'au moins $n = 3$ amers pour déterminer complètement le mouvement de la caméra entre deux images, sans compter la difficulté de l'exigence supplémentaire que ces amers aient suffisamment convergé pour pouvoir faire ce calcul, ce qui n'est pas garanti du fait de l'observabilité partielle des amers.

En revanche nous avons à notre disposition un filtre de Kalman qui est capable de formuler une hypothèse de mouvement de la caméra à partir de l'observation d'un seul amer, ce qui permet de ramener n à 1 et de ne pas avoir à implémenter d'autre algorithme de modélisation.

En théorie le filtre fournit même les conditions précises d'adéquation d'un amer au modèle généré, en calculant la matrice de covariance de l'état qui permet de calculer la distance de Mahalanobis. En pratique cependant il est beaucoup plus efficace de ne pas calculer la matrice de covariance lors de la correction, qui représente l'essentiel du coût de calcul d'une correction du filtre de Kalman, et de tester la compatibilité avec un seuil minorant, inférieur au bruit de mesure. On obtient ainsi un comportement conservateur, et les amers rejetés à tort seront finalement corrigés dans un second temps s'ils passent l'étape de contrôle de compatibilité, qui est devenue suffisante une fois plusieurs amers corrigés dans l'image.

Le dernier avantage de la méthode est qu'elle permet d'effectuer une correction groupée des amers initialement compatibles, ce qui est dans les conditions habituelles plus rapide qu'une correction séquentielle, et qui compense largement le surcoût lié au test RANSAC.

4.4.4 Gestion des amers

La gestion des amers se fait à deux niveaux :

— au niveau de l'image, pour réguler le nombre et la répartition des amers utilisés sur le moment. Dans RT-SLAM, ce sont les objets de type `FeatureManager` qui assurent cette fonction.

— au niveau de la carte, pour réguler le nombre et la répartition des amers utilisés tout au long de la séquence. Dans RT-SLAM, ce sont les objets de type `MapManager` qui assurent cette fonction. Deux politiques sont implémentées : une politique globale, et une politique locale.

La gestion des amers dans l'image guide la création des amers, tandis que la gestion des amers dans la carte guide leur destruction.

a Politique dans l'image

Il est important si l'on veut éviter des singularités (mauvais conditionnement des équations) et maximiser l'observabilité de notre état pour rendre le système le plus robuste possible, de répartir correctement les amers dans l'image, en exploitant au mieux le champ de vue. Ceci est d'autant plus important que la quantité d'amers observables est limitée par les capacités de calcul.

Pour cela on définit une grille devant l'image, et on essaie de maintenir au moins un amer visible dans chaque cellule de cette grille. Il est inutile d'essayer d'imposer plus d'un amer par cellule, puisque augmenter le nombre de cellules aura le même effet sur le nombre total

d'amers, mais en améliorant l'uniformité de leur répartition. De plus, afin d'empêcher des amers d'être trop proches les uns des autres, on impose une zone interdite autour des lignes de la grille, et afin de ne pas créer de zones mortes dans l'image, on déplace aléatoirement cette grille sur l'image à chaque nouvelle image. Les cellules vides de cette grille guident ainsi la création de nouveaux amers. La figure 4.7 illustre ce processus.

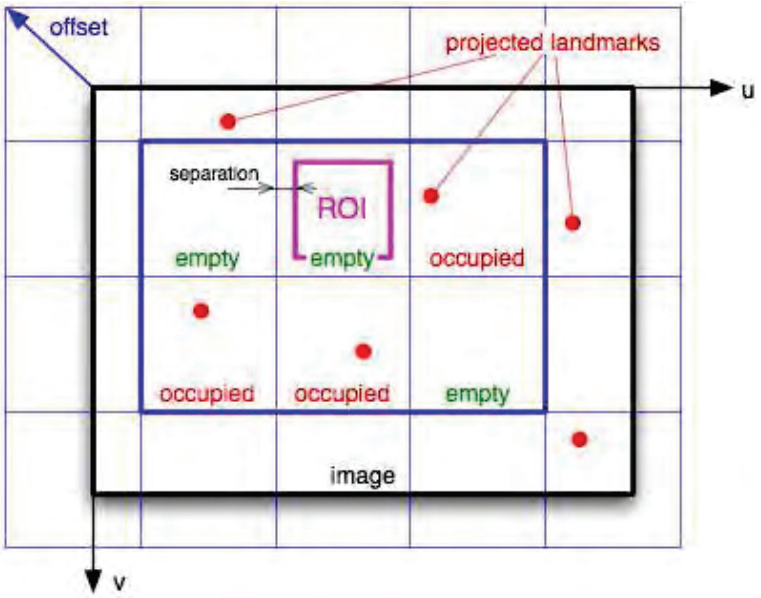


FIGURE 4.7 – [a] Illustration de la grille réalisant la politique de gestion des amers dans l'image. Le décalage *offset* de la grille change aléatoirement à chaque image, et les nouveaux amers sont crées dans la région d'intérêt *ROI* des cases ne contenant aucun amer projeté.

a. Illustration issue de la documentation de RT-SLAM, et créée par Joan Solà.

b Politique globale dans la carte

La complexité du filtre de Kalman étant critique vis à vis du nombre d'amers présents dans la carte, il est nécessaire de limiter leur nombre, mais tout en garantissant un certain nombre pour permettre de se localiser suffisamment précisément quand on réobserve des endroits déjà observés.

Plusieurs règles ont donc été mises en place à cet effet, portant d'une part sur la qualité des amers, et d'autre part sur leur répartition géométrique, mais qui diffèrent selon que l'amer est couramment visible ou pas. En effet il est nécessaire d'avoir en permanence un nombre relativement important d'amers pour garantir stabilité et robustesse au bruit, alors que les amers non visibles se comportent simplement comme des ancres pour éviter la dérive lorsqu'il sont à nouveau observés.

Les critères portant sur la qualité des amers sont les suivants. Un amer sera supprimé si l'une de ces conditions est vraie :

1. le ratio entre le nombre fois où l'amer a été trouvé et le nombre de fois où il a été recherché est trop faible, pour tous les capteurs.

2. le ratio entre le nombre fois où l'amer a été compatible et le nombre de fois où il a été trouvé est trop faible, pour tous les capteurs.
3. l'amer n'est visible par aucun capteur, et le ratio entre l'incertitude de sa distance et sa distance est trop important.
4. l'amer n'est visible par aucun capteur, et n'a été observé assez de fois dans aucun capteur.
5. l'amer est visible par un capteur mais sa zone de recherche dans ce capteur est devenue trop grande.
6. l'amer demande sa destruction à cause de critères propres à son type particulier (par exemple une contrainte dans sa paramétrisation non respectée).

Par ailleurs en vue de limiter la densité des amers dans l'espace, des grilles volumiques sphériques centrées sur le robot sont crées, partageant l'espace en secteurs angulaires eux-mêmes subdivisés en différentes cellules par un découpage en distance. Une première grille est créée pour les amers visibles et observés, une deuxième grille pour les amers visibles et non observés, et enfin une troisième grille pour les amers invisibles. Chaque cellule d'une grille ne peut contenir qu'un amer au maximum. Lorsque plusieurs amers sont présents, seul l'amer de meilleure qualité, c'est-à-dire avec l'incertitude de localisation la plus faible, est conservé. La résolution des grilles est initialisée en cohérence avec la résolution de la grille utilisée dans l'image, mais la grille des amers invisibles peut voir sa résolution dynamiquement réduite si la carte est pleine, afin de libérer de l'espace en supprimant des amers. Cela permet de ne pas avoir de limite sur la taille de l'espace explorable, simplement plus il sera grand plus la densité d'amers conservés sera faible. Cette contrainte n'est pas appliquée aux grilles gérant les amers visibles pour ne pas diminuer la robustesse sur le moment.

Remarque La discrétisation régulière d'une sphère est un problème difficile. Nous nous sommes contenté d'une implémentation approchée simple, qui à partir d'une résolution angulaire α souhaitée, travaille en coordonnées sphériques pour discrétiser régulièrement l'angle d'assiette ϕ selon α, fait de même pour l'angle de cap θ pour la tranche contenant l'équateur de la sphère, puis discrétise ensuite dynamiquement l'angle de cap θ pour les autres tranches de façon à obtenir des cellules d'aire la plus proche possible de celles obtenues pour la tranche contenant l'équateur.

c Politique locale dans la carte

La politique locale de gestion des amers dans la carte est destinée aux applications où l'espace à explorer est trop grand pour pouvoir conserver une densité suffisante d'amers pour espérer parvenir à réaliser des fermetures de boucle. Seuls les amers visibles sont alors conservés dans la carte. De cette manière on peut considérer que l'on n'effectue plus du vrai SLAM, mais plutôt de l'odométrie visuelle avec une mémorisation temporaire.

Les seuls critères portent sur la qualité des amers. Un amer sera supprimé si l'une de ces conditions est vraie :

1. le nombre d'image depuis la dernière fois que l'amer était visible est trop important, pour tous les capteurs.
2. le nombre de fois que l'amer a été recherché sans succès depuis la dernière fois qu'il a été compatible est trop important, pour tous les capteurs.
3. l'amer est visible par un capteur mais sa zone de recherche dans ce capteur est devenue trop grande.
4. l'amer demande sa destruction à cause de critères propres à son type particulier (par exemple une contrainte dans sa paramétrisation non respectée).

4.4.5 Traitement d'image

a Détection des points

La détection des points est basée sur un détecteur de Harris avec cependant plusieurs optimisations.

Certaines optimisations sont des approximations : un masque de dérivation minimal $[-1, 0, 1]$ est utilisé, comme le masque de convolution qui est carré et constant. En plus de nécessiter un nombre réduit de calculs, cela permet l'utilisation d'images intégrales [Viola et Jones, 2001] pour calculer efficacement les convolutions.

Les autres optimisations sont liées à la stratégie de recherche active (section 4.4.1 page 66) et de gestion des amers (section 4.4.4 page 69) qui ont été adoptées : la recherche ne se fait que dans une région réduite de l'image, et seulement un amer est recherché, ce qui élimine les étapes coûteuses de seuillage et de suppression des sous-maxima.

b Appariemment des points

L'appariement des points est basée sur un corrélateur ZNCC (Zero-mean Normalized Cross Correlation), également avec plusieurs optimisations.

Des images intégrales sont à nouveau utilisées pour calculer efficacement les moyennes et les variances des niveaux de gris, et une recherche hiérarchique est effectuée (une passe à demi-résolution et une seconde passe à pleine résolution sont suffisantes). Nous avons également implémenté l'algorithme de *bounded partial correlation* introduit par [Stefano *et al.*, 2005] afin d'interrompre le calcul du score de corrélation lorsqu'il n'y a plus d'espoir d'obtenir un score meilleur que le seuil ou le meilleur déjà obtenu dans la recherche. Enfin la recherche ne se fait que dans l'ellipse de prédiction de l'amer, qui reste très petite par rapport à la taille de l'image.

Par ailleurs afin d'éviter la dérive d'apparence du point lors de son suivi, c'est l'apparence initiale de l'amer qui est toujours utilisée pour sa recherche dans les images suivantes. Pour être robuste aux changements de point de vue et suivre les amers suffisamment longtemps, une prédiction de l'apparence de l'amer est effectuée, grâce à l'étape de prédiction du filtre. Cette

prédiction ne tient pas compte de la normale du support physique de l'amer, et n'est donc pas parfaite, mais elle permet de tenir compte des rotations dans le plan et de la distance.

Cette prédiction, dont un exemple a été donné figure 4.6 page 66, peut se faire de deux façons.

Prédiction affine La prédiction de l'apparence de l'amer sous forme affine est présentée dans [Davison *et al.*, 2007], et se calcule directement à partir du changement de point de vue et du modèle de caméra. Il n'est cependant suffisamment précis que pour un modèle *pinhole* avec une distortion faible.

Prédiction homographique Lorsque la distortion est trop importante, ou dans le cas d'une caméra omnidirectionnelle, il est nécessaire de passer à une transformation plus complexe, par exemple homographique. Étant cependant difficile à estimer à partir du modèle de la caméra, nous la calculons numériquement de la façon suivante :

— les 4 coins de l'imagette de prédiction que l'on souhaite obtenir sont rétroprojetés dans l'espace autour de l'amer physique. Leur distance est ajustée pour qu'ils forment un plan contenant l'amer physique, et dont la normale est la moyenne entre le point de vue courant et le point de vue initial (puisque la normale réelle est inconnue, on minimise ses conséquences).

— ces 4 points sont reprojetés dans l'imagette du descripteur de l'amer obtenue lors de sa détection.

— l'homographie entre les deux quadruplets de points dans chacune des imagettes est calculée de façon exacte.

— cette homographie est alors appliquée à l'imagette du descripteur pour donner l'imagette de prédiction à l'instant courant.

4.5 Améliorations et variantes

4.5.1 Reparamétrisation des amers

Nous avons vu que les paramétrisations distance inverse des amers étaient très utiles pour initialiser directement les amers avec un système n'en fournissant qu'une observation partielle. Le principal inconvénient de ces paramétrisations est leur taille. Par exemple avec une paramétrisation AHP, un amer est représenté par 7 valeurs scalaires dans la carte stochastique, contre 3 seulement pour une paramétrisation euclidienne. Or l'occupation mémoire et la complexité temporelle du filtre de Kalman varient de façon quadratique avec la taille de la carte, ce qui signifie qu'un amer AHP est environ 5 fois plus encombrant et long à traiter qu'un point euclidien.

Nous avons donc implémenté un mécanisme de reparamétrisation des amers basés sur la distance inverse lorsqu'ils ont suffisamment convergé, en paramétrisation classique. Nous utilisons pour cela le critère de linéarité proposé par [Montiel, 2006] :

$$L = \frac{4 \cdot \sigma_d}{d} \left|\cos \alpha\right| \tag{4.33}$$

dépendant de l'incertitude sur la distance de l'amer σ_d par rapport à sa distance d au capteur, et du cosinus du changement de point de vue angulaire α que le capteur a eu depuis l'initialisation de l'amer, décrivant l'observabilité qu'il a eu de sa distance.

Lorsque L est devient suffisamment faible (par exemple inférieur à 0.1), l'amer est transformé.

4.5.2 Gestion temps réel de l'intégration des amers

Comme nous l'avons déjà mentionné dans la section 4.4.1 page 66, la méthode se prête bien à une gestion temps réel dur, en permettant d'arrêter le traitement à n'importe quel moment, par exemple quand l'image suivante est prête. Cela permet d'éviter de rater des images à cause de traitements qui ont été trop longs sur les images précédentes. Il est plus intéressant d'observer moins d'amers sur toutes les images, que plus d'amers sur un nombre réduit d'images, puisque qu'un temps allongé entre deux images traitées va augmenter la taille des zones de recherche, et diminuer la précision de la prédiction et donc des points de linéarisation.

Pour implémenter cela en pratique, on maintient en permanence une estimation basique du temps nécessaire au filtre pour effectuer la correction d'un amer dans un groupe (RANSAC) ou seul (recherche active), en supposant qu'il change peu entre une image et la suivante. Les images parvenant périodiquement, au moment d'effectuer la correction du groupe RANSAC on sait combien de temps reste disponible, et on peut limiter la taille du groupe d'observations à corriger pour ne pas déborder. On continue ensuite de la même façon en corrigeant les amers un par un en vérifiant si on aura le temps de terminer la correction.

4.5.3 Observations significatives

Nous avons vu que fonctionner à une fréquence élevée était important entre autres pour améliorer la précision des points de linéarisation et ainsi la consistance du filtre. Ceci est d'autant plus vrai que la dynamique du système est élevée, mais lorsque ce n'est pas le cas, trop de mises à jour du filtre peuvent avoir un effet négatif sur sa consistance.

En effet d'une part en théorie, plus on observe, meilleure est notre estimée. Mais ceci n'est vrai que dans un cas idéal, notamment avec des bruits d'observation parfaitement gaussiens. Or il reste toujours dans le système des biais de calibrage à divers endroits, qui rendent le bruit d'observation non gaussien, et font converger le système vers des valeurs biaisées, non prises en compte dans l'incertitude. Le système devient alors d'autant plus surconfiant qu'il y a d'observations.

D'autre part, dans le cas particulier de l'observabilité partielle d'amers, le bruit d'observation va générer un bruit dans l'état, y compris dans la position du robot (aussi faible qu'il soit, il est non nul). Ces changements de position provoquent alors une observabilité virtuelle de la profondeur des amers. Chaque observation apportant une information non nulle (si l'innovation n'est pas parfaitement nulle, ce qu'elle n'est jamais à cause du bruit réel, et si le bruit théorique n'est pas infini, ce qu'il n'est pas non plus), cela fait diminuer l'incertitude de localisation des amers, y compris la composante non observable puisque du bruit sur la position l'a rendue un peu observable. Les incertitudes se réduisent donc progressivement

au fur et à mesure des observations, même si l'observabilité théorique est nulle, ce qui le rend inconsistant. Ce phénomène est particulièrement visible par exemple lorsque le système est à l'arrêt, et n'observe donc rien de la distance des amers, et que l'on constate que les amers convergent tout de même. Le phénomène survient en réalité également pour des amers complètement observables, à cause des erreurs des jacobiennes utilisées, comme l'ont montré [Julier et Uhlmann, 2001] puis [Huang *et al.*, 2008].

Il semble donc intéressant d'adapter la fréquence des observations à la dynamique et aux besoins du système. Une observation n'est particulièrement significative que si l'incertitude de la mesure est petite devant l'incertitude de sa prévision, ce qui signifie que l'on est sûr que l'innovation (différence entre la mesure et la prévision) utilisée pour faire la correction contient essentiellement de l'information, et pas du bruit de la mesure.

L'idée est donc de n'intégrer une observation que si elle est suffisamment significative en ce sens. Le problème est que selon comment est fixé le seuil, on perd toujours un peu d'information. On ne soumet donc à cette condition que la mise à jour de la matrice de covariance $\hat{\mathbf{P}}_\mathbf{k}$, et on continue à corriger systématiquement la moyenne $\hat{\mathbf{x}}_\mathbf{k}$. Cela revient à toujours donner l'information au filtre (et donc ne pas en perdre), mais à ne pas le lui dire quand on estime qu'elle n'est pas significative (et donc lui éviter de croire qu'il en a eu plus qu'en réalité et devenir inconsistant).

En pratique cela est très efficace pour éviter les convergences en profondeurs non fondées lorsque le robot est à l'arrêt, et en mouvement cela améliore parfois significativement la consistance (entre 10 et 30%), et parfois légèrement la précision (voir tableau 7.4 page 181 pour des résultats détaillés). Il faut cependant maintenir le seuil assez bas (entre 1 et 1.5 pour le rapport des variances) sous peine de dégrader la précision.

4.5.4 Amélioration du suivi des amers

Afin de limiter la dérive de la localisation, il est intéressant de suivre les amers le plus longtemps possible. Bien sûr le domaine duquel un amer reste visible est limité, mais les capacités de suivi sont souvent plus limitantes, à cause du changement de point de vue et en conséquence de l'apparence de l'amer.

On a vu en effet section 4.4.5.b page 72 que l'amer suivi était toujours comparé à son descripteur initial, afin d'éviter une dérive progressive du point réel suivi. Une correction dépendant de la distance et de l'angle de gîte de la caméra est appliquée, mais sans prise en compte du changement de point de vue car on ne connait pas la direction de la normale de la surface support de l'amer. Ainsi quand le point de vue a trop changé, il n'est plus possible d'apparier l'amer à son descripteur et il est perdu. Il existe des méthodes qui estiment la normale de l'amer, permettant de compenser correctement le changement de point de vue, mais elles sont plus complexes à implémenter en monoculaire [Molton *et al.*, 2004] ou nécessitent un banc stéréo [Berger et Lacroix, 2008], relativement coûteuses à l'exécution, et ne sont pas non plus parfaites quand le support de l'amer n'est pas un plan ou que le changement d'échelle est trop important.

Nous avons donc implémenté une méthode relativement simple consistant lorsqu'un amer est

perdu à ajouter sa dernière apparence correctement détectée au descripteur, et à essayer de continuer à le suivre avec celle-ci. On a donc un descripteur *multivues*, et à chaque nouvelle image on choisit la vue dont le point d'observation est le plus semblable selon l'estimation courante de la position.

L'inconvénient lors du changement de vue de référence pour réaliser le suivi, est que la nouvelle vue comportera nécessairement une légère erreur de positionnement de son centre sur l'amer physique (correspondant au bruit d'observation), et cette erreur se retrouvera de façon constante sur tous les appariements à venir, constituant de ce fait un biais, provoquant un bruit d'observation non gaussien car non centré. Ce phénomène est de plus cumulable avec les différentes vues successives utilisées. En pratique les résultats ne sont pas vraiment améliorés, voire parfois dégradés (le suivi plus long des amers a tendance à réduire la dispersion, mais les biais engendrés à augmenter un peu l'erreur).

4.5.5 Carte de visibilité

Il arrive régulièrement, notamment en environnement encombré intérieur, que des amers soient toujours dans le champ de la caméra mais masqués par des éléments de la scène. Comme vu section 4.4.4 page 69, la politique de gestion des amers va persévérer à essayer de les chercher, puis au bout d'un certain temps considérer qu'ils ont disparu et les supprimer. Cela pose plusieurs problèmes :

— il y a un gaspillage de temps de calcul à rechercher systématiquement des amers qui sont masqués,

— on peut se retrouver à supprimer des amers qui auraient en fait été retrouvés en changeant de point de vue,

— ces amers empêchent d'en créer de nouveaux sur les obstacles qui les masquent, ce qui nuit à la précision et robustesse du système.

Nous avons donc implémenté une carte de visibilité pour chaque amer afin de déterminer les endroits de l'espace d'où il est observable ou non. Cette carte est sphérique et définit des secteurs discrétisés (il s'agit de la même discrétisation que celle utilisée pour la politique globale de gestion des amers, décrite section 4.4.4 page 69). Une fois qu'un secteur a été marqué comme ne permettant pas d'observer un amer, cet amer est ignoré mais pas supprimé, ce qui permet d'en créer d'autres à son voisinage attendu, et d'arrêter de le chercher inutilement. Quand on change de point de vue, on adapte la politique au secteur dans lequel on se trouve :

— secteur marqué comme visible : on le cherche.

— secteur marqué comme non visible : on l'ignore, tout en le recherchant occasionnellement au cas où, afin de gérer la dynamique de l'environnement.

— secteur marqué comme inconnu : on le cherche jusqu'à ce que le secteur se détermine.

Il est cependant difficile de correctement gérer la politique de catégorisation face à des amers qui peuvent être perdus à cause d'un échec de l'appariement, et on peut parfois obtenir des zones avec trop d'amers, voire des amers dupliqués si cela n'est pas vérifié.

4.5.6 Amers virtuels

Les amers que l'on appelle virtuels sont des amers qui remplissent parfaitement les conditions d'extraction dans l'image, mais qui en fait ne correspondent pas à un amer physique dans l'espace.

L'exemple le plus courant, en particulier en environnement intérieur, est la détection de coins dans l'image qui sont en fait l'intersection d'une ligne verticale et d'une ligne horizontale situées à des distances différentes, qui peuvent être fréquents comme le montre la figure 4.8. Ainsi selon que le déplacement de la caméra se fera verticalement ou horizontalement, ce sont différentes distances du point qui ressortiront et rendront l'estimée de position de l'amer inconsistante, comme l'illustre la figure 4.9.

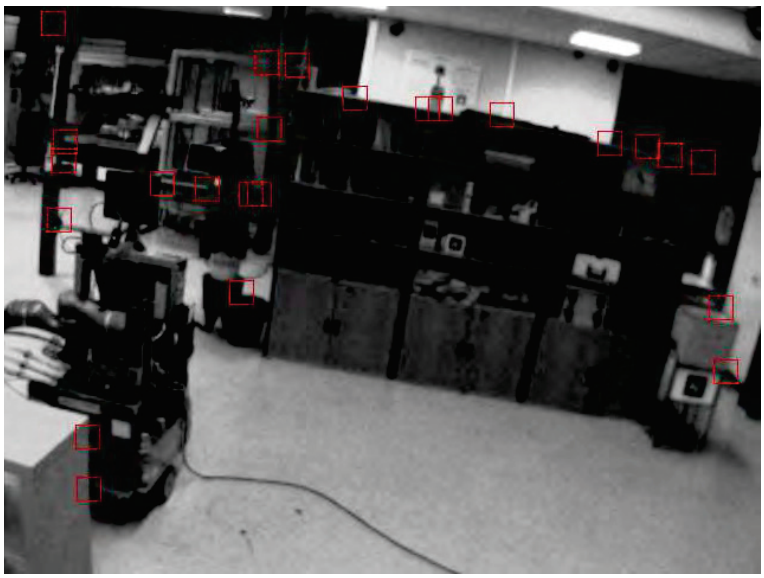


FIGURE 4.8 – Les sources possibles d'amers virtuels sont nombreuses dans un environnement intérieur, comme le montre cet extrait de l'une de nos séquences de test.

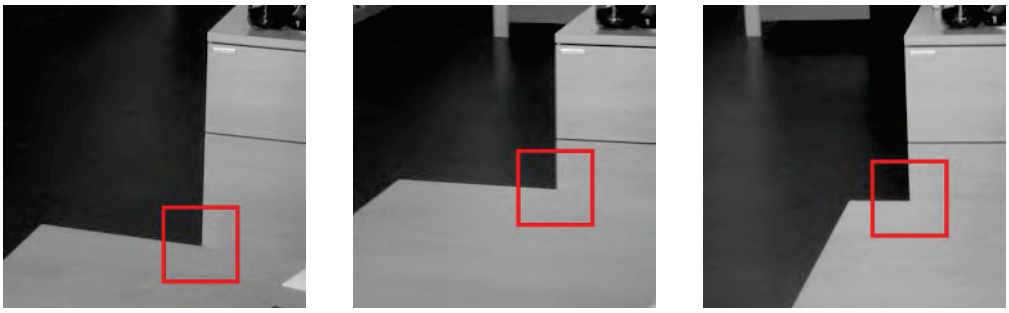


FIGURE 4.9 – Exemple d'incohérence de profondeur d'un amer virtuel. Le déplacement entre la première et la deuxième image laisse penser que sa profondeur est celle du meuble à l'avant-plan, tandis que le déplacement entre la deuxième et la troisième image laisse penser que sa profondeur est celle du meuble à l'arrière- plan

On peut imaginer différentes solutions pour traiter ce problème, mais aucune n'est très simple à implémenter :

— Dès la détection, vérifier avec l'apparence de l'amer que l'on n'a pas affaire à quelque chose qui ressemble à une intersection de lignes. La difficulté est d'identifer ces axes, et ce n'est pas infaillible puisque cela échoue quand les deux objets ont des textures similaires, comme dans la figure 4.9 page précédente.

— Estimer séparément la profondeur selon deux axes orthogonaux d'exploration, et vérifier leur compatibilité. La difficulté est de choisir ces axes (les choisir a priori ne fonctionnera pas dans tous les cas), et l'inconvénient est que l'on se rend compte avec du retard du défaut de l'amer, alors qu'il a déjà été utilisé.

En pratique on se contente d'exiger une certaine symétrie entre les deux valeurs des points de Harris détectés, ce qui élimine les cas les plus grossiers, puis sans autre traitement particulier l'instabilité intrinsèque de ces amers provoque leur suppression à terme. En effet d'une part leur apparence est plus susceptible de changer brutalement, et d'autre part leur incompatibilité finit par être détectée par le mécanisme de contrôle de compatibilité. Il est cependant nécessaire d'observer un nombre suffisant d'amers dans l'image pour que leur effet ne soit pas compromettant.

4.6 Résultats

Les résultats présentés ici sont corrigés du facteur d'échelle, défini par le premier maximum de distance de la trajectoire par rapport à l'origine (le facteur d'échelle est calculé de façon à ce que l'éloignement par rapport à l'origine à ce moment là soit égal à celui de la vérité terrain).

Les figures 4.10 à 4.12 page 80 montrent les résultats obtenus sur la séquence MOCAP_LOOP. On constate que même s'il n'y a aucun décrochage complet suivi de divergence (ce qui peut arriver avec un filtre mal réglé ou une trajectoire trop dynamique ou pas assez observable), la précision est très mauvaise. On distingue trois catégories de comportements :

— Une première partie des trajectoires suit effectivement la trajectoire de la vérité terrain (une fois le facteur d'échelle corrigé), avec cependant une assez grande dispersion, en partie due à des changements de facteur d'échelle au cours de la séquence. Les erreurs de ces trajectoires sont assez faibles, et permettent de distinguer la fermeture de boucle qui intervient entre les abscisses 9 et 10 m, et qui se manifestent par une réduction abrupte de l'erreur et de la dispersion. L'erreur normalisée est également en général inférieure à 3, et montre donc une consistance du filtre raisonnable.

— Les deux autres groupes de trajectoires voient à un moment leur erreur augmenter brutalement, et se décalent toutes de façon similaire, soit en angle, soit en position, soit les deux. Cela arrive dès le début, et est dû au fait qu'il y a peu d'amers proches et que l'observabilité de la trajectoire est donc limitée, ne garantissant pas la convergence vers la bonne trajectoire. L'observabilité revient par la suite puisque la fin de la trajectoire a la bonne forme, mais pas au bon endroit. On constate également avec l'erreur normalisée que le filtre n'est pas conscient de ces erreurs car il ne les intègre pas dans son incertitude, ce qui le rend extrêmement inconsistant. L'augmentation du nombre d'amers observé

permet de limiter fortement ces erreurs, mais il n'est alors plus possible de les traiter en temps réel.

La figure 4.13 page 82 montre que sur la séquence MOCAP_FAST, qui démarre avec des translations répétées dans le plan, il n'y a presque plus d'erreur de convergence initiale, à part quelques unes qui se retrouvent en opposition de phase dès les premiers mouvements sur l'axe y. Même sans considérer ces cas, la précision globale étant médiocre, mais surtout dès que la dynamique du mouvement devient trop importante on assiste à des divergences complètes : la vision ne parvient plus à suivre aucun amer, le modèle à vitesse constante est alors intégré sans être ajusté. Augmenter la dynamique du modèle n'améliore pas le comportement car les zones de recherche deviennent très grandes et les appariemments d'amers difficiles.

On observe donc très clairement les limites d'utilisabilité (facteur d'échelle à corriger), de précision, et de robustesse du SLAM visuel pur, ce qui nous amène à considérer l'intégration de capteurs complémentaires.

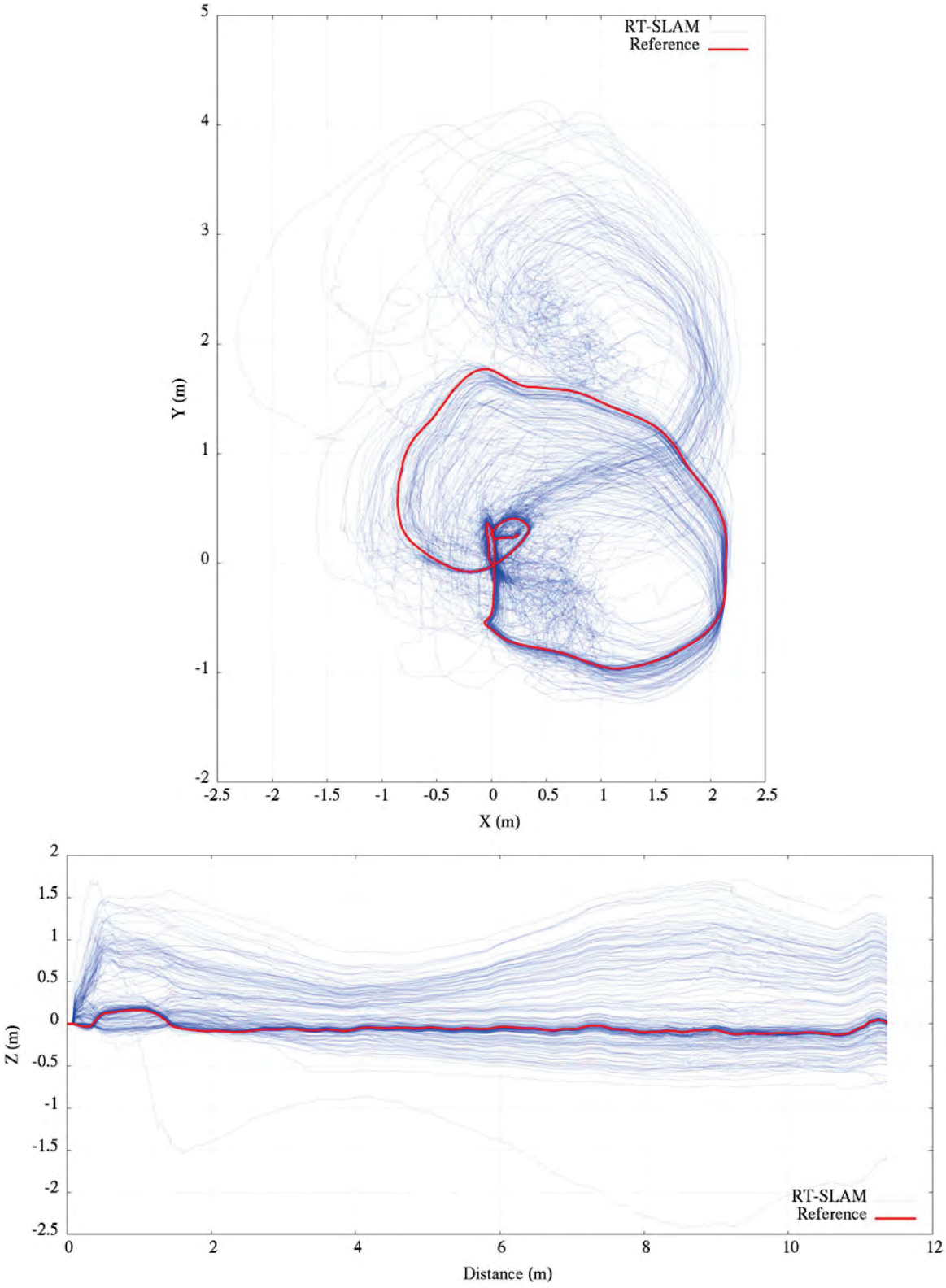


FIGURE 4.10 – Résultat de trajectoire du SLAM visuel-inertiel sur la séquence MOCAP_LOOP.

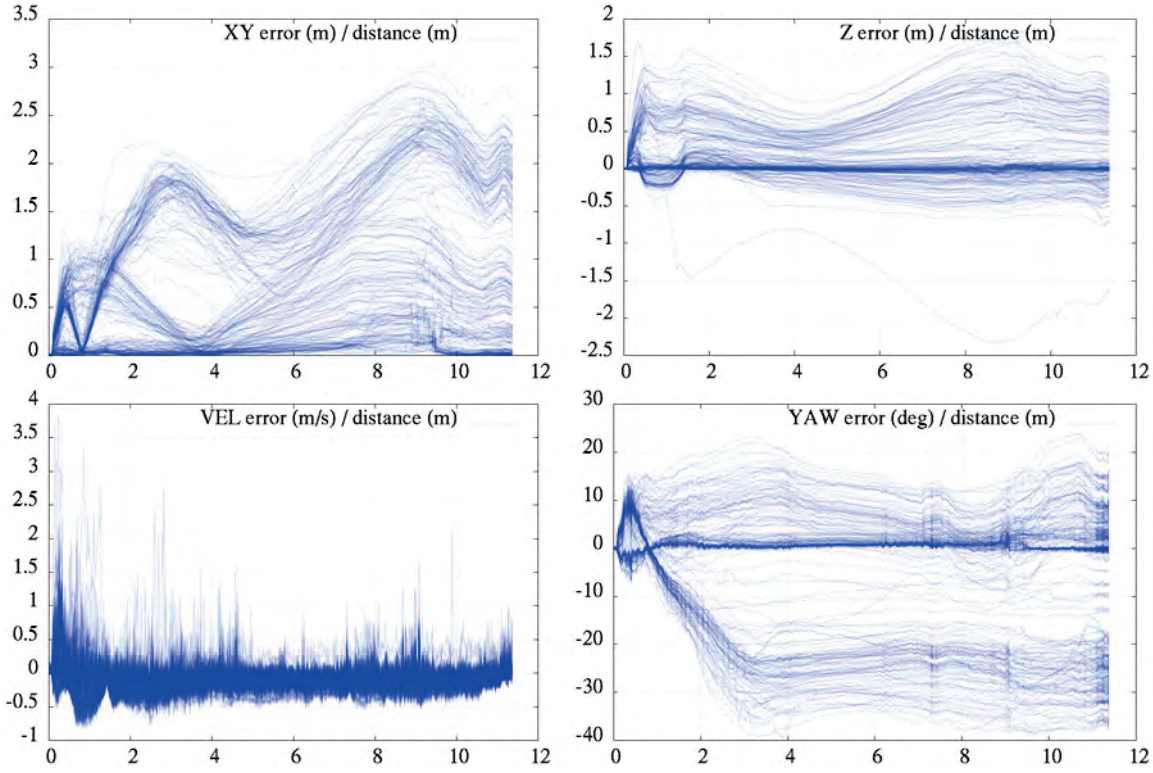


FIGURE 4.11 – Résultat d'erreur du SLAM visuel sur la séquence MOCAP_LOOP.

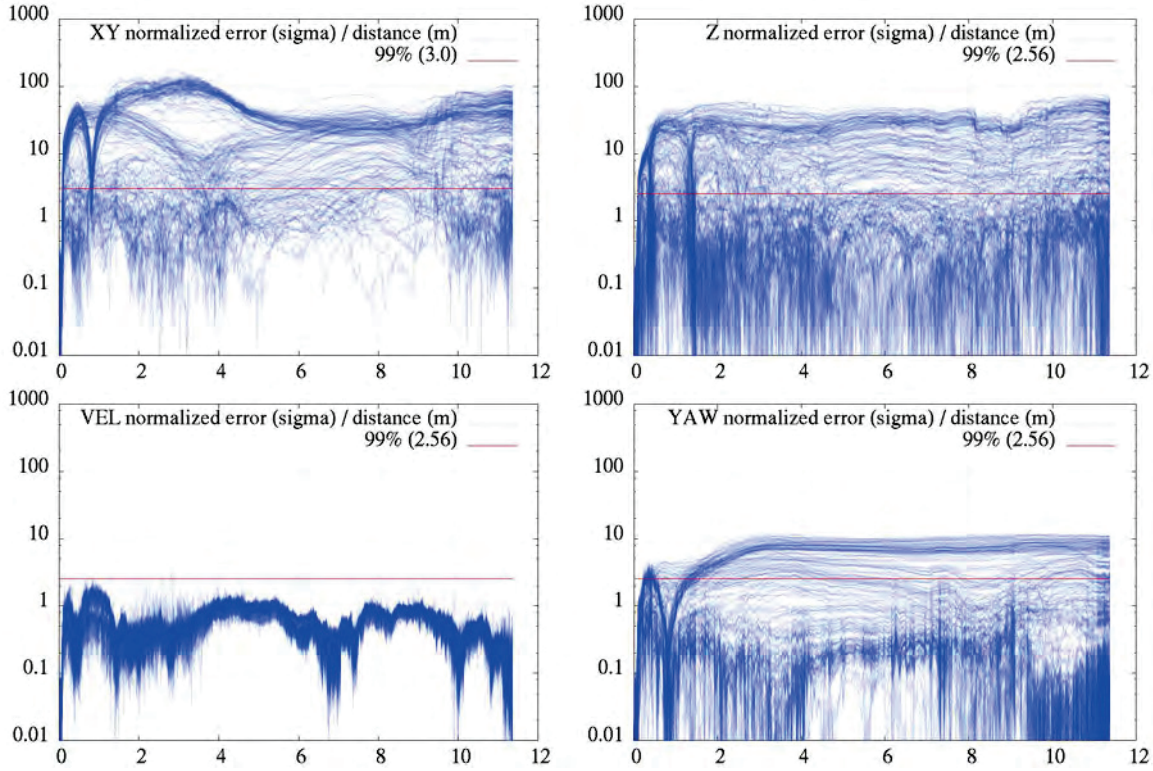


FIGURE 4.12 – Résultat d'erreur normalisée du SLAM visuel sur la séquence MOCAP_LOOP.

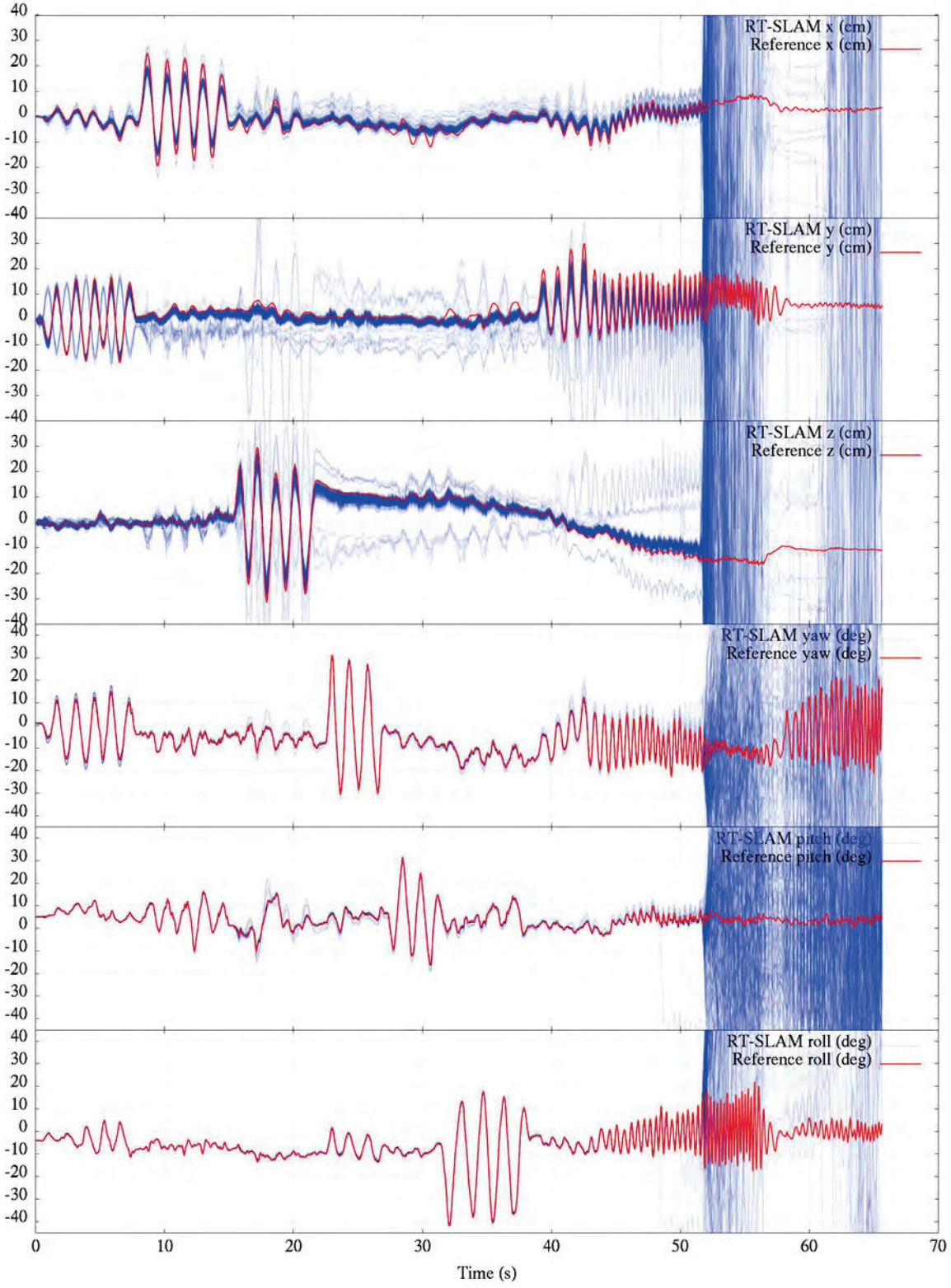


FIGURE 4.13 – Résultat de trajectoire du SLAM visuel sur la séquence MOCAP_FAST.

Introduction d'une centrale inertielle dans le système pour améliorer ses propriétés, après avoir expliqué l'intérêt et les façons d'utiliser un capteur supplémentaire. Présentation des techniques rendues nécessaires par l'utilisation de plusieurs capteurs, analyse des nouvelles limitations du système et résultats.

Chapitre 5

SLAM visuel-inertiel

Sommaire

Nous avons vu dans le chapitre précédent que le SLAM monoculaire pur n'était pas utilisable de façon pratique. Nous commencerons donc ce chapitre en rappelant de façon exhaustive ses différentes limitations et en analysant leurs origines, avant d'étudier pourquoi des capteurs supplémentaires pourraient être utilisés pour lever ces limitations (section 5.1). Nous verrons en particulier comment il est le plus intéressant de les exploiter, dans l'étape de prédiction ou de correction du filtre. Nous expliquerons ensuite comment exploiter une centrale inertielle dans l'étape de prédiction du filtre, avec la modélisation de l'intégration de ses mesures, mais également la modélisation des erreurs de ses mesures (section 5.2). Nous poursuivrons avec une étude de l'observabilité de ce modèle d'erreur en condition expérimentales (section 5.3), et par une présentation des nouveaux problèmes posés par l'utilisation de plusieurs capteurs et de leurs solutions (section 5.4), comme la définition de leurs positions relatives ou leur synchronisation temporelle, avant de conclure par une présentation de résultats (section 5.5).

5.1 Intérêt de capteurs supplémentaires

5.1.1 Limitations du SLAM mono-caméra

Un premier problème est qu'une caméra seule ne permet pas d'observer en absolu des distances. C'est un capteur angulaire, et l'utiliser pour déterminer une distance nécessite de faire de la triangulation, et donc de connaître le déplacement de la caméra. Dans le problème du SLAM où l'on ne connait pas ce déplacement que l'on cherche à estimer simultanément, il est impossible de faire la différence entre un petit déplacement devant des amers proches, ou un grand déplacement devant des amers éloignés (la relation entre le déplacement et la distance des amers étant proportionnelle). La conséquence est que le système va converger vers certaines valeurs, dépendant notamment de la valeur d'initisalisation de la distance des amers par rapport à leur distance moyenne réelle, mais le facteur d'échelle réel de la trajectoire et de la carte obtenues est complètement inconnu, ce qui limite l'utilisabilité pratique du système. De plus de la même manière que la trajectoire dérive lorsque l'on parcourt une grande distance et que les amers sont progressivement renouvelés, ce facteur d'échelle peut dériver et ne pas être constant sur l'ensemble de la trajectoire.

Un second problème est l'imprécision de la prédiction avec un modèle de mouvement, qui est d'autant plus grande que le rapport entre la dynamique du mouvement (et donc l'infidélité au modèle de mouvement) et la fréquence des capteurs utilisés en observation (ici la caméra) est grand. D'une part l'incertitude sur la précision du modèle génère des ellipses d'incertitude des amers dans l'image qui peuvent être grandes, et dans lesquelles la recherche nécessite donc un certain temps de calcul. D'autre part l'imprécision en elle-même de la prédiction a pour conséquence un point de linéarisation erroné, une linéarisation également imprécise, et une correction imprécise. La fréquence faible oblige également à extrapoler cette linéarisation sur un plus grand domaine pour convertir les incertitudes, où elle est encore moins précise, et donc à commettre des erreurs sur l'estimation des incertitudes, rendant l'estimation inconsistante. On notera également que les capteurs extéroceptifs, qui doivent acquérir, transmettre, et traiter de plus grandes quantités de données, sont en général plus limités en fréquence que les capteurs proprioceptifs.

Enfin un dernier problème est que la dynamique limitée du système crée des périodes de temps plus ou moins longues où le modèle fournit une prédiction biaisée. Par exemple avec un modèle à vitesse constante, pendant les périodes d'accélération le modèle va constamment sous-estimer le déplacement, et au contraire pendant les périodes de décélération il va constamment les surestimer. Cela pénalise également la correction de l'état, en terme de précision et de consistance.

En résumé une approche basée uniquement sur une caméra comme capteur extéroceptif présente les limitations suivantes :

1. Absence de facteur d'échelle absolu, et dérive du facteur d'échelle relatif.
2. Temps de calcul des appariements d'amer relativement important.
3. Corrections peu précises et inconsistantes (dues à des linéarisations imprécises, appliquées sur un grand domaine, et des erreurs du modèle de mouvement non gaussiennes).

5.1.2 Utilisation d'un capteur proprioceptif pour l'observation

L'intégration des données d'un capteur proprioceptif peut améliorer certains points cités plus haut, s'il remplit certaines conditions :

1. il fonctionne à une fréquence plus élevée que les capteurs extéroceptifs, afin de réduire la durée d'application du modèle de mouvement,
2. ses données sont suffisamment peu bruitées pour améliorer significativement l'estimation de l'état et donc le point de linéarisation utilisé pour l'intégration des données extéroceptives par la suite (si ce n'est pas le cas cela limite son intérêt, mais n'est pas rhédhibitoire).

Ces deux conditions permettent au capteur, dont les mesures sont intégrées comme des observations de l'état du système, d'augmenter la précision et la robustesse du système en améliorant les points de linéarisation, et pourra améliorer la vitesse en réduisant la taille des zones de recherche active des amers.

À la condition supplémentaire que les données du capteur proprioceptif fournissent l'information de facteur d'échelle (une position, vitesse ou accélération absolues), ce dernier sera également correctement estimé, avec une précision dépendant de ce capteur.

5.1.3 Utilisation d'un capteur proprioceptif pour la prédiction

Sous d'autres conditions plus restrictives, il est également possible d'exploiter les informations issues d'un capteur proprioceptif dans l'étape de prédiction du filtre de Kalman étendu en lieu et place du modèle de mouvement :

1. en plus de fonctionner globalement à une fréquence plus élevée que les capteurs extéroceptifs, il doit également fournir des données de façon suffisamment régulière pour garantir qu'il y aura toujours une prédiction entre des données extéroceptives (idéalement le capteur est tout simplement périodique). Un modèle de mouvement peut-être utilisé pour interpoler les données proprioceptives au moment des observations, sauf si l'on décide de synchroniser de manière matérielle les capteurs.

2. il est cette fois absolument indispensable et non plus seulement préférable que ses données soient peu bruitées, sinon on pourra avoir un fonctionnement pire qu'un simple modèle de mouvement avec des points de linéarisation très faux, et des incertitudes très grandes.
3. il fournit des données complètes, qui permettent de prédire les évolutions de l'ensemble de l'état. Si ce n'est pas le cas il est toujours possible d'adopter une solution hybride en le complétant par un modèle de mouvement.
4. il n'est jamais fautif, c'est-à-dire que ses erreurs sont toujours compatibles avec son incertitude. En effet il n'est pas possible de réaliser l'étape de contrôle de compatibilité pour détecter des données aberrantes quand on utilise le capteur en prédiction, et une prédiction incohérente peut avoir de graves conséquences sur la réussite des appariements d'amers et donc sur le fonctionnement du filtre.
5. il ne mesure pas une grandeur redondante avec un autre capteur à intégrer en observation. En effet cette grandeur devrait nécessairement être inclue dans l'état du filtre pour réaliser la fusion des données des différents capteurs. Mais alors les équations de prédiction ne permettraient au capteur utilisé en prédiction que d'écraser cette partie de l'état, et les mesures de l'autre capteur seraient simplement ignorées au lieu d'être fusionnées. Il est nécessaire dans ce cas d'utiliser les deux capteurs en prédiction en les fusionnant ou les hybridant au préalable. L'exemple typique est une centrale inertielle bas coût utilisée en prédiction, et un gyromètre de cap plus précis en observation.

Un capteur qui satisfait toutes ces conditions apporte de nombreux avantages :

1. l'intégration des données de ce capteur nécessite moins de ressources processeur, car seul le bloc de la matrice de covariance qui dépend directement de l'état du robot a besoin d'être mis à jour, qui est significativement plus petit que la matrice de covariance complète qui contient l'état de tous les amers de la carte. Cet avantage est d'autant plus intéressant que le capteur proprioceptif est rapide.
2. il n'est pas nécessaire d'inclure dans l'état du filtre les grandeurs mesurées par ce capteur, ce qui diminue également un peu les calculs.
3. un tel capteur fournit en général une information avec un bruit plus gaussien qu'un modèle de mouvement, qui aura tendance à sous-estimer sur certaines périodes et surestimer sur d'autres le mouvement, créant ainsi un biais.

5.2 Modèle inertiel

Une centrale inertielle, qui mesure l'accélération et les vitesses angulaires selon trois axes remplit les critères qui viennent d'être énoncés pour être utilisée pour l'étape de prédiction, et est de plus particulièrement complémentaire à une caméra. En effet elle observe la dérivée d'ordre 2 de la position et d'ordre 1 de l'orientation, et mesure donc la dynamique du système. L'inconvénient qui en découle et donc que la position calculée par intégration de ses données dérive très rapidement, mais cet effet est compensé par la caméra qui observe directement la position et l'orientation.

5.2.1 Modélisation d'une centrale inertielle

Une centrale inertielle mesure l'accélération est la vitesse angulaire dans un référentiel inertiel (galiléen), et l'exprime dans son repère local.

Ainsi sur Terre, les accéléromètres mesurent l'intensité de la gravité $\mathbf{g}$, et les gyromètres mesurent la rotation de la terre. Avec des centrales inertielles bas coût comme celles utilisées pour nos applications, la rotation de la Terre ($< 15\,\mathrm{deg/h}$) est bien inférieure à la précision du capteur, et peut donc être négligée. La gravité $\mathbf{g}$ en revanche n'est pas du tout négligeable, et doit donc être estimée. Si sa norme est connue sur Terre avec une certaine précision (environ $9.81\,m/s^2$), tout comme son orientation dans un repère terrestre (verticale et vers le bas), son orientation dans le repère du capteur dépend de l'orientation $\mathbf{q}$ du capteur.

Au-delà de la réalité physique, les mesures des capteurs sont également entâchées d'erreur. Il y a toujours un certain bruit gaussien centré $\mathbf{a_n}$ et $\mathbf{w_n}$, mais également des biais additifs $\mathbf{a_b}$ et $\mathbf{w_b}$ (il peut y également y avoir une erreur de gain mais qui est souvent trop faible pour être correctement estimée).

Les mesures réelles $\tilde{\mathbf{a_m}}$ et $\tilde{\mathbf{w_m}}$ du capteur s'expriment donc ainsi en fonction de l'accélération $\mathbf{a_t}$ et de la vitesse angulaire $\mathbf{w_t}$ réelles dans le référentiel terrestre qui nous intéresse (toujours exprimé dans le repère du capteur) :

$$\tilde{\mathbf{a_m}} = \mathbf{a_t} + \mathbf{a_b} + \mathbf{a_n} + \mathrm{q2R}\left(\mathbf{q}^{-1}\right) \cdot \mathbf{g} \tag{5.1}$$

$$\tilde{\mathbf{w_m}} = \mathbf{w_t} + \mathbf{w_b} + \mathbf{w_n} \tag{5.2}$$

Remarque Si le fait qu'un accéléromètre mesure également la gravité peut-être vu, à raison, comme une contrainte, c'est également un avantage certain, en ce que cela permet d'observer la verticale et donc les deux paramètres d'attitude (assiette et gîte) de manière absolue.

5.2.2 Equation d'évolution

Afin de simplifier grandement les équations et leur linéarité, on suppose que la transformation entre le repère du robot et le capteur inertiel est nul : le robot est donc littéralement le capteur inertiel. Ceci n'est pas une contrainte lorsqu'il n'y a pas d'autre capteur qui exige un placement particulier par rapport au repère du robot, mais de toute façon lorsqu'il y a plusieurs capteurs sur un robot il est plus stable et il y a rarement des difficultés à placer le capteur inertiel au centre du robot.

L'état complet du robot est donc :

$$\mathcal{R} = \left(\mathbf{p}\ \mathbf{q}\ \mathbf{v}\ \mathbf{a_b}\ \mathbf{w_b}\ \mathbf{g}\right)^T \tag{5.3}$$

On déduit alors du modèle du capteur donné équations 5.1 et 5.2 l'équation d'évolution du

système :

$$\begin{aligned}
\mathbf{p}^+ &= \mathbf{p} + \mathbf{v} \cdot \mathrm{d}t && (5.4)\\
\mathbf{q}^+ &= \mathbf{q} \otimes \mathrm{v2q}\left((\mathbf{w_m} - \mathbf{w_b}) \cdot \mathrm{d}t\right) && (5.5)\\
\mathbf{v}^+ &= \mathbf{v} + (\mathrm{q2R}(\mathbf{q}) \cdot (\mathbf{a_m} - \mathbf{a_b}) + \mathbf{g}) \cdot \mathrm{d}t && (5.6)\\
\mathbf{a_b}^+ &= \mathbf{a_b} && (5.7)\\
\mathbf{w_b}^+ &= \mathbf{w_b} && (5.8)\\
\mathbf{g}^+ &= \mathbf{g} && (5.9)
\end{aligned}$$

Variantes sur l'estimation de la gravité De la façon dont est écrite l'équation d'évolution, le repère dans lequel est définie la trajectoire estimée est défini par l'orientation initiale du système. Cela permet dans les cas où l'orientation absolue initiale est parfaitement inconnue, de gérer cette incertitude dans la direction de la gravité, qui intervient de façon linéaire dans les équations, plutôt que dans l'orientation du robot, qui intervient quant à elle de façon non linéaire dans les équations, et qui ne peut donc pas démarrer avec une incertitude trop grande (on serait alors trop sensible au point de linéarisation).

Cependant la trajectoire constituée par l'ensemble des positions estimées est exprimée dans le repère défini par la direction de la gravité, qui est variable au cours du temps si cette dernière continue à changer pendant la séquence, ce qu'elle fait en général. Il est alors très difficile de reconstituer une trajectoire dans un repère fixe, cela nécessitant de réintégrer les déplacements en corrigeant avec la direction courante de la gravité.

De plus grâce à la centrale inertielle, on peut très souvent obtenir une estimation raisonnable de l'orientation absolue initiale, en supposant que le système est à peu près statique avant le démarrage, ce qui est le cas pour un robot en général. On peut alors n'estimer que la norme de la gravité, en imposant sa direction verticale, ce qui simplifie grandement l'interprétation postérieure de la trajectoire obtenue, ou l'intégration de capteurs de position absolue dans le filtre.

On estime alors simplement g, et on utilise dans l'équation de prédiction pour $\mathbf{v}^+$ le vecteur $\mathbf{g} = -g \cdot (0\ 0\ 1)^T$.

En outre la valeur de g peut en réalité être connue selon la position géographique sur Terre avec une précision supérieure à celle des capacités d'estimation du système, d'autant qu'il y a ambiguïté d'observabilité entre les biais des accéléromètres et g lorsque les angles d'attitude changent peu. Il est donc plus simple et plus précis d'utiliser une valeur statique si l'on ne souhaite pas estimer sa direction.

5.2.3 Calcul des paramètres de bruit

Le réglage manuel des paramètres d'un modèle est une tâche toujours fastidieuse, et qui ne garantit pas de trouver les valeurs optimales. Dans certains cas il n'est pas possible de faire autrement, comme dans le cas des erreurs des modèles de mouvement avec des capteurs qui n'observent pas le facteur d'échelle, même si une compréhension de leur signification physique permet de déterminer un ordre de grandeur et facilite le processus. Mais dans d'autres cas,

notamment comme ici quand on dispose d'un capteur spécifié par le fabricant, ou que l'on peut caractériser, il est plus efficace et plus fiable de calculer ces paramètres à partir de ces spécifications.

La table 5.1 montre un exemple de spécifications fournies par le fabricant de la centrale inertielle que nous avons exploitée dans la documentation technique. On constate que toutes les informations sont fournies en modèle continu, quant bien même les données sont fournies de façon discrète, et il est donc nécessaire d'effectuer des conversions pour déterminer les paramètres à renseigner dans le filtre.

		Units	Accelerometer m/s^2	Gyrometer °/s	Gyrometer rad/s
Dimensions			3 axes	3 axes	
Full Scale			± 50	± 300	± 5.24
Noise density	n_d	[units/$\sqrt{\text{Hz}}$]	0.002	0.05	$9{\cdot}10^{-4}$
Bandwidth	B_f	[Hz]	30	40	
Bias stability	σ_b	[units 1σ]	0.02	1	0.017
ADC resolution		[bits]	16	16	
Sampling frequency	f_s	[Hz]	100	100	

TABLE 5.1 – Spécifications de performance de la XSens MTi.

On cherche à déterminer les paramètres suivants :

— bruits de mesure : σ_{a_n} et σ_{w_n},

— dérive des biais, caractérisée par le bruit blanc responsable du *random walk* des biais : σ_{a_w} et σ_{w_w},

— incertitude initiale des biais : $\sigma_{a_b}(0)$ et $\sigma_{w_b}(0)$,

— incertitude initiale des autres paramètres de l'état : position $\sigma_p(0)$, vitesse $\sigma_v(0)$, orientation $\sigma_q(0)$ et gravité $\sigma_g(0)$.

a Bruits de mesure

Les premiers paramètres à calculer sont les écarts-types des bruits de mesure. La spécification constructeur fournit seulement la densité spectrale de bruit n_d. Dans un capteur qui échantillonne un signal continu, on place un filtre passe-bas de fréquence de coupure au plus égale à la moitié de la fréquene d'échantillonnage, conformément au théorème de Shannon. Le capteur a alors une bande passante B_f limitée également indiquée dans la spécification. La puissance totale du bruit qui se retrouve dans la mesure est donc l'intégrale du carré de la densité de bruit sur la plage de fréquence correspondant à la bande passante, et la valeur RMS de ce bruit est la racine carrée de la puissance :

$$\sigma_n = \sqrt{\int_{B_f} n_d^2} \tag{5.10}$$

On considère en général la densité spectrale de bruit constante, et on ajoute un coefficient multiplicateur pour tenir compte du fait que le filtre n'est pas parfait et déborde de la fréquence

de coupure, en l'occurrence 1.6 pour un filtre du premier ordre :

$$\sigma_n = n_d \cdot \sqrt{1.6 \cdot B_f} \tag{5.11}$$

Avec les valeurs de la table 5.1 page précédente, on obtient les valeurs suivantes :

$$\sigma_{a_n} = 14 \cdot 10^{-3}\,\mathrm{m/s^2} \qquad \sigma_{w_n} = 7 \cdot 10^{-3}\,\mathrm{rad/s} \tag{5.12}$$

Si la spécification constructeur n'est pas disponible, on peut également caractériser expérimentalement le capteur en calculant l'écart-type des mesures, le capteur restant parfaitement statique. Des mesures nous ont permis d'obtenir :

$$\sigma_{a_n} = 8 \cdot 10^{-3}\,\mathrm{m/s^2} \qquad \sigma_{w_n} = 6 \cdot 10^{-3}\,\mathrm{rad/s} \tag{5.13}$$

Les valeurs obtenues sont bien du même ordre de grandeur, et même légèrement inférieures aux spécifications. En revanche rien ne garantit que l'on ne pourra pas avoir un bruit supérieur à celui mesuré dans d'autres conditions environnementales.

b Dérive des biais

Connaître approximativement la vitesse de dérive des biais est nécessaire pour augmenter l'incertitude des biais lors de chaque prédiction, afin de les autoriser à changer. Elle n'est en revanche pas indiquée dans la documentation technique du constructeur (la grandeur *bias stability* correspondant au domaine de variation du biais).

Cette valeur n'est cependant pas critique, car même une valeur nulle n'empêchera pas l'estimation de ce biais de changer très lentement, mais il est tout de même intéressant de connaître un ordre de grandeur. On peut pour cela caractériser le capteur.

On modélise les variations du biais comme issues de l'intégration d'un bruit blanc continu $\mathbf{a_w}$ (resp. $\mathbf{w_w}$), d'écart-type σ_{a_w} (resp. σ_{w_w}) :

$$\mathbf{a_b}^+ = \mathbf{a_b} + \int_t^{t+\mathrm{d}t} \mathbf{a_w} \cdot \mathrm{d}t \tag{5.14}$$

$$\mathbf{w_b}^+ = \mathbf{w_b} + \int_t^{t+\mathrm{d}t} \mathbf{w_w} \cdot \mathrm{d}t \tag{5.15}$$

La mise à jour de l'incertitude des biais lors de la prédiction consiste alors simplement à y ajouter une valeur :

$$\mathbf{P}(\mathbf{a_b})^+ = \mathbf{P}(\mathbf{a_b}) + \sigma_{a_w}^2 \cdot \mathbf{I} \cdot \mathrm{d}t \tag{5.16}$$

$$\mathbf{P}(\mathbf{w_b})^+ = \mathbf{P}(\mathbf{w_b}) + \sigma_{w_w}^2 \cdot \mathbf{I} \cdot \mathrm{d}t \tag{5.17}$$

Il s'agit donc d'estimer σ_{a_w} et σ_{w_w}.

Pour cela, on enregistre les données de la centrale inertielle pendant plusieurs heures, en statique, et on calcule le biais au cours du temps sur des périodes de durée T. On calcule ensuite l'écart-type des variations de ce biais au bout d'un temps $n \cdot T$, et on ajuste une

évolution en racine carrée du temps sur cette courbe (puisque la variance évolue linéairement avec le temps, l'écart-type évolue avec la racine carrée du temps). Le coefficient de la racine carrée correspond alors aux écarts-types σ_{a_w} et σ_{w_w} recherchés.

La figure 5.1 montre un exemple de telle caractérisation pour $T = 100\,\mathrm{s}$, et n variant de 1 à 6. Si on souhaite identifer la vitesse maximale de dérive du biais il est en effet nécessaire de ne pas considérer une période trop grande, où le fait que le biais est contraint à un domaine réduit réduira en moyenne la dérive. Les points ne sont pas parfaitement alignés sur la courbe d'une part à cause du bruit de mesure, et d'autre part car notre modèle ne correspond pas exactement à la réalité ; mais nous souhaitons seulement trouver les meilleures valeurs des paramètres pour notre modèle. On retiendra donc que la vitesse maximale de dérive des biais des accéléromètres est de l'ordre de $\sigma_{a_w} = 1{\cdot}10^{-4}\,\mathrm{m/s^2/\sqrt{s}}$, et celle des biais des gyromètres de l'ordre de $\sigma_{w_w} = 2{\cdot}10^{-5}\,\mathrm{rad/s/\sqrt{s}}$, valeurs que l'on pourra utiliser dans le filtre.

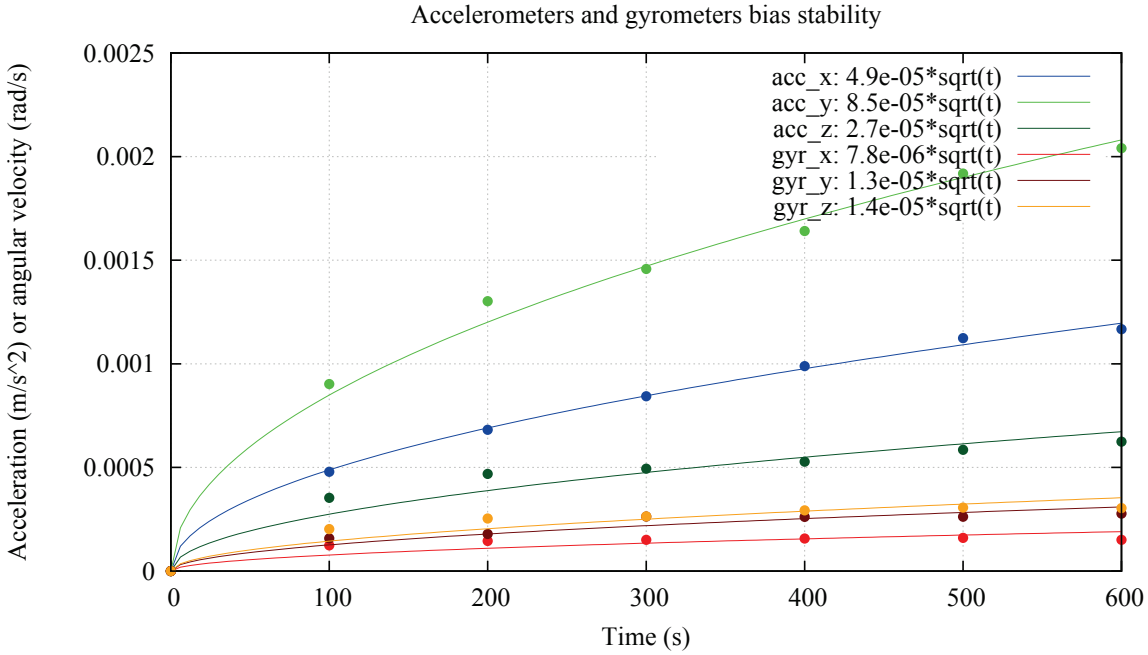


FIGURE 5.1 – Exemple de caractérisation de la vitesse de dérive du biais des accéléromètres et gyromètres.

Notons qu'il existe des modèles de biais plus complexes comme expliqué par exemple dans [Gebre-Egziabher, 2004], comportant un terme d'atténuation avec constante de temps pour tenir compte du fait que le biais reste borné, et dont les paramètres peuvent être calculés à partir de la variance d'Allan et de l'autocorrélation du signal en statique sur de très longues durées.

c Incertitudes initiales

Biais L'incertitude initiale des biais est donnée par la grandeur *bias stability* indiquée dans la table de spécifications fournie dans la documentation technique du constructeur. En effet puisque cette grandeur correspond au domaine de variation des biais, c'est la meilleure estimée

qu'on a en l'absence d'autre information.

Position La position absolue n'est pas observable avec une caméra et une centrale inertielle, donc il convient de fixer une incertitude initiale nulle. Si un capteur observant la position absolue est intégré dans le filtre, il faut alors initaliser la position absolue initiale et son incertitude comme il se doit.

Orientation Si la gravité est estimée comme un vecteur à 3 dimensions, l'orientation est estimée de façon relative comme la position, et on peut donc l'initialiser à 0 avec une incertitude nulle. Si en revanche seule la norme de la gravité est estimée, l'orientation doit être initialisée le plus précisément possible et l'incertitude en adéquation (en pratique une erreur d'initialisation de $10\,^\circ$ est correctement corrigée). En supposant que l'on est à peu près statique avant le démarrage, il est possible de l'initialiser en mesurant la gravité par une moyenne de l'accélération sur quelques secondes pour rendre la gravité verticale. L'incertitude s'obtient par la transformation de la précision de cette mesure de gravité.

Vitesse initiale La vitesse initiale est en général inconnue, mais il est préférable de démarrer avec une vitesse faible afin de faciliter le fonctionnement des algorithmes de suivi visuel. L'écart-type de l'incertitude peut alors être initialisé à environ la moitié de la vitesse initiale maximale envisageable. Dans le cas d'un robot, on peut raisonnablement s'imposer de démarrer l'algorithme pendant que le robot est à l'arrêt, et ainsi donner une incertitude très faible sur la vitesse.

Gravité Si l'orientation initiale est parfaitement inconnue, la gravité doit être estimée comme un vecteur à 3 dimensions, et on ne peut alors l'initialiser qu'avec une valeur nulle et une incertitude sur chaque axe couvrant sa norme sur Terre (par exemple $\sigma_g = 5\,\mathrm{m/s^2}$). S'il est possible de maintenir un mouvement faible pendant quelques secondes ou moins avant le démarrage du système, il est possible d'initialiser la gravité avec la valeur moyenne mesurée sur les accéléromètres pendant cette période, et l'incertitude appropriée (on peut aussi imposer la norme à une valeur standard). Dans ce cas l'orientation doit être être également initialisée en conséquence.

5.3 Étude expérimentale de l'observabilité des biais

L'estimation correcte des biais est principalement utile pour limiter la dérive de la localisation lorsqu'elle n'est plus stabilisée par la vision, afin d'être robuste aux perturbations de la vision les plus longues possibles (qui peuvent provenir de masquages, d'éblouissements ou au contraire de baisse de luminosité, etc). En effet leur valeur étant relativement faible (quelques centièmes de $\mathrm{m/s^2}$ pour les accéléromètres par exemple), leur effet n'est visible que sur une période d'intégration approchant la seconde, et est négligeable entre deux images.

On doit cependant se poser la question de l'observabilité de ces biais, qui nécessitent d'intégrer des données sur une longue durée pour s'affranchir du bruit, qui dans le cas des accéléromètres

sont combinés à la gravité d'une manière dépendant de l'orientation, obligeant à explorer plusieurs orientations pour espérer parvenir à séparer la contribution de chacun. On cherche donc à analyser dans cette section comment un filtre de Kalman y parvient.

5.3.1 Évaluation de l'estimation des biais

Nous utilisons dans cette section la séquence MOCAP_FAST, présentée section 3.3.2.b page 44, qui est intéressante car elle stimule tous les degrés de liberté, et a été enregistrée avec une vérité terrain par capture de mouvement qu'il est possible d'injecter dans le filtre en observation à la place de la caméra, comme cela est détaillé section 6.4.1 page 152. Le but est de s'affranchir des problèmes de vision pour étudier l'observabilité des grandeurs souhaitées avec un capteur de position parfait. L'inconvénient de cette séquence est qu'aucun calibrage des biais n'a été effectué au moment de son enregistrement, mais cela n'est pas nécessaire pour ce que nous voulons montrer.

a Gyromètres

La figure 5.2 montre que l'estimation des biais des gyromètres de la centrale inertielle converge correctement. Si on remplace l'intégration de la caméra dans le filtre par l'intégration de la vérité terrain issue du système de capture de mouvement, afin de garantir une observabilité parfaite, on obtient des courbes quasiment identiques.

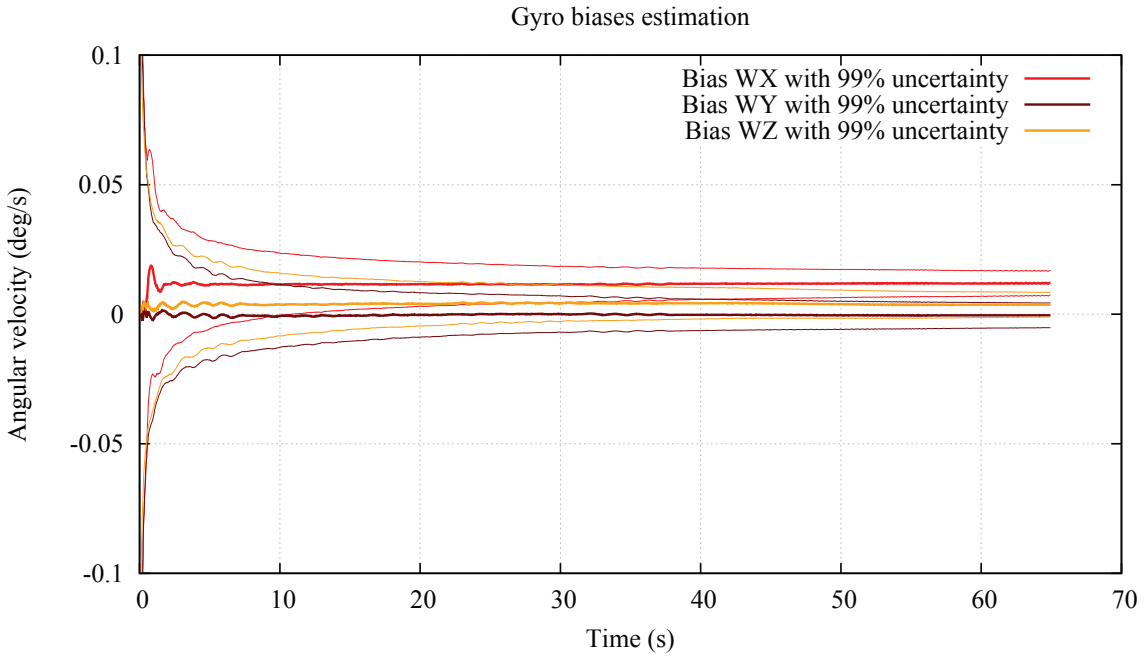


FIGURE 5.2 – Estimation des biais des gyromètres avec une IMU+caméra.

b Accéléromètres

Si on représente la valeur des biais estimée au cours du temps, ainsi que la norme et l'orientation du vecteur gravité comme figure 5.3 , on constate plusieurs choses :

- L'estimation ne converge pas vers des valeurs stables.
- L'estimation est temporellement inconsistant. En effet les biais sont définis comme constants dans le filtre, donc la courbe délimitante supérieure de la plage d'incertitude ne devrait jamais passer en dessous du maximum dans le passé de la courbe délimitante inférieure, et la courbe inférieure jamais au dessus du minimum dans le passé de la courbe supérieure. On peut interpréter ce phénomène comme un échec du filtre à fusionner toute l'information dans l'état courant pour respecter l'hypothèse markovienne, ou de façon plus imagée à un oubli du passé.
- On observe une corrélation entre la norme de la gravité et le biais en z, et entre le *pitch* de la gravité et le biais en x. Ceci est lié au manque de consistance temporelle évoqué plus tôt : pour correctement observer la gravité et les biais, le filtre doit se souvenir de l'ensemble de la séquence avec des attitudes différentes.
- Les valeurs estimées pour ces biais ne sont pas réalistes, car si nous n'avons pas pu les calibrer précisément au moment de l'enregistrement de cette séquence, nous savons que leur ordre de grandeur est de quelques centièmes de $\mathrm{m/s^2}$.

On peut alors faire plusieurs tests :

- Si on remplace la caméra par une vérité terrain issue d'un système de capture de mouvement (*motion capture* abbrévié en *mocap*) (figure 5.4), on obtient des données moins bruitées mais le filtre ne parvient toujours pas à estimer correctement les biais (pas de convergence, inconsistance, corrélations entre gravité et biais).
- Si toujours avec le système de capture de mouvement à la place de la caméra, on n'estime cette fois que la norme de la gravité (figure 5.5 page 98), alors on obtient une convergence et consistance de l'estimation des biais en x et y, mais toujours les mêmes phénomènes entre la norme de la gravité et le biais en z.
- En revanche si on utilise à nouveau la caméra à la place du système de capture de mouvement (figure 5.6 page 98), le système ne converge plus pour aucun axe. Cela se comprend par le fait qu'en estimant seulement la norme de la gravité, la gravité définit la verticale, et le système de capture de mouvement observant directement l'orientation il n'y a plus d'ambiguïté. En revanche avec une caméra l'orientation n'est plus observée directement mais seulement à travers la gravité, et il y a donc à nouveau ambiguïté entre les biais et l'orientation.

On peut donc conclure qu'un EKF n'a pas les propriétés nécessaires de consistance pour estimer précisément des grandeurs qui ne sont pas observables instantanément, *i.e.* qui nécessitent une mémoire sur le long terme pour être observables. De la même façon qu'on a vu section 4.4.2 page 67 qu'un objet lentement mobile pouvait en introduisant un biais dans l'erreur faire dériver progressivement le filtre, ici un manque d'observabilité conduit le filtre à dériver lentement.

Intégrer au filtre la vérité terrain permet de rendre observable instantanément les biais des accéléromètres, à condition qu'il ne demeure pas d'ambiguïté entre deux grandeurs estimées

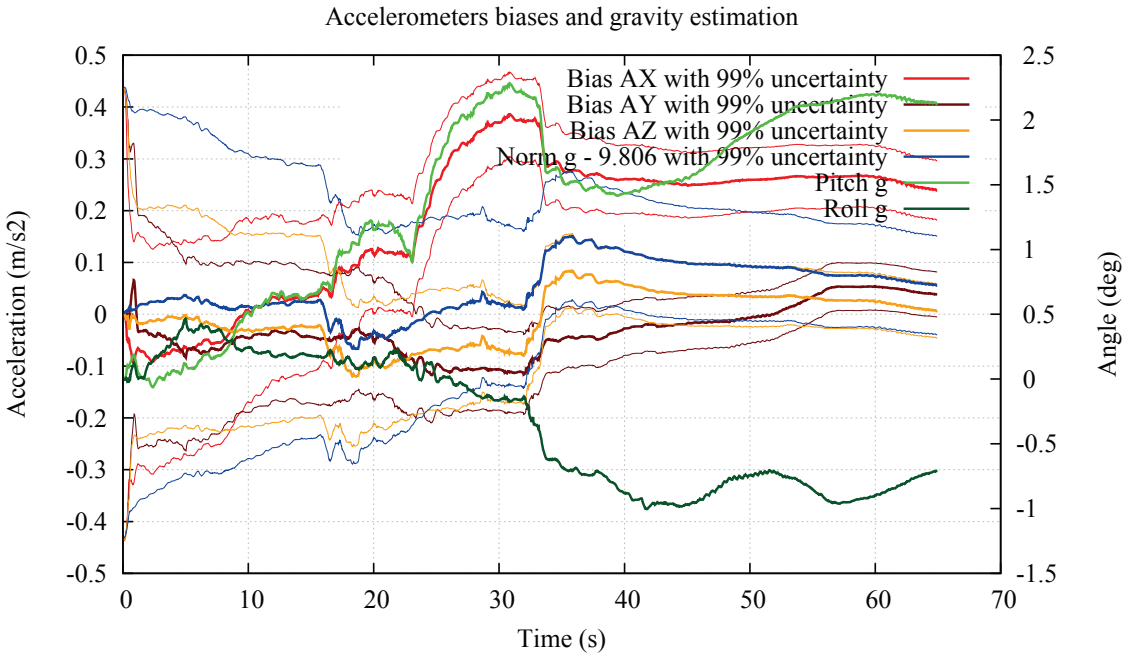


FIGURE 5.3 – Estimation des biais des accéléromètres et de la gravité avec une IMU+caméra.

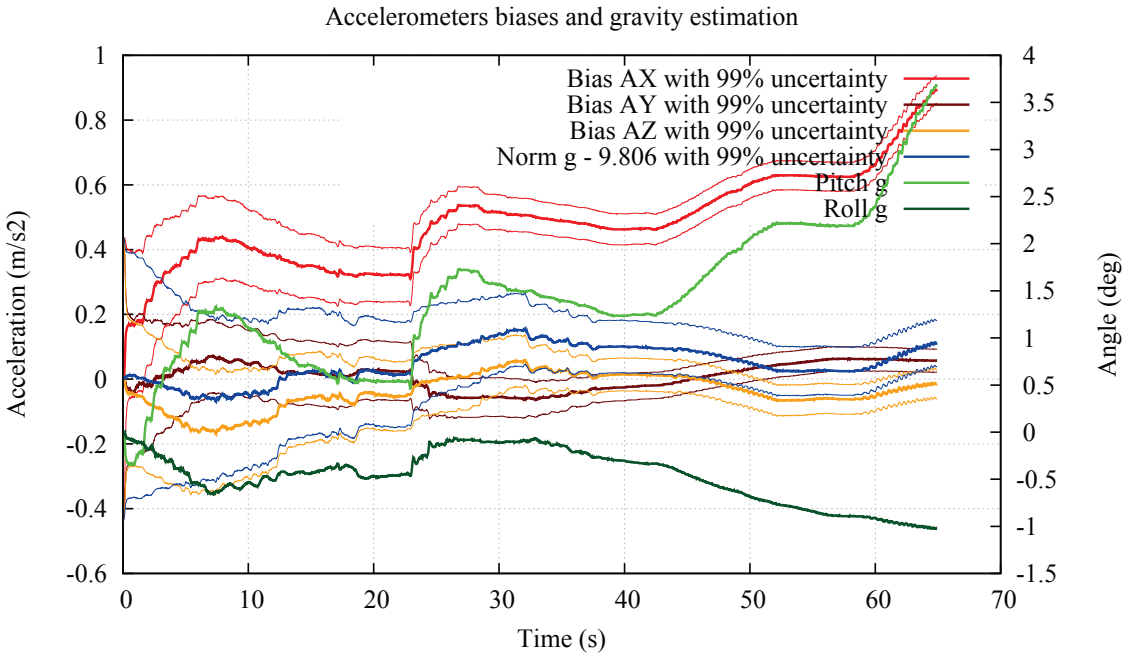


FIGURE 5.4 – Estimation des biais des accéléromètres et de la gravité avec une IMU+mocap.

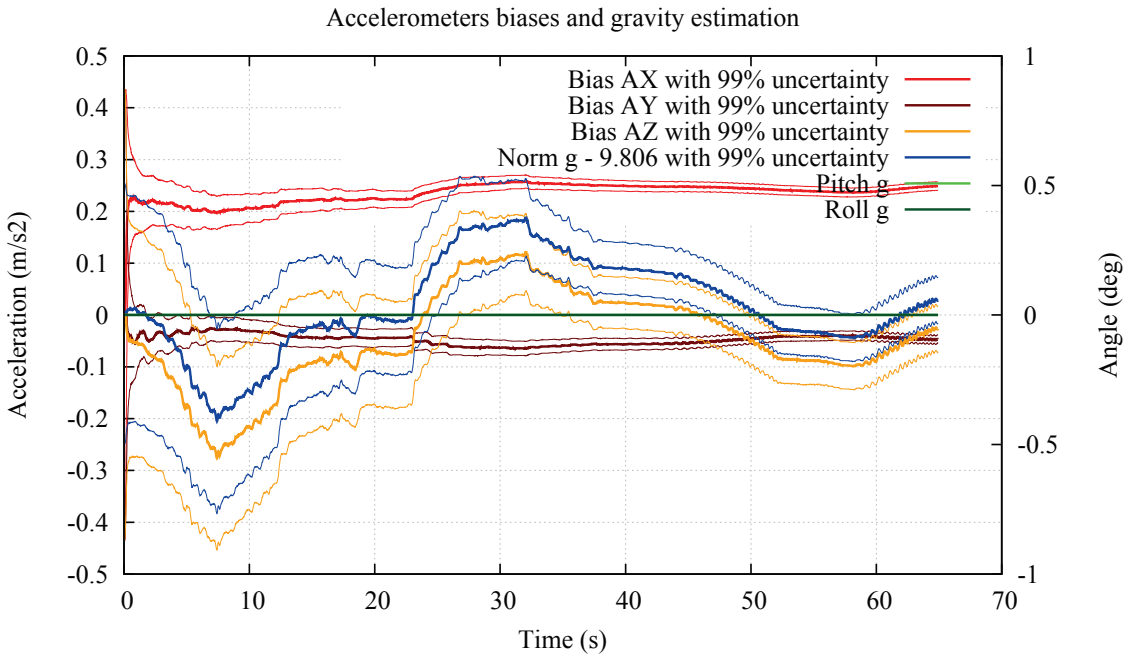


FIGURE 5.5 – Estimation des biais des accéléromètres et de la norme de la gravité avec une IMU+mocap.

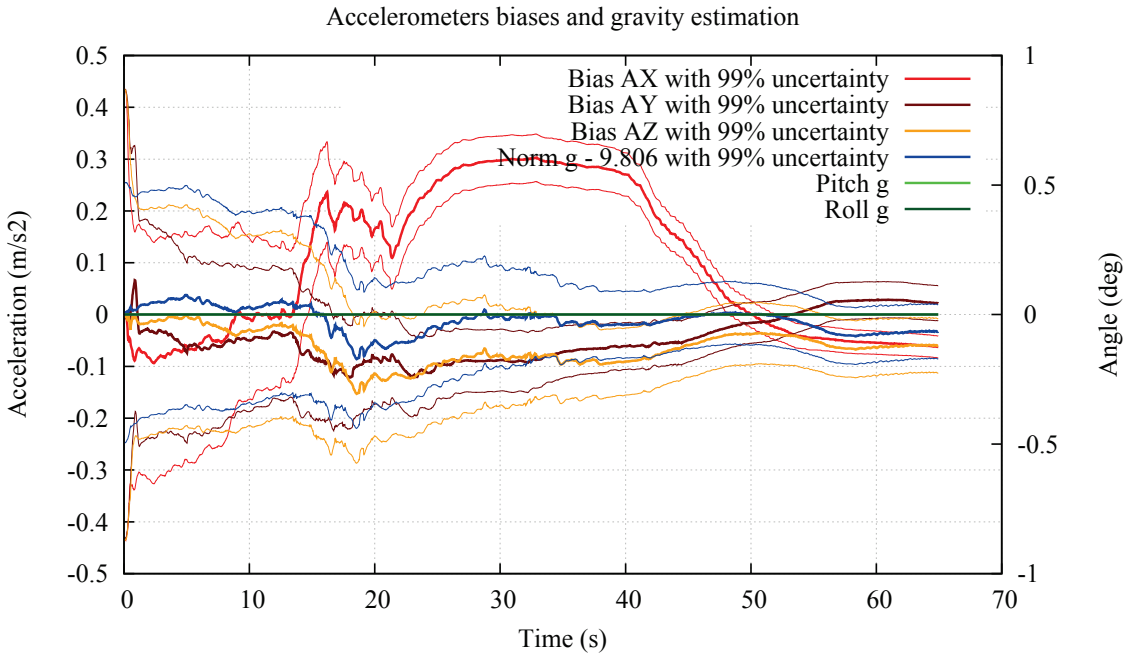


FIGURE 5.6 – Estimation des biais des accéléromètres et de la norme de la gravité avec une IMU+caméra.

pendant trop longtemps, comme ce peut être le cas entre les biais des accéléromètres et la gravité si les angles d'attitude ne changent pas suffisamment.

5.3.2 Calibrage hors ligne des biais

Afin de connaître la valeur précise des biais de la centrale, nous avons mis en place une procédure de calibrage hors ligne.

Afin d'identifier les biais, il sera toujours nécessaire d'atténuer fortement l'influence du bruit, dont l'amplitude n'est pas du tout négligeable devant celle des biais. On réalise pour cela des moyennages des mesures en position statique. Afin de savoir sur quelle durée doivent être réalisées ces moyennages pour obtenir une atténuation suffisante, on peut utiliser la loi des grands nombres :

$$x_i \sim \mathcal{N}(\bar{x}, \sigma_x) \;:\quad p\left(\left|\frac{1}{n}\sum_{i=1}^{n} x_i - \bar{x}\right| \geq \varepsilon\right) \leq \frac{\sigma_x^2}{(n \cdot \varepsilon^2)} \tag{5.18}$$

Ainsi, dans l'unité qui convient, avec un bruit d'écart-type 0.007, une précision souhaitée de 0.001 sur le calcul des biais (soit moins de 10% de précision pour des biais de l'ordre de quelques centièmes), on obtient avec 1000 mesures (soit 10 s avec une centrale XSens Mti) une probabilité de 5% de ne pas obtenir la précision souhaitée. Si on a le temps on peut par sécurité prendre 10000 mesures (soit 100,s) pour diviser par 10 cette probabilité et la rendre négligeable. Il ne faut cependant pas trop augmenter le nombre de mesures car on se heurte ensuite au problème de stabilité des biais, qui ne sont pas parfaitement constants.

a Gyromètres

Les biais des gyromètres sont simples à estimer, car les phénomènes mesurés extérieurs à leur mouvement dans le référentiel terrestre, à savoir la rotation de la Terre, sont très faibles :

— soit ils sont directement négligeables à côté des erreurs des gyromètres, ce qui est le cas pour la plupart des capteurs de basse et moyenne gamme.

— soit l'erreur de leur modélisation grossière est négligeable à côté des erreurs des gyromètres. Une modélisation grossière serait de placer le capteur à peu près horizontalement et orienté vers le nord, et de corriger avec les biais suivants dus à la rotation de la Terre :

$$b_x = 15 \cdot \cos(lat) \qquad b_y = 0 \qquad b_z = 15 \cdot \sin(lat) \qquad \text{deg/h} \tag{5.19}$$

Dans le cas de notre capteur, le biais maximal dû à la rotation de la Terre (15 deg/h ou $7.3 \cdot 10^{-5}$ rad/s) est bien inférieur aux valeurs de biais classiques du capteur, et à la précision qu'il est possible d'obtenir du biais avec un temps de moyennage raisonnable.

Nous réalisons donc le calibrage des biais sans tenir compte de la rotation de la Terre, avec un moyennage sur 10000 mesures, ce qui est tout à fait praticable puisque n'ayant besoin d'être fait qu'une seule fois. Nous obtenons les valeurs :

— biais : $(0.0002, -0.0029, -0.0060)$ rad/s.

Par curiosité, nous avons également effectué ce calibrage en continu pendant près de trois jours, afin d'avoir une représentation de la stabilité de ces biais. Les résultats, présentés figure 5.7, montrent une relative stabilité au cours du temps.

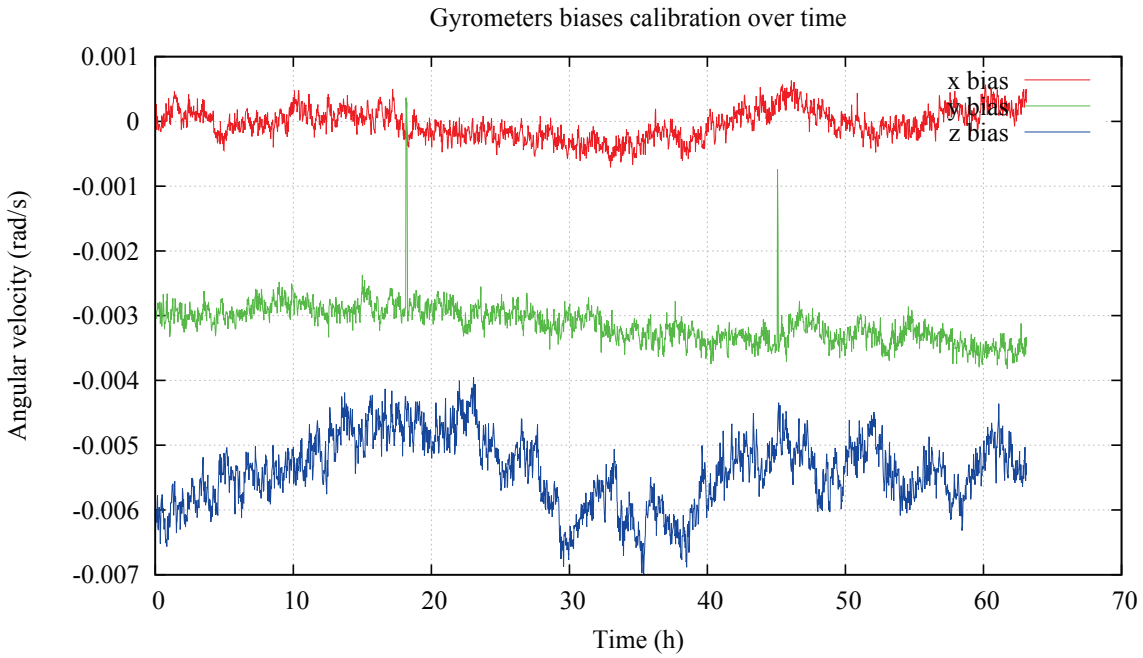


FIGURE 5.7 – Calibrage des biais des gyromètres en continu pendant 63h, avec une période de moyennage de 100s.

b Accéléromètres

Le calibrage des biais des accéléromètres est plus délicat car les phénomènes mesurés extérieurs à leur mouvement dans le référentiel terrestre, à savoir la gravité, sont cette fois prépondérants. Il est nécessaire de séparer les contributions des biais de celle de la gravité, en fonction de l'orientation du capteur, mais la gravité étant très grande devant les biais à mesurer il faut connaître cette orientation très précisément.

Nous avons donc pour cela enregistré 14 séquences dans des orientations différentes, correspondant aux 8 échantillonnages à 45° de l'angle d'assiette, additionné aux 8 échantillonnages à 45° de l'angle de gîte, et diminué de 2 de ces orientations qui sont redondantes (avec un angle de 0° et de 180°). Cela permet de projeter la gravité sur chacun des axes dans un sens et dans l'autre, ainsi que de la projeter sur des paires d'axes. Cette fois nous n'utilisons que des séquences de 10 s pour rendre la procédure pratiquable, mais nous avons vu que en préambule de cette section que cela permet d'atteindre une précision suffisante.

Pour chacune de ces séquences, les valeurs moyennes de chacun des axes sont calculées, ainsi que l'écart-type moyen des mesures afin de s'assurer de l'absence de mouvements ou vibrations perturbateurs qui auraient pu fausser les mesures. La table 5.2 présente un exemple de résultat obtenu.

Nous effectuons ensuite une optimisation numérique à l'aide du logiciel Octave (le script

roll (°)	**pitch (°)**	$\mathbf{a_x}$ **(m/s²)**	$\mathbf{a_y}$ **(m/s²)**	$\mathbf{a_z}$ **(m/s²)**	$\sigma_\mathbf{a}$ **(m/s²)**
0.	0.	-0.0759	-0.0400	9.8356	0.0072
0.	-45	5.3434	-0.0024	8.2226	0.0090
0.	-90.	9.7314	0.4094	0.4511	0.0084
180.	-45.	6.2766	0.4724	-7.4345	0.0079
180.	0.	0.2067	0.1643	-9.7698	0.0087
180.	45.	-7.0682	-0.3652	-6.8259	0.0082
0.	90	-9.8605	-0.2053	-0.1294	0.0081
0.	45.	-6.1075	-0.0853	7.7383	0.0077
45.	0.	0.0838	6.8837	6.9749	0.0079
90.	0.	0.0735	9.7709	0.0557	0.0085
135.	0.	0.2188	5.1221	-8.3122	0.0085
-135.	0.	0.1048	-6.8482	-7.0127	0.0083
-90.	0.	0.0272	-9.8383	0.0460	0.0077
-45.	0.	-0.1663	-6.1510	7.6959	0.0082

TABLE 5.2 – Exemple de valeurs obtenues pour le calibrage des accéléromètres d'une centrale XSens MTi.

`calib_inertial.m` qui utilise la fonction de minimisation par descente de gradient `fminunc` est fourni avec la bibliothèqe RT-SLAM). Il y a 6 inconnues globales de calibrage : les erreurs de biais et de gain sur chacun des axes. Chaque orientation fournit 3 équations indépendantes (une par axe) mais aussi 2 inconnues supplémentaires (les angles d'assiette et de gîte précis, initialisés à 45 ° près pour éviter les minima locaux). Il faut donc au minimum des mesures dans 6 orientations différentes. La valeur de la gravité est fixée à $9.80659\,\mathrm{m/s^2}$ (valeur fournie pour Toulouse – France par le modèle EGM2008 utilisé par Wolfram Alpha [Wolfram, 2013]), car elle est directement liée aux erreurs de gain et ne peut donc pas être estimée simultanément.

Avec les données présentées table 5.2, on obtient les résultats suivants :

— biais : $(-0.056343, -0.032987, 0.029774)$ m/s²

— erreurs de gain : $(1.000047, 0.999821, 0.999972)$

— écart-type de l'erreur résiduelle avec calibrage : 0.0036020 m/s²

— écart-type de l'erreur résiduelle sans calibrage : 0.071857 m/s²

— écart-type du bruit de mesure : 0.008 m/s²

Le fait que l'erreur résiduelle avec calibrage soit inférieure au bruit de mesure (et de l'ordre de grandeur de ce à quoi on pouvait s'attendre en moyennant sur 10 s d'après la loi des grands nombres), et très inférieure à l'erreur résiduelle sans calibrage (biais nuls et gains unitaires) valide le protocole.

Ces résultats confirment que les erreurs de gain peuvent légitimement être négligées (leur effet est environ 10 fois plus faible que les biais). Le fait que leur moyenne soit également très proche de 1 (à 10^{-4} près) montre également que la valeur de g introduite statiquement est correcte (à 10^{-3} près).

Les erreurs de biais semblent également relativement stables au cours du temps, comme le montrent les résultats des différents calibrages obtenus à quelques jours d'intervalle table 5.3.

	Biais			**Gains**		
J=0	-0.056343	-0.032987	0.029774	1.000047	0.999821	0.999972
J+1	-0.056415	-0.041164	0.031331	0.999874	0.999870	1.000130
J+7	-0.055306	-0.034379	0.030392	0.999897	0.999942	1.000119

TABLE 5.3 – Résultats de calibrage obtenus différents jours sur le même capteur.

Avec les mêmes données que celles acquises pour étudier la stabilité des biais des gyromètres présentées figure 5.7 page 100, on peut étudier de même la stabilité des biais des accéléromètres, figure 5.8, pour constater que les changements ne sont en effet pas très rapides.

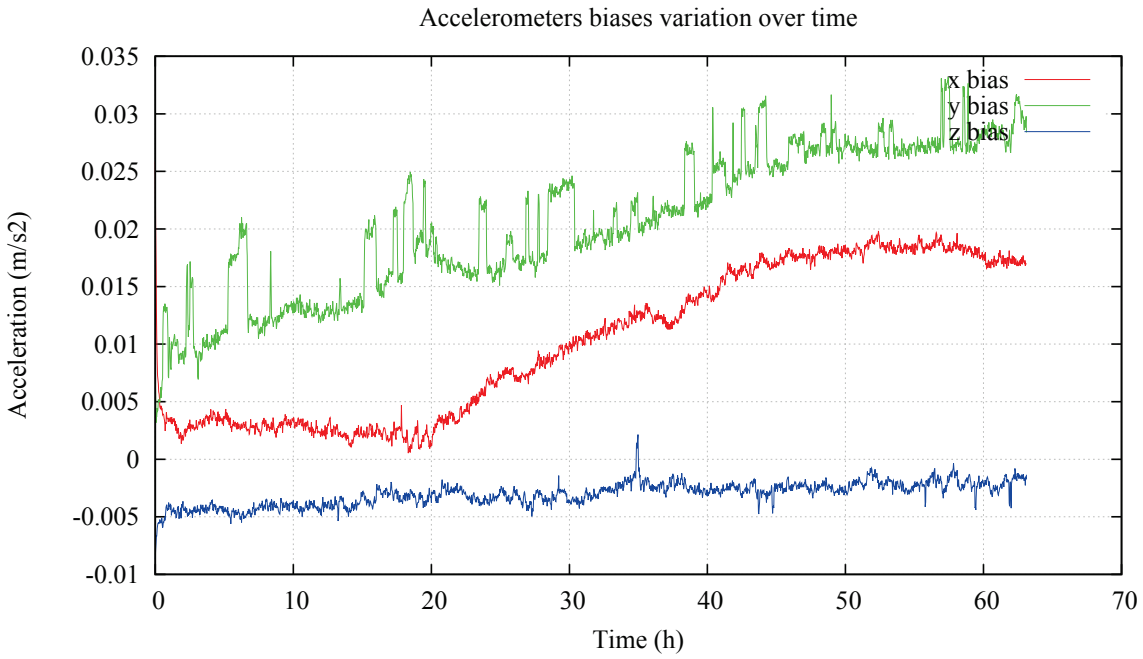


FIGURE 5.8 – Évolution des biais des accéléromètres pendant 63h, avec une période de moyennage de 100s. Ces valeurs sont elles-mêmes biaisées puisque l'orientation n'est pas précisément connue et la participation de la gravité n'a pas pu être éliminée correctement, mais les variations sont tout de même représentatives.

5.3.3 Évaluation des effets des bruits et biais

L'intérêt d'estimer finement le modèle de la centrale inertielle, notamment les biais des capteurs, est de limiter la dérive lors de la perte de la vision qui la stabilise. La dérive a deux types de causes : l'intégration du bruit (le *random walk*), et l'intégration des biais. Nous évaluons dans cette section ces causes pour les accéléromètres et gyromètres, ainsi que leur combinaison, afin de déterminer lesquels sont prédominants.

a Effets de l'intégration du biais de l'accéléromètre

Le biais de l'accéléromètre, constant, s'intègre en dt. La dérive en vitesse entre les temps 0 et T est donc :

$$\sigma_v(T) = \sigma_{a_b} \cdot T \tag{5.20}$$

Si on intègre à nouveau entre 0 et T on obtient, la dérive en vitesse étant toujours déterministe :

$$\sigma_p(T) = \int_0^T \sigma_v(t) \cdot \mathrm{d}t = \frac{1}{2} \cdot \sigma_{a_b} \cdot T^2 \tag{5.21}$$

b Effets de l'intégration du bruit de l'accéléromètre

Le bruit de l'accéléromètre, gaussien, s'intègre lui en $\sqrt{t}$. La dérive en vitesse entre les temps 0 et T est donc :

$$\sigma_v(T) = \sigma_{a_n} \cdot \sqrt{T} \tag{5.22}$$

La dérive en vitesse étant un mouvement brownien, son intégrale est la suivante :

$$\sigma_p(T) = \int_0^T \sigma_v(T) \cdot \mathrm{d}t = \frac{\sqrt{3}}{3} \cdot \sigma_{a_n} \cdot T^{3/2} \tag{5.23}$$

c Effets de l'intégration du biais du gyromètre

Le biais du gyromètre s'intègre donc en dt. La dérive en angle entre les temps 0 et T est donc :

$$\sigma_e(T) = \sigma_{w_b} \cdot T \tag{5.24}$$

Cependant on ne s'arrête pas là, car une erreur sur l'orientation implique une erreur dans la suppression du vecteur gravité. En effet, s'il est possible d'empêcher la dérive angulaire en observant la gravité, c'est en faisant l'hypothèse que l'accélération propre est en moyenne nulle sur le long terme, et que l'accélération moyenne mesurée sur le long terme est $\mathbf{g}$. Mais nous ne faisons pas cette hypothèse, puisqu'en fonctionnement normal c'est la caméra qui sert à séparer l'accélération propre du système et $\mathbf{g}$. Lors d'une perte de la vision, la dérive angulaire va donc créer des biais sur les accéléromètres par projection de $\mathbf{g}$:

$$\sigma_{a_b}(T) = g \cdot \sin(\sigma_e(T)) \approx g \cdot \sigma_e(T) = g \cdot \sigma_{w_b} \cdot T \tag{5.25}$$

L'approximation $\sin(\sigma_e(T)) \approx \sigma_e(T)$ reste valable tant que $\sigma_e(T)$ reste faible, ce qui sera le cas car quand la dérive angulaire sera devenue grande, une proportion significative de g aura été intégrée, et la dérive en position sera devenue énorme. Cette approximation est donc valide pour étudier le début de la dérive, mais pas quand le temps tend vers l'infini. On intègre ensuite deux fois ce biais sur l'accéléromètre :

$$\sigma_v(T) = \int_0^T \sigma_{a_b}(t) \cdot \mathrm{d}t \approx \frac{1}{2} \cdot g \cdot \sigma_{w_b} \cdot T^2 \tag{5.26}$$

$$\sigma_p(T) = \int_0^T \sigma_v(t) \cdot \mathrm{d}t \approx \frac{1}{6} \cdot g \cdot \sigma_{w_b} \cdot T^3 \tag{5.27}$$

d Effets de l'intégration du bruit du gyromètre

Le bruit du gyromètre s'intègre donc en $\sqrt{t}$. La dérive en angle est donc :

$$\sigma_e(T) = \sigma_{w_n} \cdot \sqrt{T} \tag{5.28}$$

De la même façon que pour le biais, cette dérive angulaire va créer des biais sur les accéléromètres par projection de $\mathbf{g}$:

$$\sigma_{a_b}(T) = g \cdot \sin(\sigma_e(T)) \approx g \cdot \sigma_e(T) = g \cdot \sigma_{w_n} \cdot \sqrt{T} \tag{5.29}$$

La dérive de ce biais étant un mouvement brownien, son intégrale vaut :

$$\sigma_v(T) = \int_0^T \sigma_{a_b}(t) \cdot \mathrm{d}t \approx \frac{\sqrt{3}}{3} \cdot g \cdot \sigma_{w_n} \cdot T^{3/2} \tag{5.30}$$

Si on intègre encore une fois comme un signal déterministe [1] :

$$\sigma_p(T) = \int_0^T \sigma_v(t) \cdot \mathrm{d}t \approx \frac{2\sqrt{3}}{15} \cdot g \cdot \sigma_{w_n} \cdot T^{5/2} \tag{5.31}$$

e Application numérique

Les résultats d'une application numérique sur une centrale XSens MTi sont présentés figure 5.4 . On constate plusieurs choses :

— Pour ce capteur, l'influence des biais devient à partir de quelques secondes supérieure à l'influence du bruit (elle l'est toujours à partir d'un certain temps pour n'importe quel capteur du fait de l'évolution proportionnelle au temps pour les biais et proportionnelle à la racine carrée du temps pour le bruit).

— La seconde chose remarquable est que pour ce capteur les biais et bruits sur les gyromètres sont les causes les plus importantes de dérive en position, ce très rapidement, et de plus en plus quand la durée d'intégration augmente (on vérifie que la dérive angulaire n'a pas dépassé les conditions de validité de l'approximation utilisée dans les calculs). Ceci peut sembler surprenant, mais nos constatations expérimentales (corriger précisément les biais des accéléromètres ne réduit pas significativement la dérive en l'absence de vision) confirment ce résultat théorique.

Ces résultats montrent l'importance d'estimer les biais des gyromètres, ainsi que l'intérêt moindre d'estimer les biais des accéléromètres qui ont un effet sur la dérive plus limité que le bruit des gyromètres, qui n'est lui pas corrigeable. Ceci a l'avantage d'atténuer largement l'inconvénient de notre système vu section 5.3.1 page 95, qui, s'il estime précisément les biais des gyromètres, a plus de difficultés à le faire pour les biais des accéléromètres.

1. Nous n'avons pas trouvé les éléments de théorie pour savoir comment intégrer ce type de signal, mais une simulation de type Monte Carlo montre que ce calcul est correct à quelques pourcents près ; la différence lors du calcul précédent avec un signal déterministe était déjà un simple facteur différant de moins de 15%, c'est-à-dire $\sqrt{3}/3$ au lieu de $2/3$).

(a) Valeurs typiques de bruit et biais pour une centrale XSens MTi, utilisés dans les calculs.

	Accéléromètre (m/s^2)	**Gyromètre (rad/s)**
Biais	0.02	0.01
Bruit	0.008	0.006

(b) Résultat. Les angles sont exprimés en degrés, et les positions en mètres.

	T=1s		**T=5s**		**T=10s**	
Cause	**Angle**	**Position**	**Angle**	**Position**	**Angle**	**Position**
Biais Accéléromètre	–	0.010	–	0.250	–	1.000
Bruit Accéléromètre	–	0.005	–	0.060	–	0.169
Biais Gyromètre	0.6	0.016	2.9	2.044	5.7	16.350
Bruit Gyromètre	0.3	0.014	0.8	0.760	1.1	4.299

TABLE 5.4 – Influence des bruits et biais sur la dérive des grandeurs intégrées. Les symboles – signifient que le bruit ou biais en question n'a pas d'influence sur la grandeur.

5.3.4 Évaluation sur une longue séquence

Les biais des accéléromètres n'ont cependant pas que pour conséquence de provoquer une dérive rapide en cas de perte de la vision. On a en effet vu que l'observation de la gravité et donc de la verticale via les accéléromètres permettait de stabiliser les angles d'attitude, et donc de limiter la dérive en z. Or l'observation de cette verticale est directement victime des biais des accéléromètres. On peut ainsi constater figure 5.9 page suivante, sur une séquence de SLAM effectuée avec ou sans estimation des biais, que si le filtre a du mal à estimer précisément ces biais (on voit par exemple au début que l'erreur en z croît rapidement), il parvient tout de même à en éliminer la majeure partie afin de retrouver une verticale peu biaisée. L'estimation des biais des accéléromètres garde donc tout son intérêt.

5.4 Considérations pratiques d'implémentation

5.4.1 Calibrage extrinsèque

Lorsque plusieurs capteurs sont utilisés dans un système de localisation, il est nécessaire de connaitre les positions relatives de ces capteurs, dont la procédure de détermination est appelée calibrage extrinsèque. Ce calibrage est important car l'orientation relative des capteurs définit comment les rotations autour des axes d'un capteur seront projetées sur l'autre, et la translation relative des capteurs a pour effet de convertir des rotations mesurées par un capteur en translations pour l'autre capteur.

Dans certains cas, lorsque l'on peut garantir un alignement manuel précis (multiples de 90 °), il est assez facile de garantir une précision sur les angles inférieure au degré, et de mesurer les translations avec une précision de l'ordre du millimètre. Il est nécessaire pour cela de localiser précisément les origines des capteurs. On a vu section 4.2.3.a page 57 comment procéder pour une caméra. Pour une centrale inertielle, c'est la position des accéléromètres qui détermine

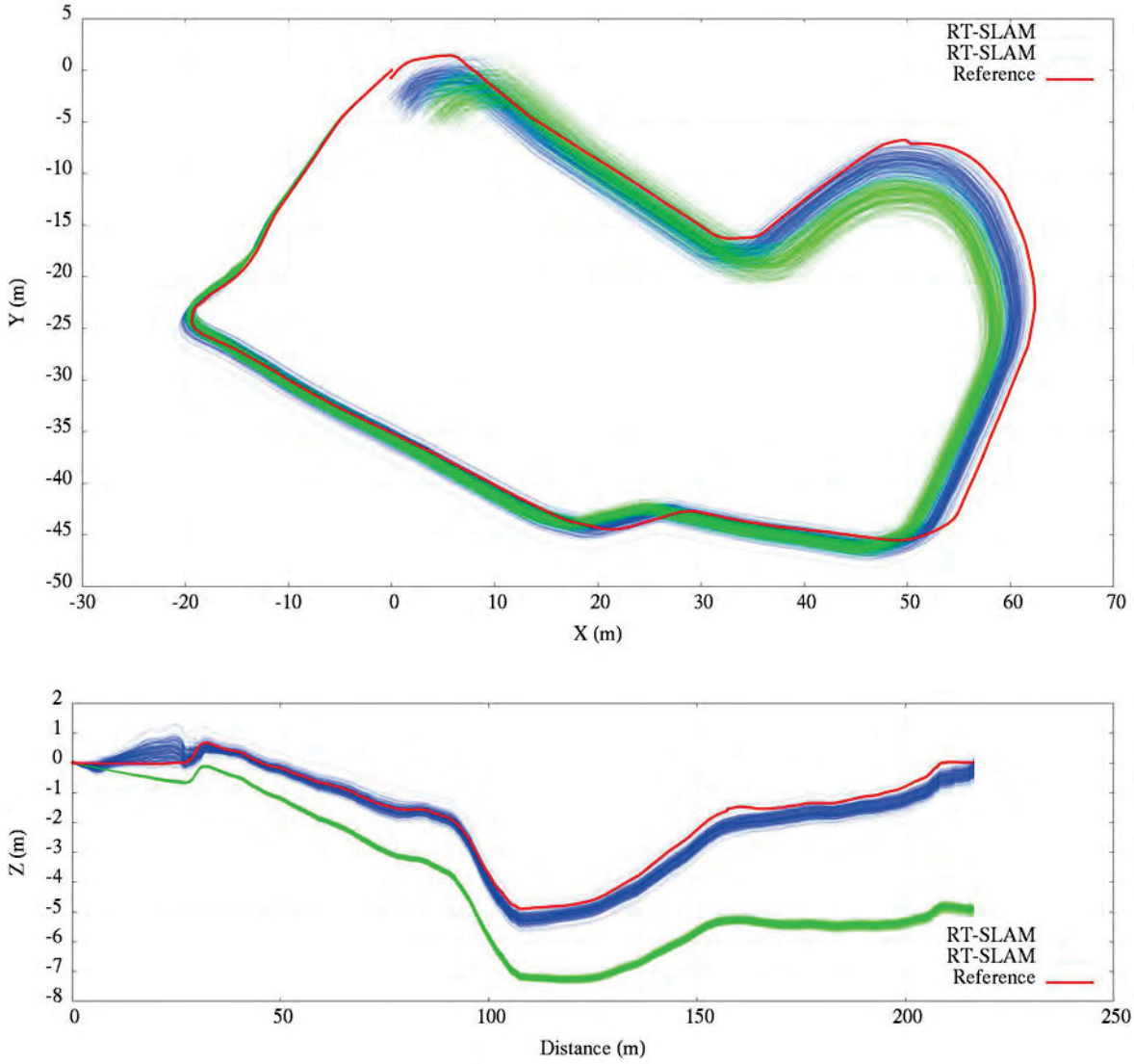


FIGURE 5.9 – Comparaison sur la séquence ESPERCE du slam inertiel avec estimation des biais des accéléromètres (en bleu) et sans (en vert).

l'origine du capteur, qui est normalement indiquée dans la documentation constructeur.

Cette précision sur les translations est en général suffisante, car il faut garder en tête que les accéléromètres physiques sont situés dans des composants de taille finie, qui est de l'ordre de quelques millimètres, et qui limite la précision de l'origine du capteur. Dans le cas de notre centrale XSens MTi, il s'agit de deux composants accéléromètres 2 axes, chacun de taille environ 5 mm, et distants d'autant l'un de l'autre.

Une précision de l'ordre du degré pour le positionnement angulaire n'est cependant pas nécessairement suffisant, provoquant des projections d'accélérations sur d'autres axes du même ordre de grandeur que les biais. De plus lorsque le positionnement relatif des capteurs est plus arbitraire, il peut être difficile de mesurer précisément les angles et les translations, et plus généralement l'axe de profondeur de la caméra n'est pas nécessairement parfaitement aligné avec le boîtier de la caméra [2]. On peut donc avoir intérêt à estimer dans le filtre le calibrage extrinsèque, ce que nous avons prévu dans RT-SLAM. Il suffit pour cela d'inclure la position de la caméra relative à la centrale inertielle, et de compléter les jacobiennes de la fonction d'observation pour ces nouveaux paramètres. L'application de la fonction d'observation reste la même, utilisant simplement la position relative stockée dans l'état au lieu de celle stockée statiquement dans l'objet capteur.

Comme toujours, estimer des paramètres nécessite de l'observabilité, qui sera d'autant meilleure que les autres paramètres du système sont connus précisément. Ainsi fournir une vérité terrain au filtre via des observations, par exemple de position, est un bon moyen d'améliorer l'observabilité des paramètres des modèles. Le calibrage extrinsèque ne changeant en général pas (sauf peut-être dans des contextes très particuliers de très long cours ou de conditions adverses), on peut donc réaliser un calibrage à l'aide du filtre et d'une vérité terrain, que l'on réinjecte ensuite en dur dans les fichiers de configuration. Par rapport à une estimation en ligne en opération, cela permet d'économiser du temps de calcul, de bénéficier dès le début de la séquence d'une valeur correcte, et de bénéficier d'une valeur plus précise car obtenue avec un système plus observable.

La figure 5.10 page 109 montre l'évolution du calibrage extrinsèque estimé sur les séquences MOCAP_FAST et MOCAP_LOOP, qui ont été réalisées avec le même système à 40 minutes d'intervalle (le calibrage est donc identique). Sur la séquence MOCAP_FAST figure 5.10(a), en intégrant la vérité terrain au filtre pour améliorer l'observabilité, on perçoit une certaine convergence, notamment angulaire, les translations semblant moins stables, et surtout prenant des valeurs non réalistes (on sait que l'on n'a pas fait une erreur de 2 cm dans les mesures à la main). Sur la séquence MOCAP_LOOP figure 5.10(b) en revanche, la convergence est nettement moins bonne, et les valeurs prises sont assez différentes des résultats sur la séquence MOCAP_FAST. Ceci se comprend facilement, la séquence MOCAP_FAST fournissant une bien meilleure observabilité des biais en stimulant tous les degrés de liberté, que la séquence MOCAP_LOOP qui est plus douce et monotone. En désactivant l'estimation des translations (deuxième colonne) afin de ne pas perturber l'estimation des angles, on obtient une convergence légèrement meilleure, et on a également des résultats bien plus répétables avec la séquence MOCAP_FAST ou en n'intégrant plus la vérité terrain (troisième colonne), même si des différences notables demeurent. Un autre moyen d'estimer hors ligne ce calibrage

2. Cela dépend du positionnement du capteur et de l'objectif

est de chercher à minimiser les corrections d'orientation effectuées par la vision sur la centrale inertielle, comme illustré figure 5.10(c). On obtient des corrections cap/assiette/gîte de $(-0.8, -0.4, -0.7)$ degrés, qui sont compatibles avec les valeurs de convergence du graphique du milieu de la figure 5.10(a).

La figure 5.11 page 110 montre cette fois le résultat de l'estimation en ligne du calibrage extrinsèque pour les séquences CAYLUS et ESPERCE. L'estimation des positions est toujours très chaotique, mais celle des angles converge vers des résultats exploitables, et qui peuvent à nouveau être confirmés par une minimisation de la correction. Sans utiliser la vérité terrain l'estimation est cependant toujours plus imprécise, voire ne converge pas du tout et est très inconsistant temporellement dans le cas de la séquence CAYLUS.

La figure 5.12 page 111 compare pour la séquence ESPERCE le résultat sur la trajectoire obtenue de cette correction, qui montre une légère amélioration des alignements en cap, mais qui montre cependant une influence assez limitée de ce calibrage.

En conclusion, l'estimation de la partie translation du calibrage extrinsèque entre la centrale inertielle et la caméra n'est pas suffisamment observable pour être estimée par le filtre, quelle que soit la trajectoire. L'estimation en ligne de la partie orientation sans intégration de la vérité terrain ne fonctionne en général pas non plus, mais il est possible l'utiliser pour réaliser un calibrage, soit en intégrant une vérité terrain, soit en prenant soin de stimuler suffisamment tous les degrés de liberté pour obtenir une observabilité suffisante. Il suffit ensuite d'introduire les paramètres obtenus dans le fichier de configuration.

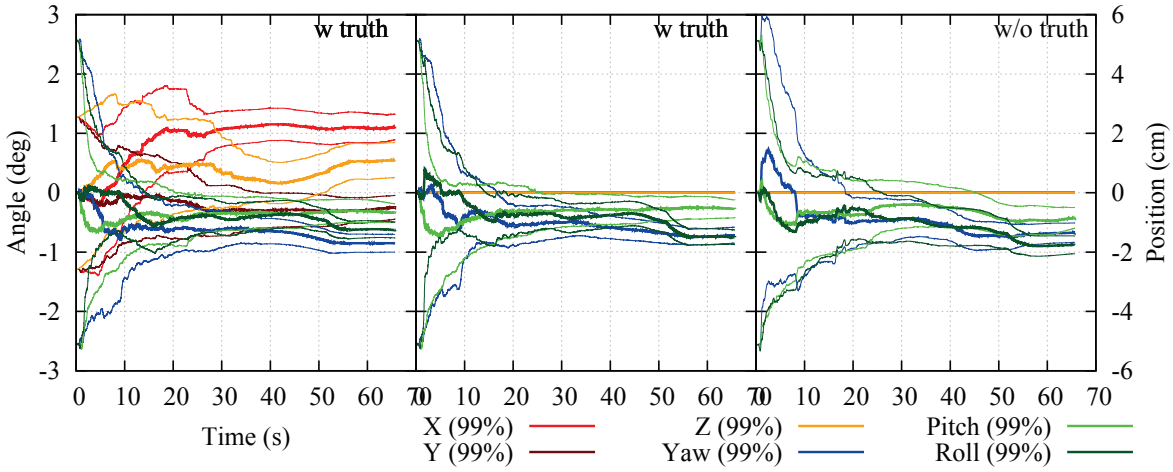
5.4.2 Horodatage des données

Avec un seul capteur comme présenté dans le chapitre 4 page 51, l'horodatage des données est important afin de calculer assez précisément l'intervalle de temps entre les images pour le modèle de mouvement.

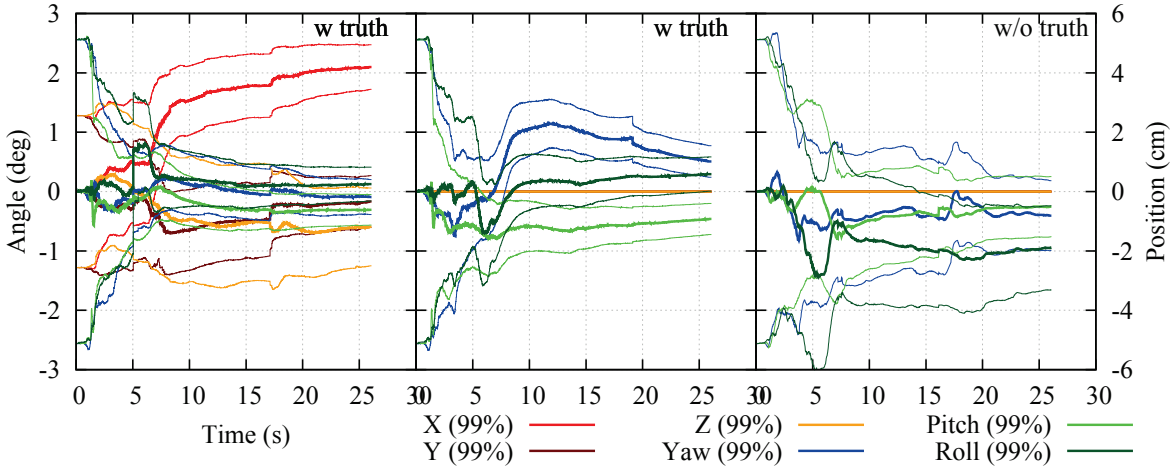
Mais le problème ne se présente pas directement en utilisant des caméras avec une connectivité firewire, car le pilote horodate les images à un niveau très bas et fournit donc un horodatage très stable et précis sur un système Linux classique (de l'ordre de quelques dizaines de microsecondes d'après nos constatations).

En revanche lorsque plusieurs capteurs sont utilisés, l'horodatage précis des données devient critique, et tous les capteurs ne fournissent pas un horodatage aussi précis que les caméras firewire sans effort, et ce n'est notamment pas le cas de la centrale inertielle XSens MTi que nous utilisons.

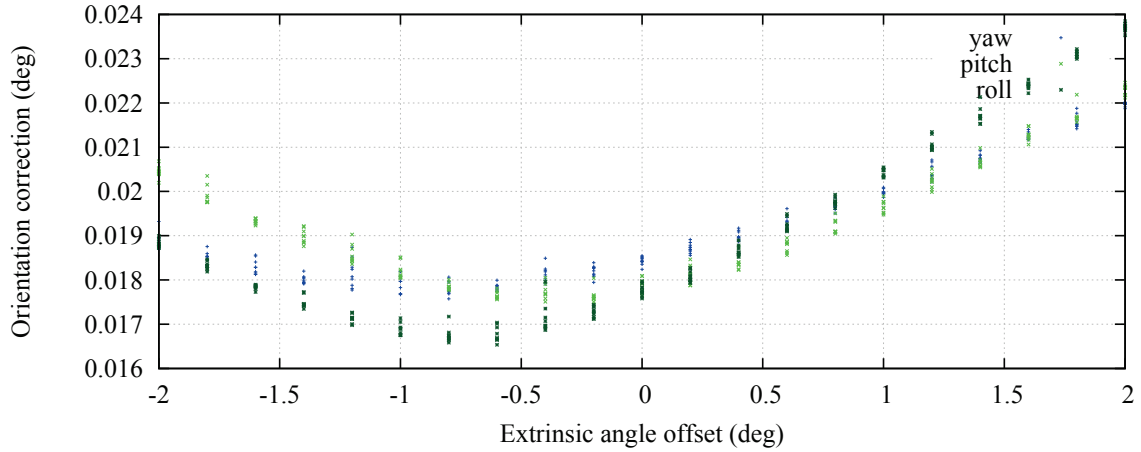
Le problème avec les capteurs utilisant une connectivité port série ou port USB et que ces ports ne gèrent pas l'horodatage des données au niveau du pilote. La seule façon d'horodater les données est donc dans le processus utilisateur, au moment de leur récupération. Mais dans un système multitâches, les processus sont ordonnancés selon la méthode de temps partagé, et un processus en particulier peut ne pas être exécuté pendant un temps arbitrairement long, selon la charge du système. Ainsi il n'existe aucune garantie sur le délai entre le moment où le capteur envoie ses données et où le processus utilisateur récupère les données et les horodate. Dans le pire des cas, avec un capteur rapide, on peut se retrouver avec plusieurs données



(a) Séquence MOCAP_FAST.

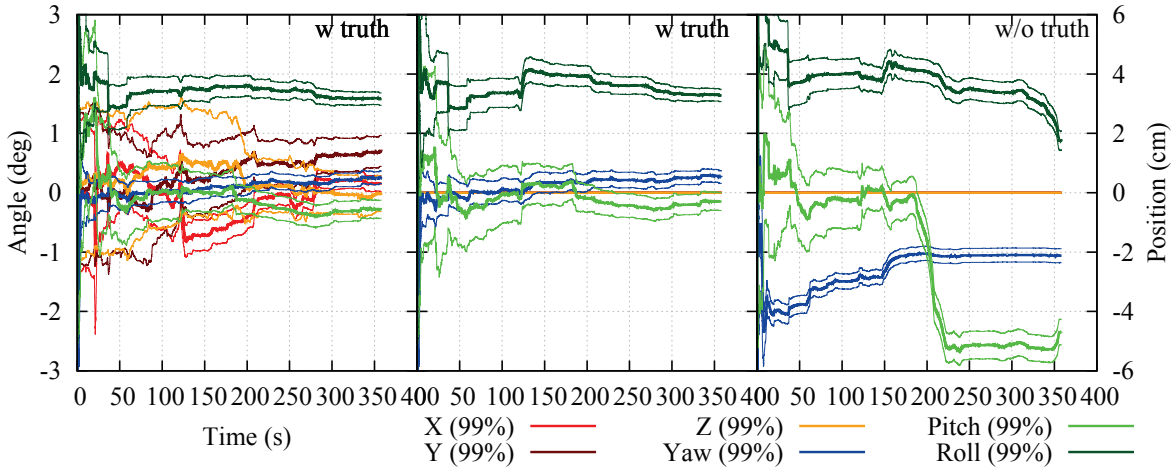


(b) Séquence MOCAP_LOOP.

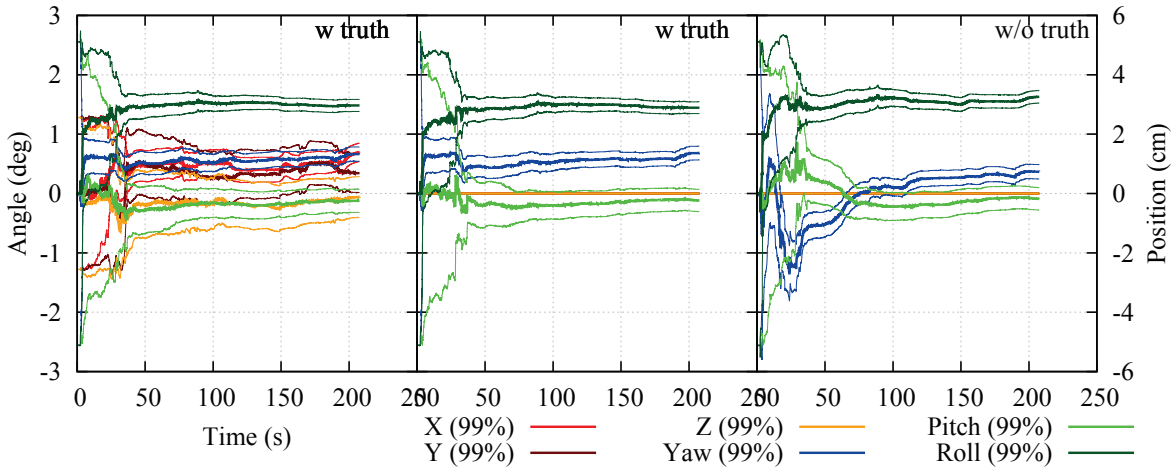


(c) Minimisation de la correction d'orientation sur la séquence MOCAP_FAST.

FIGURE 5.10 – Résultat de l'estimation du calibrage extrinsèque entre la caméra et l'IMU sur les différentes séquences (différence avec la valeur initiale).



(a) Séquence CAYLUS.



(b) Séquence ESPERCE.

FIGURE 5.11 – Résultat de l'estimation du calibrage extrinsèque entre la caméra et l'IMU sur les différentes séquences (différence avec la valeur initiale).

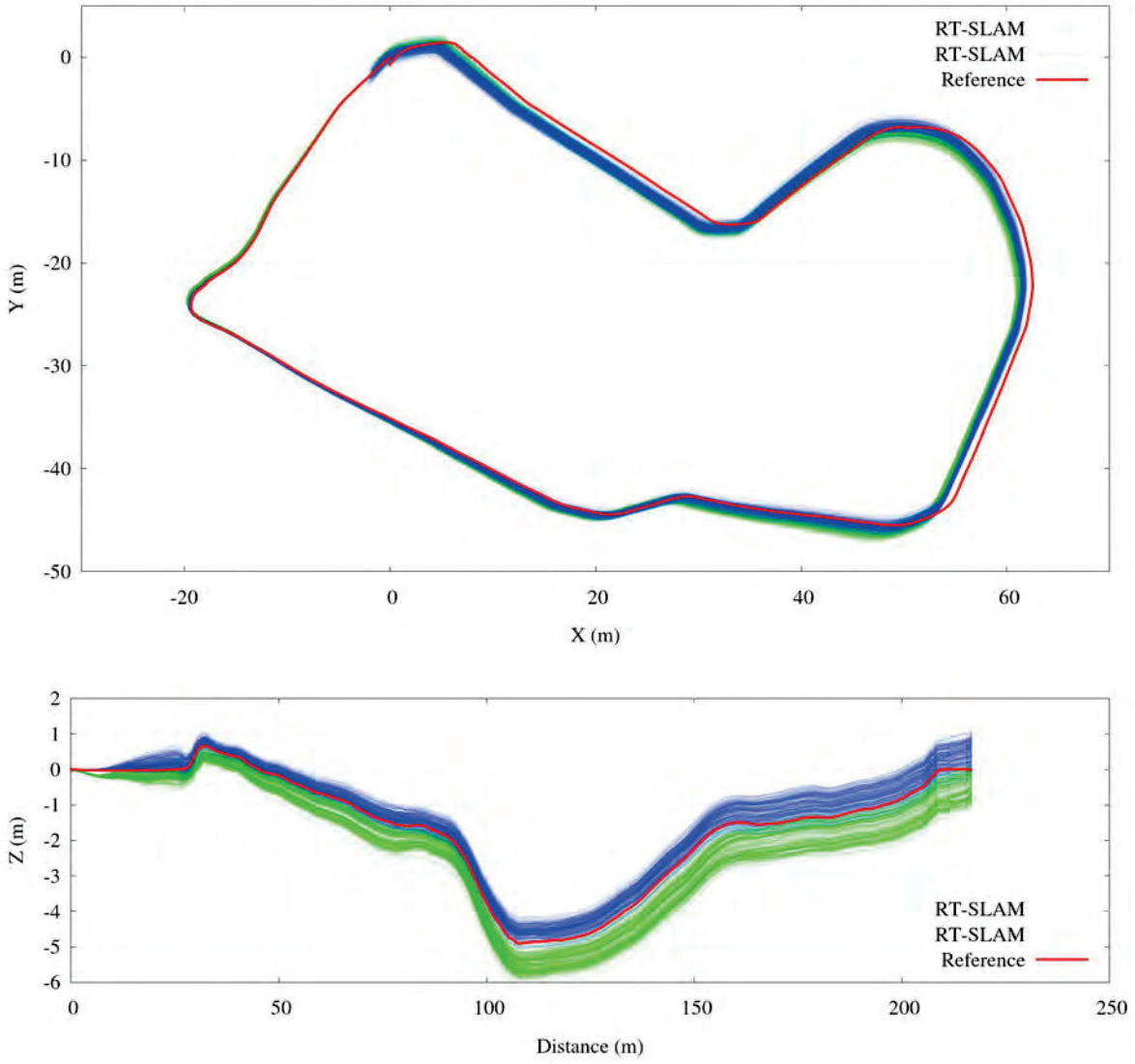


FIGURE 5.12 – Comparaison sur la séquence ESPERCE du slam visuel-inertiel avec calibrage extrinsèque issu de l'estimation (en bleu, correspondant aux angles $(-88.5, -0.1, -89.4)$) et sans (en vert, correspondant aux angles $(-90, 0, -90)$). L'odométrie, décrite au chapitre suivant, a également été inclue afin de limiter la dispersion et permettre de mieux comparer. On constate que l'alignement dans la dernière ligne droite est légèrement meilleur, tout comme l'erreur globale (on a une erreur RMS en position de 1.1 m au lieu de 1.4 m, et en orientation de 0.5 °au lieu de 1.1 °).

reçues en même temps, en retard, et donc datées de la même heure.

Nous avons étudié plusieurs solutions possibles à ce problème.

a Système d'exploitation temps réel

La première solution envisageable est d'utiliser un système d'exploitation temps réel dur, c'est-à-dire qui permet d'offrir des garanties (bornes) sur les délais.

Nous avons testé le système Xenomai, qui consiste en un noyau remplaçant le noyau Linux standard, et est donc léger à mettre en œuvre puisqu'il permet de ne pas changer le reste des applications utilisées (environnement de bureau, logiciels, etc).

Cependant quand il s'agit d'entrées-sorties, il n'est pas suffisant d'avoir un noyau temps réel, il est également nécessaire d'utiliser un pilote temps réel. Un tel pilote est disponible pour le port série RS232, et les résultats sont présentés figure 5.13 pour un système peu chargé et figure 5.14 pour un système très chargé. L'axe des ordonnées correspond à l'erreur constatée entre l'horodatage des données, et leur date théorique basée sur la fréquence nominale du capteur (le capteur ayant lui un fonctionnement temps réel il n'y a pas lieu de le remettre en cause, à part une légère différence d'échelle des horloges que l'on constate avec la pente globale des courbes).

Le premier graphique correspond à un noyau classique configuré de façon standard, le second graphique à un noyau classique configuré pour optimiser la latence (ordonnanceur temps réel *Round Robin*) avec processus haute priorité, et le troisième graphique à un noyau Xenomai avec pilote temps réel. On constate que la configuration d'un noyau classique apporte très peu d'amélioration. Avec un noyau classique, on voit les effets de l'ordonnancement par des paliers, et dès que le système devient chargé la dispersion est très forte (plusieurs millisecondes). Avec un noyau et pilote temps réel, l'horodatage est considérablement plus précis, la plupart du temps de l'ordre de quelques microsecondes, et exceptionnellement de quelques dixièmes de millisecondes. Le pilote temps réel fournit également un horodatage du contrôleur d'interruptions (IRQ, *Interrupt ReQuest*), qui est encore plus précis et ne souffre d'aucune fluctuation, mais qui n'est pas contrôlé par la correction d'horloge réseau (NTP, *Network Time Protocol*), et n'est donc pas exprimé dans le même référentiel (cette correction NTP est pourtant indispensable quand plusieurs machines sont présentes sur le robot, ou quand plusieurs robots coopèrent).

Une remarque importante ici est qu'il est indispensable de réaliser l'horodatage le plus tôt possible après la réception des données, y compris sur un système non temps réel, car plus il y a de code entre la réception des données et l'horodatage (correspondant en général au décodage du message), plus il y a de risque que le processus soit interrompu par l'ordonnançeur entre la réception et l'horodatage. La figure 5.15 page 114 montre que en fonctionnement normal si le retard reste en général inférieur à une période lorsque l'horodatage est pris immédiatement à la réception des données (comme on avait déjà pu le constater figures 5.13 et 5.14), il est régulièrement égal à plusieurs périodes quand on attend la fin du décodage du message pour horodater (ce qui signifie que les données n'ont pas été traitées au fur et à mesure mais toutes en même temps).

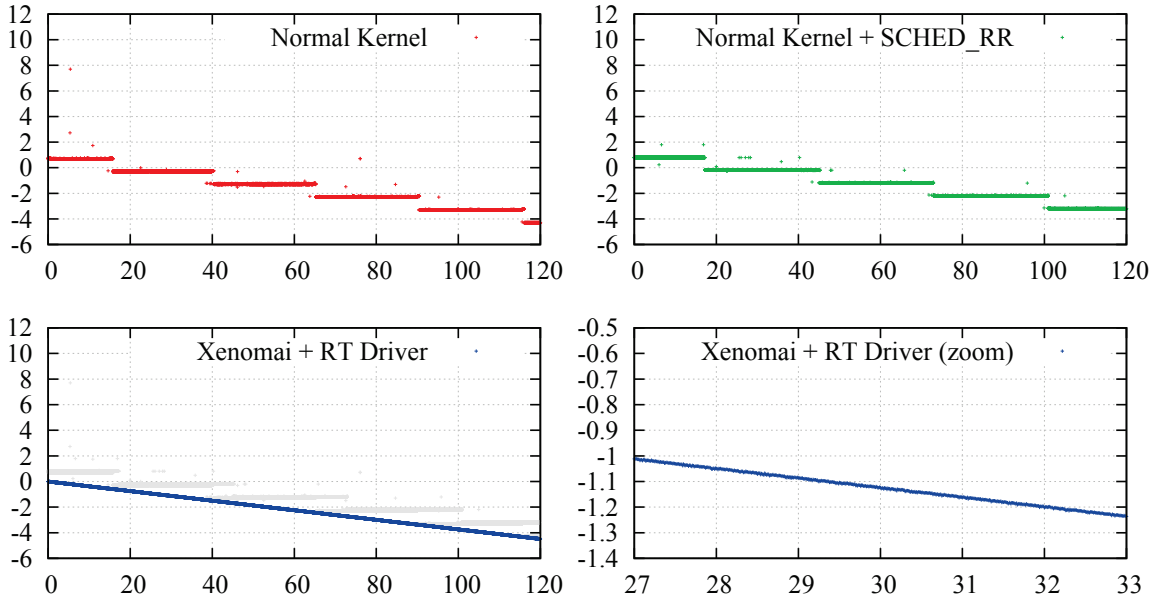


FIGURE 5.13 – Erreur d'horodatage (ms) en fonction du temps (s), pour un système peu chargé, avec différents noyaux.

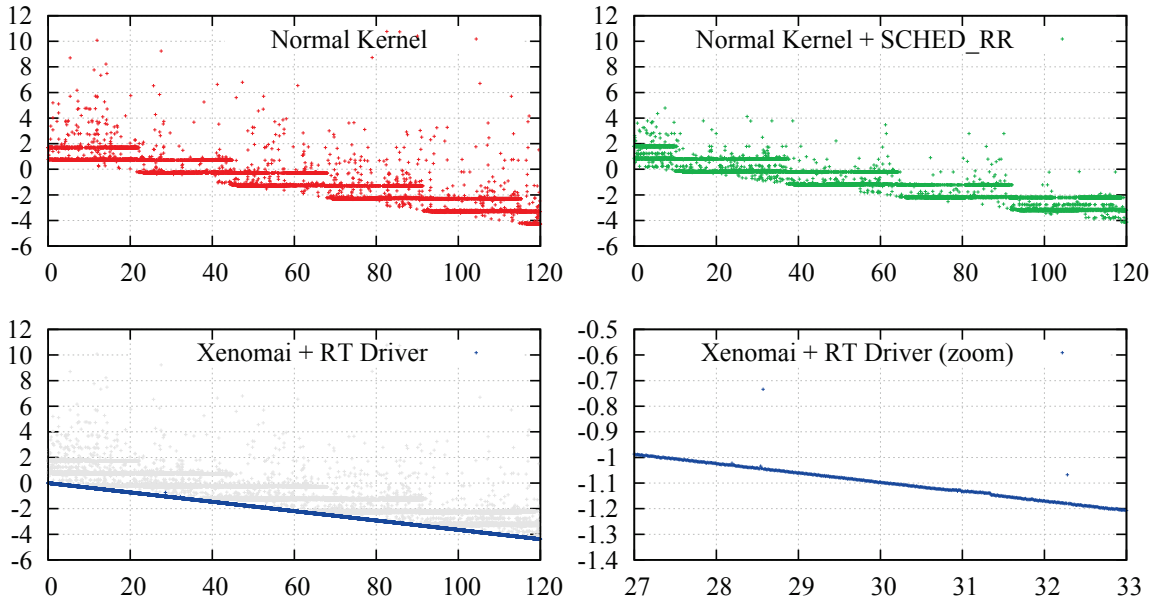


FIGURE 5.14 – Erreur d'horodatage (ms) en fonction du temps (s), pour un système très chargé, avec différents noyaux.

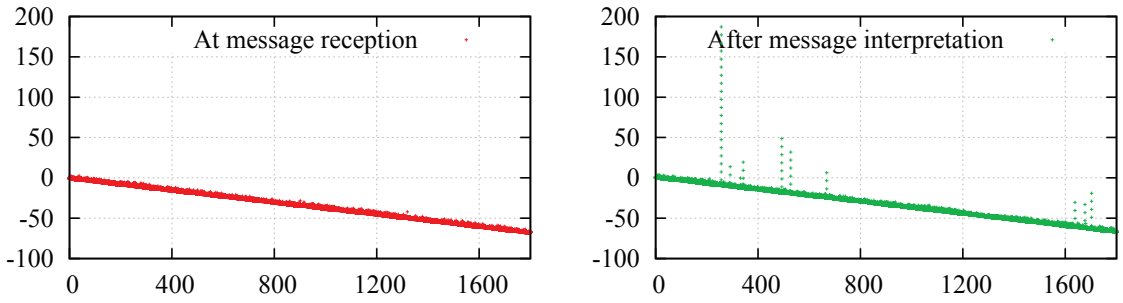


FIGURE 5.15 – Erreur d'horodatage (ms) en fonction du temps (s), pour un horodatage à réception des données ou après décodage des données, sur un système très chargé.

Ces résultats sont donc extrêmement positifs, mais il reste les bémols suivants :

— Seul le pilote RS232 est disponible, maintenu et fiable. Il existe un pilote USB1.1 mais qui n'est plus maintenu et bogué, et ne supporte pas les périphériques USB2.0. Cela limite donc fortement l'usage des convertisseurs RS232-USB (nous ne les avons cependant pas testés) ce qui peut poser problème lorsque le nombre de ports série natifs est insuffisant, ainsi que des autres capteurs interfacés sur USB, ou encore sur Ethernet.

— Cela nécessite de modifier toutes les bibliothèques pilotant des capteurs pour les rendre temps réel.

— Si l'on peut envisager d'installer un système temps réel sur un robot, cela alourdit considérablement les possibilités de test du logiciel sur une machine personnelle.

C'est pourquoi nous avons également exploré d'autres solutions moins lourdes à mettre en œuvre.

b Synchronisation matérielle

Le besoin n'est en fait pas nécessairement dans un horodatage absolu précis, mais dans un horodatage relatif précis entre les différents capteurs et entre les données consécutives.

Ainsi dans certains cas il peut être suffisant de synchroniser matériellement les différents capteurs, soit avec une horloge externe, soit avec l'un des capteurs faisant office d'horloge. On saura alors par exemple qu'une donnée inertielle sur deux est associée à une image.

En l'occurrence notre IMU XSens MTi possède une sortie de synchronisation configurable logiciellement pour émettre une impulsion de la durée souhaitée, et à la fréquence souhaitée, mais synchronisée avec ses propres mesures, et les caméras possèdent en général une entrée de synchronisation. Il est également possible par ce moyen de contrôler l'exposition avec la durée de l'impulsion comme expliqué section 4.2.3.d page 60.

Cette solution n'est cependant pas parfaite :

— Elle demeure invasive et lourde à mettre en œuvre.

— Il reste une ambiguïté dans l'association des données lorsque les capteurs ne fonctionnent pas à la même fréquence. Il reste donc un traitement logiciel à réaliser pour lever cette

ambiguïté, ainsi que pour calculer la durée entre les différentes données à partir des fréquences théoriques lorsqu'aucun capteur ne fournit d'horodatage précis.

— Il y a une contrainte forte sur les fréquences des capteurs, qui doivent être des multiples les unes des autres.

— Ce n'est pas facilement généralisable à beaucoup de capteurs, car tous doivent être munis d'une entrée de synchronisation, ce qui n'est pas toujours le cas.

Une solution un peu plus souple pourrait aussi être de faire générer les signaux de synchronisation par la machine hôte, mais il serait dans ce cas fortement préférable d'utiliser sur cette machine un système temps réel afin de pouvoir générer ces signaux de façon suffisamment régulière.

c Gestionnaire d'horodatage

Génèse Les capteurs sont toujours constitués de systèmes temps réels, et peuvent quasiment toujours être configurés pour réaliser leurs mesures et envoyer les données correspondantes de façon périodique. Une solution peut donc être d'horodater les données à la réception sur un système non temps réel (on obtient des horodatages plus ou moins fluctuants, comme on l'a constaté dans le premier graphique des figures 5.13 page 113 et 5.14 page 113), et de filtrer ces horodatages par rapport à la contrainte que l'écart entre les données consécutives est connu avec une très bonne précision.

C'est l'idée que nous avons retenue dans un premier temps pour la centrale XSens MTi, et implémenté dans sa bibliothèque pilote. La fréquence réelle d'émission des données est estimée dynamiquement par la moyenne des dernières données reçues (on a vu figures 5.13 page 113 et 5.14 page 113 qu'elle était un peu différente de la théorie à cause d'une légère différence d'échelle entre l'horloge du capteur et celle de la machine hôte, et cette différence d'échelle peut varier lentement au cours du temps, à cause de la température ou des corrections de type NTP sur la machine hôte), et on impose que les horodatages respectent cette fréquence.

Cette méthode se retrouve cependant mise en défaut lorsqu'un problème de communication se produit et qu'une donnée n'est pas reçue, puisque toutes les données suivantes seront horodatées avec une période d'avance. Cependant beaucoup de capteurs, et c'est le cas de notre centrale inertielle, utilisent un compteur de données et envoient le numéro avec les données, ce qui permet de détecter les données manquantes et de ne pas se décaler dans l'horodatage.

A côté de cela, certains capteurs fournissent un horodatage réalisé en interne, de façon très fiable donc, mais qui pose toujours un problème de synchronisation entre l'horloge interne du capteur et l'horloge de la machine hôte : décalage et facteur d'échelle. Il est alors possible d'une façon identique d'estimer ces paramètres grâce aux dates de réception sur la machine hôte. On se rend alors compte que dans les deux cas, que le filtre fournisse un comptage des données, ou un horodatage interne, le problème est exactement le même : un compteur peut être considéré comme une horloge, seulement liée à celle de la machine hôte par un décalage et un facteur d'échelle.

Fonctionnement Le modèle temporel d'un capteur est constitué des grandeurs suivantes :

- la *correction d'échelle* γ de l'horloge : l'horloge de l'hôte avance γ fois plus rapidement que l'horloge du capteur.
- le *décalage* τ_s et τ_h entre les horloges : la date τ_s pour le capteur correspond à la date τ_h pour l'hôte. Il suffirait normalement de définir le décalage à l'origine, à savoir τ_h pour $\tau_s = 0$, mais puisque le facteur d'échelle γ peut changer il est nécessaire de définir le décalage au moment où la valeur courante de γ est valide.
- la *latence* λ : une donnée mesurée à la date t_s par le capteur est reçue par la machine hôte λ plus tard (définie en unités de temps de l'horloge de l'hôte). Elle est due au traitement interne des données par le capteur et à la communication des données.
- le *retard* δ entre la réception de la donnée par l'hôte et son horodatage, qui fluctue donc aléatoirement.

La relation entre la date du capteur t_s à laquelle est effectuée la mesure, et la date de la machine hôte t_h correspondant au même instant est alors :

$$t_h - \tau_h = (t_s - \tau_s) * \gamma \tag{5.32}$$

Et on cherche à identifier t_h, n'ayant à notre disposition que les horodatages h_s sur le capteur et h_h sur l'hôte :

$$h_s = t_s \tag{5.33}$$
$$h_h = t_h + \lambda + \delta \tag{5.34}$$

L'algorithme, donc le cœur est présenté de façon plus formelle table 5.5 page 118, est basé sur les principes suivants :

- La latence λ est calculée statiquement avant le démarrage. Étant calculée à partir de paramètres bas niveau, comme la durée de traitement interne des données parfois indiquée dans la documentation constructeur, et la durée de communication dépendant de la taille des messages et de la vitesse de communication, elle est idéalement calculée par la bibliothèque pilote et déjà retranchée de l'horodatage à la réception h_h, et la valeur résiduelle à utiliser ici est ensuite nulle. On peut noter que si toutes les informations ne sont pas disponibles pour effectuer ce calcul théoriquement, il est également possible de le déterminer expérimentalement en étudiant la synchronisation avec d'autres capteurs (par exemple il est facile de voir dans l'image si la prédiction réalisée avec la centrale inertielle a de l'avance ou du retard). Plus de détail est donné à ce sujet section 5.4.3 .
- Le retard δ est aléatoire et dynamiquement estimé à chaque mesure pour être supprimé, et on suppose qu'il est régulièrement négligeable (on essaie de « câler » notre correction sur ces données).
- La correction d'échelle γ est initialisée à sa valeur théorique. L'horloge hôte fournissant des horodatages en secondes, si le capteur fournit un compteur et fonctionne à la fréquence f, alors $\gamma = 1/f$. Si le capteur fournit un horodatage par exemple en millisecondes, alors $\gamma = 1000$.

— γ varie lentement, et est dynamiquement mise à jour par moyennage. On doit pour réaliser ce moyennage utiliser des mesures avec un δ le plus faible possible. On va donc découper les données en secteurs de durée p, en maintenant itérativement pour chaque secteur k la donnée m_k qui a le retard δ le plus faible, et utiliser ces données pour calculer γ. En pratique la période p sera petite au départ pour rapidement corriger des erreurs grossières d'initialisation de γ, puis augmentera pour améliorer la précision d'estimation de γ, les anciens secteurs étant fusionnés pour que les trois derniers secteurs couvrent la plus grande période de temps possible dans la limite de la taille nominale, qui ne doit pas être trop grande pour permettre à γ de suivre les variations du capteur (quelques dizaines de secondes en général).

— Le décalage d'horloge τ_h et τ_s est dynamiquement estimé. Il est corrigé lorsque l'on reçoit une donnée plus tôt que prévu (ce qui signifie qu'il était surestimé auparavant), et réinitialisé lorsqu'on corrige l'échelle γ.

Des exemples de résultats sont présentés figure 5.16 page 119. Une fois le régime permanent atteint, on a des erreurs maximales de l'ordre du dixième de milliseconde, dues au suivi de la dérive du facteur d'échelle des horloges (deuxième graphique). Si on initialise le facteur d'échelle plus loin de la réalité, le filtrage reste stable. On constate tout de même qu'il vaut mieux le surestimer un peu (troisième graphique) auquel cas l'erreur sera vite compensée par les données au retard faible, que le sous-estimer (quatrième graphique) auquel cas rien ne pourra corriger l'erreur avant la première mise à jour de la fréquence. On obtient dans tous les cas une erreur maximale de l'ordre de la milliseconde pendant le régime transitoire. Idéalement il faut donc laisser tourner quelques secondes l'algorithme sur un système peu chargé pour lui permettre d'évaluer la fluctuation et de fixer le décalage avec la donnée la moins retardée. L'estimation précise de la correction d'échelle est plus longue, mais le fonctionnement reste acceptable avant sa stabilisation, et si le capteur tourne en permanence sur le système cela devient transparent.

Les performances sont donc un peu moins bonnes qu'avec un système temps réel ou une synchronisation matérielle, mais elles se sont montrées tout à fait suffisantes pour des applications comme les nôtres. Il y a également paramétrage supplémentaire, mais la mise en œuvre est tout de même incomparablement plus souple et simple.

Notons qu'une approche similaire a été développée auparavant et publiée après nos travaux dans [Olson, 2010].

5.4.3 Estimation du décalage temporel

Comme on vient de le voir, que l'on utilise un système temps réel ou un filtrage des horodatages, on obtient toujours une date de réception sur le système hôte, et on remonte à la date à laquelle la donnée a été acquise par soustraction d'un décalage λ dû aux temps de traitement interne et de communication. On peut envisager plusieurs solutions pour le calculer.

A l'initialisation :

- λ est initialisé à la valeur calculée a priori.
- γ est initialisé à l'échelle théorique du capteur.
- $p_{\min}$, $p_{\max}$ et $p = p_{\min}$ définissant la période des secteurs sont initialisés.
- k le numéro de secteur est initialisé à 0.

Lors de la réception de la première donnée, d'horodatages $h_{s,0}$ et $h_{h,0}$:

- $\tau_s = h_{s,0}$ (eq. 5.33)
- $\tau_h = h_{h,0} - \lambda$ (eq. 5.34 avec $\delta = 0$)

Pour chaque nouvelle donnée qui arrive, d'horodatages $h_{s,n}$ et $h_{h,n}$:

- $t_{h,n} = \tau_h + (h_{s,n} - \tau_s) * \gamma$ (eq. 5.32)
- $\delta = h_{h,n} - (t_{h,n} + \lambda)$ (eq. 5.34)
- $m_k = n$ si δ est le plus faible de ce secteur k.
- Si $\delta < 0$, alors on met à jour le décalage et on corrige la date obtenue :
  - $\tau_h = \tau_h + \delta$
  - $t_{h,n} = t_{h,n} + \delta$
- Si on termine le secteur k de taille p, on met à jour la fréquence et le décalage :
  - $\gamma = \dfrac{h_{h,m_j} - h_{h,m_i}}{h_{s,m_j} - h_{s,m_i}}$ (eq. 5.32)
  - $\tau_s = h_{s,m_j}$ (eq. 5.33)
  - $\tau_h = h_{h,m_j} - \lambda$ (eq. 5.34 avec $\delta = 0$)
  - On calcule les trois jeux γ, τ_s, τ_h pour les paires (m_i, m_j) obtenables par combinaison des trois derniers secteurs : $(i,j) \in \{(k{-}2,k{-}1),(k{-}2,k),(k{-}1,k)\}$; on élimine ceux qui sont incompatibles avec le secteur non utilisé (nouveau δ négatif) ; on conserve celui qui a le γ le plus élevé.
  - $p = p \cdot \varphi$, et si $p > p_{\max}$ alors $p = p_{\max}$, sinon on fusionne les secteurs $k{-}2$ et $k{-}1$ en gardant la donnée m_{k-1} ou m_{k-2} de nouveau δ le plus faible.
  - $k = k + 1$

TABLE 5.5 – Algorithme du gestionnaire d'horodatage.

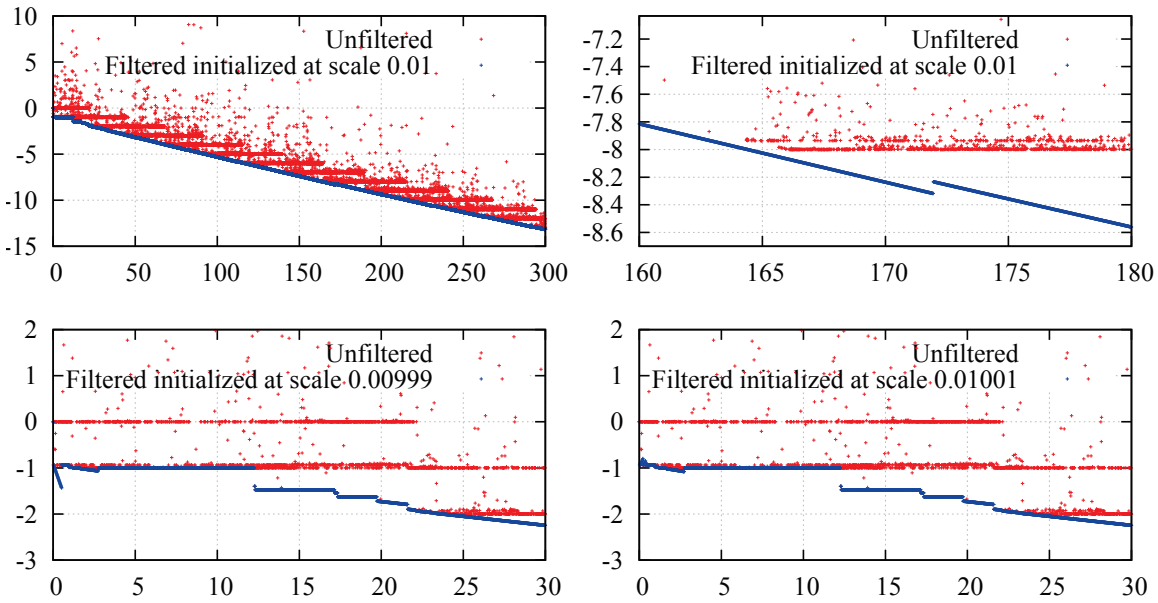


FIGURE 5.16 – Comparaison de l'erreur d'horodatage brute et filtrée (ms) en fonction du temps (s), sur un système très chargé, avec une taille de secteurs finale de 30 s.

a Calcul théorique

Ce décalage peut être calculé théoriquement d'après les spécifications fournies dans la documentation technique du constructeur : durée de traitement interne des données (dépendant parfois du mode dans lequel se trouve le capteur), durée de communication dépendant de la taille des messages et de la vitesse de communication.

Si parfois toutes les informations nécessaires sont effectivement fournies dans la documentation constructeur (par exemple pour l'IMU XSens MTi), c'est malheureusement rarement le cas. De plus il demeure toujours une certaine incertitude au moment de la réception sur la machine hôte, et la possibilité de faire une erreur, d'oublier un paramètre, ou que la documentation soit erronée.

b Calibrage hors ligne

Il est donc également possible et souvent préférable de le déterminer expérimentalement, en étudiant sa synchronisation avec d'autres capteurs. Le principe consiste à ajuster expérimentalement ce décalage sur un jeu de données hors ligne pour minimiser la moyenne de la norme des corrections ou des innovations.

Par exemple la figure 5.17 page 121 montre à quoi ressemblent ces valeurs en fonction du décalage des horodatages de l'IMU, mais le procédé fonctionne de même avec d'autres capteurs utilisés en observation (étudiés au chapitre 6 page 135), comme le montrent les figures 5.18 page 121 et 5.19 page 121. Les latences importantes constatées sur ces capteurs sont dues au fait que l'on utilise le middleware Genom du LAAS pour accéder à ces données, et l'implé-

mentation des modules de ces capteurs ne gère pas correctement la latence et l'horodatage. Dans le cas de l'odométrie le minimum est peu net sur le graphiques à cause d'une faible force d'influence du capteur sur la localisation (il est simplement censé stabiliser en moyenne sur le long terme), et car les corrections sont diluées avec celles de la vision.

Même si cette solution est suffisante pour la plupart des applications, elle n'est toujours pas parfaite pour une utilisation intensive en ligne, car il n'y a pas de garantie que ce décalage ne change pas selon les conditions (par exemple temps d'exposition des caméras).

c Estimation dynamique

Idéalement, on voudrait donc comme l'on fait pour tous les calibrages, estimer ce biais dans le filtre. Malheureusement les équations du filtre de Kalman ne permettent pas de modéliser cette erreur d'horodatage.

On pourrait cependant estimer dynamiquement ce décalage par un processus extérieur au filtre, faisant le lien entre les variations des mesures de la centrale inertielle, et l'amplitude des corrections dues à chaque observation des données d'un capteur.

En effet si ce décalage est nul, les prédictions de la centrale inertielle ne seront pas biaisées, et les corrections de moyenne nulle, quelles que soient les variations des mesures de la centrale inertielle (augmentation ou diminution). En revanche si la centrale inertielle a de l'avance, on constatera lorsque l'accélération augmente des corrections négatives, et lorsqu'elle diminue des corrections positives. Si au contraire la centrale inertielle a du retard, on constatera inversement lorsque l'accélération augmente des corrections positives, et lorsqu'elle diminue des corrections positives.

Notons que le même raisonnement s'applique pour les vitesses angulaires de la centrale inertielle et la correction de l'orientation. L'amplitude de ce lien permet ensuite de remonter à une estimation du décalage, que l'on peut appliquer, avant de recommencer pour affiner ou ajuster.

Formulation mathématique Plus concrètement, si l'on considère les grandeurs suivantes :

- Δ le décalage temporel recherché entre l'IMU et le capteur en observation considéré.
- δ la période des mesures du capteur en observation.
- δ_m la durée la plus proche de δ correspondant à un nombre entier de périodes des mesures de l'IMU.
- $a(t)$ les mesures d'accélération fournies sur un axe par l'IMU.
- $p(t)$ l'estimée de position sur le même axe (on suppose ici pour simplifier que le repère estimé dans le filtre est celui de l'IMU, ce qui est notre cas en pratique).

Alors à l'instant t, l'erreur sur l'accélération due au décalage Δ est approximativement :

$$a'(t) \cdot \Delta \tag{5.35}$$

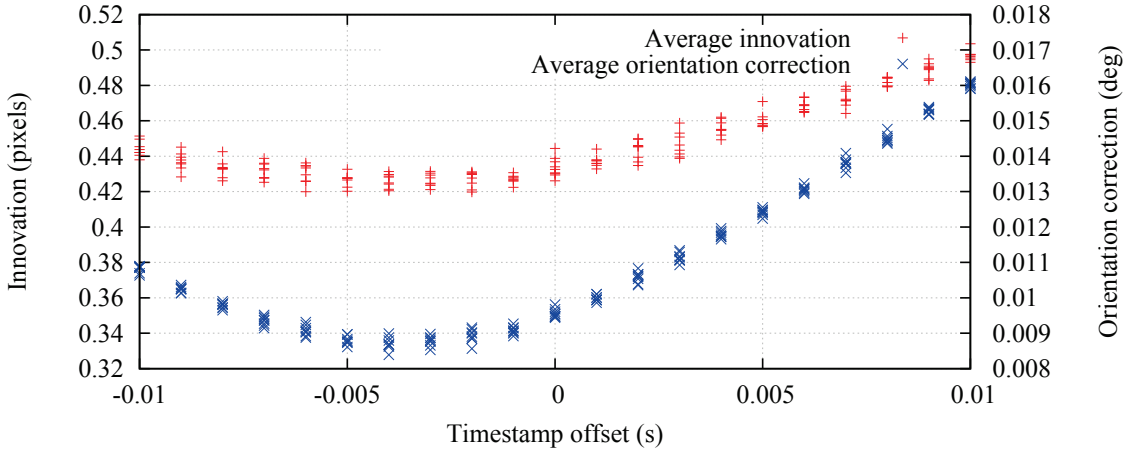


FIGURE 5.17 – Exemple de minimisation du décalage des horodatages entre l'IMU et la caméra. On retient un décalage de -4 ms.

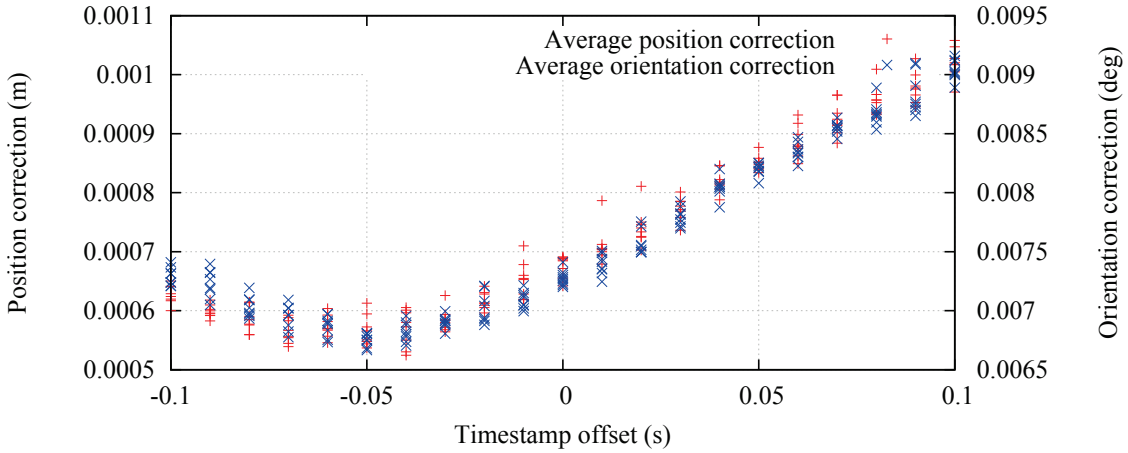


FIGURE 5.18 – Exemple de minimisation du décalage des horodatages entre le GPS et le système caméra-IMU déjà synchronisé. On retient un décalage de -50 ms.

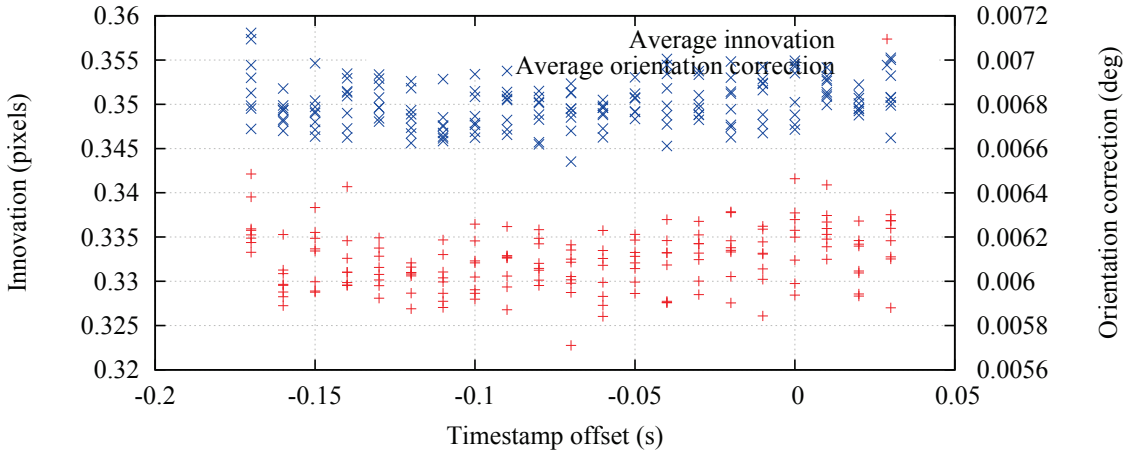


FIGURE 5.19 – Exemple de minimisation du décalage des horodatages entre l'odométrie et le système IMU-caméra déjà synchronisé. On retient un décalage de -70 ms.

En dehors de tout bruit, cette erreur intégrée deux fois sur la période δ est égale à la correction de la position par le capteur :

$$a'(t) \cdot \Delta \cdot \delta^2 = p(t) - p^-(t) \tag{5.36}$$

On peut exprimer $a'(t)$ en fonction des mesures de l'IMU pendant δ :

$$a'(t) \approx \frac{\tilde{a}(t) - \tilde{a}(t-\delta_m)}{\delta_m} \tag{5.37}$$

On obtient alors une expression du décalage Δ en fonction des mesures de l'IMU et de la correction de position avec le capteur en observation :

$$\tilde{\Delta} = \frac{(p(t) - p^-(t)) \cdot \delta_m}{(\tilde{a}(t) - \tilde{a}(t-\delta_m)) \cdot \delta^2} \tag{5.38}$$

On trouve de la même façon pour les angles d'euler $e(t)$ et les vitesses angulaires $w(t)$ de l'IMU :

$$\tilde{\Delta} = \frac{(e(t) - e^-(t)) \cdot \delta_m}{(\tilde{w}(t) - \tilde{w}(t-\delta_m)) \cdot \delta} \tag{5.39}$$

Ainsi on obtient à chaque correction de ce capteur six hypothèses de valeur de Δ différentes et indépendantes (une par axe de l'IMU). Chaque valeur est en réalité extrêmement bruitée et non significative en elle-même, à part si les mesures ont changé très rapidement, mais ce bruit est de moyenne nulle. Il suffit donc de moyenner toutes les mesures $\tilde{\Delta}$ obtenues sur une certaine période de temps pour faire ressortir du bruit un biais qui est la vraie valeur de Δ.

Il est de plus possible de déterminer l'incertitude de chaque mesure $\tilde{\Delta}$ à partir des incertitudes de la correction et des mesures de l'IMU, et de l'amplitude des variations des mesures de l'IMU, afin d'effectuer une moyenne pondérée, et d'estimer l'incertitude de la moyenne $\hat{\Delta}$ pour pouvoir décider quand l'utiliser.

Nous n'avons cependant pas eu le temps d'implémenter cette fonctionnalité, mais pas ressenti non plus fortement sa nécessité, le calibrage sur données hors ligne étant suffisamment stable dans notre cas.

5.5 Résultats

Les figures 5.20 à 5.22 page 126 montrent le résultat du SLAM visuel-inertiel sur la séquence MOCAP_LOOP. Par rapport aux résultats du SLAM visuel pur présentés section 4.6 page 78, le constat est sans appel : les différentes trajectoires sont bien plus groupées, et précises. On distingue à nouveau la fermeture de boucle pour la majorité des trajectoires, entre les abscisses 9 et 10 m. On constate cependant une erreur significative, qui est très corrélée avec l'éloignement à l'origine comme on peut le voir figure 5.21, qui a toutes les caractéristiques d'une erreur de facteur d'échelle. Le biais des accéléromètres étant inconnu au départ, la convergence initiale vers la vitesse est imprécise, et le mouvement de faible dynamique ne permet pas de l'observer correctement par la suite. Cette erreur de facteur d'échelle a également tendance à rendre l'estimation un peu inconsistante.

Les figures 5.23 à 5.25 page 128 montrent cette fois les résultats pour la séquence MOCAP_FAST. Encore une fois la différence avec le SLAM visuel pur est frappante. Le système est parfaitement robuste à la dynamique, et d'une très bonne précision. D'une part la forte dynamique de la séquence permet de mieux estimer les biais de la centrale inertielle, et d'autre part le faible renouvellement des amers prévient la dérive. Le système est de plus consistant de façon très satisfaisante. L'erreur de cap qui oscille avec le cap lui-même est dû à des imprécisions de calibrage, notamment une petite erreur de synchronisation entre la vérité terrain et le système de SLAM conjugué à la très grande dynamique du mouvement. La forte consistance en cap et extrême en vitesse sont en réalité dues à l'incertitude de la vérité terrain. La précision de l'ordre du millimètre sur la position des marqueurs (donc plutôt bonne) de la capture de mouvement fournit en effet une précision seulement de l'ordre du degré sur les angles (donc plutôt faible). De plus pour le calcul de vitesse une fusion avec un modèle de mouvement à vitesse constante est utilisé, auquel on a dû pour cette séquence donner une très grande liberté afin de ne pas dégrader la dynamique réelle du mouvement, donnant une vérité terrain de vitesse très peu filtrée et avec une grande incertitude.

À partir de maintenant nous nous intéressons aux séquences CAYLUS et ESPERCE acquises par notre robot. Ces deux séquences sont trop longues pour être gérées en mémorisant tous les amers dans le filtre de Kalman, aussi les résultats présentés ici et dans les chapitres suivants sont obtenus avec le mode odométrie visuelle décrit section 4.4.4.c page 71, qui oublie définitivement les amers perdus, et n'est donc pas capable de fermer de grandes boucles. La contrepartie est que le traitement s'effectue dorénavant en temps constant lors de la progression du robot. Nous présentons section 7.4 page 184 la méthode selon laquelle les grandes boucles seront fermées.

Les figures 5.26 à 5.28 page 130 illustrent les résultats obtenus sur un vrai robot, sur la séquence CAYLUS. On constate à nouveau une dispersion significative, qui se manifeste à chaque virage. Elle est toujours due au problème d'estimation du facteur d'échelle, qui est un peu atténué par rapport à la séquence MOCAP_LOOP car la dynamique est plus importante (le robot fait notamment un arrêt à chaque point de passage). Un autre phénomène intervient cependant ici, car les virages sont brutaux et provoquent une renouvellement rapide des amers. Ceci laisse une petite liberté au facteur d'échelle pour changer, les biais des accéléromètres n'étant pas parfaitement estimés (on peut constater ces changements de facteur d'échelle avec les variations de l'erreur de vitesse). La consistance en revanche est très peu satisfaisante. On obtient sur cette séquence une dérive de $18\,\mathrm{cm}/\sqrt{\mathrm{m}}$, et une erreur RMS de 7.9 m.

Les figures 5.29 à 5.31 page 132 montrent les résultats sur la séquence ESPERCE. On constate une erreur maximale de l'ordre de 5 m, et un facteur d'échelle qui peut changer au cours de la trajectoire (certaines commence trop long et finissent trop courtes), ainsi qu'entre les différentes exécutions, résultant en une dispersion conséquente de l'erreur. On constate cependant que le système est robuste à toutes les conditions environnementales traversées puisqu'il n'y a aucun décrochage de la vision ni erreur très grande. L'erreur au démarrage en z est due à l'estimation des biais des accéléromètres, comme on l'a vu figure 5.9 page 106. On calcule une dérive moyenne de $23\,\mathrm{cm}/\sqrt{\mathrm{m}}$, et une erreur RMS de 1.8 m.

L'apport des données inertielles sur la robustesse et la précision du SLAM visuel est donc primordial, et rend le système utilisable en pratique en tant que tel. Sa précision n'est ce-

pendant pas idéale, la dispersion des trajectoires étant relativement conséquente, et il n'est pas non plus complètement à l'abri de fautes. En effet dans certaines conditions, lorsque la vitesse est constante et sans arrêts, et qu'un virage brusque est pris, le facteur d'échelle est encore moins observable, et peut tellement dériver qu'il mène le système à la divergence. Tous ces problèmes restants sont effectivement liés à l'estimation du facteur d'échelle, lié à l'estimation délicate des biais des accéléromètres. C'est pourquoi nous allons voir dans le chapitre suivant comment intégrer des capteurs supplémentaires, notamment des capteurs permettant de mieux observer ce facteur d'échelle, tels que l'odométrie ou d'autres caméras.

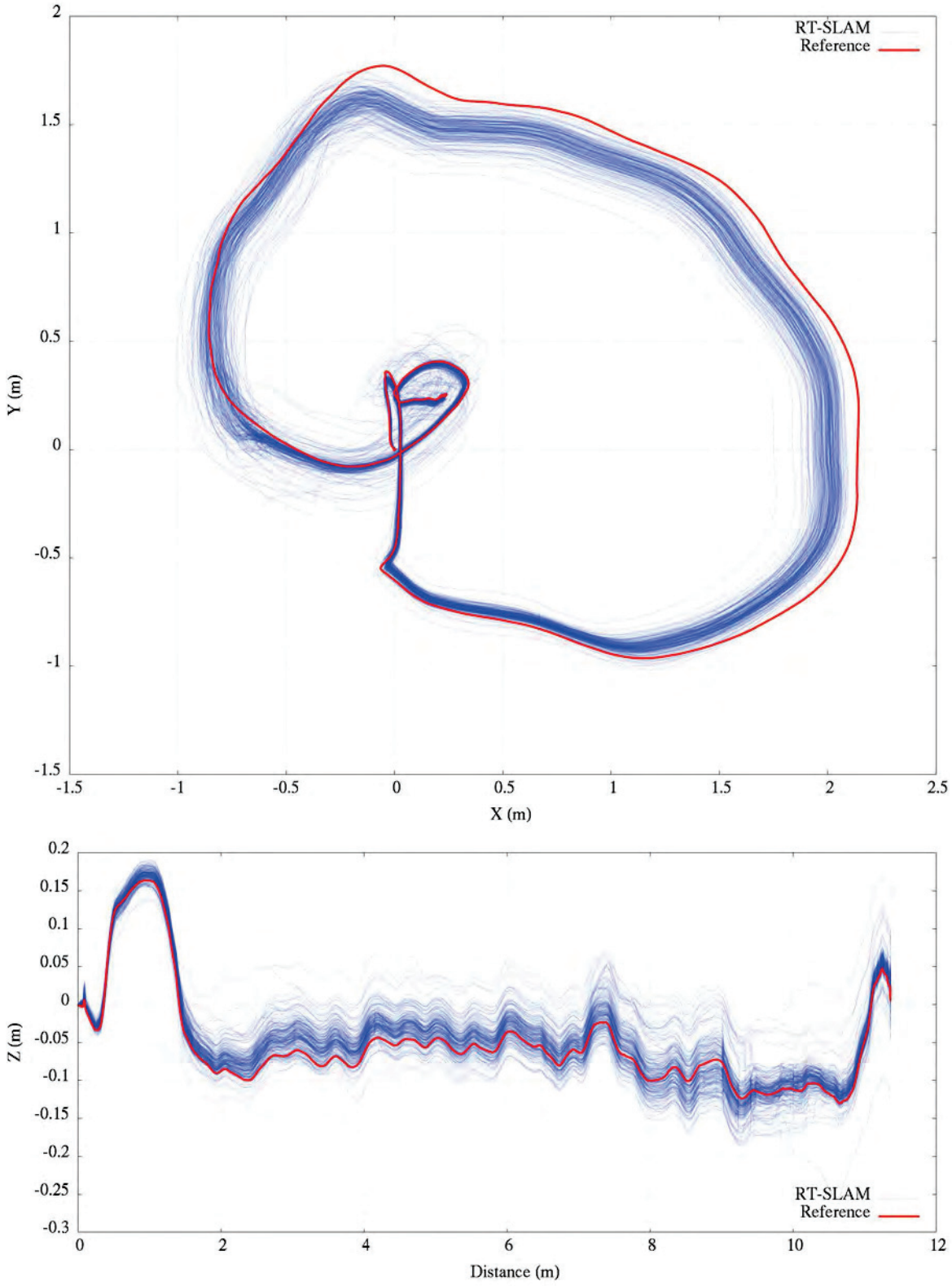


FIGURE 5.20 – Résultat de trajectoire du SLAM visuel-inertiel sur la séquence MOCAP_LOOP.

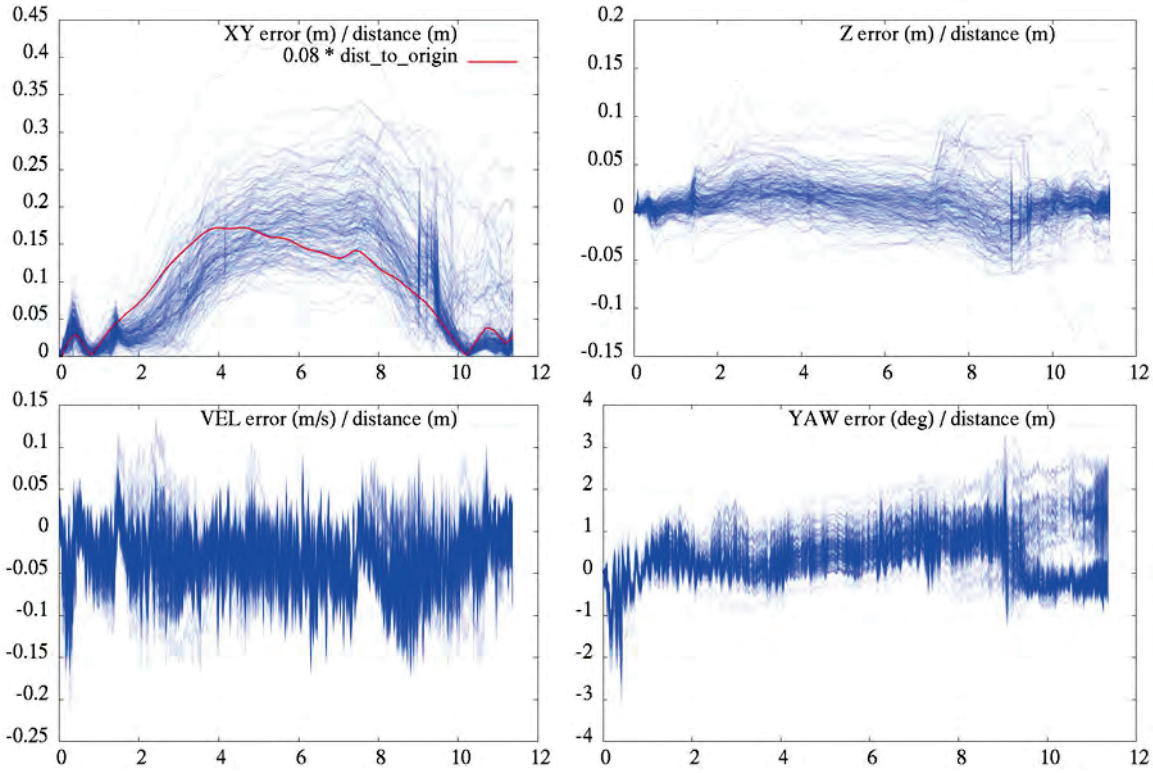


FIGURE 5.21 – Résultat d'erreur du SLAM visuel-inertiel sur la séquence MOCAP_LOOP.

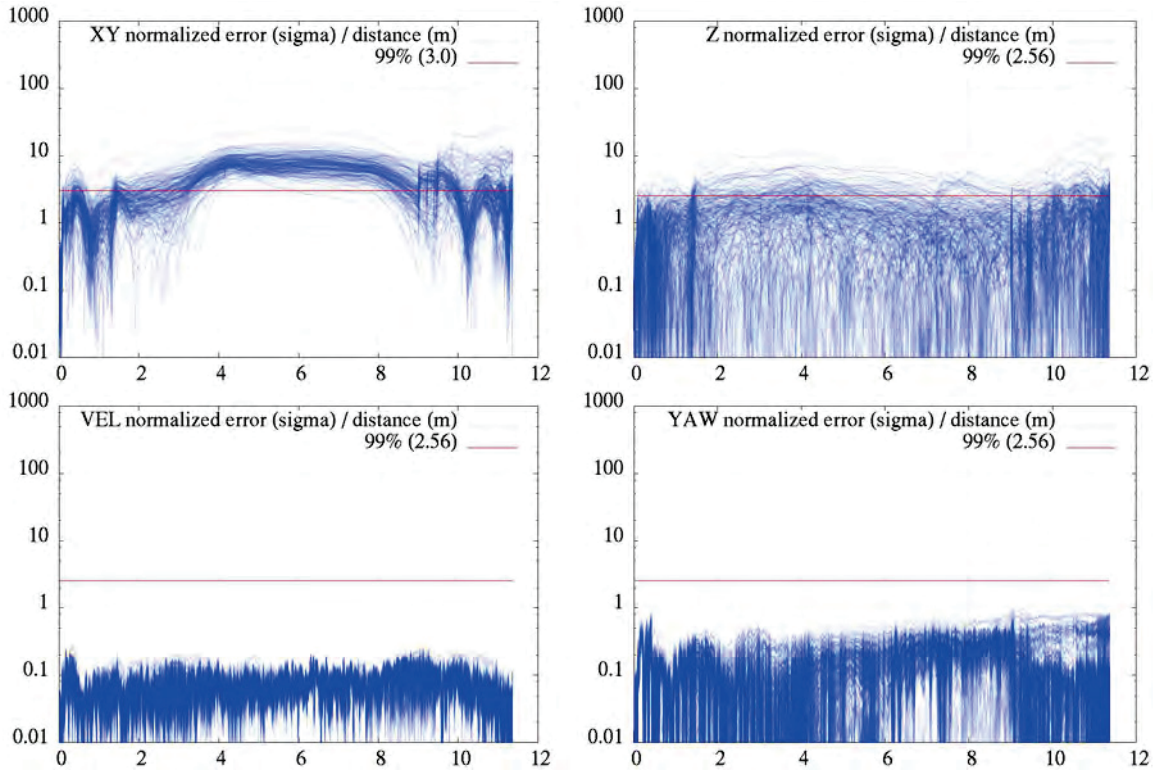


FIGURE 5.22 – Résultat d'erreur normalisée du SLAM visuel-inertiel sur la séquence MOCAP_LOOP.

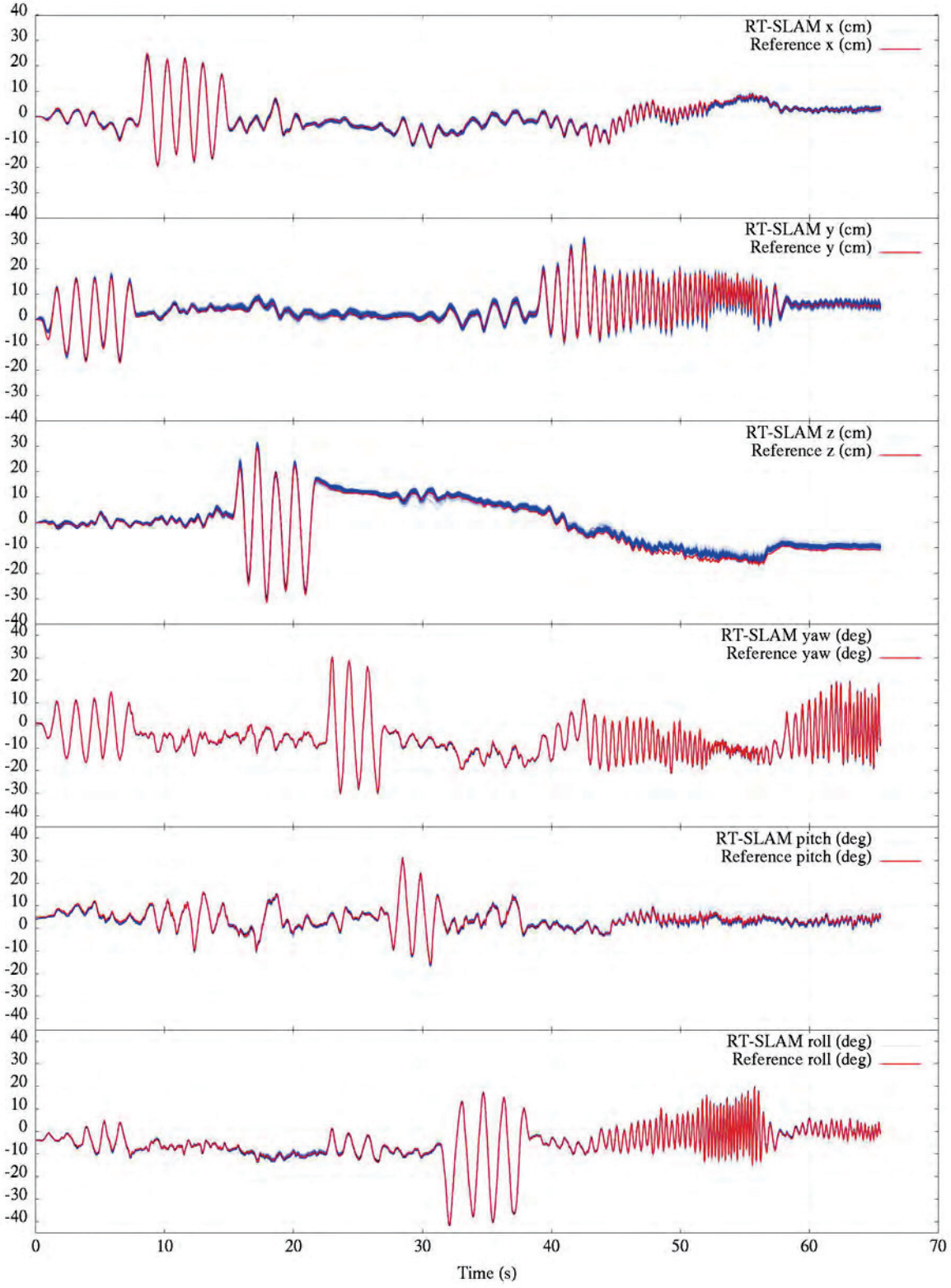


FIGURE 5.23 – Résultat de trajectoire du SLAM visuel-inertiel sur la séquence MOCAP_FAST.

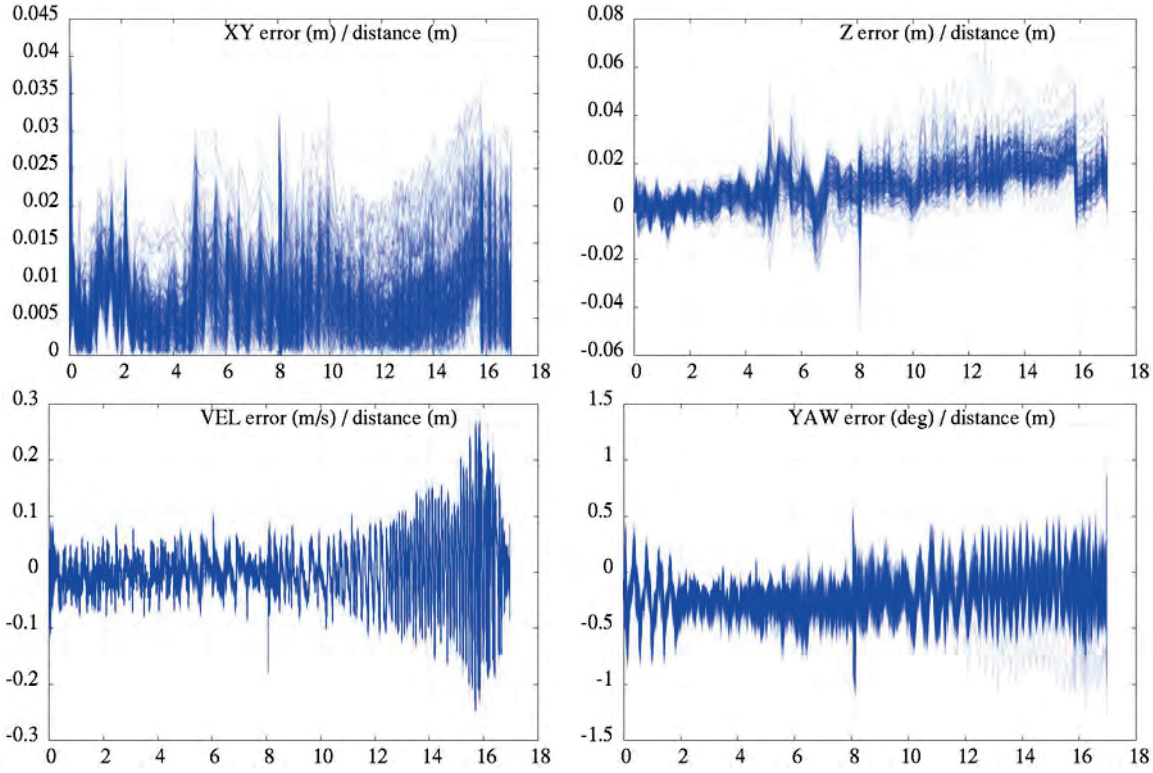


FIGURE 5.24 – Résultat d'erreur du SLAM visuel-inertiel sur la séquence MOCAP_FAST.

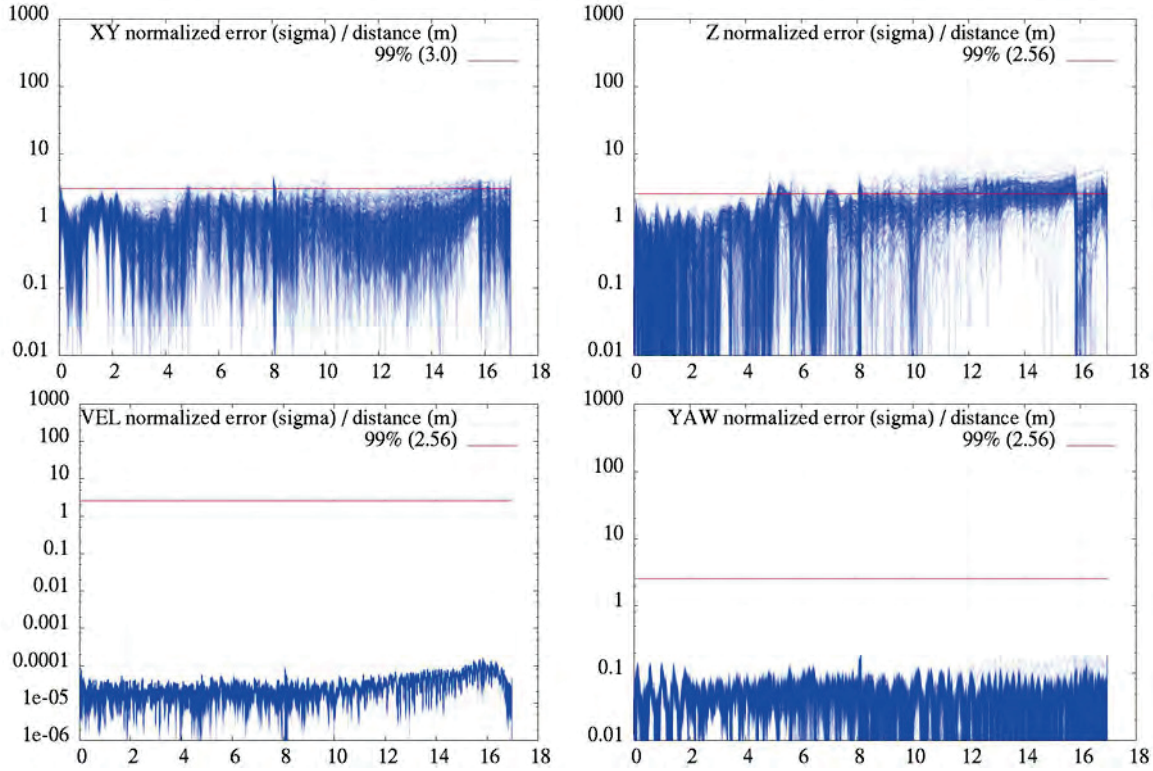


FIGURE 5.25 – Résultat d'erreur normalisée du SLAM visuel-inertiel sur la séquence MOCAP_FAST.

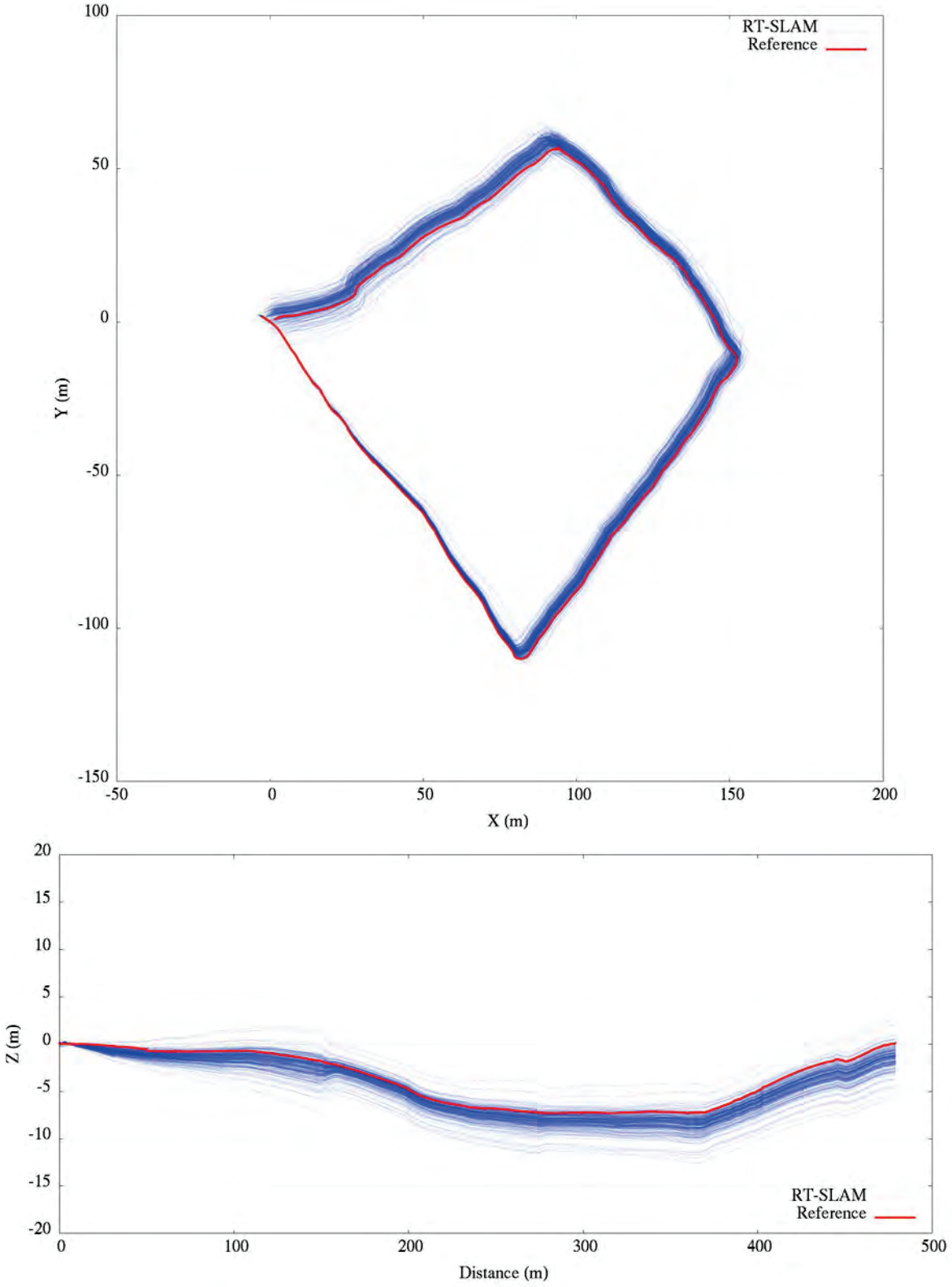


FIGURE 5.26 – Résultat de trajectoire du SLAM visuel-inertiel sur la séquence CAYLUS.

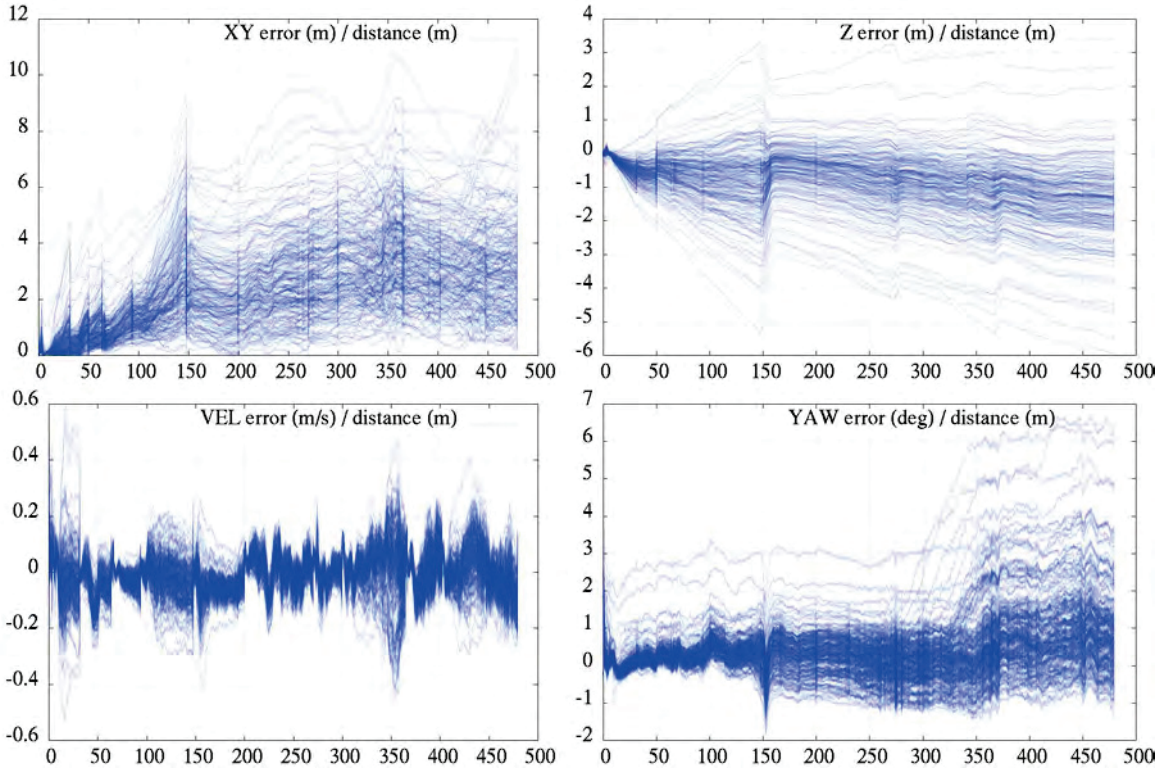


FIGURE 5.27 – Résultat d'erreur du SLAM visuel-inertiel sur la séquence CAYLUS.

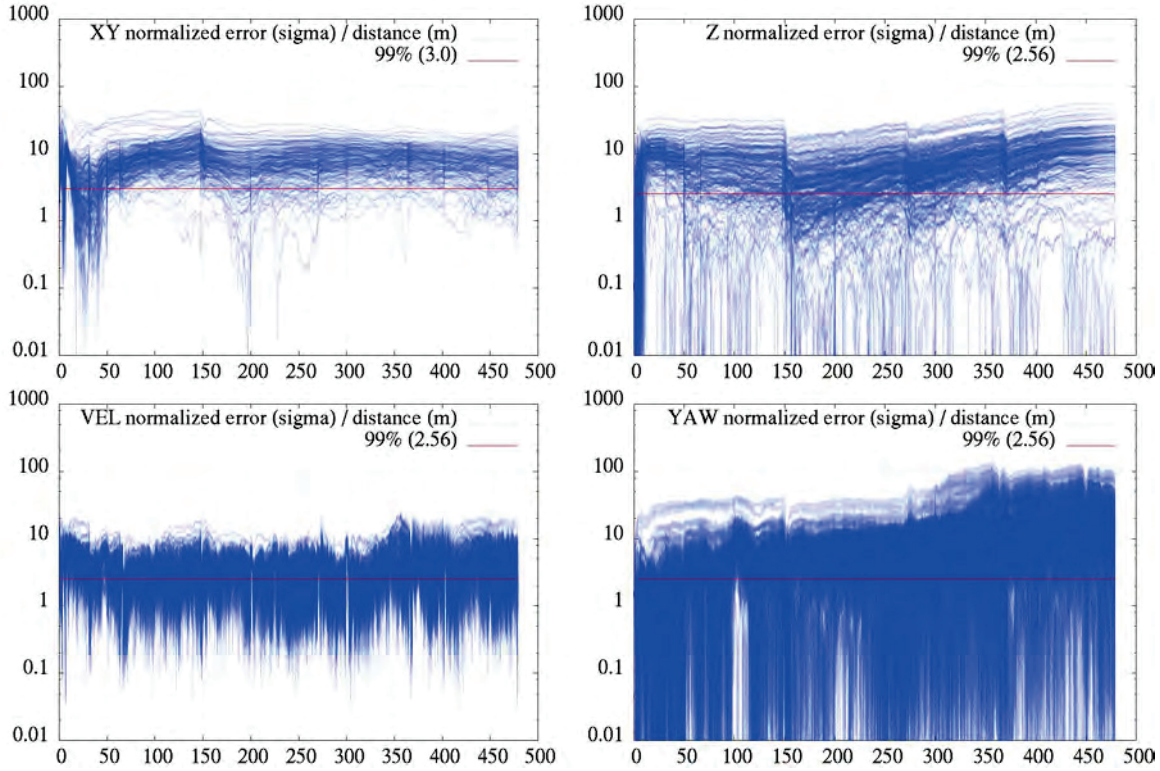


FIGURE 5.28 – Résultat d'erreur normalisée du SLAM visuel-inertiel sur la séquence CAYLUS.

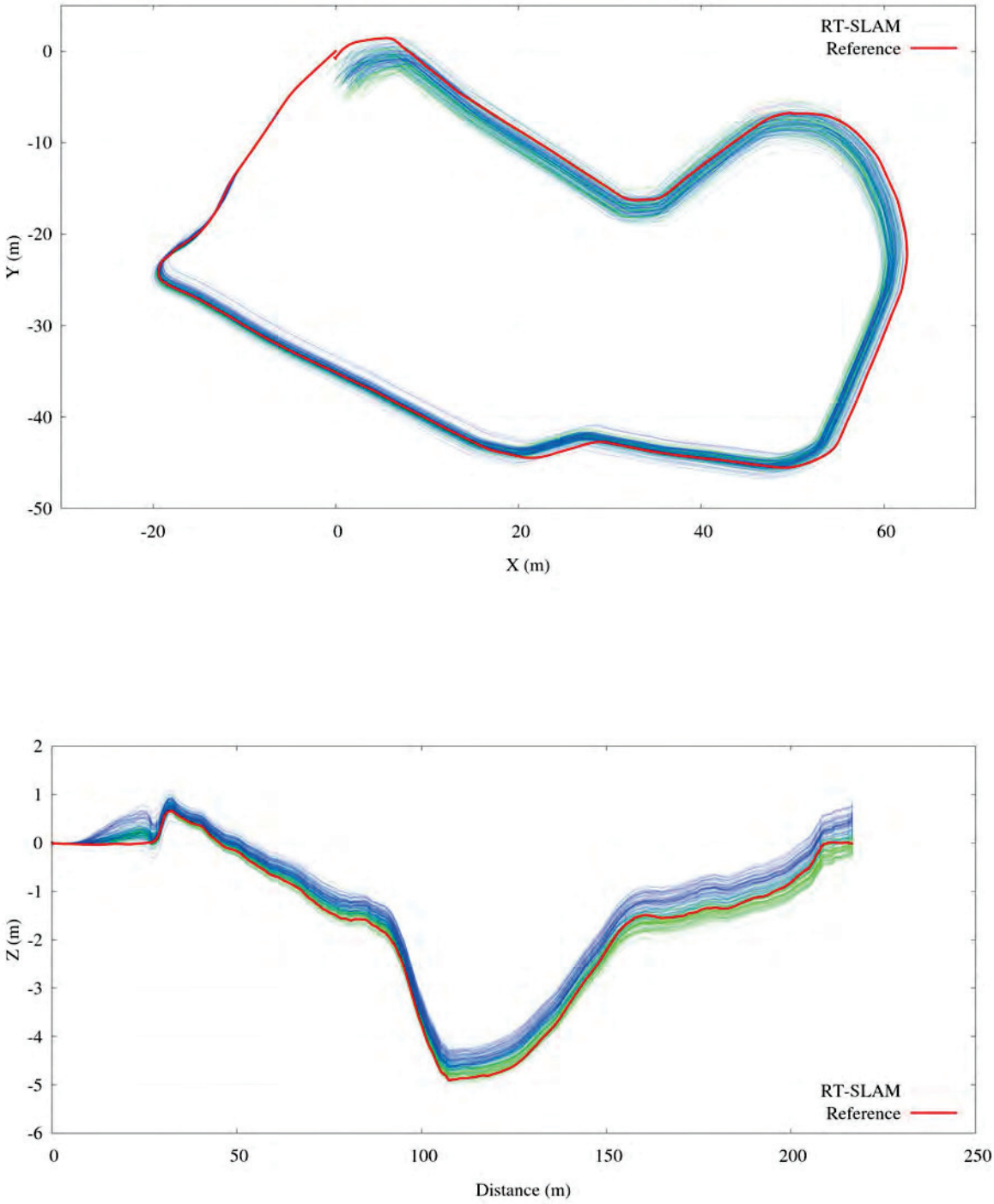


FIGURE 5.29 – Résultat de trajectoire du SLAM visuel-inertiel sur la séquence ESPERCE. Les couleurs vert et bleu correspondent aux deux caméras.

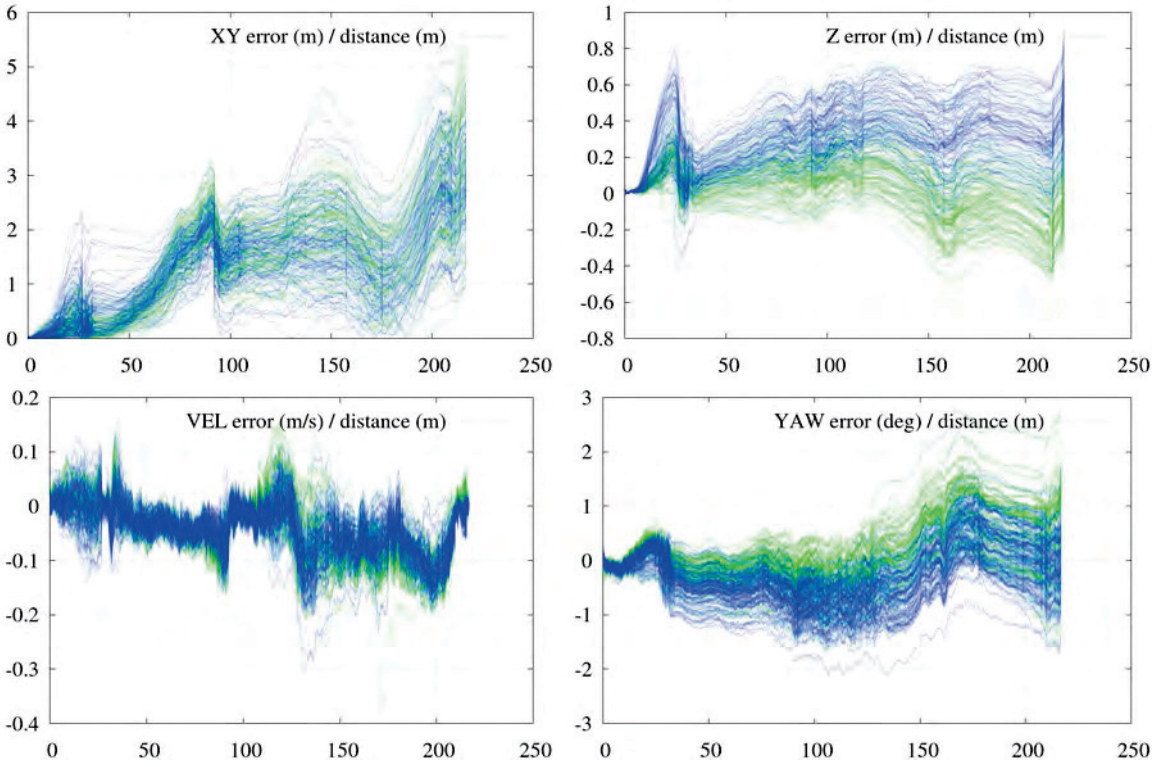


FIGURE 5.30 – Résultat d'erreur du SLAM visuel-inertiel sur la séquence ESPERCE.

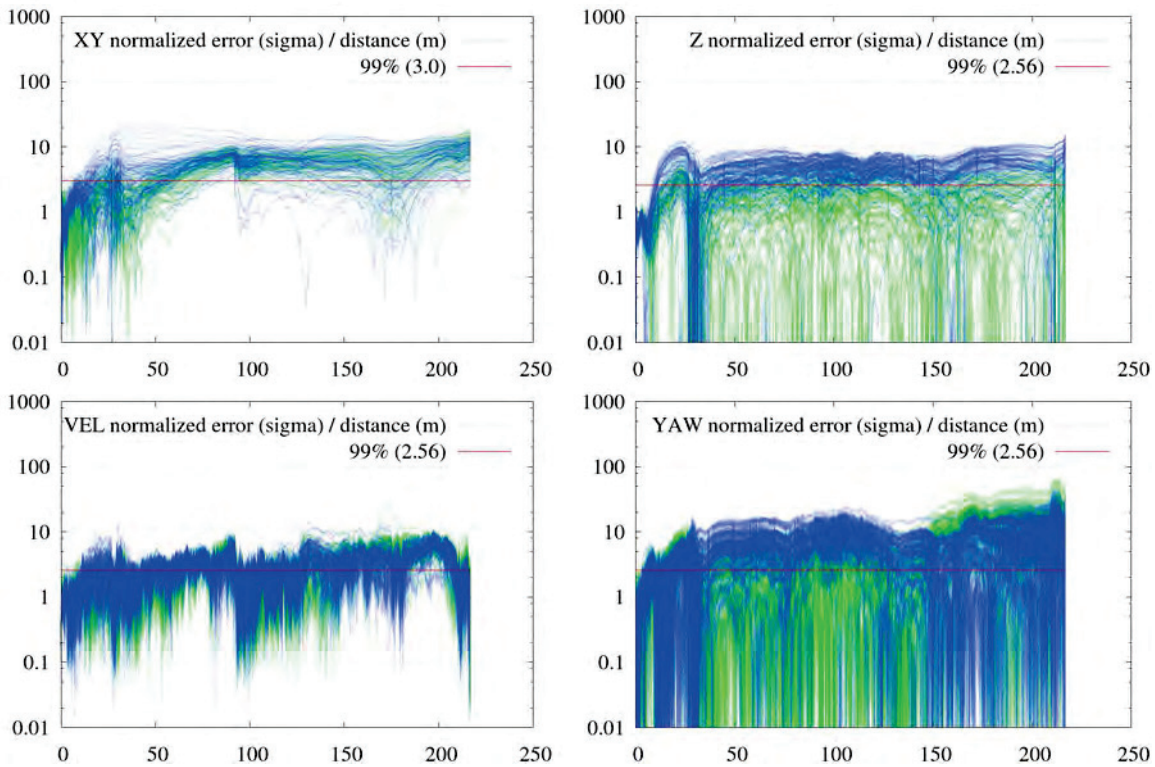


FIGURE 5.31 – Résultat d'erreur normalisée du SLAM visuel-inertiel sur la séquence ESPERCE.

L'exploitation de capteurs supplémentaires, proprioceptifs ou extéroceptifs, pour encore améliorer les performances. Seront présentés une méthode originale d'intégration de l'odométrie, une proposition d'intégration de gyromètres supplémentaires, des méthodes de localisation absolue telles que GPS ou systèmes de capture de mouvement, et enfin l'utilisations de caméras supplémentaires.

Chapitre 6

Utilisation de capteurs supplémentaires

Sommaire

Comme on vient de le voir dans le chapitre précédent, si l'utilisation d'une centrale inertielle en sus de la caméra rend le système beaucoup plus robuste et précis, il reste des cas où les performances ne sont pas suffisantes. Nous verrons donc dans ce chapitre comment exploiter des capteurs supplémentaires, qu'ils soient proprioceptifs ou extéroceptifs, dans le but de combler les dernières lacunes et encore améliorer les performances.

Nous nous intéresserons dans un premier temps à l'odométrie, où nous verrons une méthode pour l'intégrer afin de s'afranchir de ses défauts, tout en profitant de ses qualités de stabilité et d'apport de facteur d'échelle (section 6.1). Nous nous pencherons ensuite sur les gyromètres précis à fibre optique, et proposerons des solutions pour les intégrer malgré les conflits avec la centrale inertielle (section 6.2), avant de passer aux systèmes de localisation absolus, comme le GPS (section 6.3), qui peut fournir des positions et des vitesses absolues, les systèmes de capture de mouvement (section 6.4), qui peuvent également fournir l'orientation absolue, ainsi que des observations partielles de position par un observateur extérieur. Enfin nous nous intéresserons à l'exploitation de caméras supplémentaires (section 6.5), en étudiant les différentes possibilités, avant de conclure avec un aspect d'implémentation critique lors de l'utilisation de plusieurs capteurs en temps réel avec un filtre récursif, qui est la gestion temporelle de l'intégration des capteurs (section 6.6).

6.1 Odométrie

Tous les robots terrestres à roues sont dotés de codeurs odométriques ou tachymétriques, nécessaires pour la commande individuelle des moteurs. L'intégration de leurs données est peu coûteuse en temps de calcul, et procure certains avantages :

— elle stabilise le facteur d'échelle lorsque rien d'autre ne tient ce rôle (par exemple en complément d'une caméra et une centrale inertielle),

— elle rend le système plus robuste en l'empêchant de diverger dans des cas de décrochages (qui peuvent arriver lors de la perte de la vision avec uniquement des informations inertielles pour prendre le relais).

6.1.1 Approche

L'intégration de l'odométrie pose cependant quelques difficultés qu'il faut traiter de façon adéquate :

— Elle ne répond pas à l'ensemble des critères énumérés en section 5.1.3 page 87 pour être utilisée dans l'étape de prédiction du filtre, car elle peut être fautive en cas de glissement des roues. Il faut donc l'utiliser en observation, ce qui permet d'utiliser le contrôle de compatibilité pour détecter ses fautes. Il est en effet important de ne pas prendre en compte du tout les données franchement fautives, et pas seulement d'englober les possibles fautes dans l'incertitude des données, car ces fautes sont des biais sur les mesures et constituent donc un bruit absolument non gaussien.

— L'odométrie ne fournit en réalité que la norme de la vitesse du véhicule, et pas sa

direction. S'il est effectivement souvent supposé approximativement que le déplacement du véhicule se fait le long de son axe longitudinal, ceci n'est pas tout à fait vrai dès lors que le véhicule dispose d'une transmission non rigide avec le sol (suspensions ou pneus). En effet le phénomène de transfert de masse fait que le véhicule hausse le nez lors des accélérations, et pique du nez lors des décélérations, et si on utilise l'angle d'attitude du véhicule pour intégrer l'altitude on aura des erreurs (qui ont de plus tendance à ne pas se compenser mais au contraire à plus s'accumuler vers le haut, comme illustré section C.2 page 221). C'est donc la norme de la vitesse qui doit être observée. On peut cependant ajouter l'information que la vitesse latérale du véhicule est nulle, du moins si l'on a exclu que l'on pouvait être en train de glisser latéralement.

— La vitesse angulaire fournie par l'odométrie est très peu fiable sur les véhicules avec quatre roues fixes comme notre plateforme Mana, pour lesquelles un glissement des roues est indispensable pour réaliser les rotations (*skid-steering*). Mieux vaut donc ne pas exploiter ces informations, d'autant que cela n'est pas nécessaire car la vision est beaucoup plus précise pour estimer les orientations.

— Enfin bien sûr elle n'est pas disponible sur les plateformes aériennes, et très partiellement sur les plateformes terrestres sans roues.

6.1.2 Modèle

Notre robot étant muni de quatre roues fixes commandables indépendamment, l'odométrie du robot fournit la vitesse angulaire de chacune des roues. On calcule donc la vitesse linéaire du robot par la moyenne des vitesses correspondant à chaque roue, en fonction de leurs vitesses angulaires $(\omega_{lf}, \omega_{lb}, \omega_{rf}, \omega_{rb})$ et de leurs rayons respectifs $(r_{lf}, r_{lb}, r_{rf}, r_{rb})$:

$$\tilde{v}_o = \frac{1}{4} \cdot (r_{lf} \cdot \omega_{lf} + r_{lb} \cdot \omega_{lb} + r_{rf} \cdot \omega_{rf} + r_{rb} \cdot \omega_{rb}) \tag{6.1}$$

Il serait bien sûr possible d'observer de façon brute chacune des vitesses des quatre roues, ce qui permettrait de prendre en compte plus précisément les différentes incertitudes, mais en pratique l'incertitude dépend plus des conditions de terrain et de navigation et de la cohérence des informations entre les roues que des incertitudes individuelles de chacune des roues, donc il est à la fois préférable et plus simple de résumer ces informations à une seule vitesse.

L'odométrie fournit par définition la vitesse du centre du robot physique. Mais on a vu section 5.2.2 page 89 que le repère du robot du SLAM était le repère de la centrale inertielle. Dans le cas général où la centrale inertielle n'est pas placée au centre physique du robot, on doit donc prendre en compte la position de l'odométrie (l'origine du robot physique) par rapport au robot du SLAM (l'origine du capteur inertiel), exprimée par la transformation constituée d'une translation et d'un quaternion $(\mathbf{t_o}, \mathbf{q_o})$.

Puisque l'on a décidé de n'observer que la norme de la vitesse, l'équation d'observation est donc :

$$\hat{v}_o = \|\mathbf{v} + \mathrm{q2R}(\mathbf{q}) \cdot (\mathbf{w} \wedge \mathbf{t_o})\|_2 \tag{6.2}$$

où pour rappel $\mathbf{q}$, $\mathbf{v}$ et $\mathbf{w}$ sont respectivement l'orientation, et les vitesses linéaire et angulaire du robot, et $\wedge$ représente le produit vectoriel.

En pratique nous plaçons la centrale inertielle au centre du robot, car cela permet d'observer plus précisément la vitesse linéaire du robot, qui est l'objectif de l'odométrie, en levant toute ambiguïté avec les vitesses angulaires. Nous utilisons donc l'équation simplifiée suivante :

$$\hat{v}_o = \|\mathbf{v}\|_2 \tag{6.3}$$

6.1.3 Aspects d'implémentation

a Calculs d'incertitudes

Comme on l'a vu section 6.1.1 page 136, le problème de l'odométrie est qu'elle peut facilement être fautive, notamment lors de glissements des roues.

Une première solution est donc de calculer l'incertitude de l'odométrie dans son fonctionnement nominal, sans faute, et de compter sur le contrôle de compatibilité de l'EKF pour détecter les fautes et les éliminer. Le problème est que lorsque l'incertitude de la vitesse dans l'état du robot est un peu trop grande il n'y a aucune garantie de ne pas intégrer une vitesse odométrique fautive.

Cependant certaines plateformes robotiques proposent des informations riches d'odométrie. Par exemple les plateformes SegWay RMP400 fournissent les vitesses, couples et commandes indépendantes des quatre roues. Il est donc possible de détecter certaines fautes ou risques de fautes, et même d'estimer l'incertitude des données en examinant leur cohérence, voire d'essayer de les corriger.

Exemple Pratiquement nous avons implémenté une solution très simple, qui calcule un score de fiabilité à partir de différents critères : cohérence des données des différentes roues, constance de la vitesse, faiblesse de la vitesse angulaire, faiblesse des couples, cohérence avec la commande. Ce critère est ensuite converti en une incertitude pour prendre en compte un risque de faute légère, et à partir d'un seuil correspondant à une faute avérée la donnée est complètement ignorée. Il serait cependant intéressant d'effectuer un apprentissage du risque et de l'ampleur de l'erreur à partir de ces données.

Remarque Une précaution particulière doit être prise lors de l'opération de contrôle de compatibilité, lorsque l'incertitude des données a été calculée pour prendre en compte le risque de faute. En effet le contrôle s'effectue normalement sur l'innovation, qui prend donc en compte à la fois l'incertitude de la mesure et celle de la mesure attendue. Ceci est correct lorsqu'on a un capteur qui exhibe des fautes, puisque le but du contrôle de compatibilité est justement de détecter ces fautes. En revanche si on a calculé l'incertitude de manière à intégrer la faute maximale qui peut être commise, on se retrouve avec un capteur qui n'est jamais fautif, et le contrôle de compatibilité ne pourra rien détecter. Il n'est cependant toujours pas souhaitable d'intégrer ces données puisque leur erreur est biaisée. Il faut alors utiliser seulement l'incertitude de la mesure attendue dans le test de compatibilité pour calculer la distance de Mahalanobis, au lieu de l'incertitude de l'innovation (qui contient en plus l'incertitude de la mesure), si l'on veut éliminer les données non compatibles.

b Calibrage

Le calibrage de l'odométrie revient à calculer précisément le rayon des roues. Pour cela le plus simple et le plus efficace est de comparer la distance parcourue par chaque roue à une vérité terrain (GPS centimétrique, mesure manuelle, ...). On choisira de préférence un terrain plat et régulier pour pouvoir se contenter d'utiliser les deux extrémités de la trajectoire, et le plus long possible pour améliorer la précision en diluant les incertitudes de mesure aux extrémités.

Il est important de calibrer séparément chaque roue si cela est possible, car les différences peuvent être significatives et cela permet de mieux analyser leur cohérence. Par exemple sur notre plateforme, après un calibrage sur une distance de 50 m, on obtient des facteurs correctifs pour chaque roue de l'ordre du pourcent : $(0.990, 1.000, 0.997, 1.009)$.

6.1.4 Résultats

Les figures 6.1 à 6.3 page 140 montrent les résultats obtenus sur la séquence CAYLUS, à comparer avec les résultats sans odométrie présentés page 130. Par rapport au SLAM visuel-inertiel sans odométrie, la dérive est tombée à $9\,\mathrm{cm}/\sqrt{\mathrm{m}}$ (divisée par 2), et l'erreur RMS à 1.7 m (divisée par plus de 4). On constate clairement sur la trajectoire que la dispersion a désormais tendance à diminuer lors des virages au lieu d'augmenter, ce qui montre qu'il n'y a plus de problème de facteur d'échelle, et que la dispersion restante est angulaire. La dérive angulaire plus importante sur le dernier segment de la trajectoire est due au contre-jour qui pénalise les ancres visuelles. L'erreur sur la vitesse a également significativement diminué.

Les figures 6.4 page 142 à 6.6 142 montrent les résultats sur la séquence ESPERCE. Là encore l'estimation du facteur d'échelle est nettement amélioré, et en conséquence la dispersion grandement réduite. On calcule une dérive moyenne d'environ $8\,\mathrm{cm}/\sqrt{\mathrm{m}}$ (divisée par 3), et une erreur RMS à 1.1 m (réduction de 40%). Le facteur d'échelle semble cependant s'être détérioré pendant dans la partie basse du graphique de la trajectoire, puisqu'elle se retrouve un peu trop court alors qu'elle était bien dans le virage précédent. Cette partie correspond à une pente forte dans des hautes herbes, où l'odométrie est peu fiable et donc peu utilisée par le système.

6.2 Gyromètre

Si trois gyromètres sont déjà utilisés dans la centrale inertielle, ce sont des gyromètres très bas coût qui dérivent très rapidement. Dans notre système, les angles d'assiette et de gîte sont stabilisés par l'observation de la gravité, et ne dérivent donc pas, mais rien n'empêche l'angle de cap de dériver.

Il existe cependant des gyromètres beaucoup plus précis, appelés FOG (pour *Fiber Optic Gyrometer*), qui exploitent le comportement relativiste de la lumière en lui faisant parcourir dans les deux sens une longue fibre optique enroulée sur elle-même (« effet Sagnac »). Ces gyromètres sont très précis, avec des dérives de l'ordre de quelques degrés par heure, pour un prix de quelques milliers d'euros qui reste accessible.

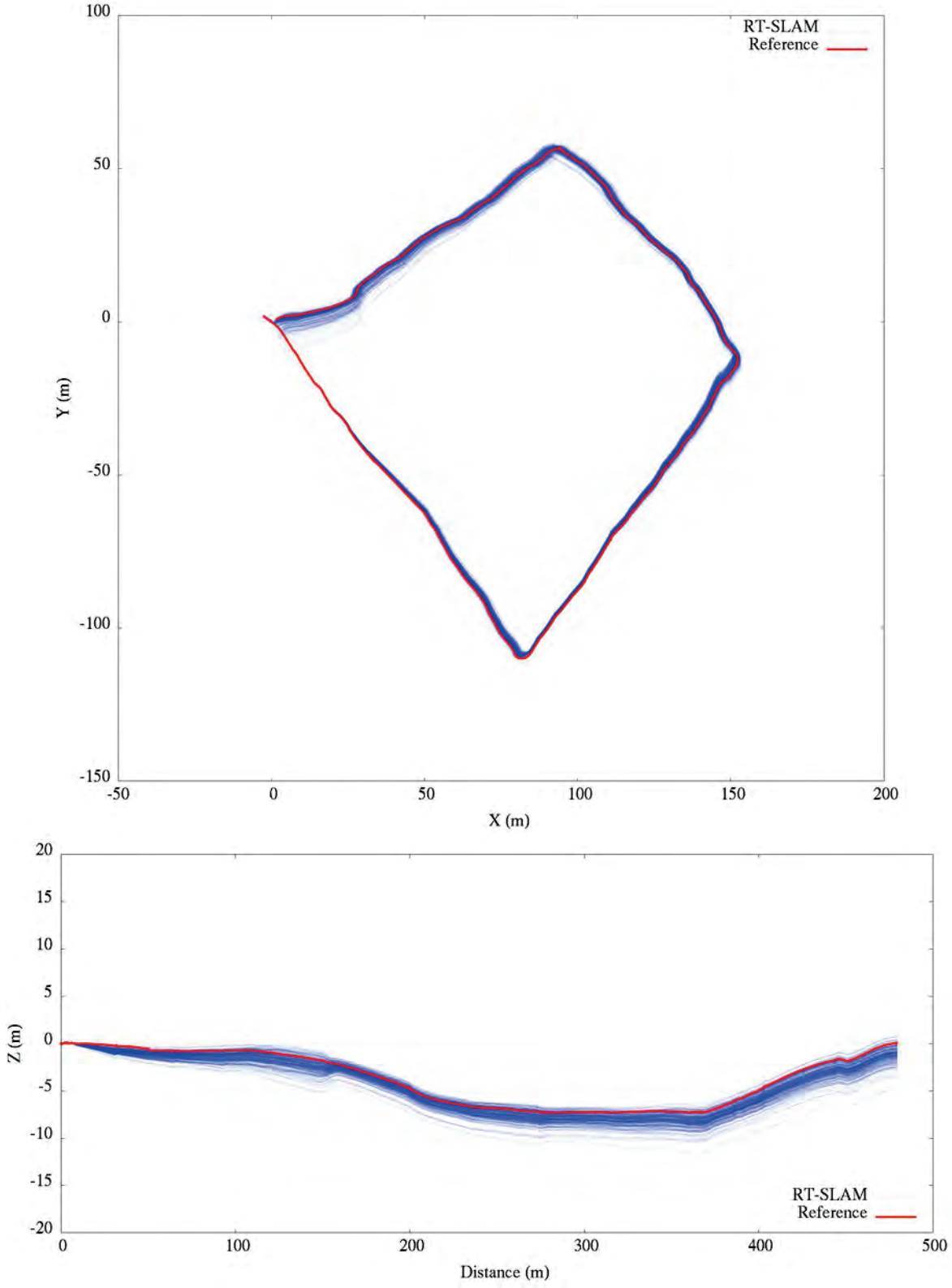


FIGURE 6.1 – Résultat de trajectoire du SLAM visuel-inertiel avec odométrie sur la séquence CAYLUS.

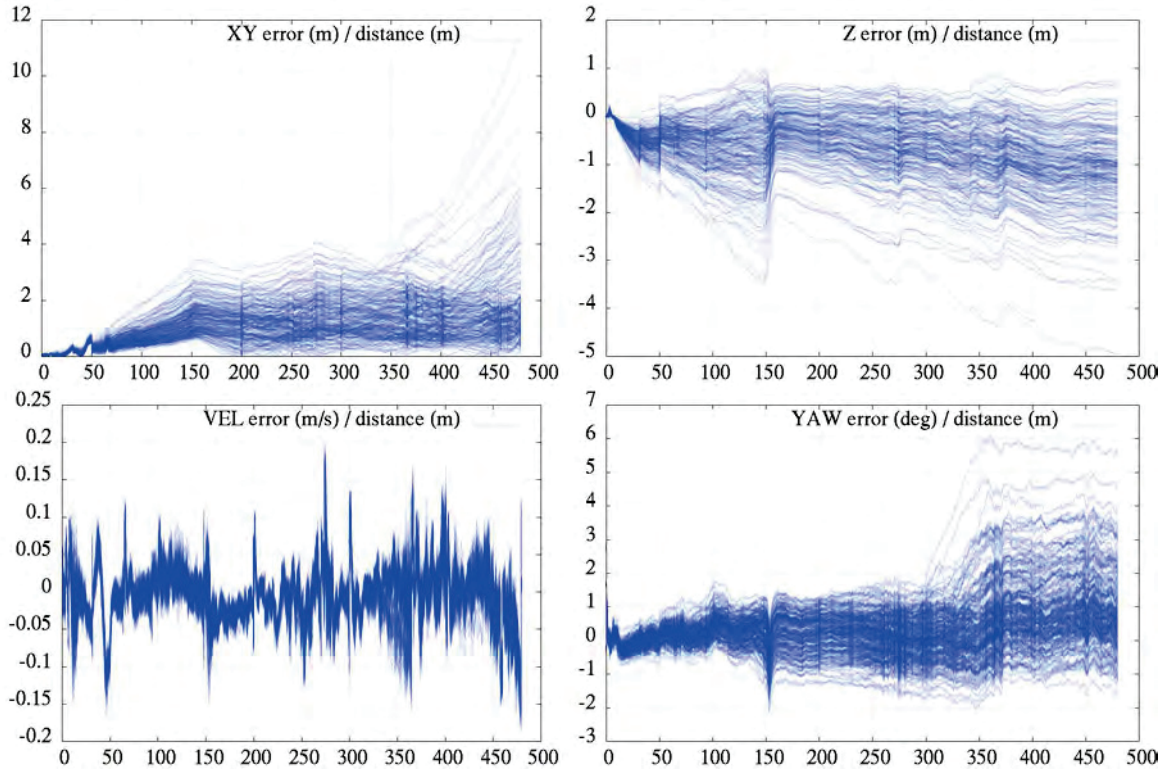


FIGURE 6.2 – Résultat d'erreur du SLAM visuel-inertiel avec odométrie sur la séquence CAYLUS.

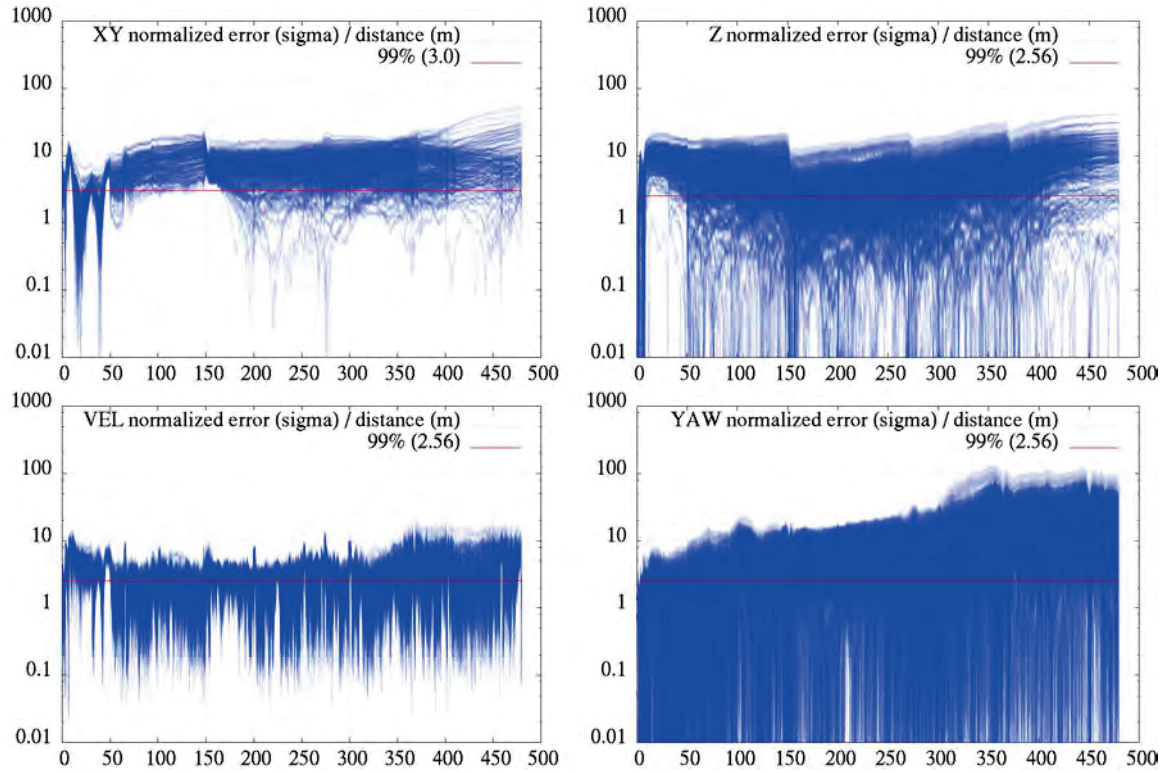


FIGURE 6.3 – Résultat d'erreur normalisée du SLAM visuel-inertiel avec odométrie sur la séquence CAYLUS.

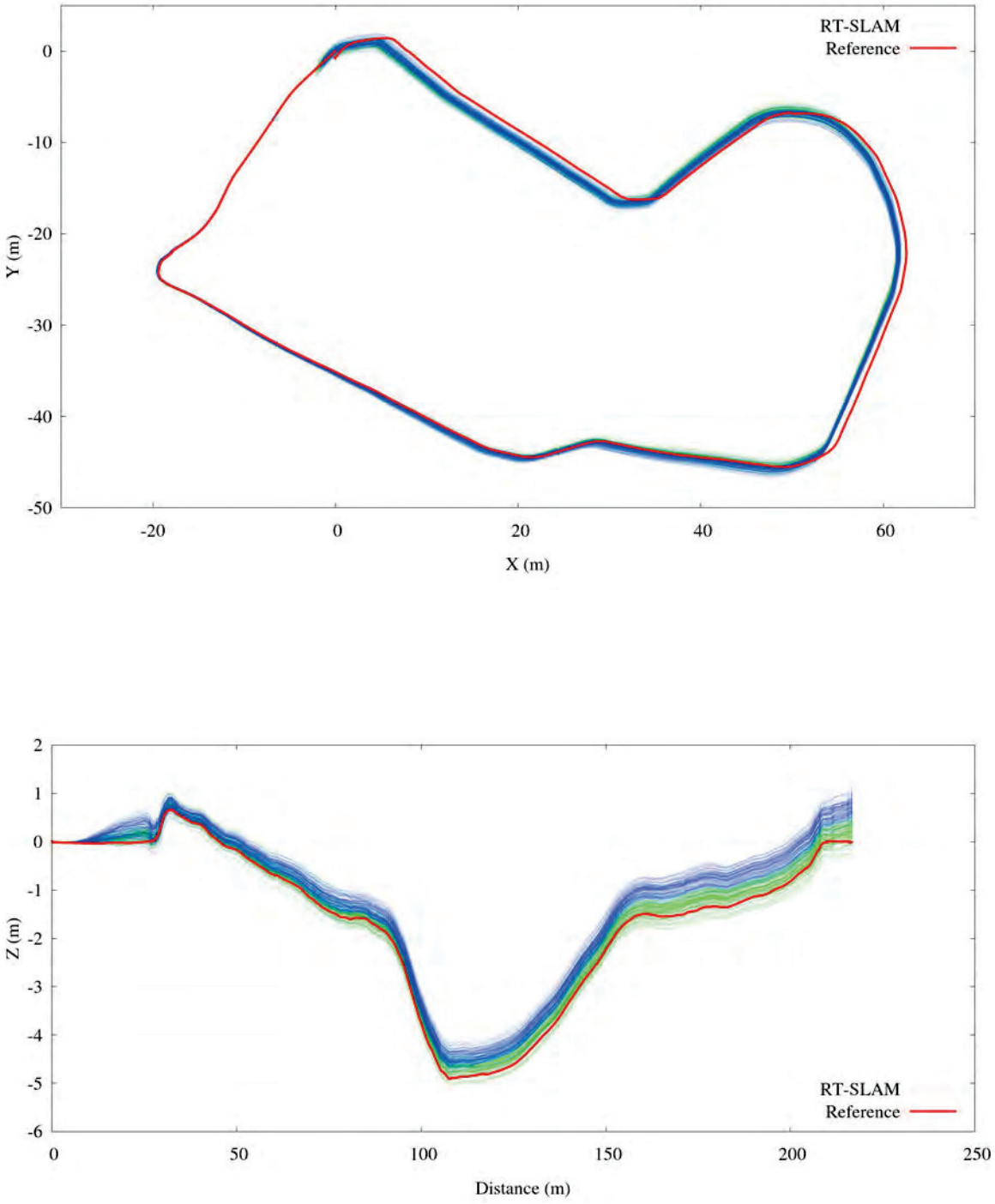


FIGURE 6.4 – Résultat de trajectoire du SLAM visuel-inertiel avec odométrie sur la séquence ESPERCE.

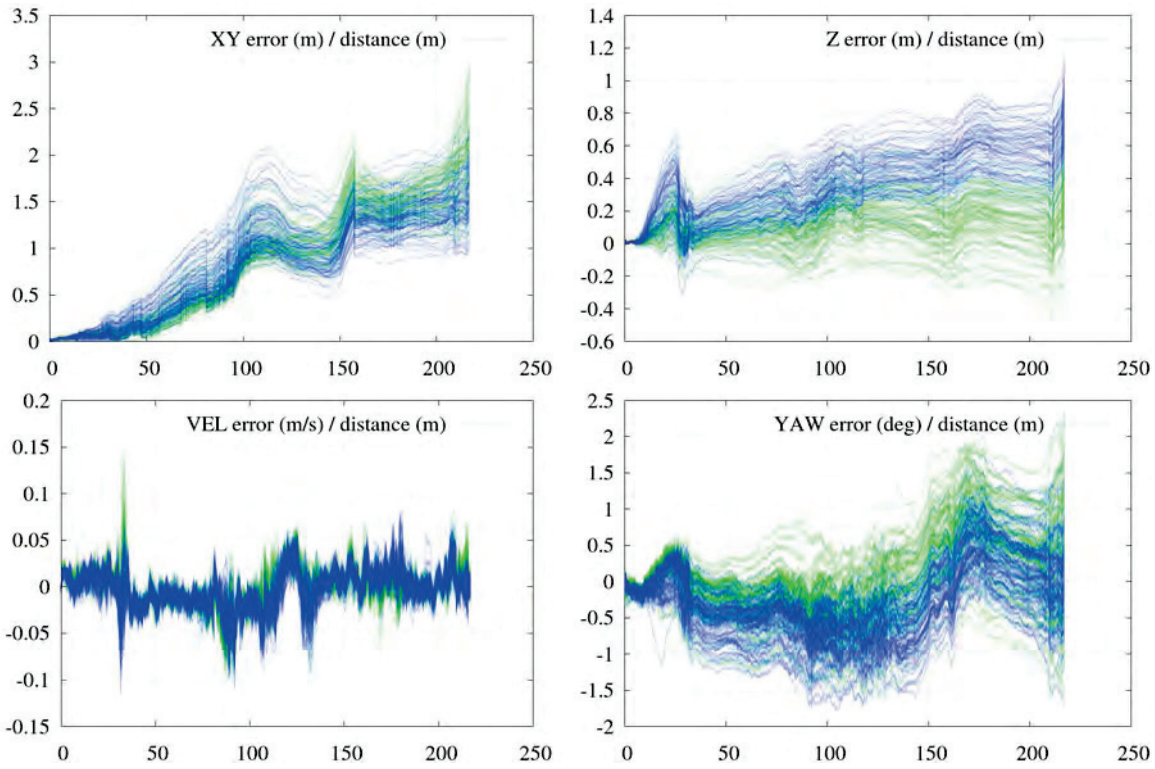


FIGURE 6.5 – Résultat d'erreur du SLAM visuel-inertiel avec odométrie sur la séquence ESPERCE.

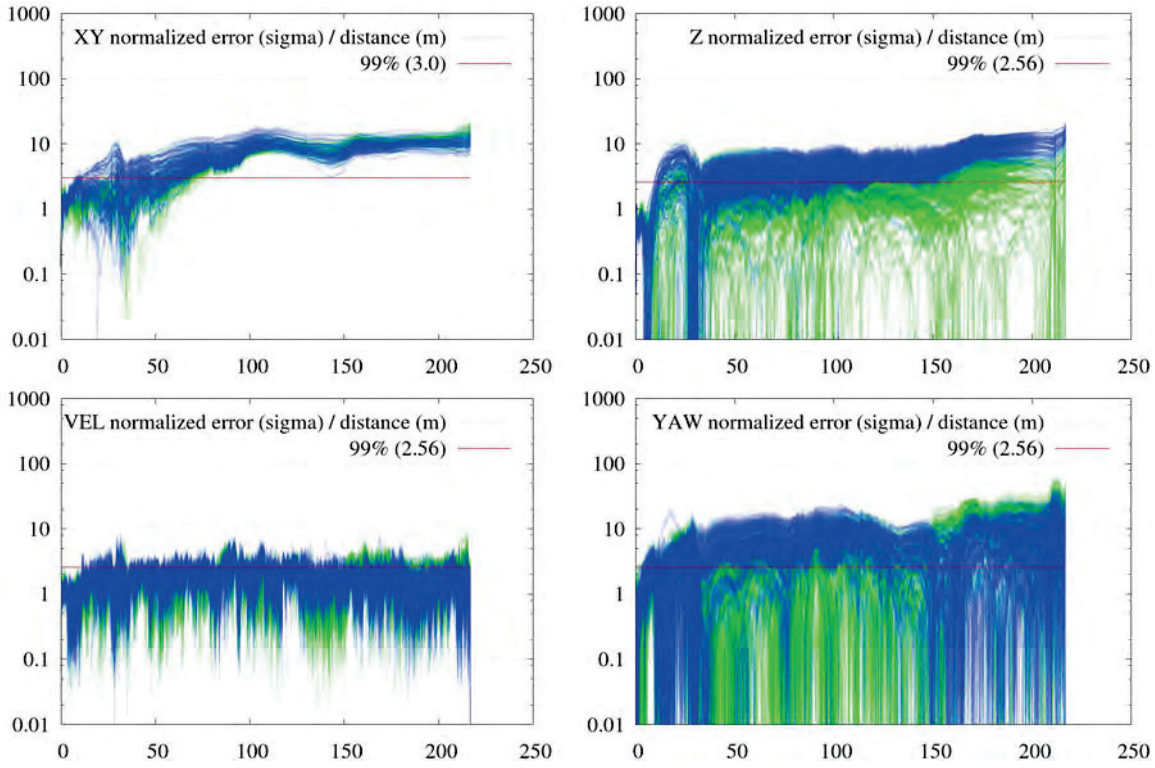


FIGURE 6.6 – Résultat d'erreur normalisée du SLAM visuel-inertiel avec odométrie sur la séquence ESPERCE.

Il peut donc être intéressant d'exploiter un tel gyromètre pour limiter la dérive de cap du système. Cela pose cependant une difficulté de compatibilité avec la centrale inertielle utilisée en prédiction, comme cela a été évoqué section 5.1.3 page 87. Nous allons voir dans cette section les solutions possibles, même si faute de temps aucune n'a été implémentée en pratique.

6.2.1 Intégration

Toute grandeur mesurée par un capteur intégré en observation doit être estimée dans le filtre (les équations de Kalman imposent d'être capables de définir la fonction d'observation qui transforme l'état à l'instant courant en une prédiction de la mesure du capteur ; en l'occurrence ici si la vitesse angulaire n'est pas dans l'état, on ne pourra pas la prédire à partir de l'état). Or si l'on ajoute la vitesse de lacet dans l'état, l'équation de prédiction avec l'IMU aura pour effet d'écraser la valeur estimée, annulant ainsi l'essentiel de l'information du FOG intégré en observation.

a IMU en observation

Une solution pour remédier à cette difficulté consiste à passer l'IMU en observation, et utiliser un modèle de mouvement pour la prédiction. Le filtre fusionnera alors automatiquement le modèle avec les mesures des deux capteurs, selon les incertitudes de chacun.

Dans le modèle, on enrichit alors l'état des accélérations linéaires $\mathbf{a}$, des vitesses angulaires $\mathbf{w}$, et du biais du FOG f_b :

$$\mathcal{R} = (\mathbf{p}\ \mathbf{q}\ \mathbf{v}\ \mathbf{w}\ \mathbf{a}\ \mathbf{a_b}\ \mathbf{w_b}\ f_b\ \mathbf{g})^T \tag{6.4}$$

Avec un modèle de mouvement à vitesse angulaire et accélération linéaire constantes, on a alors l'équation d'évolution suivante :

$$\begin{aligned}
\mathbf{p}^+ &= \mathbf{p} + \mathbf{v} \cdot \mathrm{d}t && (6.5)\\
\mathbf{q}^+ &= \mathbf{q} \otimes \mathrm{v2q}\,(\mathbf{w} \cdot \mathrm{d}t) && (6.6)\\
\mathbf{v}^+ &= \mathbf{v} + (\mathrm{q2R}(\mathbf{q}) \cdot \mathbf{a} + \mathbf{g}) \cdot \mathrm{d}t && (6.7)\\
\mathbf{a}^+ &= \mathbf{a} && (6.8)\\
\mathbf{w}^+ &= \mathbf{w} && (6.9)\\
\mathbf{a_b}^+ &= \mathbf{a_b} && (6.10)\\
\mathbf{w_b}^+ &= \mathbf{w_b} && (6.11)\\
{f_b}^+ &= f_b && (6.12)\\
\mathbf{g}^+ &= \mathbf{g} && (6.13)
\end{aligned}$$

Les équations d'observation des accéléromètres et gyromètres étant ensuite triviales :

$$\begin{aligned}
\tilde{\mathbf{a}}_\mathbf{m} &= \mathbf{a} + \mathbf{a_b} && (6.14)\\
\tilde{\mathbf{w}}_\mathbf{m} &= \mathbf{w} + \mathbf{w_b} && (6.15)\\
\tilde{f}_m &= (1\ 0\ 0) \cdot \mathbf{w} + f_b && (6.16)
\end{aligned}$$

On perd cependant les avantages d'utiliser l'IMU en prédiction plutôt qu'en observation, présentés section 5.1.3 page 87 ; le temps de calcul dans le filtre sera en particulier significativement plus long puisqu'il devra mettre à jour l'ensemble de la matrice de covariance du filtre.

b Fusion des gyromètres

On préfèrera en général fusionner en amont les informations du FOG et du gyromètre de lacet de l'IMU, en supposant bien sûr qu'ils sont parfaitement parallèles. Si le FOG est suffisamment rapide pour tenir lieu de prédiction, il suffit de remplacer le gyromètre de lacet de l'IMU par le FOG. Dans ce cas il n'y a absolument rien à changer aux équations de prédiction, à part correctement mettre à jour les paramètres de bruits avec ceux du FOG.

Si le FOG est au contraire trop lent (par exemple 10 Hz), on pourra alors fusionner les deux gyromètres avec un modèle de mouvement dans un filtre de Kalman dédié (dont l'état ne sera constitué que d'une variable donc le problème du temps de calcul ne se posera pas), afin de conserver la précision du FOG tout en bénéficiant de la dynamique du gyromètre de l'IMU. La sortie de ce filtre sera alors utilisée directement en prédiction du filtre principal. L'inconvénient est cependant que l'on détériore un peu la précision du FOG, et qu'il n'est plus possible d'estimer les biais des gyromètres, et il est donc préférable d'éviter cette solution.

6.2.2 Aspects d'implémentation

a Synchronisation temporelle

Intégrer un gyromètre très précis impose une synchronisation temporelle elle aussi très précise, au risque de provoquer des incompatibilités dans la recherche des amers.

Il est donc très important de filtrer correctement les horodatages du FOG, comme expliqué section 5.4.2 page 108, ainsi que de calibrer précisément le décalage temporel comme expliqué section 5.4.3 page 117.

b Rotation de la Terre

La précision des FOG, de l'ordre de quelques degrés par heure, est cependant désormais telle que la mesure de la rotation de la Terre présentée section 5.2.1 page 89 n'est plus du tout négligeable. Elle crée un biais dans les mesures, qui dépend de la latitude, mais également de l'orientation absolue du gyromètre. Cependant si on considère que l'attitude du robot reste d'amplitude limitée et de moyenne nulle, on peut négliger l'influence de l'orientation, et retenir le biais suivant en fonction de la latitude ℓ :

$$w_e = 15 \cdot \sin(\ell) \qquad \text{deg/h} \tag{6.17}$$

6.3 GPS

Lorsqu'il est disponible, le signal GPS est très intéressant pour un système de localisation car il permet de borner l'erreur et de limiter la dérive.

Un capteur GPS peut être intégré de deux façons :

— Intégration dite à couplage fort (*tight integration*), qui réimplémente les équations du GPS pour utiliser les données brutes de pseudodistances aux satellites dans le filtre. On peut alors considérer un récepteur GPS intégré ainsi comme un capteur extéroceptif (on observe des satellites qui sont des amers).

— Intégration dite à couplage faible (*loose integration*), qui utilise directement la position finale calculée par le GPS. On peut alors considérer qu'on a affaire à un capteur proprioceptif à notre sens (section 2 page 9).

L'intégration à couplage fort est plus précise, car elle permet de prendre en compte des modèles d'erreur à la fois plus simples et plus justes, et évite de mettre en cascade plusieurs filtres, mais est en revanche considérablement plus difficile à implémenter, à cause de la complexité des équations du GPS.

Nous nous sommes donc pour des questions de facilité orientés vers une intégration faiblement couplée du GPS, n'ayant ni les compétences ni le temps à consacrer à une intégration fortement couplée.

Nous nous plaçons également dans le cadre simplifié du SLAM inertiel avec vecteur gravité forcé à la verticale. En effet d'un point de vue pratique le SLAM inertiel est beaucoup plus robuste, et le fait qu'il observe le facteur d'échelle et l'orientation absolue le rend moins dépendant du GPS. Ceci permet de n'exploiter le GPS qu'occasionnellement, ce qui est intéressant car le signal GPS n'est pas toujours disponible. Imposer le vecteur gravité vertical simplifie également les équations d'observation du GPS puisque cela enlève la nécessité d'une rotation pour le ramener à la verticale et obtenir l'orientation absolue, sans contraindre particulièrement l'utilisation du robot quant à l'initialisation de l'orientation, ni être pénalisant pour la précision comme on l'a vu dans le chapitre 5 page 85.

On peut noter que la déviation du vecteur gravité avec la verticale locale définie par le géoïde ellipsoïdique utilisé par le GPS pour modéliser la Terre étant de l'ordre de quelques millièmes de degré, elle est largement négligeable devant la précision de nos mesures (même si la déviation avec la verticale théorique supposant que la Terre est une sphère serait elle de l'ordre de quelques dixièmes de degré).

6.3.1 Intégration du GPS-RTK centimétrique

Le GPS différentiel utilise des corrections transmises par une station de base locale, qui permettent de compenser les perturbations atmosphériques dans la transmission des signaux des satellites. Le GPS RTK mesure en plus la phase de la porteuse de ces signaux pour atteindre une précision de l'ordre du centimètre.

L'intégration des mesures d'un tel GPS ne pose pas de difficulté car son bruit est gaussien,

et les fonctions d'observations sont de plus assez immédiates, puisque ce capteur observe directement l'état du robot. La seule subtilité est que l'antenne GPS n'est jamais située au centre du robot, et qu'il faut donc prendre en compte sa position par rapport au robot, exprimée par la transformation constituée d'une translation et d'un quaternion $(\mathbf{t_g}, \mathbf{q_g})$ (en réalité l'orientation n'est pas utilisée car le signal GPS est indépendant de l'orientation du capteur).

a Intégration de la position

Le GPS est configuré pour renvoyer des coordonnées cartésiennes UTM, car elles sont plus intuitives à manipuler par l'opérateur que des coordonnées LLA (Latitude Longitude Altitude). Leur intégration dans le filtre en est également simplifiée.

La fonction d'observation de la position $\tilde{\mathbf{p}}_\mathbf{g}$ fournie est alors :

$$\hat{\mathbf{p}}_\mathbf{g} = \mathbf{p} + \mathrm{q2R}(\mathbf{q}) \cdot \mathbf{t_g} \tag{6.18}$$

où pour rappel $\mathbf{p}$ et $\mathbf{q}$ sont la position et le quaternion dans l'état du robot estimé par le filtre.

b Intégration de la vitesse

Un récepteur GPS fournit également des mesures de vitesse, obtenues par effet Doppler, et donc indépendantes des mesures de position obtenues par temps de vol. Ces vitesses sont exprimées dans le repère global, et ne dépendent pas de l'orientation de l'antenne GPS. En revanche la vitesse angulaire du robot se répercute sur la vitesse linéaire de l'antenne déportée du centre.

La fonction d'observation de la vitesse $\tilde{\mathbf{v}}_\mathbf{g}$ fournie est donc :

$$\hat{\mathbf{v}}_\mathbf{g} = \mathbf{v} + \mathrm{q2R}(\mathbf{q}) \cdot (\mathbf{w} \wedge \mathbf{t_g}) \tag{6.19}$$

où pour rappel $\mathbf{v}$ et $\mathbf{w}$ sont la vitesse linéaire et angulaire du robot, et $\wedge$ représente le produit vectoriel.

Nous n'avons en réalité pas implémenté l'intégration des mesures de vitesse du GPS, d'une part car l'exploitation des mesures de position suffisait à nos applications, et d'autre part car nous n'étions pas certains de l'indépendance des positions et des vitesses fournies par notre GPS qui réalise un traitement poussé des données.

La simple observation d'une séquence de positions permet de toute façon de corriger l'estimation de la vitesse et surtout de l'orientation.

6.3.2 Intégration du GPS naturel

Revenons sur les notions de signal blanc et de signal gaussien :

— Un signal continu est dit blanc si sa densité de fréquence est uniforme.

— Un signal discret est dit gaussien si sa distribution de probabilité suit une loi normale.

— Un signal continu blanc, échantillonné à n'importe quelle fréquence, donne un signal discret gaussien.

La difficulté de l'intégration du GPS naturel (sans corrections) est qu'il est entâché d'un bruit non blanc. En effet l'erreur des mesures est constituée de la combinaison :

— d'un biais, provoqué par les perturbations dans l'atmosphère, qui dérive lentement, et dont l'ordre de grandeur est de plusieurs mètres.

— d'un bruit blanc lié à l'électronique de mesure, d'ordre de grandeur de plusieurs centimètres seulement.

— de sauts brutaux et aléatoires, liés à des changements dans l'ensemble de satellites utilisés ou à des réflexions multiples dans des environnements encombrés, d'ordre de grandeur de plusieurs mètres.

Ainsi si on laisse un récepteur GPS immobile pendant plusieurs dizaines de minutes, on verra ses mesures dériver lentement de plusieurs mètres autour de sa position réelle, dans une trajectoire plutôt continue. Il n'est donc pas possible d'intégrer directement à plusieurs hertz ces mesures dans un filtre qui fait l'hypothèse de mesures aux bruits gaussien, car le filtre convergerait vers la moyenne locale de ces mesures, quelle que soit l'incertitude donnée à ces mesures.

a Intégrer à basse fréquence

Il est cependant possible d'échantillonner un signal non blanc en un signal approximativement gaussien, si on choisit une fréquence d'échantillonnage de l'ordre de grandeur de la fréquence des changements les plus lents de ce signal.

Si la figure 6.7 montre que la distribution sur le long terme de l'erreur d'un GPS naturel statique est bien gaussienne, la figure 6.8 montre elle qu'il y a temporellement des périodes de biais de l'ordre de l'heure (le récepteur fournissant ses données à 20 Hz). Cela signifie que la période de sous-échantillonnage nécessaire pour obtenir un signal gaussien est du même ordre de grandeur.

Cette solution est correcte et très simple, mais ne permet d'exploiter qu'une très petite partie de l'information fournie par le GPS. Elle évite la dérive à très long terme, mais nécessite d'avoir un système qui ne dérive que peu sur la durée entre deux intégrations de mesures (de l'ordre de l'heure), ce qui est très difficile.

b Modéliser le bruit

Une autre solution classique au problème de l'intégration de mesures aux bruits non gaussiens est de modéliser ce bruit pour corriger ce qui le rend non gaussien. L'exemple typique est un bruit entâché d'un biais fixe ou variant très lentement. Il suffit alors d'estimer ce biais dans le filtre, et d'observer la mesure corrigée de ce biais.

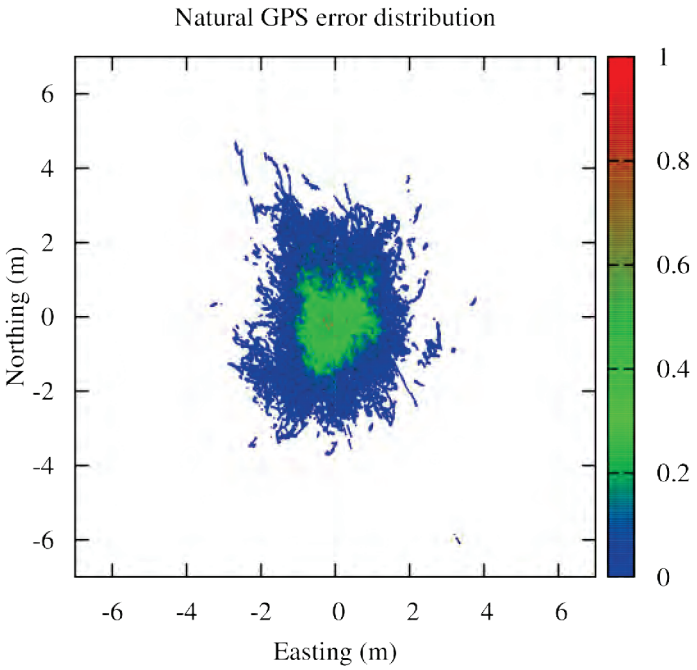


FIGURE 6.7 – Distribution de l'erreur d'un récepteur GPS Novatel OEM4, statique pendant environ 140 heures.

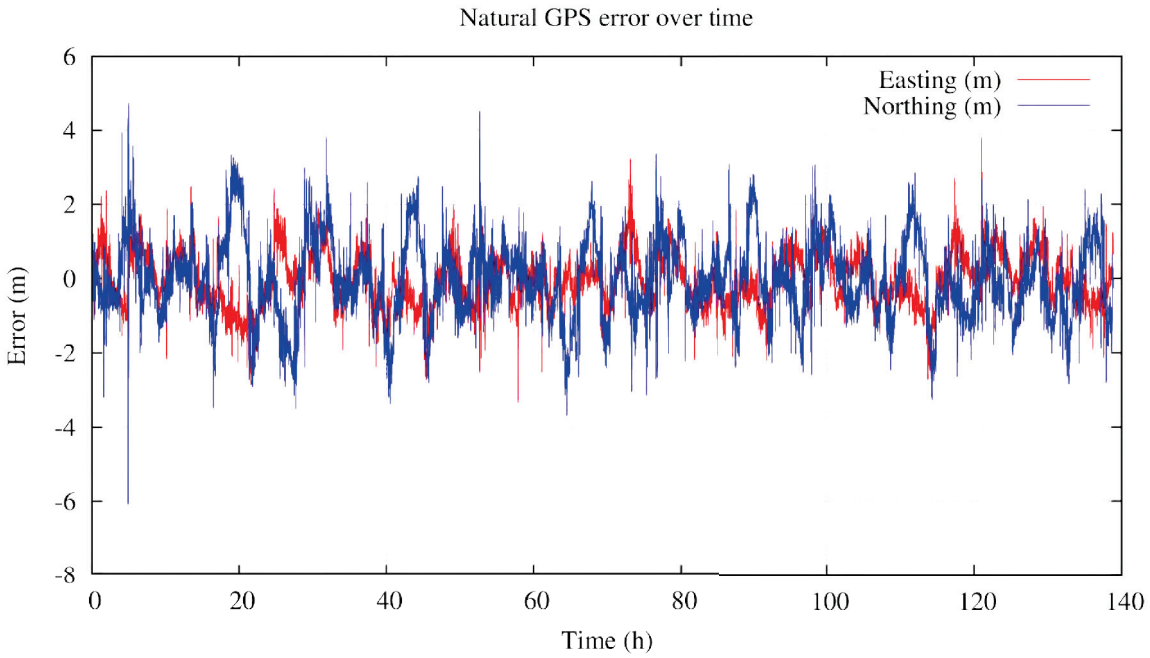


FIGURE 6.8 – Évolution de l'erreur d'un récepteur GPS Novatel OEM4 statique au cours du temps.

Dans le cas du GPS, plusieurs problèmes se posent :

— estimer les variations de ce biais nécessite d'avoir un système qui dérive moins que lui sur un ordre de grandeur de temps comparable à celui des variations du biais, qui est en l'occurrence de l'ordre de l'heure.

— estimer ce biais en absolu nécessite d'avoir un système qui reste intègre sur un temps très grand devant les temps de variation du biais, afin d'en extraire une valeur moyenne (plusieurs heures ou dizaines d'heures). Ceci n'est pas réaliste avec un EKF.

— les éventuels sauts brutaux doivent également être détectés et pris en compte comme des variations brutales du biais.

c Intégrer avec une grande incertitude

Cette méthode est incorrecte, mais est immédiate et peut donner des résultats satisfaisants en l'absence de sauts brutaux. L'inconvénient est que l'estimation va suivre le biais et sera donc entâchée du même biais. Mais ce biais variant lentement il ne va pas beaucoup perturber les autres paramètres estimés comme la direction du vecteur vitesse et la cohérence avec les autres capteurs, et il est de toute façon très difficile d'obtenir une précision absolue meilleure que celle donnée par le GPS en instantané, donc il n'est pas très pénalisant qu'elle varie avec lui.

Il est en revanche souhaitable d'éliminer les sauts brutaux qui eux posent des risques d'incohérence avec les autres capteurs, ce qui peut se détecter assez facilement avec un traitement amont d'élimination de discontinuités, et en estimant la valeur de biais discontinu par celle qui est présente majoritairement en temps.

Il est également nécessaire de ne pas faire de contrôle de compatibilité. En effet d'une part si la position estimée a dérivé et est devenue inconsistante, on cesserait alors complètement d'accepter les mesures GPS qui sont pourtant un moyen de réduire cette dérive. D'autre part à condition de ne pas utiliser les mesures utilisant seulement 4 satellites, il y a une redondance dans l'information qui permet au système de détecter les fautes telles que les réflexions multiples et de fournir une incertitude consistante.

d Intégrer de façon fortement couplée

Il est tout de même utile de préciser qu'une intégration fortement couplée du GPS permettrait de modéliser le bruit du GPS plus fidèlement pour l'intégrer proprement, en considérant les biais indépendants liés à chaque pseudodistance.

6.3.3 Aspects d'implémentation

a Initialisation

Le GPS fournissant des mesures dans le repère global, tandis que le SLAM estime une localisation relative à la position de départ, il est nécessaire d'ajuster les deux repères.

La solution la plus simple est d'initialiser la position du SLAM aux valeurs absolues dans le repère global, avec les incertitudes associées. La position peut-être facilement obtenue en moyennant les premières mesures GPS reçues avant de démarrer, mais le cap est plus délicat à obtenir. En pratique, par simplicité nous avons conservé une procédure qui consiste avant le démarrage de la mission à avancer tout droit de quelques mètres pour l'estimer à partir du GPS, et le donner en initialisation au SLAM, et n'avons pas implémenté d'autre solution.

On peut cependant vouloir s'abstraire d'une telle procédure. Il n'est pas possible en SLAM-EKF de définir une incertitude totale sur le cap et de l'estimer en commençant à avancer, car l'orientation est une fonction complètement non linéaire, dans le sens ou des jacobiennes en différents points peuvent être parfaitement opposées, et la jacobienne calculée en un angle ne permettra pas de corriger pour atteindre un angle opposé. Ce type de problème est en général résolu en faisant plusieurs hypothèses, et en faisant tourner un filtre différent par hypothèse en parallèle jusqu'à pouvoir déterminer lequel est le meilleur. Cette solution n'est pas praticable dans notre cas pour des raisons de temps de calcul. Une solution plus simple est de laisser le SLAM dans son repère local qui dépend de sa position et son orientation de départ, avec une incertitude de départ nulle dans ce repère, mais d'estimer par la suite et par ailleurs ce repère local afin de transformer les mesures du GPS dans ce repère, et exporter une position dans le repère global.

Ainsi au départ les mesures GPS reçues peuvent être utilisées pour déterminer l'origine du repère local. Quand le robot commence à bouger, les mesures ne sont plus utilisées directement dans le filtre, mais par un processus externe au filtre principal qui estime l'orientation initiale du repère local minimisant l'erreur aux moindres carrés entre la trajectoire locale issue du filtre et les mesures GPS, en tenant compte des incertitudes de chaque point (on peut aussi cette fois faire tourner plusieurs petits filtres avec différentes hypothèses). Dès que l'incertitude de l'orientation initiale est passée sous un certain seuil, on l'utilise pour transformer les mesures GPS dans le repère local, en modifiant leur incertitude pour tenir compte de celle sur l'orientation et du déplacement du robot depuis le démarrage. Cette convergence de l'orientation et la réexploitabilité directe des mesures GPS intervient d'autant plus rapidement que les mesures GPS sont précises (quelques dizaines de centimètres suffisent en mode centimétrique, quelques mètres en mode naturel).

Le problème principal de cette méthode est qu'on ne peut pas utiliser la localisation globale obtenue du SLAM pour construire les modèles d'environnement, car elle ne sera pas exacte de façon relative (si on part avec une orientation opposée, on va commencer à partir dans une direction avant de retourner le robot et le faire partir dans l'autre direction ; utiliser cela pour construire un modèle d'environnement serait catastrophique). Il faut donc soit fournir également dans l'interface la localisation locale, et l'utiliser pour construire les modèles d'environnement, soit corriger les modèles au moment où l'orientation initiale du repère local est corrigée grâce au GPS, ce qui complique encore l'implémentation.

b Communication des corrections

La principale difficulté dans l'utilisation du GPS-RTK en ligne est la communication des corrections de l'antenne de référence au récepteur sur le robot. Même lorsque l'objectif n'est

que d'obtenir une vérité terrain, nous n'avons pas les logiciels permettant de d'intégrer ces corrections a posteriori, seul notre récepteur sachant les intégrer, et nous devons tout de même transmettre ces corrections.

Pour des trajectoires courtes une transmission par réseau WiFi est tout à fait suffisante, mais sa portée est limitée à quelques centaines de mètres en environnement dégagé, et bien inférieure en environnement encombré (relief, végétation, bâtiments, ...).

Nous avons donc essayé dans un premier temps d'utiliser des modems dédiés Aerocomm AC4868 fonctionnant dans la bande des 868 MHz, d'une portée théorique de 15 km, mais ils se sont révélés en pratique avoir une portée limitée à quelques centaines de mètres en environnement semi-urbain, et à accumuler du retard dans les transmissions quand les conditions se dégradaient.

Nous avons donc finalement basculé sur des clés 3G, qui présentent l'inconvénient d'être payantes et de ne pas offrir une couverture garantie sur les lieux d'expérimentation, mais qui en pratique sont en général fonctionnelles et permettent une réception sans limite de portée.

6.4 Observations extérieures

Des observations extérieures peuvent être intéressantes à intégrer dans certains contextes, y compris pratiques.

6.4.1 Capture de mouvement

Les systèmes de capture de mouvement permettent par la détection d'au moins trois marqueurs positionnés sur le système à localiser de mesurer à la fois sa position et son orientation, en général de très bonne précision et haute fréquence (pour notre système une précision de l'ordre du millimètre, la précision de l'orientation dépendant alors de l'écartement des marqueurs, et une fréquence de 100 Hz). Nous avons présenté section 3.3.1.a page 41 la plateforme utilisée à des fins de tests pour les séquences MOCAP_LOOP et MOCAP_FAST. L'utilité principale d'un système de capture de mouvement pour une application de localisation est de faire office de vérité terrain, et n'est pas réaliste à être intégrée dans un système de localisation puisqu'il s'agit d'un capteur déporté nécessitant une lourde infrastructure, mais nous avons tout de même implémenté son intégration dans le filtre pour faire des tests d'observabilité de certaines grandeurs et de convergence du filtre, comme expliqué section 5.3 page 94 pour l'estimation des biais de la centrale inertielle.

L'équation d'observation pour la partie position est donc identique à celle du GPS présentée équation 6.18 page 147, et nous présentons ici l'équation d'observation de la partie orientation seulement.

On a vu que l'on représente les orientations par des quaternions dans l'état, afin de ne pas avoir de valeur singulière dans la représentation des angles. Le degré de liberté supplémentaire est géré en assurant la normalisation du quaternion, ce qui ne pose pas de problème dans l'espace de l'état. En revanche lorsque l'on veut utiliser un quaternion en observation, le fait

que sa matrice de covariance soit de taille 4 mais de rang 3 et donc non inversible pose problème, puisque la matrice de covariance de l'innovation doit être inversée. En pratique on a rarement une erreur fatale car le déterminant calculé n'est pas parfaitement nul à cause d'erreurs numériques, mais le résultat de la correction est complètement erroné.

On doit donc utiliser en observation une représentation dont la covariance n'est pas singulière. Nous avons choisi les angles d'Euler, car il s'agit d'une représentation commune que nous manipulons déjà. Les angles d'Euler sont bien sûr quant à eux toujours victimes d'une singularité pour des attitudes proches de $\pm\pi$, mais elles ne sont en pratique jamais atteintes, et surtout les capteurs renvoient quasiment toujours leurs données sous forme d'angle d'Euler donc ce n'est pas une contrainte supplémentaire. Si on souhaitait tout de même traiter ce problème, il suffirait de changer de représentation dans ces cas singuliers, par exemple en utilisant un vecteur rotation (qui lui serait singulier pour des angles proches de zéro).

En revanche il est indispensable lors du calcul de l'innovation avec les angles d'Euler, comme différence de la mesure et de la mesure attendue, de réaliser cette différence modulo 2π.

En considérant que les mesures sont celles de l'orientation d'un capteur dont la transformation par rapport au repère du SLAM est constituée d'une translation et d'un quaternion $(\mathbf{t_m}, \mathbf{q_m})$, on a alors pour équation d'observation de l'orientation $\tilde{e}_m$:

$$\hat{e}_m = \text{q2e}(\mathbf{q} \otimes \mathbf{q_m}) \tag{6.20}$$

où $\otimes$ représente le produit de quaternion, q2e l'opération de conversion de quaternion à euler, et $\mathbf{q}$ toujours l'orientation dans l'état du robot.

6.4.2 Observation de faisceaux

On appelle rayon une demi-droite définissant une direction depuis un point de vue, et faisceau (*bundle* en anglais) un rayon assorti d'une incertitude angulaire sur cette direction (que l'on peut donc se représenter par un cône). L'observation d'un faisceau est donc adaptée à l'observation d'un amer dans un capteur angulaire pur tel qu'une caméra. Elle peut être exploitée dans deux contextes :

— Un observateur, fixe ou mobile, connaissant sa position, observe la direction du robot par rapport à lui, et lui transmet l'information (s'il observe également la distance on peut en déduire la position complète du robot et utiliser les équations du GPS ; s'il observe également l'orientation du robot on peut utiliser les équations de la capture de mouvement). Un exemple d'application pratique est l'observation avec une caméra par un robot aérien bénéficiant d'une très bonne localisation du fait de sa situation privilégiée pour la réception de signaux (signaux GPS, signaux de balises de localisation, signaux de communication pour des corrections, etc).

— Le robot observe un amer ponctuel dont il connait la position absolue a priori. Cela est en fait similaire à l'observation d'un amer en SLAM, sauf que l'on connait déjà la position de l'amer et qu'on ne cherche pas à l'estimer.

On s'abstrait ici du modèle du capteur qui a permis l'observation du rayon, qui a déjà été traité section 4.2 page 55 pour les caméras par exemple, et on se place à un niveau plus générique

où une mesure est constituée d'un point et d'un vecteur direction : $(\mathbf{r_r}, \tilde{\mathbf{u}}_\mathbf{r})$. En réalité il est inutile de considérer que le point $\mathbf{r_r}$ fait partie de la mesure, puisqu'il est constant (dans le sens connu, même doté d'une incertitude), et ne fait donc pas partie de la fonction d'observation. Par convention, on décide également que le vecteur direction $\mathbf{u_r}$ doit être orienté du robot vers le point distant (il suffit de mettre le bon signe sur la mesure dans les deux cas d'utilisation considérés).

On a donc pour équation d'observation :

$$\hat{\mathbf{u}}_\mathbf{r} = (\mathbf{r_r} - \mathbf{p}) / \|\mathbf{r_r} - \mathbf{p}\| \tag{6.21}$$

Ce modèle a été implémenté dans RT-SLAM, et le travail d'intégration pour un hélicoptère qui observe le robot au sol dans une procédure de relocalisation est en cours. La principale difficulté se trouve au niveau de la communication entre les deux véhicules. En effet idéalement une série de telles mesures alors que les deux robots sont en mouvement permet d'observer l'ensemble des degrés de liberté de la localisation (position et orientation complètes), les latences de traitement et de communication étant gérées par le filtre avec le mécanisme d'extrapolation présenté section 6.6.1 page 165. Cependant l'architecture multirobots de communication utilisée ne permet pour le moment pas de communiquer en direct avec de faibles latences. Dans un premier temps nous effectuons et intégrons donc ces mesures avec le robot au sol exécutant RT-SLAM à l'arrêt, ce qui élimine les problèmes temporels, l'inconvénient étant que l'orientation n'est plus observable. Il est bien sûr possible de rendre les observations plus complètes, en estimant la distance et le cap à partir de la taille et de l'orientation perçues, mais ces grandeurs sont délicates à mesurer précisément, comme cela est évoqué dans l'annexe B page 197. Une autre difficulté à résoudre avant de pouvoir présenter des résultats utilisables est l'imprécision de l'estimation de l'orientation de l'hélicoptère, qui doit être résolue par l'intégration de RT-SLAM sur l'hélicoptère. Enfin à terme cette fonction ne devrait plus être intégrée directement dans le filtre, mais au sein de l'architecture de SLAM hiérarchique multirobots présentée section 7.4 page 184.

6.5 Multi-caméras

L'utilisation d'autres capteurs peut également simplement passer par l'ajout d'autres caméras. Comme on l'a déjà évoqué, si les champs de vue de deux caméras se recouvrent et que leur position relative est connue, on peut observer directement la profondeur de certains amers, ce qui permet de gagner en précision, en robustesse, et apporte l'information de facteur d'échelle. Plusieurs approches différentes sont cependant disponibles pour intégrer les caméras supplémentaires.

6.5.1 Stéréovision

L'approche la plus ancienne et la plus classique est de faire de la stéréovision. Cela consiste à synchroniser les deux caméras et les considérer comme un capteur unique qui produit des paires d'images synchrones. On cherche alors tous les amers dans chaque paire d'images synchrones,

et on déduit pour ceux trouvés dans les deux images leur position euclidienne (x, y, z) qui est ensuite directement observée dans le filtre.

On peut donc utiliser directement une paramétrisation euclidienne, l'espace de l'état et de mesure sont confondus, et l'équation d'observation est alors trivialement la fonction identité.

6.5.2 Bicam

Le bicam, proposé par [Solà *et al.*, 2007], consiste au contraire à considérer que l'on a deux capteurs complètement indépendants, mais partageant les amers de la carte, et cherchant à les observer sans savoir quel capteur les a initialisés.

Les avantages sont nombreux :

- aucun besoin de synchronisation entre les différentes caméras, même si on peut tout de même vouloir contrôler la répartition temporelle des images.
- possibilité d'exploiter des amers très éloignés, dont l'incertitude en profondeur serait trop grande pour être utilisés par la stéréovision, mais dont l'exploitation permet une excellente observabilité des angles.
- possibilité d'exploiter des amers masqués dans une des caméras.
- possibilité d'exploiter des amers hors du champ de vue d'une des caméras, et laisse donc plus de liberté sur le positionnement des caméras ou le type des caméras (champs de vue très différents par exemple).
- possibilité d'utiliser souplement et génériquement plus de deux caméras.
- possibilité d'estimer en ligne le calibrage extrinsèque entre les caméras.
- inutilité de rectifier les images.

a Synchronisation des caméras

Une première possibilité est bien sûr de mimer le comportement de la stéréovision, et de synchroniser les caméras, afin d'avoir une observabilité parfaite des amers (proches) à chaque instant.

Mais comme on vient de le souligner, cela n'est plus obligatoire, et le système s'adpatera automatiquement si les images sont désynchronisées, en continuant à faire des prédictions entre les images avec un modèle de mouvement ou un capteur inertiel.

On peut alors envisager au contraire de forcer les deux caméras à être déphasées d'une demi-période, ce que l'on appellera *bicam alterné*. L'avantage est que l'on peut en traitant le même nombre d'images doubler la résolution temporelle, en regagnant les avantages d'une fréquence élevée, d'amélioration de la précision du point de linéarisation et de réduction du domaine d'application de la linéarisation. L'inconvénient est que l'on perd l'avantage de la rigidité de la position relative des deux caméras qui peut être estimée très précisément avec un calibrage stéréo hors ligne.

Nous allons donc comparer les deux approches dans les résultats.

La figure 6.9 résume les différentes approches que l'on peut avoir pour l'intégration des images de deux caméras. On ne considèrera que les modes à 25 Hz qui utilisent des ressources équivalentes au monoculaire à 50 Hz, facilitant la comparaison. La bande passante du bus de communication avec les caméras et le temps de calcul peuvent également être des motifs de se limiter à 25 Hz, mais il est possible dans certains cas d'acquérir les images de deux caméras à 50 Hz, et en traitant moins de points par image le temps de calcul peut être équivalent, donc cela est parfaitement envisageable.

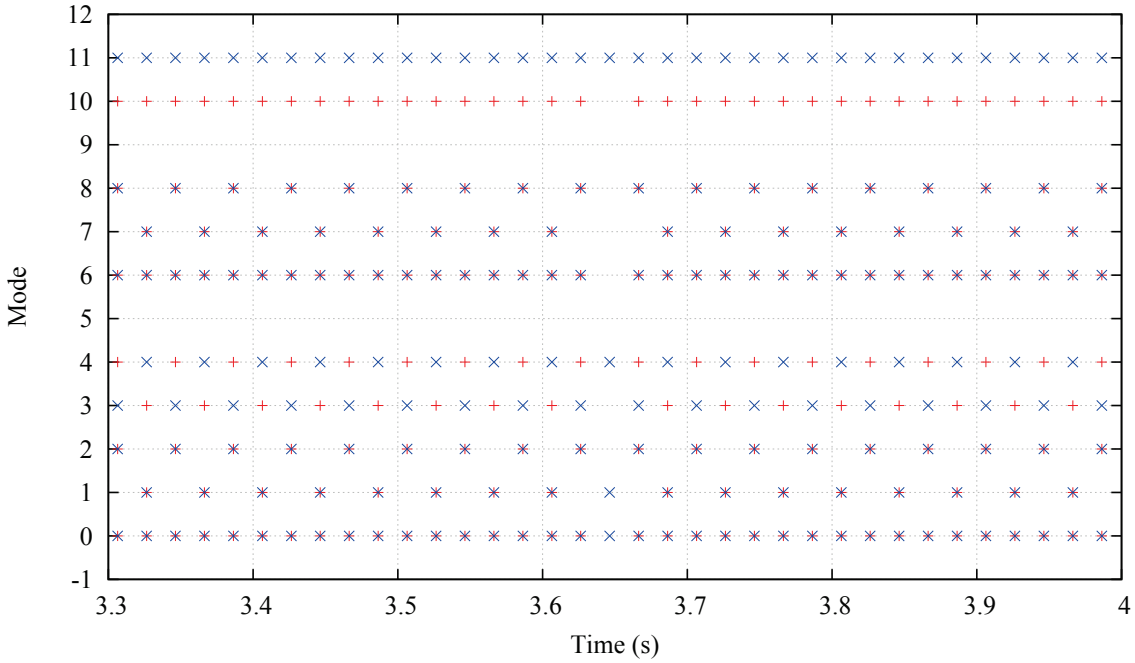


FIGURE 6.9 – Illustration des dates des images utilisées dans différents modes. 0 : bicam à 50 Hz ; 1 et 2 : bicam synchronisé à 25 Hz ; 3 et 4 : bicam alterné à 25 Hz ; 6 : stéréo à 50 Hz ; 7 et 8 : stéréo à 25 Hz ; 10 et 11 : mono à 50 Hz. On constate au temps 3.65 les effets d'une image manquante, en particulier la stéréo qui ne fera rien à cette instant, alors que le bicam pourra exploiter l'autre image.

b Choix des caméras

Cette méthode d'intégration de plusieurs caméras laisse une très grande liberté sur le type de caméras à utiliser, leur positionnement, et leur nombre. La prise en compte des modèles de chaque caméra et la prédiction d'apparence des amers avec transformation homographique (section 4.4.5.b page 72) permet en effet d'apparier des amers entre une caméra perspective et une caméra panoramique catadioptrique ou *fisheye*. Cela ouvre la voie à la recherche du meilleur compromis entre champs de vue et précision, stabilité et observabilité du mouvement, tout en conservant une réalisation simple.

Ainsi on aura une meilleure observabilité du mouvement et donc une meilleure précision locale en observant des amers proches, et avec une bonne résolution, par exemple en utilisant une caméra perspective plutôt orientée vers le sol. Au contraire on pourra suivre les amers plus longtemps, et les retrouver plus facilement en cas de fermeture de boucle, si l'on suit des amers loins et dans un grand champ de vue, ce qui donnera une meilleure précision à long terme en

limitant la dérive.

On peut aussi envisager avantageusement la combinaison d'une caméra panoramique et d'une caméra perspective orientée vers le sol, avec la possibilité selon les capacités de calcul de doubler la caméra panoramique (idéalement les deux caméras étant à la verticale l'une de l'autre) et/ou doubler la caméra perspective afin d'améliorer la convergence des amers lointains ou proches. Dans le cas où l'on n'utilise que deux caméras perspectives, on pourrait avoir intérêt à en orienter une plutôt vers le sol, et l'autre plus vers l'horizon, afin de conserver un champ de vue commun à mi-distance permettant d'initialiser rapidement des amers, mais également de suivre des amers très proches sur le sol pour améliorer la précision locale, tout en suivant plus longtemps des amers éloignés pour améliorer la précision globale.

Nous n'avons malheureusement pas pu tester ces possibilités, par manque de temps et n'ayant à disposition que des caméras de type *fisheye*, qui n'ayant un champ de vue que de quelques degrés au dessus de 180° ne permettent pas d'observer l'horizon de façon robuste aux changements d'attitude du robot, ce qui est pourtant l'intérêt d'une caméra panoramique. Les caméras panoramiques catadioptriques semblent donc plus adaptée, mais seuls certains modèles avec une bonne tenue mécanique peuvent conserver un calibrage fiable sur un robot en mouvement sur des terrains parfois accidentés.

6.5.3 Aspects d'implémentation

a Zones de recherche

Les régions d'intérêt utilisées par la vision jusqu'à maintenant pour décrire les zones de recherche ne posaient pas particulièrement de problème, car elles étaient de taille réduite et plutôt symétriques et donc étaient correctement modélisables par des ellipses.

En multi-caméras en revanche on se retrouve en permanence confrontés à des amers qui ont été initialisés dans une caméra, et sont recherchés dans une autre, et qui ont donc une ellipse de recherche très fine et allongée, correspondant à la ligne épipolaire linéarisée relative aux deux caméras).

Le problème est qu'en présence de distortion dans l'image (avec le cas extrême des caméras omnidirectionnelles), la ligne épipolaire n'est pas une droite, et la linéarisation qu'est l'ellipse n'est valable que localement. Le point que l'on recherche pourra alors souvent être en dehors de l'ellipse de recherche, ce qui fait qu'on ne le trouvera pas (et il serait de toute façon incompatible, et non corrigeable avec la jacobienne trop fausse issue de la prédiction). Ce phénomène est illustré figure 6.10 page suivante.

On ne souhaite pas rectifier entièrement les images (c'est-à-dire corriger la distortion pour rendre les lignes épipolaires droites, et corriger l'alignement pour rendre les droites épipolaires horizontales) car cela contreviendrait au principe de la recherche active dont l'esprit est d'éviter les traitements sur l'image entière afin de pouvoir fonctionner à haute fréquence. De plus rectifier les images empêcherait un positionnement libre des caméras, ne fonctionnerait pas pour plus de deux caméras, et ne fonctionnerait pas pour des caméras omnidirectionnelles.

La solution adoptée est de linéariser par morceaux la ligne épipolaire, en faisant différentes hy-

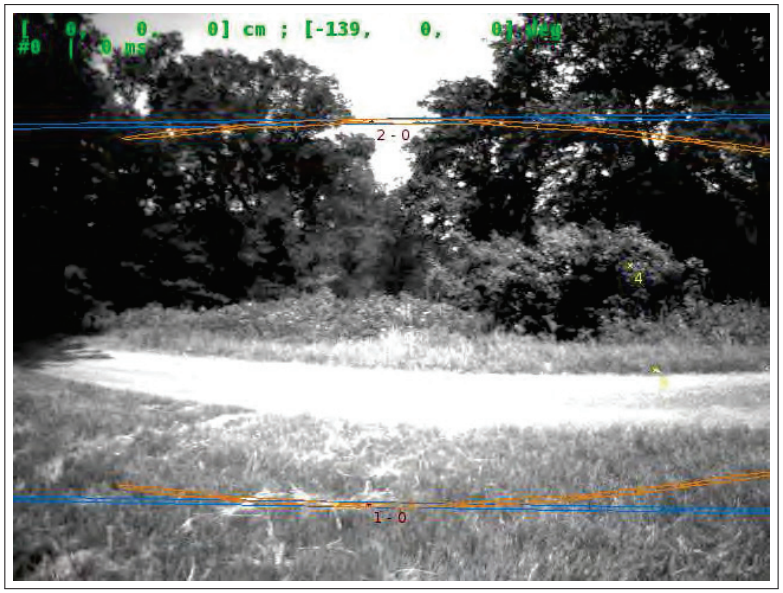


FIGURE 6.10 – Comparaison de zones de recherche constituées d'une ellipse linéarisée (en bleu), et de zones de recherche autour de la vraie ligne épipolaire en tenant compte de la distortion de la caméra (en orange).

pothèses sur la profondeur de l'amer nouvellement initialisé. En pratique on effectue plusieurs tests incrémentaux, de plus en plus précis et coûteux, pour vérifier la nécessité de décomposer la ligne épipolaire en plusieurs morceaux, puis la décomposer progressivement, avant de rechercher l'amer dans l'ensemble de ces hypothèses et le mettre à jour avec l'hypothèse la plus proche :

- Si l'amer n'a pas une modélisation en distance inverse, on ne fait rien de particulier.
- Sinon si l'incertitude de la mesure attendue ne dépasse pas un certain seuil très bas, on ne fait rien de particulier.
- Sinon si son asymétrie est trop faible, on ne fait rien de particulier.
- Sinon on décompose la partie inconnue en plusieurs hypothèses jusqu'à ce que la ligne épipolaire soit suffisamment linéaire sur le domaine de chaque hypothèse (si l'amer n'a pas été observé depuis la dernière décomposition on garde les mêmes points) :
  - On démarre avec une hypothèse, d'espérance et incertitude habituelles.
  - On vérifie si ses extrémités sont visibles dans l'image. Si ce n'est pas le cas, on cherche les limites sur la partie non observable de l'état pour que l'amer soit visible, et on ne cherche à couvrir avec les hypothèses que cette zone (cela est indispensable car le modèle de distortion n'étant pas valide en dehors de l'image, on ne doit jamais en sortir, et sans en tenir compte on peut avoir des hypothèses centrées – donc linéarisées – en dehors de l'image mais qui sont seules à couvrir une zone à l'intérieur de l'image). On effectue cette recherche numériquement par dichotomie, car il n'existe pas de formule analytique, et on ne peut pas utiliser les jacobiennes en dehors de l'image qui sont fausses pour faire une recherche numérique plus efficace.
  - Pour chaque hypothèse, on vérifie si la projection de ses extrémités est compatible avec les extrémités de sa projection. Autrement dit, on projette l'amer avec les

distances extrêmes de l'hypothèse (à 2.5σ pour un niveau de confiance de 99% par exemple), et on calcule la distance de mahalanobis de ces projections avec la projection de l'espérance de l'hypothèse et son incertitude associée. Si elle est inférieure à 3σ (pour un niveau de confiance de 99% en dimension 2), cela signifie que la zone de recherche permettra de les trouver et qu'elles passeront le contrôle de compatibilité, donc on considère que le modèle est suffisant et on arrête la décomposition (notons que si le test réussit avec l'hypothèse unique initiale, on n'a alors pas de décomposition).

— Sinon on augmente le nombre d'hypothèses d'une unité, et on rééchantillonne les n hypothèses. On cherche à obtenir une répartition uniforme dans l'espace de l'image, car la non linéarité vient de la distortion dans cette espace. Mais comme la disparité varie proportionnellement à l'inverse de la distance, cela revient à chercher une répartition uniforme dans l'espace de l'état sur la distance inverse. Si on note $\mathcal{N}(\rho_0, \sigma_0)$ l'hypothèse globale initiale, $\rho_{\min}$ et $\rho_{\max}$ les distances inverses minimale et maximale visibles dans l'image, et que l'on choisit de créer des hypothèses à $k \cdot \sigma$, on aura donc les n hypothèses suivantes (illustrées figure 6.11 page suivante) :

$$\left(\mathcal{N}\left(\rho_{n,i}\,,\,\sigma_n\right)\right)_{0 \leq i < n}$$

$$\text{avec} \quad \rho_{n,i} = (t + 2{\cdot}i{\cdot}k) \cdot \sigma_n \quad \text{et} \quad \sigma_n = \frac{(t+s)}{2{\cdot}n{\cdot}k - (k{-}t) - (k{-}s)} {\cdot} \sigma_0$$

$$\text{où} \quad t = \frac{\rho_0 - \rho_{\min}}{\sigma_0} \quad \text{et} \quad s = \frac{\rho_{\max} - \rho_0}{\sigma_0}$$

— On répète ensuite ce processus jusqu'à ce qu'il soit inutile d'ajouter des hypothèses. Notons qu'on pourra prendre soin de ne pas ajouter d'hypothèses de distance inverse de moyenne négatives, ni tester d'extrémité négative (borner à 0).

— Enfin on recherche indépendamment l'amer dans l'ellipse d'incertitude de chaque hypothèse et on teste sa compatibilité avec l'hypothèse, puis on corrige l'amer avec l'hypothèse la plus proche, au sens de la distance de Mahalanobis (c'est-à-dire que l'on corrige avec la jacobienne et l'espérance de l'hypothèse, correspondant à la linéarisation qu'elle effectue).

Le principe est similaire à une initialisation retardée, sauf que l'on conserve tous les avantages de l'initialisation immédiate.

On notera qu'il est important lorsque l'on modifie la distance inverse d'un amer, que ce soit pour une projection ou une correction, de bien mettre à jour la covariance en cohérence avec l'opération de modificaton, comme on le fait pour toute modification de l'état du filtre. Pour une affectation qui ne dépend pas du reste de l'état, cela revient à annuler toute la colonne de l'état correspondant.

La figure 6.12 page 161 montre un exemple où la construction d'hypothèses multiples a permis de retrouver un amer qui était sinon en dehors de la zone de recherche issue d'une linéarisation unique.

On peut noter que le principe s'applique de façon transparente en mono-caméra aux amers qui n'ont pas été trouvés pendant un certain temps et dont l'ellipse de recherche devient donc

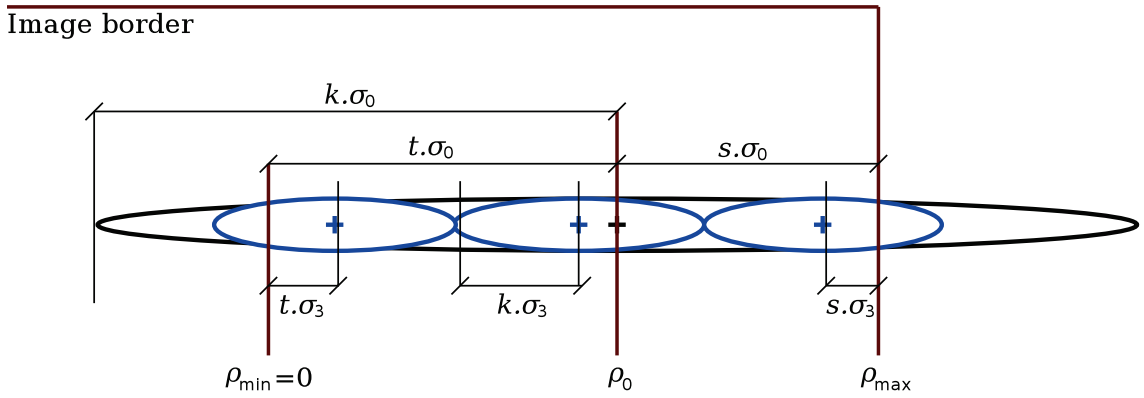


FIGURE 6.11 – Construction des hypothèses de profondeur. Ici ρ_{min} vaut 0 car on n'atteint pas le bord de l'image de ce côté. Remarquons que l'on conserve les ratios aux extrêmes : si ρ_{min} est situé à $t{\cdot}\sigma$ pour l'ellipse d'origine, il sera également à situé à $t \cdot \sigma$ pour l'hypothèse concernée, et de même pour ρ_{max}. Notons aussi que cet exemple est uniquement illustratif : il serait dans ce cas inutile de décomposer l'ellipse puisqu'il n'y a pas de non linéarité représentée pour simplifier et clarifier la représentation, et les les ellipses des hypothèses ne se recouvrent pas du tout car il s'agit d'une représentation dans l'espace des distances inverses pour illustrer leur construction, mais dans l'espace des images elles le font à cause des autres sources d'incertitude.

très grande. On remarque dans ce cas (figure 6.13 page 162) que l'on obtient une zone de recherche non symétrique, puisque l'incertitude sur la localisation a plus d'effets sur les amers situés proches de la caméra que sur ceux lointains. De plus lorsque le mouvement comprend un changement de profondeur, la relation entre la distance inverse et la disparité n'est plus linéaire, et l'ellipse initiale ne recouvre plus les distances proches souhaitées. Dans tous ces cas la construction d'hypothèses multiples améliore donc également le fonctionnement.

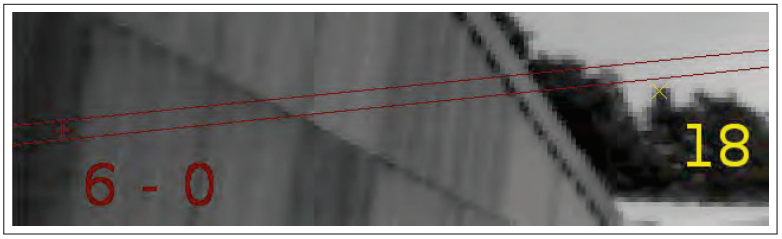
Ainsi par exemple sur les premières images de la séquence ESPERCE lorsque le robot est à l'arrêt, on initialise en moyenne 30 amers ($\sigma = 3.2$, 192 exécutions) avec les hypothèses multiples de profondeur, contre seulement 16 amers ($\sigma = 2.7$, 192 exécutions) sans, soit quasiment le double.

b Calibrage extrinsèque

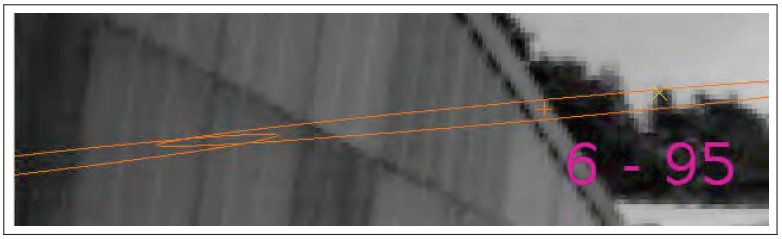
Il est possible de calibrer précisément la position relative des deux caméras en même temps que le calibrage de leurs paramètres extrinsèques, avec une mire de calibrage et une procédure d'optimisation, comme vu section 4.2.3.a page 57 (procédure de calibrage de stéréovision).

On peut également estimer en ligne ce calibrage extrinsèque, comme vu section 5.4.1 page 105. Cela fonctionne même beaucoup mieux car les paramètres définissant la position relative entre les deux caméras sont beaucoup plus sensibles que ceux définissant la position relative entre la centrale inertielle et la caméra (la disparité mesure précisément la distance, et la ligne épipolaire les orientations relatives), et donc plus observables. On le remarque sur la figure 6.14 page 162 où les incertitudes deviennent extrêmement réduites. Les translation convergent également, mais elles sont très inconsistentes temporellement, et convergent parfois vers des valeurs surprenantes, comme le 12 mm en x ici, qui semble élevée. Utiliser sa valeur a de plus plutôt tendance à dégrader les résultats que les améliorer.

(a) Initialisation d'un point dans la caméra gauche.



(b) Recherche de ce point dans la caméra droite, avec une ellipse d'incertitude classique. Le point étant à l'infini, et le système envisageant aussi que le point soit très proche de la caméra, la prédiction est éloignée, et la linéarisation n'est plus valide là où se trouve réellement l'amer. On remarquera que le système en profite pour initialiser un nouvel amer au même endroit.



(c) Recherche du point dans la caméra droite avec décomposition en deux hypothèses de profondeur, permettant de retrouver l'amer. La correction se fait de plus avec la linéarisation et la mesure attendue de cette hypothèse, plus proche que celle de l'hypothèse unique.

FIGURE 6.12 – Comparaison de la recherche d'un amer avec hypothèse simple ou hypothèses multiples.

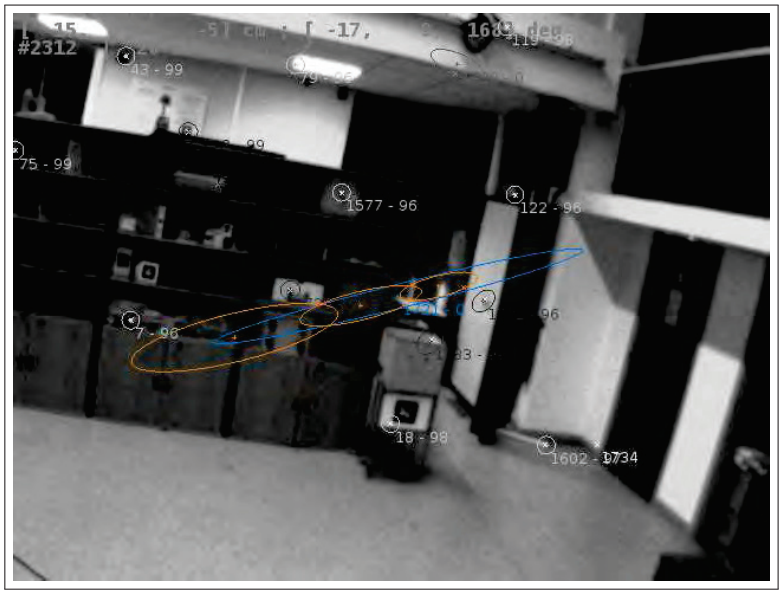


FIGURE 6.13 – Exemple de décomposition en hypothèses multiples (orange) en mono-caméra, par comparaison à une hypothèse unique (bleu).

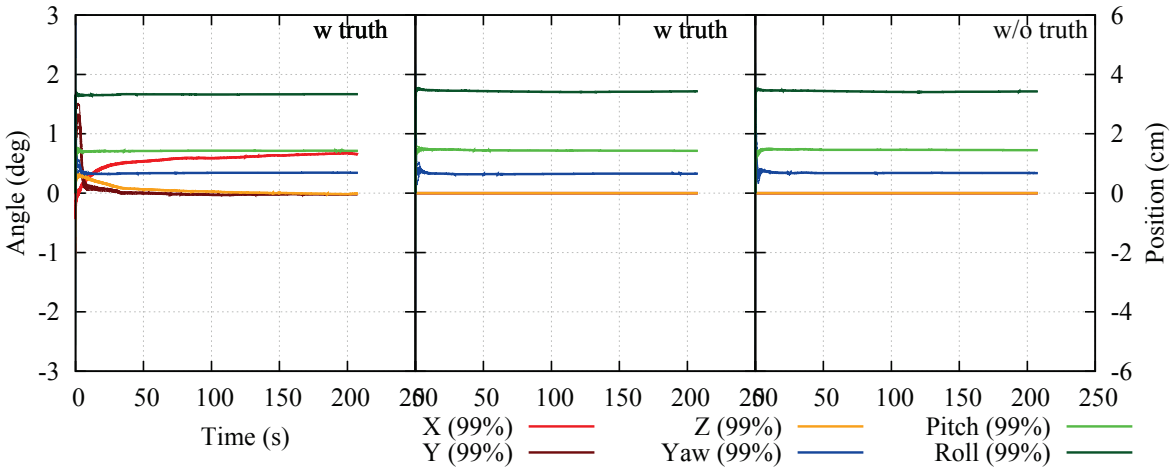


FIGURE 6.14 – Résultat de l'estimation du calibrage extrinsèque entre la caméra droite et la caméra gauche sur la séquence ESPERCE (différence avec l'état initial $(0.333, -0.1936, 0.305), (-90, 0, -90)$, incertitude initiale de 1 cm et 1 °). De gauche à droite : translations et angles estimés avec vérité terrain GPS ; angles estimés avec vérité terrain GPS ; angles estimés sans vérité terrain GPS.

Pour les angles, au contraire le calibrage réalisé par estimation est d'aussi bonne qualité voire meilleur que celui réalisé avec une mire de calibrage, auquel la plus grande attention a été pourtant été portée (utilisation de la *Camera Calibration Toolbox* de Jean-Yves Bouguet sous Matlab, avec examen minutieux des erreurs résiduelles pour éliminer les points imprécisément extraits). Les corrections apportées sur les orientations de la caméra ne sont que de l'ordre du dixième de degré, mais la précision de la convergence et des incertitudes de l'estimation est de l'ordre du centième de degré, et comme on peut le constater figure 6.15 cela a un impact significatif sur la précision de la trajectoire estimée (on a une erreur RMS en position de 0.6 m au lieu 1.0 m, et en orientation de 0.4 °au leu de 0.6 °).

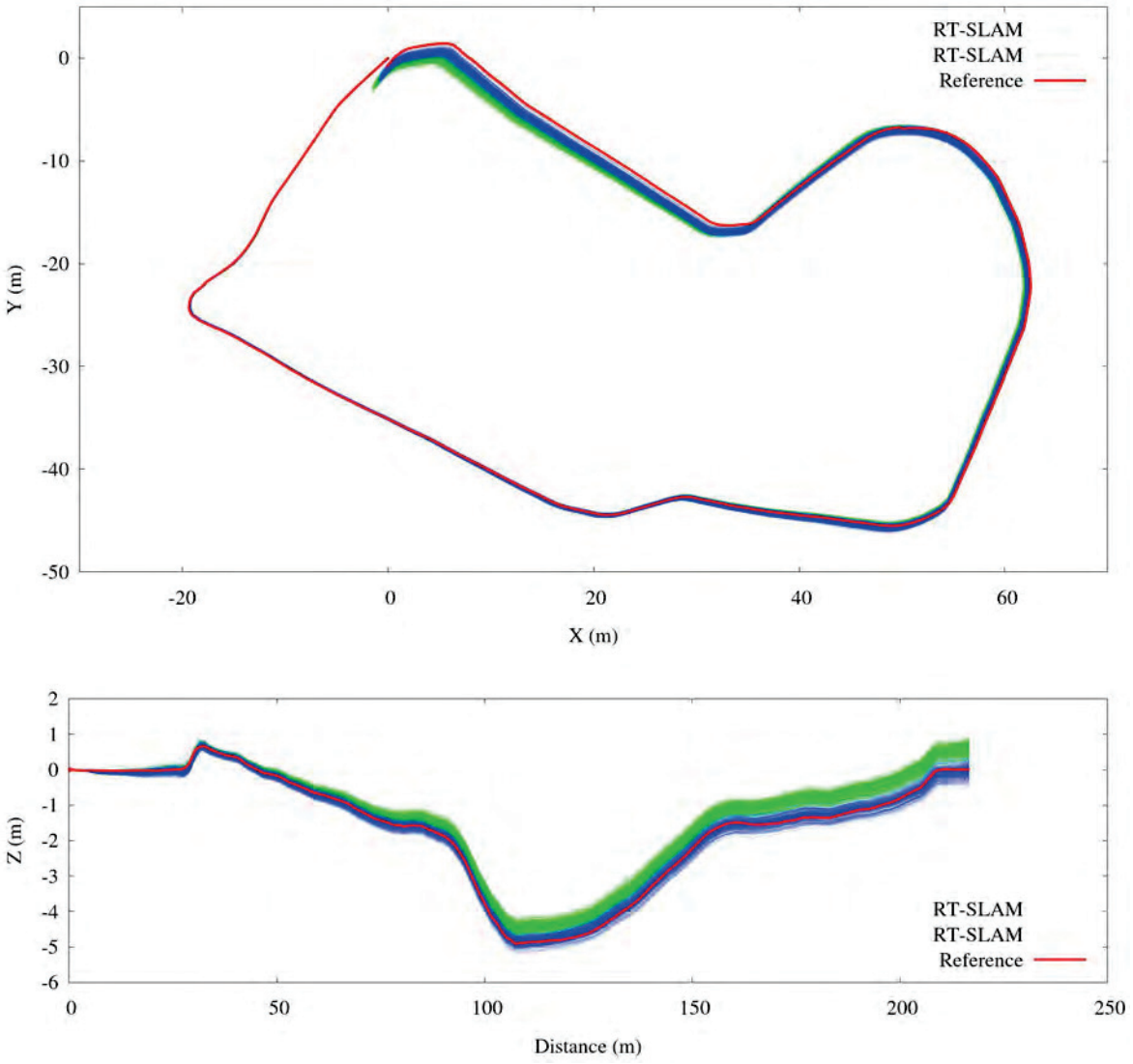


FIGURE 6.15 – Comparaison sur la séquence ESPERCE du slam visuel-inertiel bicam avec calibrage extrinsèque issu de l'estimation (en bleu, correspondant aux angles $(-88.29, 0.72, -89.67)$) ou d'un calibrage standard avec mire (en vert, correspondant aux angles $(-88.12, 0.71, -89.59)$).

De même l'estimation en ligne sans vérité terrain converge de la même façon comme on peut le voir figure 6.14 , à part au tout début où un domaine de valeur nettement plus grand est

exploré (l'erreur par rapport à l'estimation avec vérité terrain est de l'ordre du centième de degré). Cela signifie qu'il est tout à fait possible d'estimer ce calibrage entièrement en ligne (mais il est encore une fois plus précis de démarrer directement avec un calibrage correct, et moins coûteux de ne pas l'estimer en permanence s'il ne change pas).

Notre système ne permet cependant pas de définir la position de la caméra droite par rapport à celle de la caméra gauche (les positions des deux caméras sont toutes les deux nécessairement définies par rapport à l'origine du robot). Afin de pouvoir évaluer la précision du calibrage extrinsèque entre les deux caméras, il est donc préférable de procéder en deux temps : calibrer la position relative de la caméra gauche par rapport au robot, la fixer, puis faire de même avec la caméra droite. Si on estime les deux simultanément, les deux calibrages extrinsèques seront touchés par l'observabilité moindre de la position relative au robot.

c Recherche des amers

La recherche des amers se déroulant en général dans des zones de recherche très réduites, on a tendance à baisser les seuils de corrélation pour suivre les amers plus longtemps, sans que cela ne pose de problème de fiabilité. Lorsque les zones de recherche deviennent beaucoup plus grandes comme c'est le cas avec plusieurs caméras, le risque de faux appariemment augmente beaucoup. Ceci n'est pas très grave en soi puisque cela ne représente que la perte de l'amer (initialisé à une profondeur incorrecte, il sera perdu dès que l'on va se déplacer et être supprimé, sans avoir eu le temps d'injecter de la fausse information dans le reste du filtre), mais il est tout de même préférable de limiter leur nombre.

La première solution simple que nous avons implémentée est donc d'exiger un seuil de corrélation nettement plus haut quand les ellipses sont grandes.

Une autre solution classique et peu complexe est de conserver le deuxième maximum de corrélation dans la zone de recherche et de n'accepter l'appariemment que s'il n'y a pas d'ambiguité (le second maximum ne dépasse pas le seuil, ou l'écart entre les deux premiers est suffisant).

On peut également augmenter un peu la taille de la fenêtre de corrélation, afin de réduire l'ambituïté, mais avec précaution car on augmente en même temps le changement d'apparence en fonction du point de vue pour les zones comportant des éléments à différentes profondeurs ou dont la normale n'est pas du tout parallèle au plan image.

Enfin une solution classique de l'appariemment entre deux images est l'appariemment inverse, application du « critère d'unicité » utilisé en stéréovision. Si une recherche dans l'image cible donne un point B comme meilleur candidat de correspondant du point A de l'image source, on vérifie inversement qu'une recherche dans l'image source du meilleur candidiat de correspondant du point B est le point A, sinon on rejette l'appariemment pour cause d'ambiguité. Ceci n'est pas directement applicable à notre cas puisque l'on fonctionne avec une petite imagette faisant office de descripteur, mais dans le cas multicaméras on peut cependant appliquer une méthode similaire avec la dernière image de la caméra ayant initialisé l'amer. Pour cela on initialiserait simplement un nouvel amer à la position du candidat (avec une grande incertitude de profondeur donc), que l'on reprojetterait dans l'image de l'autre caméra (peu importe si

elle est dans le passé, seule sa position compte) pour faire une recherche dedans, et vérifier que le meilleur candidat est compatible avec la projection de l'amer initialement recherché dans cette caméra. L'inconvénient serait de doubler le temps de calcul de l'appariemment, mais cela resterait raisonnable surtout en limitant la vérification aux points commençant à poser des doutes d'ambiguité (par rapport au score de corrélation et au score du second candidat). Cela nécessiterait cependant de modifier significativement l'architecture pour accéder aux autres images pour le traitement d'une image et n'a pas été implémenté.

6.5.4 Résultats

Les figures 6.16 à 6.18 page 166 montrent l'exécution du SLAM bicam inertiel. On constate une dispersion encore réduite par rapport au slam visuel-inertiel avec odométrie, et une précision encore accrue par rapport à l'odométrie : dérive moyenne de $6.2\,\mathrm{cm}/\sqrt{\mathrm{m}}$ (réduction de 25%) et erreur RMS de 0.59 m (divisée par 2). Il demeure cependant des biais dans le système. Si l'odométrie n'apportait rien à la précision angulaire, le bicam l'améliore légèrement, la dérive angulaire passant de 80 à $70\,\mathrm{mdeg}/\sqrt{\mathrm{m}}$, et l'erreur angulaire RMS de 0.5 à 0.4 °.

Le bicam alterné, dont les résultats sont illustrés figures 6.19 à 6.21 page 168, est quasiment identique en terme de précision (indécelable sur les graphiques, différence non significative sur la dérive moyenne de $6.3\,\mathrm{cm}/\sqrt{\mathrm{m}}$ et l'erreur RMS de 0.60 m). Cela signifie que l'observation du facteur d'échelle n'est pas pénalisé par le fait que les images des deux caméras ne sont pas simultanées. En revanche la fréquence doublée garantira une meilleure robustesse dans des conditions difficiles, notamment les très grandes dynamiques de mouvement.

L'ajout de l'odométrie a également une influence non décelable sur les résultats, mais sa présence contribue à renforcer la robustesse du système en cas de problème avec la vision, afin de stabiliser la centrale inertielle.

Enfin le système fonctionne avec un simple modèle de mouvement à vitesse constante, mais est peu robuste et commet régulièrement de très grandes fautes, et n'est donc pas utilisable en tant que tel, même si son comportement reste bien supérieur au monocam à modèle de mouvement (taux de divergence de seulement 3% contre 30%). Ces faiblesses sont dues aux ambiguités liées aux grandes zones de recherche des amers et à la dynamique du mouvement.

6.6 Aspects d'implémentation

6.6.1 Gestion temporelle de l'intégration des capteurs

Avec l'utilisation de plusieurs capteurs en observation, un nouveau problème se pose, lié à l'obligation d'intégrer les données chronologiquement dans le filtre. Si cela est très facile à garantir lors du traitement hors ligne des données, en choisissant à chaque fois la donnée la plus ancienne restant à traiter parmi tous les capteurs, le problème se corse pour le traitement en ligne des données, qui peuvent être disponibles avec des latences différentes selon les capteurs.

En effet on ne peut pas se contenter d'intégrer la donnée la plus vieille disponible, car on ne peut garantir que l'on ne va pas recevoir ultérieurement une donnée qui est plus vieille. Il faut

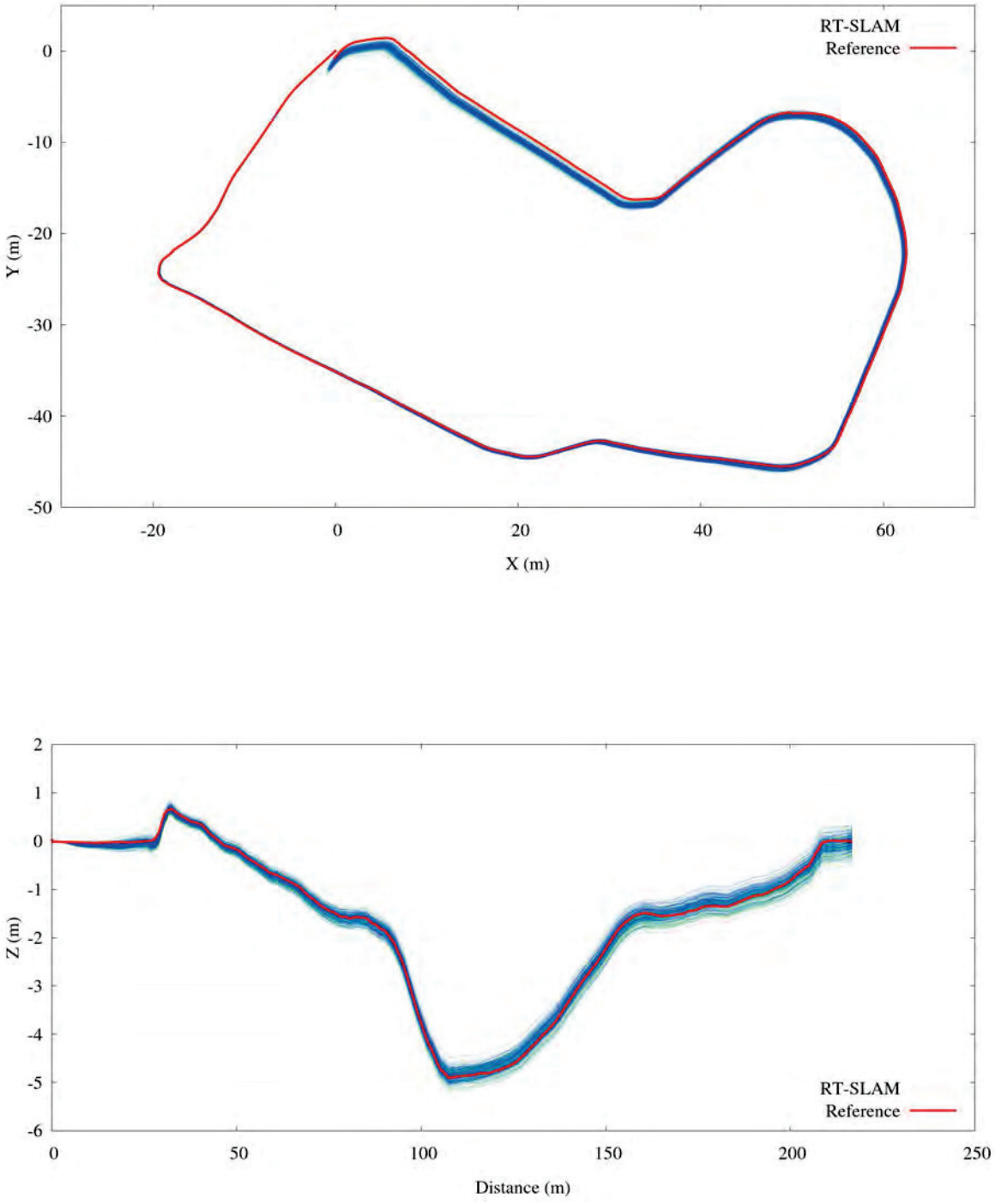


FIGURE 6.16 – Résultat de trajectoire du SLAM visuel-inertiel bicam sur la séquence ESPERCE.

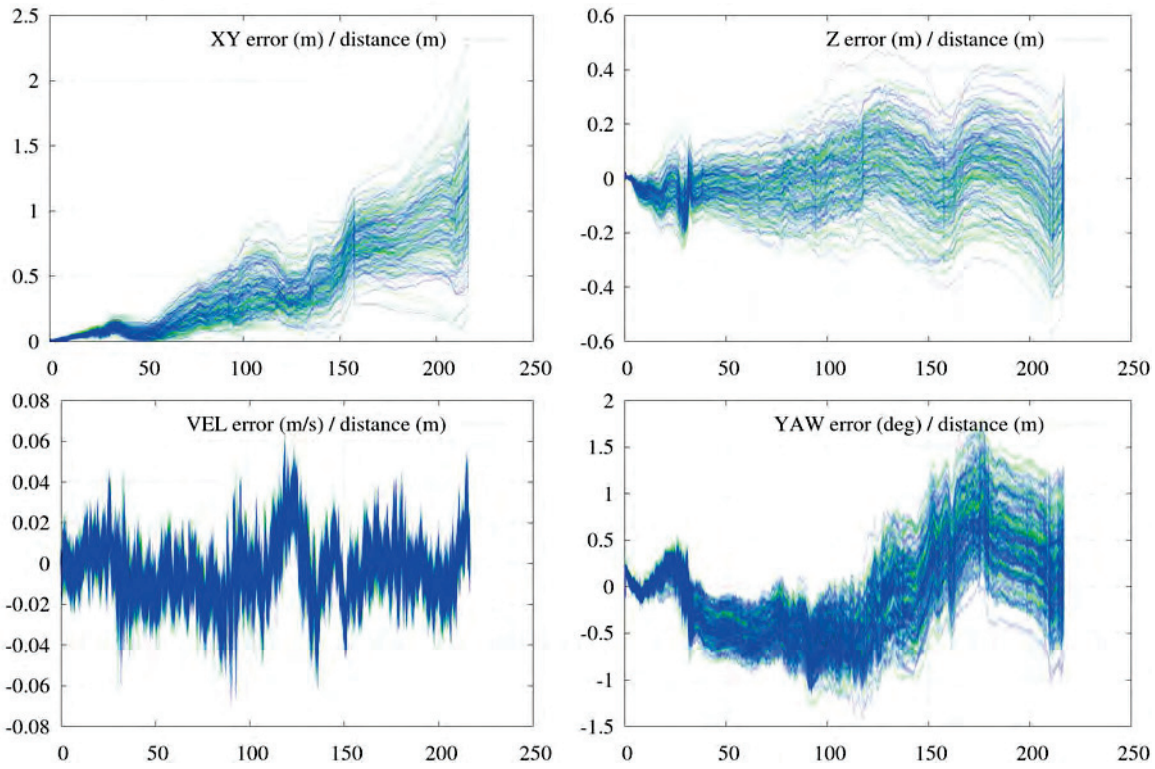


FIGURE 6.17 – Résultat d'erreur du SLAM visuel-inertiel bicam sur la séquence ESPERCE.

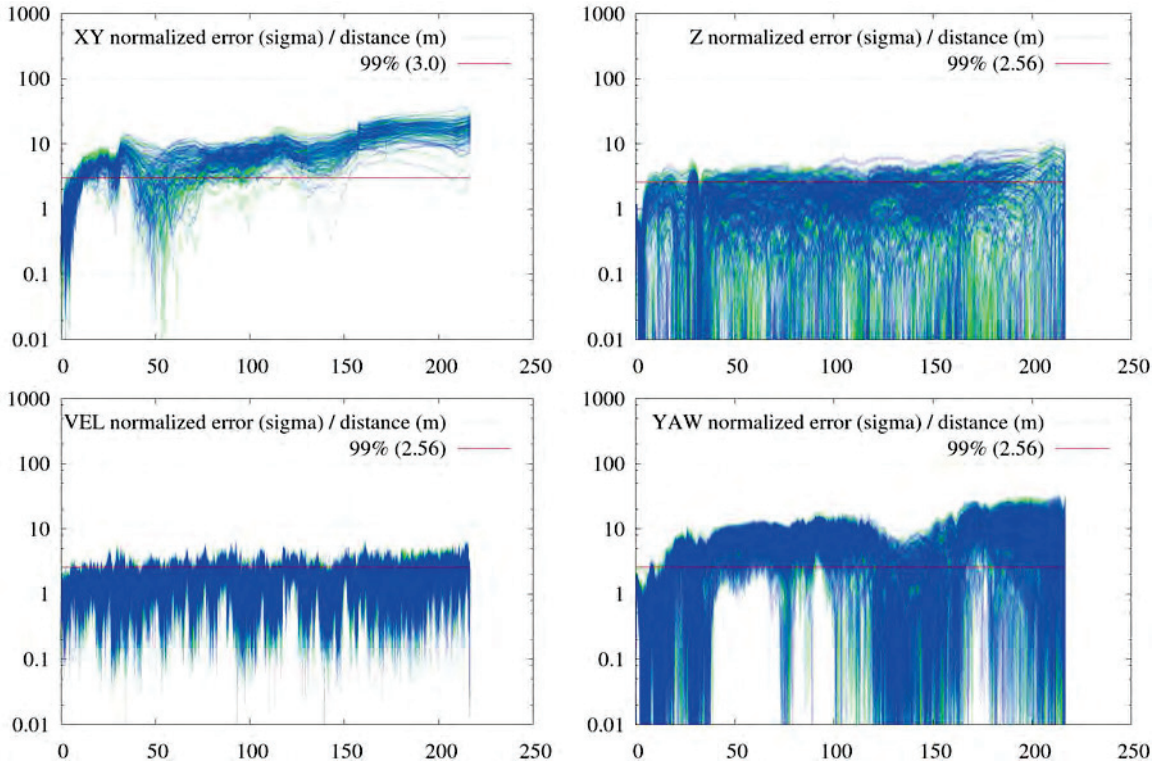


FIGURE 6.18 – Résultat d'erreur normalisée du SLAM visuel-inertiel bicam sur la séquence ESPERCE.

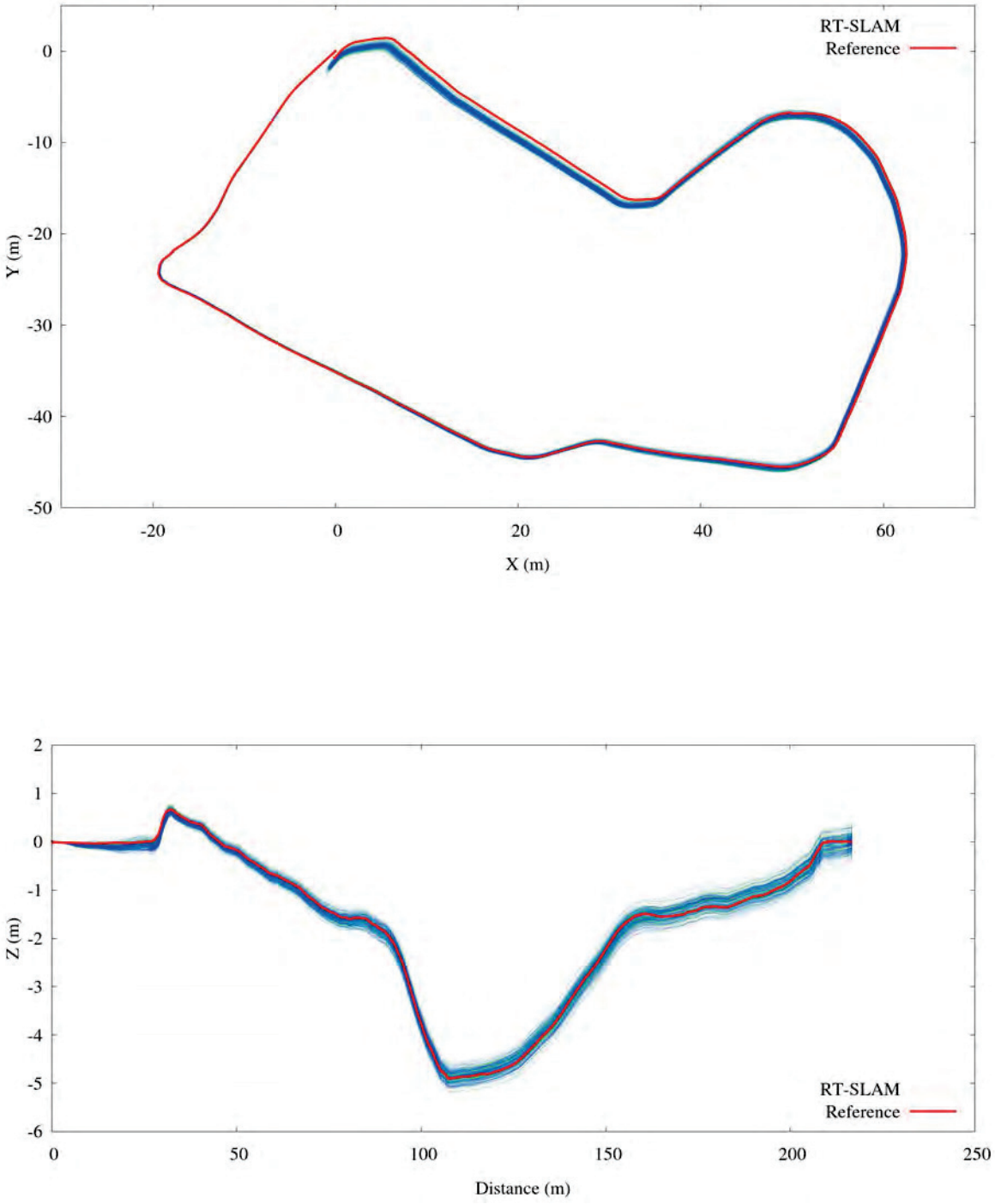


FIGURE 6.19 – Résultat de trajectoire du SLAM visuel-inertiel bicam alterné sur la séquence ESPERCE.

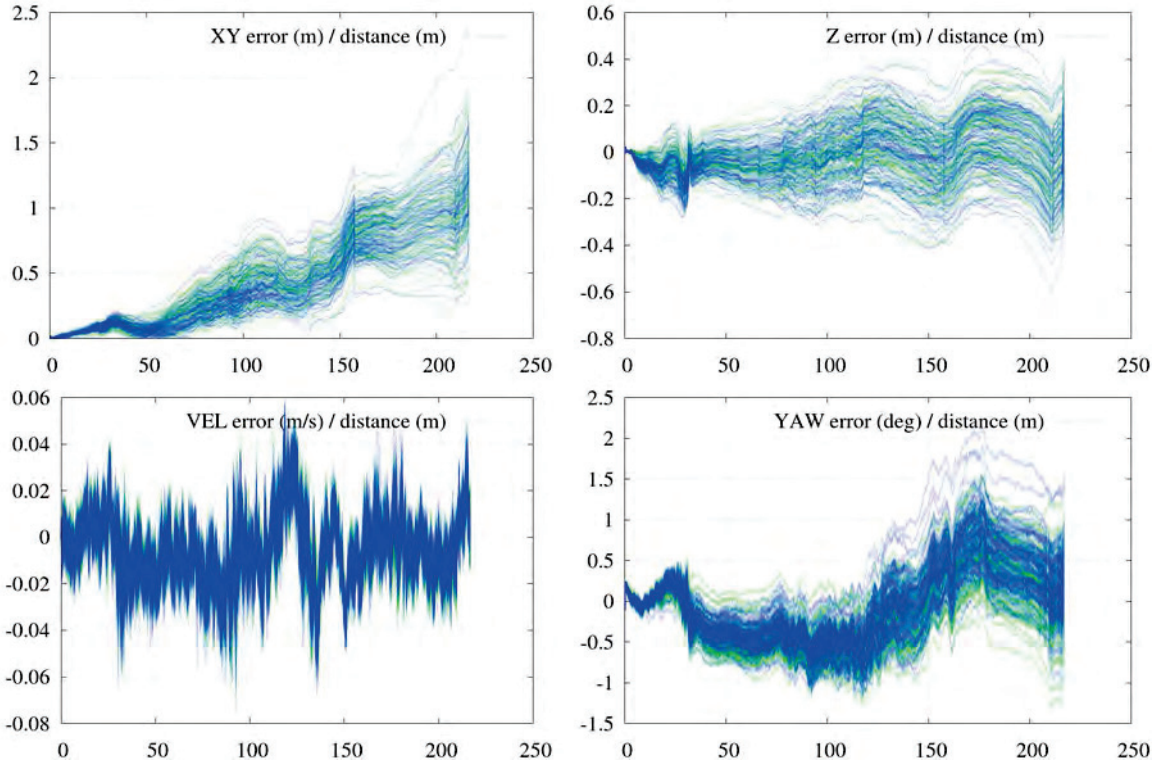


FIGURE 6.20 – Résultat d'erreur du SLAM visuel-inertiel bicam-alt sur la séquence ESPERCE.

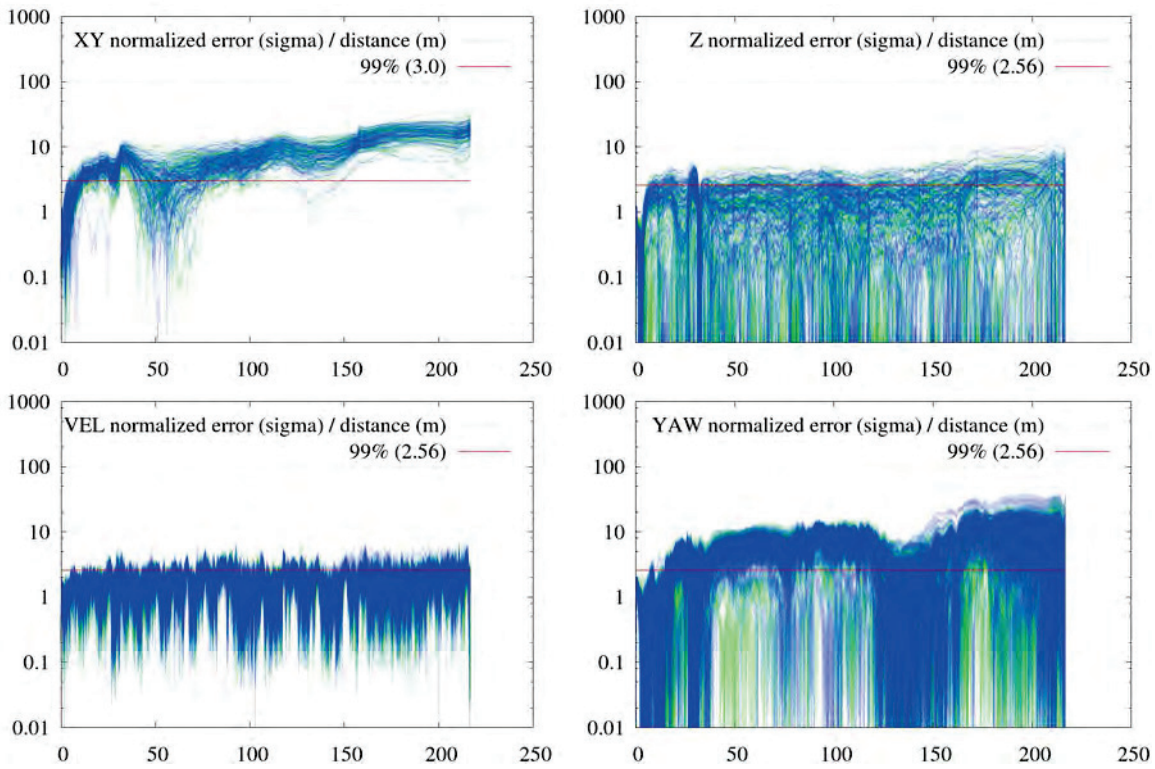


FIGURE 6.21 – Résultat d'erreur normalisée du SLAM visuel-inertiel bicam alterné sur la séquence ESPERCE.

donc considérer les dates d'arrivée attendues des données.

Par exemple soient deux capteurs s_1 et s_2, fonctionnant à la même fréquence f mais légèrement déphasés de δ (dates des données : $t_2(n) = t_1(n) + \delta$), et ayant des latences ϕ_1 et ϕ_2 (dates d'arrivée des données : $a_1(n) = t_1(n) + \phi_1$ et $a_2(n) = t_2(n) + \phi_2$) avec $\phi_1 - \phi_2 > \delta$. Si on intègre les données du capteur s_2 dès qu'on les reçoit, alors on ne pourra jamais intégrer les données du capteur s_1, plus anciennes mais reçues plus tard.

a L'algorithme

Nous avons implémenté un algorithme générique qui peut gérer un nombre arbitraire de capteurs en s'assurant que tous soient intégrés au mieux.

Il est basé sur le classement en deux catégories des capteurs, correspondant à différentes stratégies d'intégration souhaitables lors de la prise de retard dans l'intégration des données (c'est-à-dire que la donnée $n + 1$ est déjà disponible alors que la donnée n n'a pas encore pu être intégrée) :

— Rattraper le retard et intégration sans exception de toutes les données du capteur. On appliquera cette politique aux données peu coûteuses en temps à intégrer, comme les données proprioceptives. On qualifiera cette politique de « Toutes ».

— Abandonner l'intégration des données manquantes pour se consacrer aux plus récentes. On appliquera cette politique aux données lourdes à traiter, comme les images, pour éviter de prendre toujours plus de retard si on ne parvient pas à suivre le rythme. On qualifiera cette politique de « Dernières ».

Le principe général de l'algorithme est le suivant :

1. Pour chaque capteur on détermine quelle sera la date de la prochaine donnée et sa date d'arrivée prévue, par apprentissage de la fréquence et latence du capteur par expérience (en pratique la fréquence théorique est simplement calculée, et la latence utilisée est le minimum des latences des trois dernières données reçues augmentée d'une marge de 120%, afin d'accepter des capteurs qui sont lus de façon périodique et dont la latence peut varier du simple au double).
2. Afin d'être sûr de ne pas intégrer une donnée qui risque d'empêcher l'intégration de futures données à arriver, on détermine la date la plus basse de toutes les données attendues dans le futur pour chaque capteur, et on s'interdit pour le moment d'intégrer une donnée plus récente.
3. Les données candidates à l'intégration sont les plus vieilles non intégrées de chaque capteur et répondant à la condition 2, et parmis celles-ci on choisit la plus vieille.
4. La donnée sélectionnée pour être intégrée est la plus vieille non intégrée, si elle répond à la condition 2 (sinon on attend l'arrivée de la prochaine donnée).
5. On calcule le nombre minimal de données en attente d'intégration parmi tous les capteurs actifs. Si ce nombre est strictement supérieur à deux (c'est-à-dire si tous les capteurs ont au moins deux données en attente d'intégration), cela signifie que le traitement est en retard. Si la donnée sélectionnée précédemment est soumise à la politique

« Dernières », elle sera alors éliminée directement sans être intégrée, et le processus de sélection recommence immédiatement.

Remarque 1 Cet algorithme a été développé avant l'implémentation de la gestion temps réel de l'intégration des amers présentée section 4.5.2 page 74, et il pourrait être amélioré. Il serait sans doute plus robuste pour le SLAM d'utiliser la politique « Toutes » pour tous les capteurs, en gérant globalement pour tous les capteurs la répartition du temps de calcul disponible, et en limitant le temps accordé aux capteurs lourds à intégrer si du retard a été pris.

Remarque 2 L'algorithme présenté correspond au traitement des données *en ligne*, qui présente une certaine difficulté. Il existe deux autres variantes : *hors ligne*, qui intègre toutes les données puisqu'elles sont déjà disponibles, et *rejeu*, qui intègre les données exactement comme à l'exécution en ligne en se basant sur un journal, afin de reproduire à l'identique l'exécution à des fins de débogage, comme expliqué section 3.2.3.c page 39.

b Extrapolation

Un estimateur est toujours un peu en retard par rapport au temps courant, à cause de la latence des données et du temps de leur traitement. L'algorithme présenté pour s'adapter à la contrainte supplémentaire de chronologisme du filtre accentue même un peu ce retard en alignant tous les capteurs sur le capteur avec la plus grande latence. De plus l'état du robot n'est mis à jour que lorsqu'une donnée est observée, ce qui n'est pas régulier et ne fournit aucune garantie de période maximale sans mise à jour.

Ces limitations sont problématiques pour une utilisation en ligne, où l'estimation de l'état est utilisé par des algorithmes de contrôle et de construction de modèle d'environnement, qui demandent à la fois une mise à jour de l'état régulière, garantie, et de plus avec un faible retard.

Nous avons donc implémenté une extrapolation de l'état du robot à l'instant courant, à partir de l'état estimé du filtre. Cette extrapolation se fait en deux temps :

1. On commence par intégrer les données inertielles qui sont disponibles, étant peu en retard et complètes. En pratique on stocke une représentation annexe de l'état du robot, qui est réinitialisée à l'état du robot dans le filtre principal à chaque fois qu'une nouvelle donnée est observée, et qui est mis à jour avec les données inertielles au fur et à mesure de leur arrivée (on utilise les mêmes fonctions de prédiction que pour le filtre principal).
2. On complète ensuite à chaque demande par un modèle de mouvement à vitesse constante pour atteindre la date demandée (qui peut donc même être un peu dans le futur).

Cela permet ensuite d'exporter l'état extrapolé du robot à une fréquence fixe (dans notre cas 100 Hz), avec une estimation de l'incertitude cohérente tenant compte de cette extrapolation.

L'inconvénient que l'on peut relever est que les données inertielles sont donc intégrées deux fois : une fois pour l'extrapolation, et une fois dans le filtre principal. Mais seules des pré-

dictions étant réalisées, l'extrapolation ne concerne que l'état du robot et non l'état du filtre complet, et est donc une opération peu coûteuse.

c POM

Le mécanisme d'extrapolation est de plus paramétrable par un délai, afin d'exporter l'état avec un certain retard ou même un certaine avance, selon les besoins. Cela ne permet cependant pas de répondre à différents besoins de différents modules logiciels clients des informations de localisation.

En pratique sur nos robots un module extérieur, appelé POM (*POsition Manager*) se charge de maintenir un court historique de l'état du robot, afin que chaque client puisse savoir à tout moment quel était l'état du robot à un instant donné. Typiquement si les algorithmes de contrôle voudront connaître l'état actuel du robot, voire même avec un peu d'avance, les algorithmes de construction de modèles d'environnement voudront étiqueter leurs données qu'ils reçoivent avec du retard. Ces derniers auront même tout intérêt à attendre le plus possible avant d'étiqueter leurs données (juste avant leur utilisation) et non le moins possible (juste après leur réception), afin de bénéficier de la meilleure estimation possible de la position à ces dates passées.

Bilan de nos travaux. Perspectives à court terme, et analyse préliminaire d'une solution pour leur extension à la localisation au sein d'une équipe de robots.

Chapitre 7

Conclusion

Sommaire

7.1 Synthèse

7.1.1 Bilan

Au cours de cette thèse nous avons donc construit une solution opérationnelle de localisation pour nos robots, robuste, temps-réel, embarquée, et précise. Cette construction s'est faite incrémentalement, en identifiant à chaque étape les faiblesses du système afin de choisir de nouveaux capteurs à intégrer pour les combler. Nous sommes ainsi partis d'une solution classique de SLAM EKF avec une caméra aidée d'un simple modèle de mouvement. Devant les problèmes de robustesse et d'estimation du facteur d'échelle, nous avons décidé d'intégrer une centrale inertielle. Malgré la forte amélioration, il demeurait des imprécisions d'estimation du facteur d'échelle, et nous avons donc décidé d'intégrer l'odométrie, puis une deuxième caméra pour une précision encore meilleure, ainsi que des capteurs de localisation absolue afin de pouvoir en tirer profit lorsqu'ils sont disponibles. Nous avons également étudié en détail les modélisations, réglages, calibrages, et le comportement des algorithmes afin de comprendre les limitations du système. En parallèle nous avons résolu de nombreux problèmes pratiques posés par l'embarquement des algorithmes sur le robot.

Tous ces développements ont été intensivement testés, sur données réelles, puis en ligne sur le robot, et enfin utilisé de façon transparente sur le robot pour mener à bien des missions de navigation plus complexes au sein d'un système complet. RT-SLAM a ainsi déjà traité plusieurs millions d'images. Les participations de notre équipe aux challenges euRathlon (anciennement elRob) ont de plus permis de tester nos algorithmes dans des environnements très variés, et surtout inconnus et non contrôlés.

7.1.2 Contributions

Les différentes contributions de cette thèse peuvent être résumées par cette liste :

- Développement d'un système opérationnel de localisation pour un robot autonome, incluant des fonctionnalités de la littérature, mais aussi des éléments originaux, et qui reste un outil de recherche générique, permettant facilement l'intégration de nouveaux capteurs, types d'amers, etc.
- Intégration de multiples capteurs : caméras, capteurs inertiels, odométrie, GPS, capture de mouvement, observations extérieures, en réglant les paramètres du filtre par rapport à leurs spécifications et/ou leur caractérisation.
- Très nombreuses validations expérimentales du système, sur le terrain et hors ligne.
- Implémentation d'une approche de traitement des amers temps-réel dur basée sur la recherche active : section 4.5.2 page 74.
- Une technique originale pour limiter l'inconsistance du filtre lors des mouvements de faible dynamique, consistant à ne mettre à jour la covariance que pour les observations significatives : section 4.5.3 page 74.
- Une méthode originale d'intégration de l'odométrie qui s'abstrait de ses défauts d'être biaisée dans sa mesure de la direction de la vitesse (en n'utilisant que sa norme), et

d'être fautive dans sa mesure de la norme de la vitesse (en l'utilisant en observation pour appliquer le contrôle de compatibilité) : section 6.1 page 136.

— Une analyse expérimentale précise de l'observabilité de différents paramètres de calibrage avec un EKF : biais d'une centrale inertielle, calibrage extrinsèque de capteurs : sections 5.3 page 94, 5.4.1 page 105, 6.5.3.b page 160.

— Une méthode originale de faire du *bicam*, en utilisant des images temporellement alternées afin d'augmenter la fréquence temporelle de traitement à charge calculatoire équivalente : section 6.5.2.a page 155.

— Une méthode originale de faire du *bicam* en recherche active avec des caméras à forte distortion, y compris avec des caméras panoramiques, afin d'améliorer la recherche des amers et la consistance, qui décompose la profondeur en hypothèses multiples en s'adaptant dynamiquement à la non linéarité de la zone de recherche, permettant de faire du multicam hétérogène (mélangeant caméras panoramiques et perspectives) : section 6.5.3.a page 157.

— Une méthode de filtrage des horodatages des différents capteurs qui permet d'obtenir des horodatages très précis sans avoir un recours à un système d'exploitation temps-réel ou de la synchronisation matérielle : section 5.4.2.c page 115.

— Adaptation dynamique de la densité d'amers à la taille de la carte : section 4.4.4.b page 70.

— Un algorithme de gestion temporelle de l'intégration des capteurs, afin de minimiser à la fois la latence et le nombre de données manquées : section 6.6.1 page 165.

— Proposition d'un algorithme d'estimation en ligne des décalage d'horloge des capteurs : section 5.4.3 page 117.

7.1.3 Récapitulatif des performances

Types de capteurs Le tableau 7.1 page suivante récapitule les performances obtenues par les différentes combinaisons de capteurs testées sur la séquence ESPERCE (pour un calcul en temps-réel utilisant un cœur de processeur, en appliquant une politique locale de traitement des amers de type odométrie visuelle décrite section 4.4.4.c page 71).

Les systèmes n'utilisant que des caméras ne sont pas suffisamment robustes pour être utilisés en pratique. Le bicam avec modèle de mouvement se comporte mieux que la version monocam, comme attendu, mais tout de même bien en deçà des espérances. Cela semble être en particulier dû aux corruptions d'images évoquées section 3.3.2.c page 44, qui décalent complètement des parties entières de l'image, et provoquent quelques faux appariemments qui se retrouvent majoritaires et ont donc du mal à être éliminés.

L'ajout de l'odométrie n'influe que sur les erreurs de position, et pas du tout sur les erreurs angulaires, ce qui est attendu. Le bicam lui améliore très légèrement la précision angulaire, mais c'est également au niveau de la position et du facteur d'échelle qu'il fait la différence.

Le bicam alterné n'apporte pas de gain de performance, mais ne les dégrade pas non plus en observant toujours correctement le facteur d'échelle, bien que les images ne soient pas synchronisées. De la même façon l'odométrie n'a plus rien à apporter en terme de précision

Système	**Dérive**		**Erreur RMS**			**Diverg.**
	$\mathrm{cm}/\sqrt{\mathrm{m}}$	$\mathrm{mdeg}/\sqrt{\mathrm{m}}$	m	deg	σ_{m} [a]	%
Monocam [b]	151	462	76	23	233	30
Bicam sync.	37	345	15	15	192	3
Monocam + IMU	23	80	1.8	0.49	7.7	<0.5 [c]
Monocam + IMU + odo	8.1	80	1.10	0.47	9.4	0
Bicam sync. + IMU	6.2	71	0.59	0.40	11.0	0
Bicam alt. + IMU	6.4	70	0.60	0.39	10.8	0
Bicam sync. + IMU + odo	6.3	71	0.59	0.40	11.0	0

TABLE 7.1 – Récapitulatif des performances des différents systèmes sur la séquence ESPERCE avec un traitement temps-réel mono-cœur de 50 images par secondes (toutes caméras confondues), et en observant 25 amers par image.

a. Racine carrée du NEES moyen de la position (x, y, z), à comparer à une valeur attendue de 1 (ou en tout cas bien inférieure à l'enveloppe à 99% de 3.37 car les mesures ne sont pas parfaitements indépendantes à cause des données).

b. Facteur d'échelle corrigé a posteriori.

c. Des cas de divergence ont pu être observés, le système étant sensible à quelques paramètres de configuration comme l'initialisation de la distance inverse des amers.

au bicam inertiel. En revanche, bicam alterné comme odométrie apportent une plus grande robustesse au système. Le premier en doublant la fréquence des images, le deuxième en stabilisant le système lorsque les caméras sont fautives à cause de trop mauvaises conditions environnementales.

En ce qui concerne la consistance, on constate que le système est peu consistant, comme on peut s'y attendre avec un EKF sur de longues séquences. On peut noter que l'inconistance augmente quand la précision s'améliore, notamment à l'ajout de capteurs. Cette inconsistance provient d'erreurs de linéarisation, mais aussi certainement de biais non correctement modélisés dans le système.

Fréquence des images Le tableau 7.2 présente maintenant une analyse des performances similaire mais en fonction de la fréquence des images utilisées.

On constate de façon surprenante que diminuer la fréquence des images dans une certaine mesure ne dégrade que très peu les performances, voire les améliore dans certains cas. Cela est dû au fait que diminuer la fréquence des observations améliore nettement la consistance. De même augmenter la fréquence *virtuelle* à 100 images par seconde en bicam en utilisant toutes les images des deux caméras à 50 Hz détériore encore nettement la consistance, et légèrement la précision. Le paradigme d'augmenter la fréquence de la vision pour améliorer les points de linéarisation n'est valable qu'avec un modèle de vitesse ; avec une centrale inertielle en prédiction, les prédictions sont bonnes sur plusieurs dizaines voire centaines de millisecondes, et diminuer le nombre d'observations avec des jacobiennes différentes améliore la consistance. Le fonctionnement est ici encore tout à fait correct à 12.5 Hz, mais on pourra choisir de travailler à 25 Hz par sécurité pour une précision légèrement accrue. Rien ne justifie ici de travailler à 50 Hz, mais cela est à pondérer en fonction des conditions d'utilisation (notamment

Système	Dérive		Erreur RMS			Diverg.
	$\mathrm{cm}/\sqrt{\mathrm{m}}$	$\mathrm{mdeg}/\sqrt{\mathrm{m}}$	m	deg	σ_m	%
Monocam + IMU + odo @50	8.1	80	1.10	0.47	9.4	0
Monocam + IMU + odo @25	8.0	78	1.05	0.47	7.6	0
Monocam + IMU + odo @12.5	9.2	84	1.09	0.51	6.9	0
Monocam + IMU + odo @8.33	11.0	104	1.8	0.79	8.5	1.0
Monocam + IMU + odo @6.25	22.5	238	3.7	3.44	15.3	3.6
Bicam sync. + IMU @100	6.4	77	0.61	0.43	16.1	0
Bicam sync. + IMU @50	6.2	70	0.57	0.39	10.7	0
Bicam sync. + IMU @25	6.1	71	0.54	0.39	7.2	0
Bicam sync. + IMU @12.5	6.6	84	0.61	0.47	5.6	0
Bicam sync. + IMU @8.33	6.7	94	5.73	0.70	6.7	1.0
Bicam sync. + IMU @6.25	31.8	324	168	5.4	19.4	18.8
Bicam alt. + IMU @50	6.3	71	0.63	0.39	11.2	0
Bicam alt. + IMU @25	6.4	71	0.60	0.40	7.7	0
Bicam alt. + IMU @12.5	6.4	78	0.62	0.46	5.6	0
Bicam alt. + IMU @8.33	7.1	92	2.94	0.68	6.3	0.5
Bicam alt. + IMU @6.25	16.9	212	131.7	3.5	15.3	6.8

TABLE 7.2 – Analyse des performances selon les fréquences d'image. Les fréquences données pour le bicam sont toutes caméras confondues (@ 50 signifie deux caméras à 25 Hz). Notons que le calcul de dérive exclu les cas de divergence, mais pas le calcul d'erreur RMS.

la dynamique).

Le bicam alterné montre des performances globalement équivalentes, et se montre un peu plus robuste lorsque la fréquence des images devient critique, même s'il est préférable d'éviter de travailler dans ce domaine. L'augmentation du nombre d'amers à une fréquence plus basse fait remonter un peu les performances, mais aussi l'inconsistance. Il n'est donc pas particulièrement préférable d'observer plus d'amers dans chaque image et moins d'images.

Il faut par ailleurs noter qu'en raison des corruptions d'image, il y a toujours des pertes d'image (deux consécutives pour les séquences @50, une seule ensuite, sachant qu'en bicam il y a peu de chances que les deux caméras en soit victime en même temps, contrairement à la stéréo).

Nombre d'amers Le tableau 7.3 page suivante analyse cette fois l'effet de l'observation de plus ou moins d'amers dans chaque image sur les performances.

En monocam comme attendu la réduction du nombre d'amers dégrade sensiblement la précision, tandis qu'une augmentation a au contraire tendance à les améliorer, mais sans doute pas suffisamment pour que cela justifie le surcoût calculatoire.

En bicam, la sensibilité au nombre d'amers observés est moindre. Une réduction du nombre d'amers n'affecte pas excessivement la précision (ni la consistance), à part à la fréquence réduite de 12.5 images par seconde. Une augmentation n'apporte rien non plus, et au contraire a surtout pour effet d'augmenter l'inconsistence et dégrader légèrement la précision, sauf

Système	**Dérive**		**Erreur RMS**			**Diverg.**
	$\mathrm{cm}/\sqrt{\mathrm{m}}$	$\mathrm{mdeg}/\sqrt{\mathrm{m}}$	m	deg	σ_m	%
Monocam + IMU + odo @50 --	11.3	139	1.83	1.13	12.7	0
Monocam + IMU + odo @50	8.1	80	1.10	0.47	9.4	0
Monocam + IMU + odo @50 ++	7.5	72	1.01	0.48	9.7	0
Monocam + IMU + odo @25 --	10.1	105	1.21	0.72	7.3	0
Monocam + IMU + odo @25	8.0	78	1.05	0.47	7.6	0
Monocam + IMU + odo @25 ++	7.7	74	0.97	0.45	8.2	0
Monocam + IMU + odo @12.5 --	12.6	222	3.97	2.75	15.9	0.5
Monocam + IMU + odo @12.5	9.2	84	1.09	0.51	6.9	0
Monocam + IMU + odo @12.5 ++	10.4	71	1.04	0.43	8.4	0
Bicam sync. + IMU @50 --	6.8	90	0.77	0.55	10.5	0
Bicam sync. + IMU @50	6.2	70	0.57	0.39	10.7	0
Bicam sync. + IMU @50 ++	6.7	79	0.77	0.57	17.6	0
Bicam sync. + IMU @25 --	6.9	86	0.78	0.58	7.5	0
Bicam sync. + IMU @25	6.1	71	0.54	0.39	7.2	0
Bicam sync. + IMU @25 ++	6.5	76	0.75	0.53	12.2	0
Bicam sync. + IMU @12.5 --	7.7	104	0.78	0.62	4.9	0
Bicam sync. + IMU @12.5	6.6	84	0.61	0.47	5.6	0
Bicam sync. + IMU @12.5 ++	6.3	78	0.60	0.43	6.9	0

TABLE 7.3 – Analyse des performances selon le nombre d'amers observés à chaque image : 25 par défaut, 30% en moins pour - -, 50% en plus pour ++.

encore une fois à 12.5 images par secondes où l'on retrouve la précision des fonctionnements à vitesse plus élevée, y compris pour la consistance.

L'observation d'une petite trentaine d'amers semble donc un bon compromis de précision absolue, mais aussi calculatoire puisque l'on peut fonctionner ainsi à 50 Hz sur un ordinateur moderne, et à fréquence réduite sur un processeur plus léger tels que ceux rencontrés sur les quadrirotors par exemple.

Consistance Le tableau 7.4 présente enfin les résultats de variantes visant à améliorer la consistance : la non mise à jour de la covariance pour les observations non significatives (section 4.5.3 page 74), et la variante FEJ-EKF consistant à toujours linéariser à l'endroit de la première estimation [Huang *et al.*, 2008], appliquée uniquement sur les amers euclidiens (puisqu'il n'est normalement pas applicable avec des observations partielles).

On constate que FEJ-EKF améliore sensiblement la consistance (en moyenne de 15%), avec une influence plus ou moins neutre sur la précision (plutôt une légère amélioration en monoculaire, et une légère détérioration en bicam). Cette amélioration de la consistance est de plus cumulative avec celle liée à la réduction de la fréquence des images. En revanche on notera qu'en bicam l'axe z est à nouveau mal estimé au départ de la séquence, ce qui avait disparu en bicam.

Il serait intéressant d'étendre l'application de ce principe aux amers paramétrés en distance

Système	**Dérive**		**Erreur RMS**			**Diverg.**
	$\mathrm{cm}/\sqrt{\mathrm{m}}$	$\mathrm{mdeg}/\sqrt{\mathrm{m}}$	m	deg	σ_m	%
Monocam + IMU + odo @50	8.1	80	1.10	0.47	9.4	0
Monocam + IMU + odo @50 S	8.5	84	1.11	0.52	7.6	0
Monocam + IMU + odo @50 F	7.8	74	1.01	0.43	8.0	0
Monocam + IMU + odo @50 SF	8.0	78	1.04	0.48	6.7	0
Monocam + IMU + odo @25	8.0	78	1.05	0.47	7.6	0
Monocam + IMU + odo @25 S	8.3	84	1.07	0.51	7.1	0
Monocam + IMU + odo @25 F	7.8	73	0.97	0.41	6.7	0
Monocam + IMU + odo @25 SF	7.9	76	1.00	0.43	6.3	0
Monocam + IMU + odo @12.5	9.2	84	1.09	0.51	6.9	0
Monocam + IMU + odo @12.5 S	9.1	81	1.07	0.49	6.6	0
Monocam + IMU + odo @12.5 F	9.3	85	1.06	0.52	6.4	0
Monocam + IMU + odo @12.5 SF	9.0	76	1.00	0.47	6.2	0
Bicam sync. + IMU @50	6.2	70	0.57	0.39	10.7	0
Bicam sync. + IMU @50 S	6.2	76	0.54	0.53	5.9	0
Bicam sync. + IMU @50 F	6.5	76	0.59	0.44	9.0	0
Bicam sync. + IMU @50 SF	6.0	70	0.47	0.39	4.5	0
Bicam sync. + IMU @25	6.1	71	0.54	0.39	7.2	0
Bicam sync. + IMU @25 S	6.3	73	0.52	0.43	4.8	0
Bicam sync. + IMU @25 F	6.2	74	0.54	0.40	6.2	0
Bicam sync. + IMU @25 SF	6.0	71	0.48	0.42	3.6	0
Bicam sync. + IMU @12.5	6.6	84	0.61	0.47	5.6	0
Bicam sync. + IMU @12.5 S	6.9	86	0.57	0.48	4.7	0
Bicam sync. + IMU @12.5 F	6.3	81	0.56	0.44	4.8	0
Bicam sync. + IMU @12.5 SF	6.6	80	0.51	0.46	4.2	0

TABLE 7.4 – Amélioration de la consistance avec la mise à jour de la covariance pour les observations significatives uniquement (S), la variante FEJ-EKF sur les amers euclidiens (F), et leur combinaison (SF).

inverse, par exemple en ne mettant à jour le point de linéarisation que lorsqu'il a changé suffisamment pour menacer la précision du système.

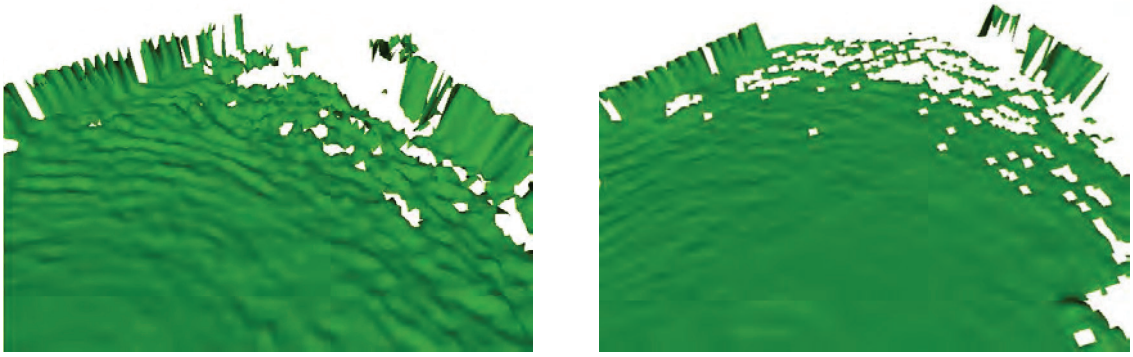
La méthode des observations significatives apporte également un gain significatif sur la consistence, mais plutôt moins que FEJ-EKF en monocam, et plus en bicam. L'effet sur la précision reste globalement neutre, avec parfois de légères améliorations, et parfois de légères détériorations.

Les deux méthodes se combinent avantageusement, et leurs effets se cumulent. On peut ainsi dans certains cas diviser l'inconsistance par deux (bicam à 25 et 50 images par seconde). Les deux méthodes ont cependant des résultats assez variables selon les conditions.

7.1.4 Conclusion

RT-SLAM est désormais la solution de localisation par défaut sur notre robot Mana lors des démonstrations générales de navigation autonome. Par rapport à la solution précédente utilisant l'odométrie, un gyromètre à fibre optique pour le cap, une centrale inertielle pour l'attitude, et du GPS centimétrique quand il était disponible, RT-SLAM a permis d'être moins dépendant du GPS, mais surtout en améliorant considérablement l'estimation de l'orientation, que ce soit en terme de précision ou de dynamique, de construire des modèles d'environnement beaucoup moins bruités, et donc de naviguer plus vite. Alors que Mana était auparavant limité à la vitesse de 1 m/s, au delà de laquelle les modèles devenaient trop bruités et généraient des obstacles imaginaires, il peut désormais se déplacer à la vitesse de 2 m/s.

À titre d'illustration, la figure 7.1 compare qualitativement le modèle numérique de terrain obtenu avec l'ancienne solution par rapport à RT-SLAM, dans des conditions similaires. Nous n'avons malheureusement pas eu le temps de réaliser une étude quantitative de cette qualité, en enregistrant en parallèle les données du Lidar Velodyne et les deux sorties de localisation, utilisées pour reconstruire les deux modèles de terrain évalués en terme de dispersion des échantillons dans chaque case du modèle.



(a) Ancienne localisation (b) RT-SLAM

FIGURE 7.1 – Comparaison des modèles numériques de terrain obtenus avec les deux solutions de localisation, sur un terrain plat herbeux à la vitesse de 2 m/s.

La figure 7.2 montre les résultats de localisation obtenus lors du challenge Eurathlon/Elrob 2013, pendant lequel le robot Mana a navigué pendant près de 3 km en autonome sur une piste forestière en utilisant uniquement la localisation fournie par RT-SLAM en bicam inertiel, dans des conditions difficiles (contre-jours, pluie, gouttes d'eau sur les objectifs, masquages parfois conséquents par les opérateurs de sécurité). L'erreur en position est limitée à quelques dizaines de mètres.

La localisation n'est maintenant plus le facteur limitant pour pouvoir naviguer à de plus hautes vitesses. Les difficultés sont plutôt dues au calibrage du Lidar Velodyne, et surtout à notre algorithme de génération des déplacements qui n'a pas été conçu pour de telles vitesses (à l'époque de sa conception les robots évoluaient entre 10 et 50 cm/s), et qui fournit des séquences de commandes de déplacements (simples arcs de cercles) qui ne sont dynamiquement pas réalisables.

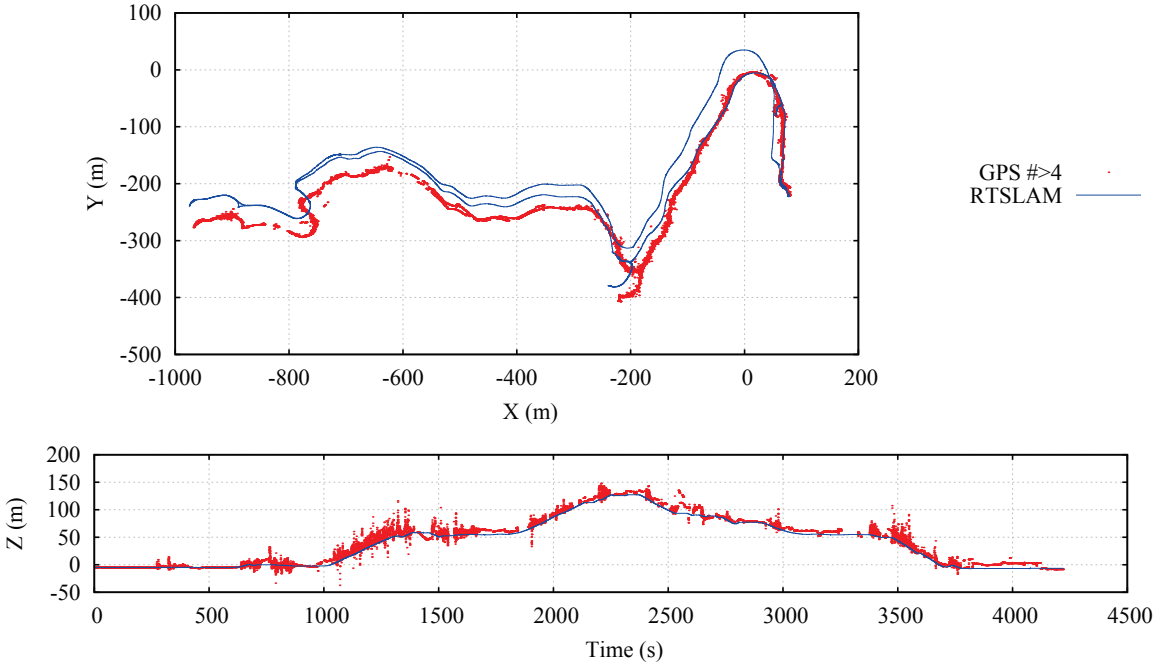


FIGURE 7.2 – Résultat en ligne du SLAM visuel-inertiel bicam lors du challenge Eurathlon/Elrob en septembre 2013 à Berchtesgaden.

7.2 Perspectives à court terme

Nous présentons ici les différentes petites améliorations possibles qui seraient rapides à faire, sans étendre significativement le fonctionnement de RT-SLAM.

— Utilisation de segments en complément des points. Un stagiaire a commencé à travailler sur le sujet et RT-SLAM a permis d'obtenir des bons résultats. Mais la difficulté porte surtout sur l'algorithme de suivi des segments préalablement détectés, qui provoque des erreurs d'appariement difficilement détectables.

— Estimation des normales des amers pour améliorer leur suivi.

— Implémentation de l'intégration du gyromètre à fibre optique.

— Implémentation de l'estimation *robotcentric* [Castellanos *et al.*, 2007] pour le mode odométrie visuelle, qui représente la carte dans le repère local du robot. Cela permet de diminuer l'incertitude de la carte et donc d'améliorer l'estimation des incertitudes avec les linéarisations, d'où une meilleure consistance.

— Contrôle dynamique de la détection des nouveaux amers lorsqu'il n'y en a pas assez d'observés, et factorisation des ancres des amers AHP initialisés durant la même image, afin d'accélérer les calculs et permettre d'observer plus d'amers.

7.3 Perspectives à moyen terme

Nous présentons ici un certain nombre d'améliorations plus profondes que ne nous n'avons pas encore eu le temps d'implémenter, mais qui pourraient apporter un gain significatif de

performance.

— Paramétrisation des orientations avec le groupe de Lie au lieu des quaternions, qui présente l'avantage de ne pas avoir de degré de liberté superflu.

— Utilisation de techniques d'optimisation globale à la place ou en complément du filtrage de Kalman.

— S'orienter vers un fonctionnement en continu des robots pour l'estimation des différents paramètres de calibrage : biais des capteurs inertiels, et calibrage des caméras par exemple. Ces paramètres évoluent en effet légèrement au cours du temps, selon différents dynamiques (par exemple dues aux variations de températures), et une mauvaise estimation de leur valeur peut considérablement affecter la précision de la localisation – voire sa consistance. Un robot endurant exploité des mois durant devrait être muni d'un système automatique d'estimation de ces paramètre, ce que permet RT-SLAM.

7.4 Prospective à long terme : localisation au sein d'une équipe de robots

7.4.1 Besoin d'une solution plus générale

Le filtre de Kalman a deux inconvénients qui le rendent inapte à estimer des trajectoires sur le long terme de façon suffisamment fiable :

— Sa complexité en temps de calcul et occupation mémoire est quadratique en le nombre d'amers dans la carte, ce qui rend les calculs non opérables en temps réel assez rapidement (quelques centaines d'amers, obtenus en général sur quelques dizaines de mètres).

— Sa consistance devient insuffisante assez rapidement également, en général avant même que le temps de calcul soit trop coûteux, comme l'ont montré [J. A. Castellanos et Tardós, 2004].

7.4.2 SLAM hiérarchique

C'est pourquoi [Estrada *et al.*, 2005] a proposé le SLAM hiérarchique, une architecture hybride de filtrage et d'optimisation comportant deux niveaux (figure 7.3) :

— Des cartes locales indépendantes, qui sont des cartes d'amer gérées par un SLAM EKF classique.

— Une carte globale qui est un graphe gérant les transformations relatives entre les cartes locales.

Le principe est donc de cantonner le SLAM EKF à des cartes de taille limitée, où il reste suffisamment consistant et les temps de calculs restent gérables. Ainsi on terminera une carte dans les cas suivants (liste non nécessairement exhaustive) :

— Trop d'amers de qualité à conserver (carte pleine)

— Incertitude angulaire sur le cap trop importante, car elle est la première cause d'inconsistance [Bailey *et al.*, 2006a].

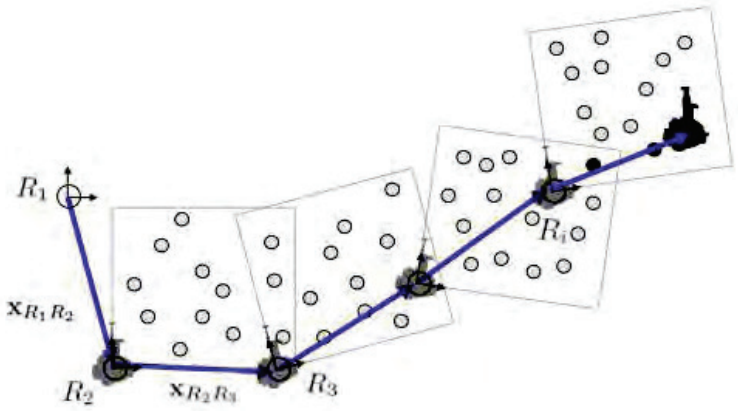


FIGURE 7.3 – [a] Illustration du SLAM hiérarchique.

a. Illustrations créées par Teresa Vidal, notamment dans [Vidal-Calleja *et al.*, 2011].

— Incertitude absolue trop importante par rapport à la précision des modèles d'environnement souhaités.

— Renouvellement trop rapide de l'ensemble des amers, correspondant probablement à un changement d'environnement (par exemple passage de porte) ou un problème technique susceptible d'engendrer une inconsistance (faute).

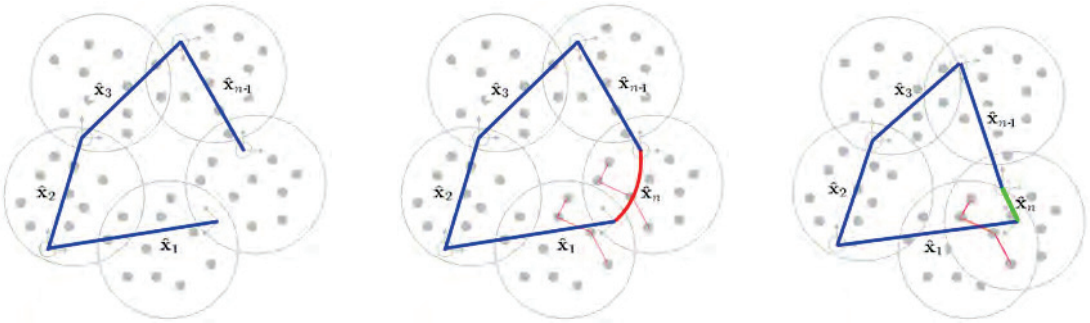
On obtient ainsi un fonctionnement en temps borné, avec une occupation mémoire linéaire, laissant cependant la possibilité d'utiliser une mémoire lente comme un disque dur pour stocker le détail des anciennes cartes.

Puis au niveau global, chaque nœud représente l'origine d'une carte locale, et chaque arête représente la position relative de la nouvelle carte par rapport à l'ancienne carte, qui est simplement la position du robot dans l'ancienne carte (avec les incertitudes associées) au moment de la création de la nouvelle carte. Les évènements de fermeture de boucle correspondent alors à la création d'un cycle topologique dans le graphe. A cause des erreurs de localisation, ce cycle topologique ne correspond cependant pas à un cycle géométrique (la composition de l'ensemble des transformations associées aux arêtes doit donner la transformation identité). Une étape d'optimisation doit alors modifier chaque arête selon son incertitude pour contraindre géométriquement l'ensemble des cycles présents dans le graphe, fournissant ainsi une position corrigée de chaque carte locale. Un exemple d'optimisation d'un cycle obtenu par appariemment de carte est donné figure 7.4 page suivante.

Le principe de la fermeture de boucle a été appliqué dans [Roussillon et Lacroix, 2010] avec appariemment de cartes automatique.

7.4.3 SLAM hiérarchique multirobots

Considérer la coopération de plusieurs robots permet d'améliorer significativement la précision de la localisation globale à travers des localisations relatives des robots, par observation directe, ou indirectement par observation d'amers communs. L'architecture de filtrage ne permet cependant pas d'exploiter pleinement ces informations, car la décentralisation du système



(a) Cartes locales successivement construites.

(b) Appariemment de deux cartes, fournissant une transformation relative entre elles.

(c) Correction après application de la contrainte géométrique de cycle.

FIGURE 7.4 – Optimisation d'une fermeture de boucle.

oblige à avoir plusieurs filtres parallèles, et parce que le partage d'information est délicat à gérer pour garantir que des informations ne sont pas utilisées plusieurs fois par le même filtre, ce qui mènerait à des inconsistances (par exemple si un robot B utilise de l'information de A pour se localiser, puis plus tard A utilise de l'information de B, alors A a réutilisé de l'information qu'il avait déjà intégrée).

En revanche une approche de SLAM hiérarchique s'étend trivialement au cas multirobots (figure 7.5), comme présenté dans [Vidal-Calleja *et al.*, 2011] : chaque robot construit ses propres cartes locales et son graphe, et partage son graphe avec les autres robots pour que chacun puisse opérer une optimisation globale. Une ébauche de cette solution a été étudiée au sein du laboratoire dans [Vidal-Calleja *et al.*, 2011], mais nous allons maintenant analyser plus précisément tous les problèmes qui se posent pour rendre cette gestion générique et automatique.

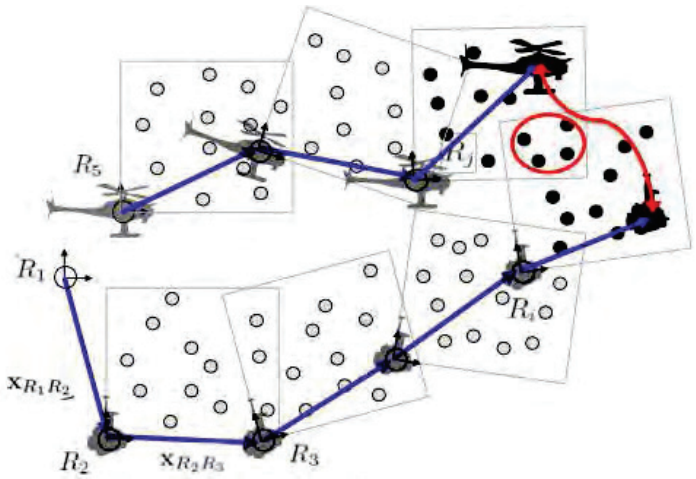
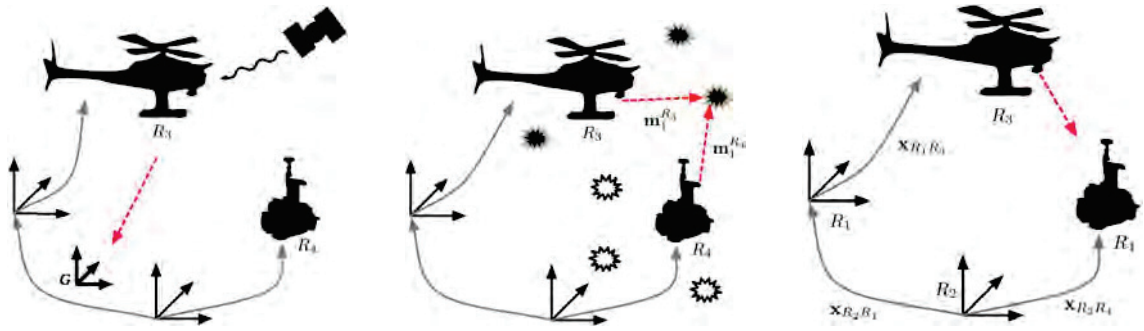


FIGURE 7.5 – [a] Illustration du SLAM hiérarchique en multirobots.

a. Illustrations créées par Teresa Vidal, notamment dans [Vidal-Calleja *et al.*, 2011].

Dans le cas multirobots, les évènements de fermeture de boucle sont multipliés. En plus

des localisations absolues (figure 7.6(a)) tels que mesure GPS ou localisation par rapport à un modèle initial, et des appariemments de carte monorobot, d'autres évènements deviennent exploitables comme des appariemments de carte interrobots (figure 7.6(b)) ou des observations directes interrobots (figure 7.6(c)).



(a) Localisation absolue (mêmes évènements qu'en monorobot)

(b) Appariemment de cartes (évènements en monorobot plus d'autres en multirobots)

(c) Observation mutuelle (évènements propres au multirobots)

FIGURE 7.6 – [a] Possibilités de fermeture de boucle en multirobots.

a. Illustrations créées par Teresa Vidal, notamment dans [Vidal-Calleja *et al.*, 2011].

L'application d'une fermeture de boucle par appariement de cartes monorobot a été appliqué de façon préliminaire et en hors ligne dans [Vidal-Calleja *et al.*, 2011], avec un robot aérien et un robot terrestre – l'appariement de cartes était assisté par un opérateur.

Des difficultés apparaissent cependant lorsque l'on souhaite généraliser ce principe avec plusieurs capteurs :

1. Quels capteurs doivent être intégrés dans le filtre des cartes locales, et quels capteurs doivent être considérés au contraire par l'optimisation globale du graphe ?
2. Quelles grandeurs doivent être optimisées dans le graphe, quelles grandeurs doivent être conservées d'une carte à la suivante ?

a Répartition des capteurs

Le but de l'optimisation globale du graphe via les fermetures de boucle est de supprimer autant que possible la dérive de la localisation due aux capteurs utilisés. Il est donc nécessaire de réserver les capteurs qui produisent une localisation qui dérive à la construction des cartes locales. À l'inverse il semble logique d'utiliser les capteurs qui ne dérivent pas au niveau de l'optimisation globale du graphe. En effet ceux-ci sont souvent moins disponibles, mais permettent régulièrement de recaler la localisation. De plus ils ont tendance à générer des discontinuités dans la localisation, qui sont gênantes pour la construction de modèles de terrain, mais sont plus faciles à gérer lorsque cantonnées aux frontières entre cartes locales, comme on le verra section 7.4.5 page 192.

On répartira donc les capteurs ainsi :

Niveau local	Niveau global
Vision	GPS
Inertiel	Localisation / modèle
Odométrie	Altimètres
	Fermetures de boucle
	Magnétomètres

b Grandeurs à optimiser ou à conserver

Si l'estimation de position dans les cartes locales tend à dériver, certaines grandeurs estimées sont cependant complètement observables et leur estimée ne dérive pas. En SLAM bicam-inertiel par exemple, c'est le cas notamment des vitesses, des biais de l'IMU, ou même des angles d'attitude stabilisés par l'observation de la gravité. Il est donc inutile d'optimiser globalement ces grandeurs, seules celles qui dérivent ont intérêt à être optimisées.

En revanche même si ces grandeurs sont observables, il est parfois préférable de démarrer l'estimation au sein d'une carte avec des valeurs initiales raisonnables, comme pour les vitesses et les biais, et parfois cela est absolument nécessaire, comme pour l'orientation lorsque la gravité est imposée verticale (les raisons et avantages ont été discutés section 5.2.2 page 89).

Le principe de SLAM hiérarchique suppose cependant que les cartes locales sont parfaitement indépendantes, afin que l'optimisation globale n'utilise pas plusieurs fois la même information provenant de différentes cartes, et ne soit donc pas biaisée. Il n'est donc pas possible de transférer de l'information d'une carte à la suivante. Cependant la quantité d'information dans un filtre dépend essentiellement des incertitudes. Ainsi transférer la valeur moyenne de grandeurs, en initialisant une incertitude correspondante très grande revient à rendre la quantité d'information transférée négligeable, et en tout cas pas supérieure à celle introduite par des valeurs initiales fixes et arbitraires.

De la même façon, on pourra lors de la création d'une nouvelle carte conserver les amers de l'ancienne carte qui sont visibles. Il convient cependant de les réinitialiser complètement sans hériter d'aucune information de la carte précédente. Mais leur présence permet d'initialiser une carte avec des amers qui se sont montrés « fiables » (aisé à mettre en correspondance et bien observables), lesquels pourront de plus faciliter les appariemments de carte monorobot pour la fermeture de boucles au niveau du graphe des cartes.

7.4.4 Implémentation pratique de fermetures de boucle

a Localisation absolue

On parle ici de mesures GPS ou de localisation par rapport à un modèle initial. Chaque simple mesure permet alors d'ajouter dans le graphe une arête vers le repère global. On comprend cependant bien que si l'on ajoute une arête par mesure, si ces mesures sont régulières le nombre d'arêtes du graphe va retomber à un ordre de grandeur similaire au nombre de positions du robot, au lieu du nombre de cartes locales, ce qui rendra l'optimisation beaucoup plus lourde en temps de calcul. De plus si les mesures sont partielles, les arêtes ajoutées le seront aussi

(par exemple les mesures GPS ne contiennent pas d'information d'orientation, à moins que le vecteur vitesse soit utilisé, avec les difficultés que cela représente comme lorsque le robot est à l'arrêt ou surtout que l'angle d'incidence du robot n'est pas nul).

Il serait donc préférable de fusionner en amont l'ensemble des mesures obtenues pendant la vie d'une carte locale. On dispose de deux trajectoires : celle du robot dans la carte locale, et celle de la localisation absolue dans le repère global sur la durée de vie de la carte locale. Il suffit alors d'ajuster la position de la carte locale pour minimiser la distance entre ces deux trajectoires en pondérant par les incertitudes afin d'obtenir une unique estimée, complète, de la transformation entre le repère de la carte locale et le repère global, et en calculant sa matrice de covariance.

Dans le cas où une seule mesure serait disponible pour une carte, on peut garder la possibilité d'ajouter une arête partielle, à condition que l'algorithme d'optimisation global le gère.

b Association de cartes basée amers

Des travaux sur l'appariemment de cartes ont été réalisés en monorobot dans [Roussillon et Lacroix, 2010], en utilisant une ancienne bibliothèque de SLAM du laboratoire. La prédiction se faisait par odométrie, des images panoramiques étaient utilisées, et les appariemments d'amers n'étaient pas faits en recherche active mais de façon globale.

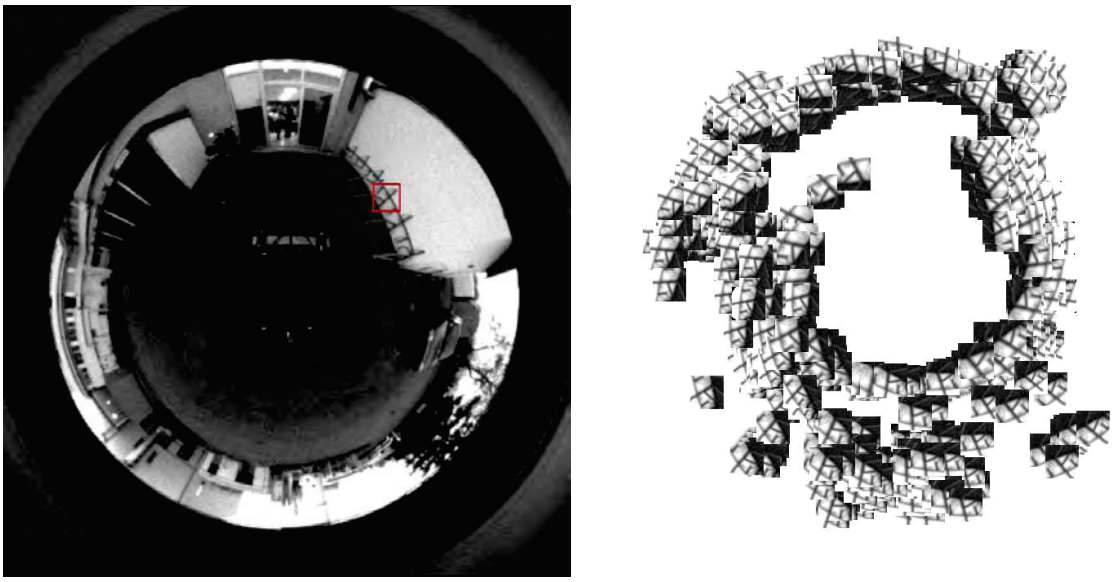
Ce dernier point a facilité l'implémentation d'un mécanisme automatique d'association de cartes. En effet pour chaque image, une centaine de points de Harris étaient extraits de l'image, et l'appariemment était fait globalement en prenant simultanément en compte leurs relations géométriques et la proximité des descripteurs de Harris (valeurs propres).

L'appariemment de cartes consistait alors à sélectionner quelques amers dans les cartes locales passées susceptibles d'en partager avec l'image courante, à savoir celles compatibles avec l'estimée courante de position et son incertitude associée, augmentée d'une marge de sécurité pour se prémunir des inconsistances. Ces amers étaient ensuite recherchés à chaque point de Harris extrait de l'image, après une prédiction par transformation homographique selon la position de l'image, afin de compenser la forte distortion des images panoramiques (figure 7.7 page suivante).

Après la fermeture de la carte courante, lorsque qu'un nombre suffisant d'amers communs avec une autre carte avait été découvert, la transformation géométrique entre les deux ensembles d'amers est estimée, avec une élimination d'éléments aberrants du type RANSAC. L'optimisation, basée sur [Estrada *et al.*, 2005] et l'extension proposée aux boucles multiples, se déroulait ensuite en arrière-plan.

La figure 7.8 page 191 montre des résultats obtenus en comparaison avec du SLAM mémoire courte où les amers perdus sont oubliés. La vérité terrain n'était pas disponible à l'époque, mais les deux segments en ligne droite dans la partie gauche doivent être confondus. Les discontinuités correspondent aux ajustements entre les cartes effectués par l'optimisation globale.

Pour une implémentation concrète dans RT-SLAM, les points suivants sont à résoudre, aborder ou améliorer :



(a) Image dans laquelle l'amer est recherché, dont l'emplacement est indiqué par le carré rouge.

(b) Apparence de l'amer à toutes les positions recherchées.

FIGURE 7.7 – Illustration de la recherche d'un amer dans une image panoramique.

— Il est nécessaire d'adapter cette méthode au fait qu'une recherche active est utilisée pour le SLAM, le plus simple étant de conserver une approche globale similaire mais effectuée en tâche de fond sur un sous-échantillon d'images. On pourra également considérer l'utilisation d'autres descripteurs.

— Il serait plus fiable et efficace de tirer partie d'algorithmes de reconnaissance de lieux tels que FabMap [Cummins et Newman, 2008] ou en utilisant le Lidar Velodyne [Muhammad et Lacroix, 2011] pour diriger la recherche des amers à retrouver.

— Il est également nécessaire d'étendre le fonctionnement au cas multirobots.

c Association de cartes basée modèle

L'association de cartes peut également se réaliser grâce à la correspondance entre des modèles d'environnement (notamment des modèles numériques de terrain).

Ce type d'association présente des avantages par rapport à l'association basée sur les amers, à savoir qu'il est plus immédiat à mettre en œuvre, et moins dépendant du type de robot et du point de vue (par exemple entre un robot terrestre et un robot aérien). En revanche il sera moins précis dans des environnements texturés mais avec peu de relief.

Les associations de cartes basée amers et basée modèle sont donc complémentaires, mais pour autant il faudra prêter attention à leur utilisation combinée. Par exemple il ne faudra pas les utiliser simultanément pour associer deux mêmes cartes, car ils ne sont pas du tout indépendants (les deux dépendent de la même localisation au sein de la carte), mais par exemple simplement conserver la meilleure estimation des deux.

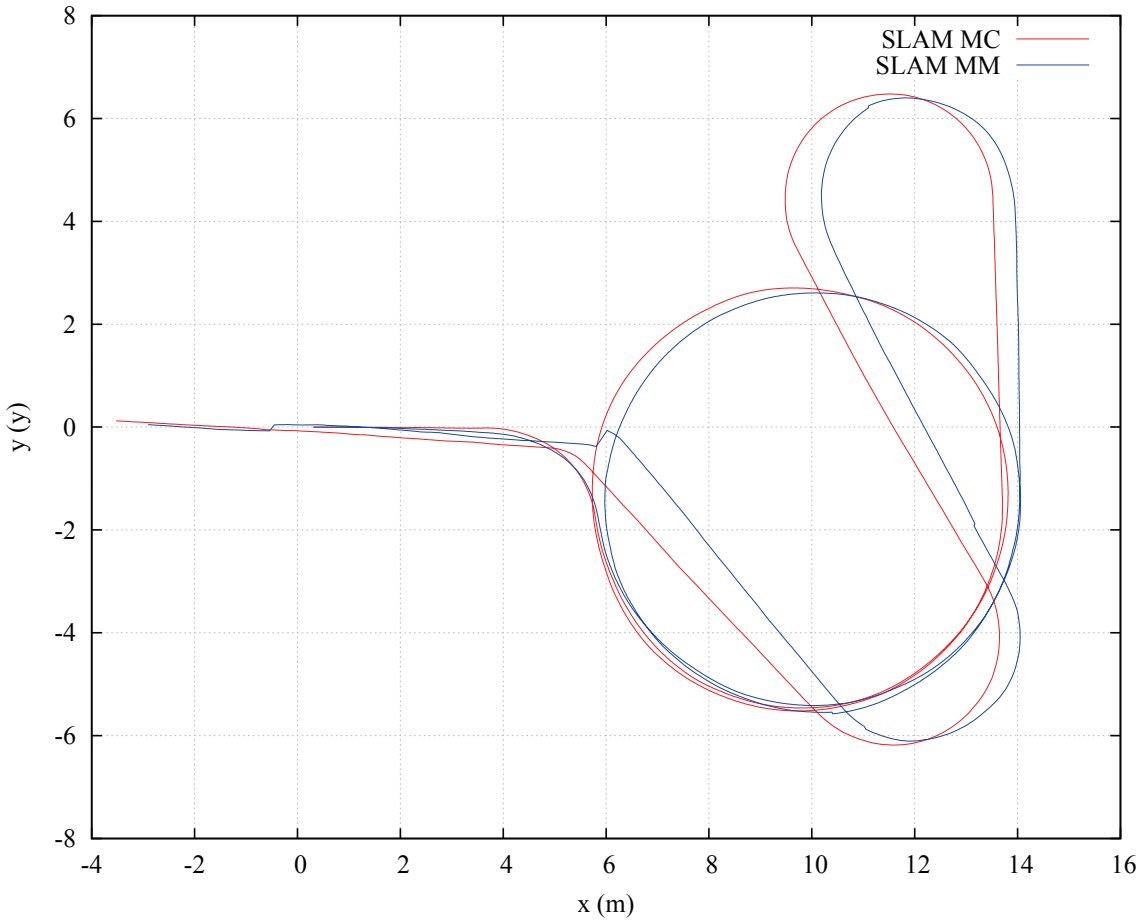


FIGURE 7.8 – Résultats de SLAM hiérarchique (MM pour MultiMap) par rapport à du SLAM de type odométrie visuelle (MC pour Mémoire Courte).

d Observation mutuelle

Ce type d'évènement a déjà été abordé pour une intégration directe dans le filtre, sections 6.3 page 146 et 6.4.1 page 152 pour des observations complètes, ou section 6.4.2 page 153 pour des observations partielles de type faisceau.

Sa gestion est plus complexe dans le cadre du SLAM hiérarchique, car on a un besoin similaire aux localisations absolues de fusionner plusieurs mesures pour obtenir une unique transformation entre deux repères, mais ces deux repères sont cette fois deux cartes locales à optimiser, et non une carte locale à optimiser et un repère global fixe.

Plusieurs cas se présentent, selon le type d'observation disponible, nécessitant des résolutions plus ou moins complexes :

— Si les observations sont complètes, c'est-à-dire qu'elles donnent une équivalence avec l'état complet du robot (position et orientation), alors chaque observation permet à elle seule de générer une hypothèse complète de transformation entre les deux cartes locales. Il suffit ensuite d'effectuer une moyenne pondérée avec les incertitudes pour toutes les observations ayant eu lieu pendant la vie commune des deux cartes locales.

— Si les observations sont partielles, mais consistent simplement en une sous-partie de l'état complet du robot (par exemple uniquement la position et pas l'orientation), il est alors nécessaire de passer par une procédure de minimisation, qui peut se faire dans l'espace de l'état du robot. Pour une hypothèse de transformation entre les deux cartes, on calcule directement la distance de Mahalanobis entre les observations et la trajectoire du robot, comme pour la localisation absolue. Il est alors possible de calculer le gradient de cette fonction de coût afin de la minimiser, pour obtenir la transformation qui maximise l'adéquation des observations avec la trajectoire. L'existence d'une valeur initiale fournie par le graphe avant optimisation devrait assurer de ne pas tomber dans un minimum local.

— Si les observations sont partielles et ne sont pas une simple sous-partie de l'état complet du robot (par exemple un faisceau, qui est une observation angulaire), il est alors nécessaire d'effectuer la minimisation dans l'espace des observations, comme le fait par exemple un filtre de Kalman. Pour une hypothèse de transformation entre les deux cartes, on transforme alors la trajectoire du robot observé en des observations attendues en appliquant la fonction d'observation, et on cherche à minimiser la distance entre ces observations attendues et celles obtenues (correspondant à l'innovation), toujours pas descente de gradient. On retombe en réalité sur une technique d'ajustement de faisceaux (*Bundle Adjustment*).

Bien entendu les méthodes les plus complexes peuvent être appliquées aux cas plus simples, y compris au cas de la localisation absolue, seule la fonction d'observation différent dans les différents cas.

Notons que si au cours des mêmes cartes locales le robot A observe le robot B, et le robot B observe le robot A, on pourra parfaitement ajouter deux arêtes distinctes dans le graphe entre les mêmes nœuds.

Rendre automatique ces observations mutuelles est un autre travail, pas nécessairement plus aisé. Nous avons à cette fin développé une bibliothèque de détection visuelle nécessitant d'équiper les robots de taches de couleur, décrite dans l'annexe B page 197.

7.4.5 Gestion des modèles d'environnement

La création directe d'un modèle d'environnement global nécessiterait une localisation globale temps-réel avec une précision de l'ordre du centimètre, qui concrètement n'est encore pas atteignable. En effet en pratique toutes les solutions de localisation globale précises reposent sur la combinaison d'une localisation relative, effectivement localement précise à l'ordre du centimètre mais qui dérive, et d'une localisation absolue qui la recale occasionnellement en générant des discontinuités (ne serait-ce qu'une fermeture de boucle), qui sont incompatibles avec la construction d'un modèle d'environnement.

L'idée pour construire des modèles d'environnement sur le long terme est donc de s'appuyer sur la structure de SLAM hiérarchique utilisée pour la localisation, en créant pour chaque modèle des sous-cartes, calquées sur les sous-cartes du SLAM, et dont la position globale est donc ajustée en même temps. Si comme évoqué section 7.4.3.a page 187 on intègre au sein des cartes locales uniquement les capteurs qui dérivent et ne produisent pas de discontinuité, on obtient alors une localisation continue, à la précision bornée, parfaitement adaptée à la construction de modèles d'environnement.

Un soin particulier devra en revanche être porté aux zones de transition entre cartes, afin de toujours permettre leur exploitation notamment pour les fonctions de navigation.

7.4.6 Communication et décision

L'extension de la localisation à la coopération multirobots pose des problèmes intéressants de communication entre les robots, en terme de volume de données et de disponibilité. L'idée générale est de n'échanger que le minimum d'information, à savoir le niveau du graphe global, afin que tous puissent profiter d'une localisation optimale à moindre coût.

Cependant la réalisation d'appariemment de cartes (basés amers ou modèles) nécessite également l'échange d'informations sur le contenu des cartes, comme précisé section 7.4.4 page 188. Ces informations sont plus lourdes, et leur échange pourra devoir être ciblé selon les capacités de l'infrastructure de communication, menant à des problématiques intéressantes.

Par ailleurs, la localisation peut devenir un processus actif. Un robot, voyant l'incertitude de sa localisation devenir trop grande, pourrait décider de mettre temporairement de côté sa mission afin d'agir pour améliorer la précision de sa localisation, en cherchant par exemple à rencontrer d'autres robots afin de fermer des boucles.

7.4.7 Conclusion

Les principaux problèmes relatifs à la localisation au sein d'une équipe de robots sont donc identifiés, et des solutions sont envisagées, mais il n'est bien sûr pas exclu que l'implémentation concrète mette en évidence de nouveaux problèmes.

Deux doctorants ont commencé à travailler sur ces aspects au sein du laboratoire : Ellon Mendes sur l'intégration du SLAM hiérarchique dans RT-SLAM, et Pierrick Koch sur la gestion des modèles d'environnement.

Annexe A

Liste des publications

C. Roussillon, A. Gonzalez, J. Solà, J-M Codol, N. Mansard, S. Lacroix, M. Devy, *RT-SLAM : a generic and real-time SLAM architecture*, International Conference on Vision Systems, Sophia Antopolis (France), 2011

R. Boumghar, C. Roussillon, A. Degroote, P. Cox, V. Delsart, B. Vandeportaele, M. Herrb, S. Lacroix, *Over the hill and far away : aerial/ground cooperation for long range navigation*, International Workshop on Robotics for risky interventions and Environmental Surveillance-Maintenance, Leuwen (Belgique), 2011

C. Roussillon, S. Lacroix, *Architecture logicielle pour la localisation des robots*, Congrès ED-SYS, Toulouse (France), 2011

C. Roussillon, S. Lacroix, *High rate-localization for high-speed all-terrain robots*, International Conference on Communication, Computing and Control Applications, Marseille (France), 2012

C. Roussillon, C. Robin, A. Maligo, A. Degroote, P. Koch, E. Mendes, S. Lacroix, *LAAS RIS team entry to the euRathlon 2013 Autonomous Navigation scenario*, euRathlon Workshop on Field Robotics, Berchtesgaden (Germany), 2013

Dans le but de faciliter les localisations inter-robots, présentation d'une bibliothèque de détection d'objets colorés par vision. Une première passe invariante en rotation basée sur les histogrammes de couleur permet d'éliminer la plupart des hypothèses. Une deuxième passe vérifie ensuite la topologie interne de l'objet pour éliminer les faux positifs.

Annexe B

Bibliothèque de détection de cibles

Sommaire

Dans le cadre du projet ACTION, nous avions besoin de détecter et identifier depuis le robot aérien à la fois le robot terrestre et d'autres cibles. Ces traitements (dits de DRI : détection, reconnaissance et identification) ne faisaient pas partie du projet, mais nous avions besoin d'une solution pratique et réaliste pour les accomplir afin de permettre la localisation de ces éléments, et à terme la mise en œuvre d'une approche de localisation distribuée multi-robots.

Le problème de reconnaître avec fiabilité un objet dans des images étant complexe, en particulier dans les conditions difficiles dans lesquelles nous nous trouvons (petite taille de l'objet – entre 15 et 50 pixels –, flous de bougé dus aux vibrations de l'hélicoptère, saturations locales et ombres lors des expositions au soleil, baisse de contraste et augmentation du bruit lorsque la luminosité diminue, etc), nous avons donc décidé de simplifier cette fonction en équipant les véhicules à détecter de couleurs vives et le robot aérien d'une caméra couleur. Une forte importance est également accordée à la vitesse de traitement car on ne souhaite pas charger le processeur du robot aérien dont ce n'est pas la tâche principale.

L'algorithme complet est constitué de trois étapes :

1. La détection initiale, qui utilise un algorithme basé sur l'appariement d'histogrammes de couleur. Il n'est pas parfait, notamment car il recherche d'un seul coup toutes les orientations possibles : il exploite donc une fenêtre carrée, et doit gérer le fait qu'il y a un peu d'arrière-plan inconnue dans sa fenêtre. Mais en contrepartie il est très rapide et permet d'éliminer une énorme proportion des pixels candidats de l'image. On le règle donc pour avoir un taux de détection proche de 100%.
2. La détection initiale fournit une image résultat constituée du score de chaque pixel, mais il reste à extraire des zones un peu plus grandes de cette image où le score a été suffisant, pour constituer des hypothèses de présence d'objet.
3. Puisque les premières étapes n'étudient que la statistique globale de teinte dans l'image, il reste quelques faux positifs qu'il convient d'éliminer. Ceci peut se faire avec un algorithme plus complexe et plus lent puisqu'on ne l'applique que sur un très petit nombre d'hypothèses. Cet affinement de la détection peut également permettre d'extraire des informations d'orientation et de taille plus précises, en étudiant la topologie interne de l'objet.

Évaluations

Les différentes évaluations comparatives qui suivent utilisent toutes un jeu de données issu de captures par l'hélicoptère Ressac de l'Onera, l'objet à détecter étant le robot Mana équipé de zones colorées. Ce jeu de données a été acquis au cours de six manipulations réparties sur 3 différentes sorties, à des lieux différents, et avec des conditions d'illumination différentes, et représentant chacune des quantités d'exemples à peu près équilibrées. Ce jeu de données représente environ 3600 images, dont 1200 contenant l'objet à rechercher. La base d'apprentissage comporte quant à elle 70 exemples. Les images sont parfois de mauvaises qualité, à cause de l'illumination ou de flous de bougé, comme l'illustre un extrait de la base d'apprentissage figure B.1 .

Deux types de taux de fausse alarme seront utilisés : pour le détecteur initial, qui teste chaque pixel, on travaillera donc au niveau du pixel. Les taux de fausse alarme correspondront à la

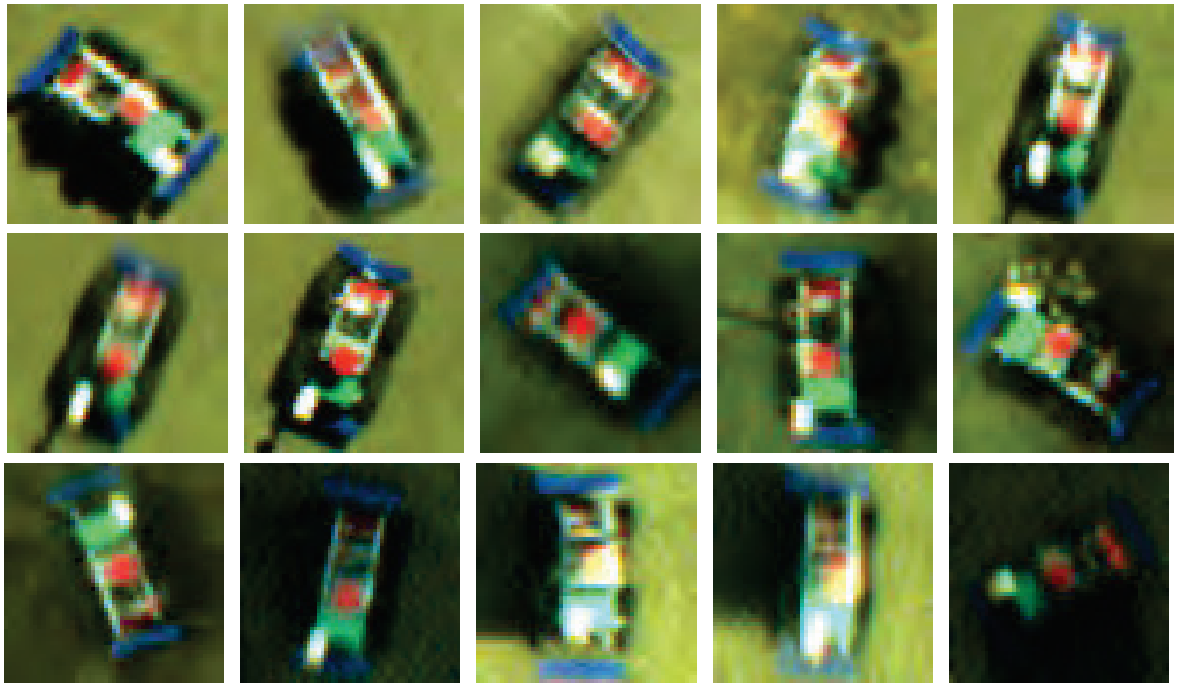


FIGURE B.1 – Échantillon de la base d'apprentissage.

proportion de pixels testés dont la fenêtre ne touche pas l'objet à détecter mais qui ont quand même donné une réponse positive. Ainsi quelques pourcents peuvent représenter un certain nombre de pixels. Pour les étapes suivantes, on travaillera au niveau de l'objet, un faux positif étant un signalement d'objet détecté alors qu'aucun n'a été étiqueté à cet endroit. On pourra cette fois avoir des taux de fausse alarme supérieurs à 100%, si plusieurs réponses fausses différentes sont données dans chaque image. Les taux de détection sont dans tous les cas au niveau objet.

B.1 Détection

B.1.1 Construction

Nous utilisons pour la construction des histogrammes l'algorithme *Distributive Histogram* qui est introduit dans [Sizintsev *et al.*, 2008] comme le plus efficace. Le principe est de construire les histogrammes dans une fenêtre glissante sur l'image, en maintenant un histogramme par colonne de l'image servant à la mise à jour de l'histogramme de la fenêtre. La figure B.2 page suivante illustre le mécanisme d'actualisation lorsque l'on décale la fenêtre d'un pixel vers la droite : on commence par mettre à jour l'histogramme de la nouvelle colonne, en ajoutant le pixel du bas et en retirant le pixel du haut, puis on retire l'histogramme de la colonne de gauche à l'histogramme de la fenêtre et on lui rajoute celui de la colonne de droite. On a ainsi une complexité algorithmique qui ne dépend pas de la taille de la fenêtre mais uniquement de façon linéaire de la taille de l'histogramme (et bien sûr également de façon linéaire de la taille de l'image pour une recherche exhaustive).

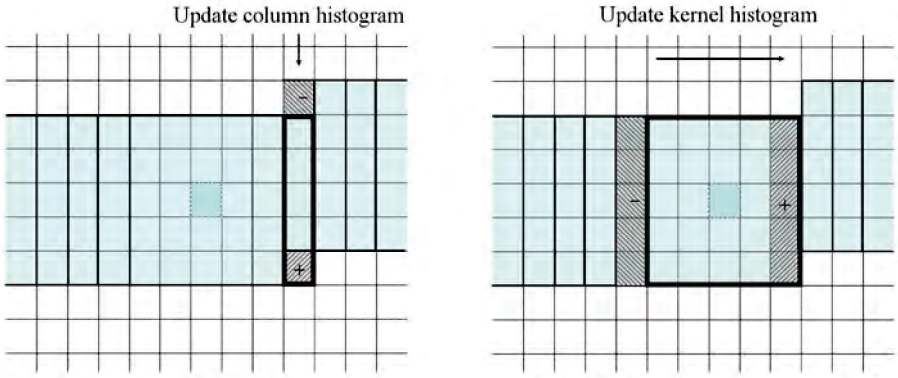


FIGURE B.2 – [a] Illustration du principe de construction d'histogramme distributif.

a. Illustration issue de [Sizintsev *et al.*, 2008].

B.1.2 Classification

Construire un histogramme nécessite tout d'abord de définir ses classes (ou cases).

Le premier paramètre est l'espace dans lequel elles seront définies. Pour travailler dans les couleurs on préfèrera se concentrer sur la teinte, qui présente la plus grande invariance aux conditions de luminosité.

Le second paramètre est la discrétisation de cet espace pour définir les classes de l'histogramme. Habituellement on divise simplement l'espace en un certain nombre de classes de taille égale, ce qui permet de classifier par une simple division entière, et on peut également choisir de préférence un nombre de classes puissance de 2, afin de pouvoir classifier par un décalage de bits encore plus rapide qu'une division.

Afin de pouvoir réaliser des classifications à la fois plus complexes et précises, tout en restant très rapides à calculer, nous avons implémenté un classifieur basé sur une table de correspondance entre la valeur du pixel et l'indice de la classe qui lui correspond. Afin de limiter la taille de la table, on commence également par diminuer la résolution de la valeur du pixel, de manière analogue à la classification directe habituelle (en choisissant une puissance de 2), donc l'étape supplémentaire de lecture de la table fait que la classification est un peu plus longue. Mais la puissance de la table de correspondance permet de compenser cela par d'autres gains de temps de calcul plus importants. Par exemple puisque l'on veut travailler dans l'espace des teintes, et que les images sont fournies dans l'espace RGB (*Red, Green, Blue*), il faut normalement réaliser une conversion, qui représente plusieurs opérations arithmétiques pour un pixel. Mais la table de correspondance peut définir ses entrées directement dans l'espace RGB et inclure la conversion en teinte (à l'approximation de la diminution de résolution près, qui est largement acceptable), ce qui élimine complètement cette étape qui était la plus coûteuse.

Mais surtout cela permet de complexifier la construction des classes autant qu'on le souhaite sans modifier le temps de calcul. Par exemple la définition de la teinte est très instable lorsque la saturation est faible (un peu de bruit peut la faire changer de beaucoup). On peut donc sans coût supplémentaire prendre également en compte la saturation, pour soit créer une classe spéciale « sans couleur », correspondant aux pixels de faible saturation (classification

appelée *HueLowSat*), soit tout simplement éliminer ces pixels de l'histogramme (classification appelée *HueHighSat*). Par rapport à une simple discrétisation de l'espace des teintes (classification appelée *Hue*), cela permet d'améliorer nettement les performances, comme le montre la figure B.3 (l'algorithme complet utilisé ici est détaillé dans la suite du chapitre).

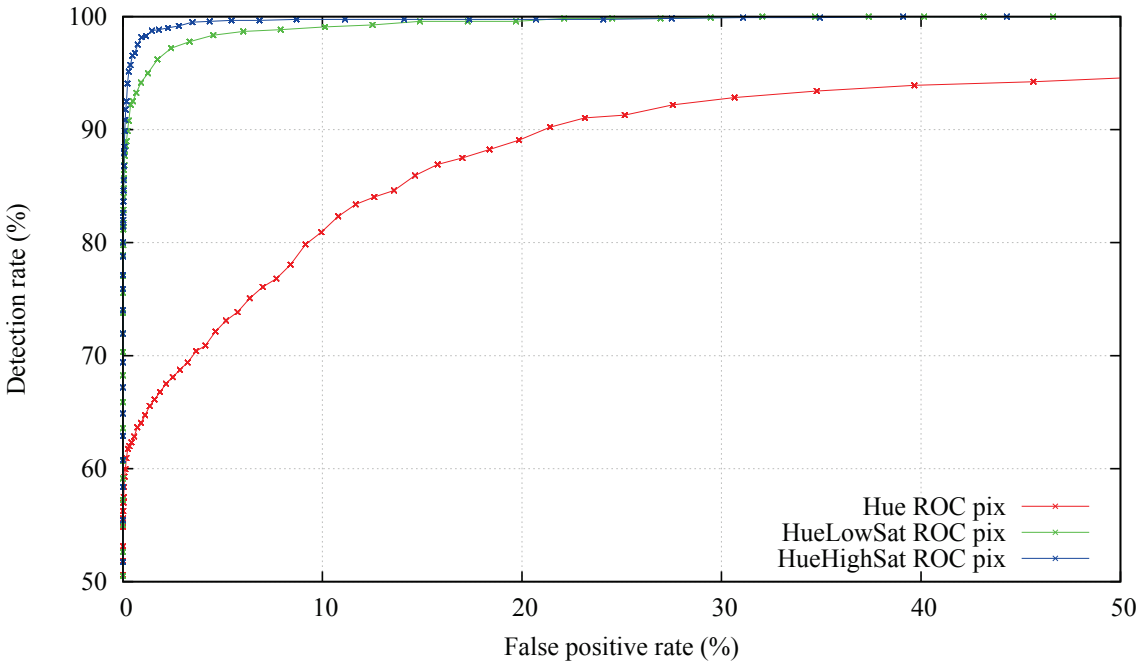


FIGURE B.3 – Comparaison des performances de l'algorithme de détection avec différentes classifications pour la construction de l'histogramme.

Afin de ne pas répéter la classification des pixels (qui arriverait deux fois sur un balayage de l'image, mais encore plus si on balaie à plusieurs échelles), on convertit directement l'image de couleurs en image de numéros de classe, et on ne travaille ensuite plus qu'avec cette dernière.

B.1.3 Distance

Une fois l'histogramme de la fenêtre obtenu, il faut le comparer à celui du modèle à détecter, avec une mesure de similarité d'histogramme telle que l'une de celles fournies dans OpenCV : corrélation, intersection, norme L1, L2 ou infinie, statistique du χ^2, ... La précision de la détection est très sensible au choix de cette distance.

Nous avons décidé pour plus de robustesse d'apprendre l'histogramme du modèle à détecter à partir de plusieurs exemples, en faisant la moyenne mais aussi surtout en calculant les écarts-types de chaque classe entre tous ces exemples, puis en l'utilisant comme coefficient de normalisation dans les distances L1 et L2. Cela permet d'ajuster l'importance accordée à chaque case de l'histogramme en fonction de l'importance réelle qu'elle doit avoir, et cela améliore nettement les performances, comme illustré figure B.4 page suivante.

On constate que les performances dépendent énormément de la distance choisie. Les distances normalisées L1n et L2n qui utilisent l'information supplémentaire d'écart-type sont évidemment beaucoup plus performantes, mais les distances L1 et L2 non normalisées sont déjà bien

placées. De plus la prise en compte des écarts-types étaient plus délicats à définir pour les autres distances. La distance L1 normalisée se montre donc la plus performante, et est par ailleurs une des plus rapides à calculer (plus rapide que L2n, la valeur absolue s'effectuant par un simple masquage de bit).

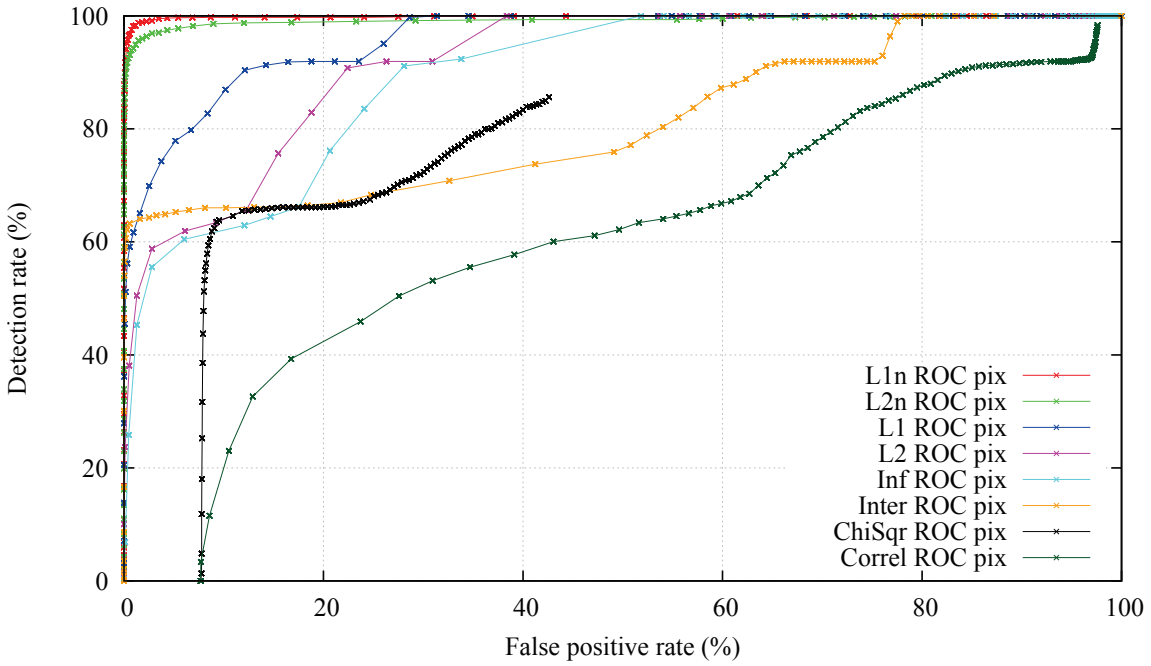


FIGURE B.4 – Comparaison des performances de l'algorithme de détection avec différentes distances utilisées pour la comparaison des histogrammes.

Remarque Afin d'éviter un surapprentissage, on impose un écart-type minimal sur les cases à l'issue de l'apprentissage. Le cas typique où cela est indispensable est lorsqu'une classe n'a jamais été présente lors de l'apprentissage, mais où elle se retrouve présente sur le fond lors de la détection. On ne veut alors pas que cela soit éliminatoire.

B.1.4 Améliorations

a Optimisations

La première optimisation a été de réimplémenter manuellement avec un soucis de performance les calculs d'histogramme et de distance entre histogrammes, notamment en prenant soin d'utiliser *templates* et fonctions *inline* quand cela était nécessaire afin d'effectuer le plus possible de décisions au moment de la compilation au lieu de l'exécution, et d'éviter les appels trop fréquents de fonctions. Par rapport à l'implémentation initiale utilisant au maximum les fonctions d'OpenCV, les temps de traitement d'une image VGA sont passés de l'ordre de 500 ms à 30 ms sur un processeur récent (Core i7 à 3.3GHz).

L'optimisation suivante a été de faire appel explicitement aux fonctions de vectorialisation SSE des processeurs afin de réaliser des calculs en parallèle, que ce soit pour la construction

de l'histogramme ou le calcul de la distance, ce qui a permis de diviser encore par deux le temps de traitement d'une image.

D'autres optimisations mineures, telles que le fait de normaliser le seuil de décision par rapport à la taille de la fenêtre plutôt que chaque pixel, ou encore le choix d'une distance plus rapide à calculer, ont permis de ramener le temps de traitement d'une image VGA à environ 8 ms. Ce temps de calcul se répartit pour environ 3 ms dans les calculs de distance, 2 ms dans les mises à jour des histogrammes des colonnes, et 3 ms dans les mises à jour des histogrammes des fenêtres.

Ces résultats montrent que prendre le temps d'écrire du code optimisé (ce qui n'est pas non plus si long, le temps total consacré au développement de ce détecteur n'étant que de quelques semaines) est largement payant, puisque le temps de calcul a été divisé par près de 60 avec le même algorithme, ce qui à cet ordre de grandeur change qualitativement ce qu'il est possible de faire.

Enfin nous travaillons toujours à pleine résolution, mais notons que travailler de façon hiérarchique, ou au moins avec une première passe ne traitant que la moitié des lignes et des colonnes, permettrait de diviser le temps de calcul par près de 4, sans dégrader significativement les résultats puisque l'on exigera par la suite qu'au moins un quart des pixels de la fenêtre de détection aient répondu positivement.

b Élimination de l'arrière-plan

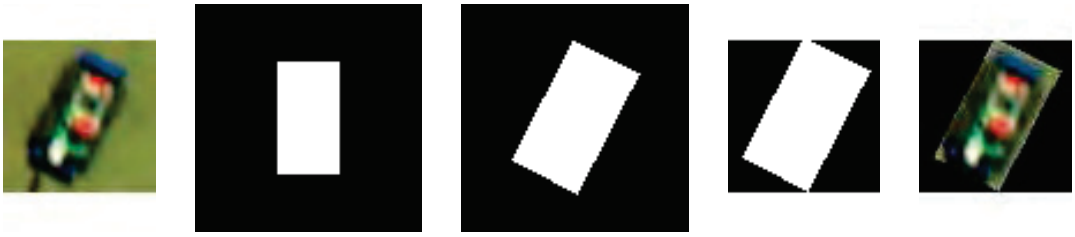
La présence de l'arrière-plan dans les fenêtres d'apprentissage et de détection est le principal facteur limitant de la précision de l'algorithme. En effet l'arrière-plan n'est pas contrôlé et son histogramme s'ajoute à celui de l'objet recherché, en éloignant l'histogramme calculé de la fenêtre entière de celui de l'objet. De plus puisque l'on souhaite faire une recherche invariante sur l'orientation de l'objet, on est obligés d'utiliser une fenêtre carrée, et selon la forme de l'objet la proportion d'arrière-plan présent dans la fenêtre pourra être très importante. Pour un objet rectangulaire deux fois plus long que large comme notre robot cette proportion atteint par exemple 50%.

Nous avons donc cherché des moyens de limiter son influence.

A l'apprentissage L'arrière-plan est plutôt simple à éliminer sur les exemples d'apprentissages, puisque l'on n'a pas de contrainte de temps, et que l'on bénéficie d'un étiquetage précis des données (présenté section B.5.2 page 216), notamment orientation et taille. Il suffit alors à partir d'un masque du profil de l'objet à détecter, de le mettre à la bonne échelle et orientation pour définir de façon propre l'arrière-plan à ne pas prendre en compte (figure B.5 page suivante).

A la détection La situation plus problématique durant l'étape de détection, d'une part car on ne dispose pas de vérité terrain, et d'autre part car on est limité en temps de calcul.

Une première idée est d'estimer l'histogramme de l'arrière-plan en calculant celui d'une bande



(a) Imagette (b) Masque (c) Après transfo. affine (d) Après découpage (e) Imagette masquée

FIGURE B.5 – Étapes de masquage pour l'élimination de l'arrière plan sur exemple d'apprentissage étiqueté.

autour de la fenêtre (pour optimiser les calculs, on peut simplement réutiliser les histogrammes colonnes de l'algorithme d'histogramme distributif – section B.1.1 page 199 – à gauche et à droite de la fenêtre). On multiplie ensuite l'histogramme obtenu par un coefficient dépendant de la proportion attendue d'arrière-plan (en fonction de la forme de l'objet), puis on le retranche de l'histogramme de la fenêtre. Cette méthode est donc basée sur l'hypothèse que le fond est relativement uniforme autour de l'objet, et représente un temps de calcul significatif.

Une autre idée est de modifier la distance pour ne pénaliser que les manques dans l'histogramme de la fenêtre, et pas les excès, en considérant qu'ils sont dus à l'arrière-plan. On peut donc s'attendre à augmenter le nombre de faux positifs, mais comme on améliore les propriétés de généralisation de l'algorithme on peut être plus exigeant sur les seuils pour compenser. Le coût en temps de calcul est de plus nul, et c'est la solution que nous avons retenue, dont les résultats sont illustrés figure B.6 , et montrent que les performances sont à nouveau fortement améliorées, par rapport à l'absence d'élimination de l'arrière-plan, ou à son élimination uniquement à l'apprentissage mais en continuant à utiliser la distance L1n. Notons qu'avec cette distance c'est désormais la classification *HueLowSat* qui donne des résultats légèrement meilleurs à cette étape, mais plus significativement pour les suivantes, et que nous allons donc désormais conserver.

c Étirement d'histogramme

Il est également possible d'utiliser l'algorithme sur des images en niveaux de gris, ce que nous avons initialement testé, même si la précision est moins bonne. Dans ce cas pour améliorer la généralisation de l'apprentissage et être plus robuste aux variations d'illumination il est intéressant d'opérer un étirement de l'histogramme avant la comparaison avec le modèle.

Pour cela on calcule le support de l'histogramme en éliminant les classes aux extrêmes qui sont vides, puis on l'étire en l'interpolant grâce aux coefficients se trouvant dans une table de correspondance précalculée (s'il y a n classes, cette table ne comporte que n^2 entrées – combinatoire entre la taille du support et le numéro de case à remplir – avec 4 valeurs pour chaque entrée – les 2 cases à interpoler avec les 2 coefficients correspondant). Le calcul ainsi optimisé n'est que deux fois plus long que sans étirement, mais on a intérêt juste après le calcul de la taille du support à ne pas faire du tout la suite des calculs (étirement proprement dit mais aussi calcul de la distance au modèle) si ce support est trop petit et donc non représentatif.

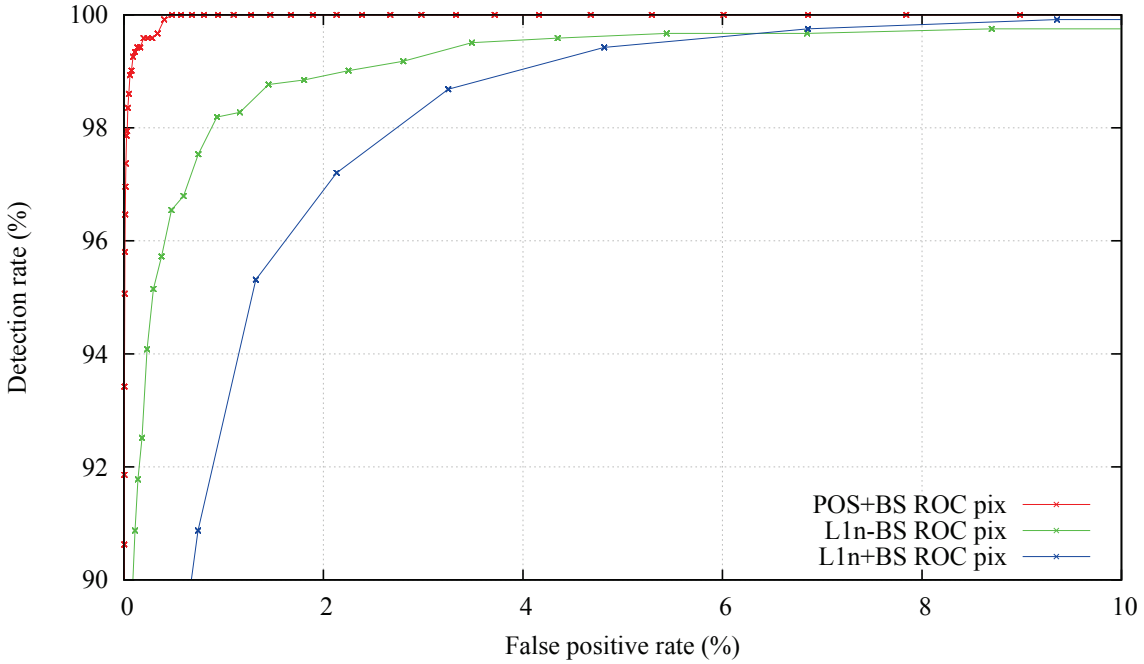


FIGURE B.6 – Comparaison des performances de l'algorithme de détection avec ou sans élimination de l'arrière-plan à l'apprentissage (BS pour *Background Substraction*).

Cela permet à la fois d'éliminer complètement le surcoût en temps de calcul, et d'améliorer encore un peu la précision en diminuant le nombre de faux positifs.

Un élément intéressant est que la précision semble meilleure si on ne renormalise pas l'histogramme après étirement, car la renormalisation génère plus de faux positifs sans améliorer le taux de détection. Cela demanderait plus ample vérification, notamment en vérifiant qu'il ne s'agit pas de surapprentissage à cause d'une variété d'exemples insuffisant, problème sur lequel nous ne nous sommes pas attardés puisque notre utilisation réelle est basée sur les histogrammes de teinte qui n'appliquent pas cette fonctionnalité.

B.2 Extraction

Le résultat de l'étape précédente est une image dont les pixels contiennent le résultat de la détection en chaque point, comme illustré figure B.7(b) page suivante. Il est donc nécessaire d'extraire de ces images les zones correspondant à la présence d'un objet. Deux algorithmes ont été développés pour répondre à cette tâche.

B.2.1 Extraction unique

La première version est simple et suppose qu'un seul objet au maximum se trouve dans l'image. Il suffit alors d'extraire la tache principale de l'image. Pour cela on démarre d'une zone correspondant à l'ensemble de l'image, et à chaque fois on calcule le barycentre des pixels ayant dépassé le seuil de détection, puis on divise la taille de la zone par deux autour de ce

barycentre, en vérifiant qu'à chaque étape on ne perd pas plus de la moitié des points, jusqu'à avoir atteint la taille de la fenêtre.

L'avantage de l'algorithme est qu'il est immédiat à implémenter et rapide à exécuter, l'inconvénient en plus du fait qu'il ne permet pas de détecter plusieurs objets dans l'image est qu'il verra son taux de détection chuter si un faux positif significatif se trouve également dans l'image, alors que l'étape d'affinement l'aurait supprimé.

B.2.2 Extraction multiple

La seconde version consiste donc à identifier les zones connexes dépassant le seuil, afin de pouvoir détecter plusieurs taches de réponse positive dans l'image. On utilise pour cela l'algorithme du remplissage par inondation (*flood fill*) :

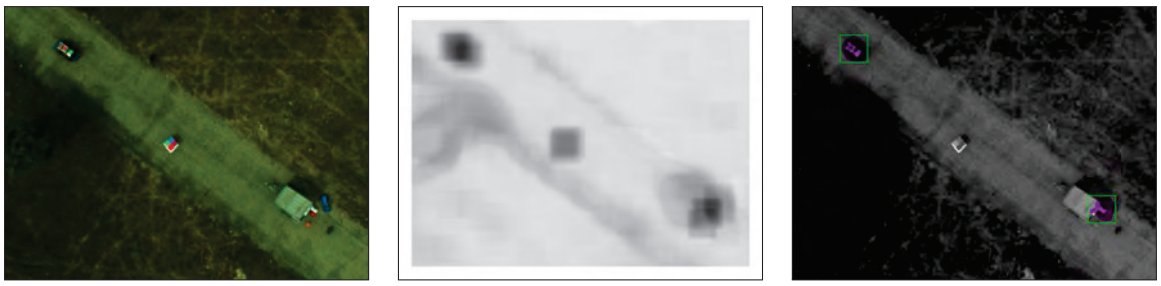
— On balaie l'image, jusqu'à trouver un pixel dépassant le seuil pour commencer à explorer une tache.

— À partir de ce pixel on entame une recherche en largeur à l'aide d'une file parmi les voisins qui dépassent également le seuil (on parle ici des quatre voisins immédiats), tout en calculant la boîte englobante (*bounding box*) de cette tache. On utilise une table pour marquer les pixels déjà explorés et éviter de les tester plusieurs fois.

— On reprend le balayage de l'image là où on l'a laissé, jusqu'à trouver un nouveau pixel (non déjà exploré) dépassant le seuil pour recommencer.

— On filtre ensuite les taches avec des contraintes de taille (minimale, et maximale par rapport à la taille de l'objet recherché) et de densité de remplissage.

Le traitement d'une image VGA prend entre 1 et 2 ms selon la quantité de pixels à extraire. Une optimisation en prégroupant les pixels par tranches contigues sur la même ligne permettrait d'accélérer un peu le traitement.



(a) L'image originale. (b) Résultat de la détection initiale. (c) Résultat de l'extraction d'objets.

FIGURE B.7 – Résultat de détection du robot Mana à différentes étapes sur un exemple. Le résultat de la détection initiale par histogramme et ensuite seuillé (pixels roses), puis les taches sont extraites (carré vert). Il y a une vraie détection, et un faux positif.

On obtient alors les résultats présentés figure B.8 . Le passage au mode objet, où le taux de fausse alarme n'est plus normalisé par le nombre total de pixels testés mais par le nombre d'images, donne une impression de dégradation des performances. Mais même à 10% de fausse

alarme (pour 90% de détection), cela ne représente qu'une image sur 10 qu'il est facile d'analyser à l'emplacement spécifique donné avec un algorithme plus évolué pour éliminer ces faux positifs. On peut également améliorer les scores si on peut faire l'hypothèse qu'il n'y a qu'un seul objet au maximum présent dans l'image. La figure B.7(c) illustre un cas de détection et de faux positif.

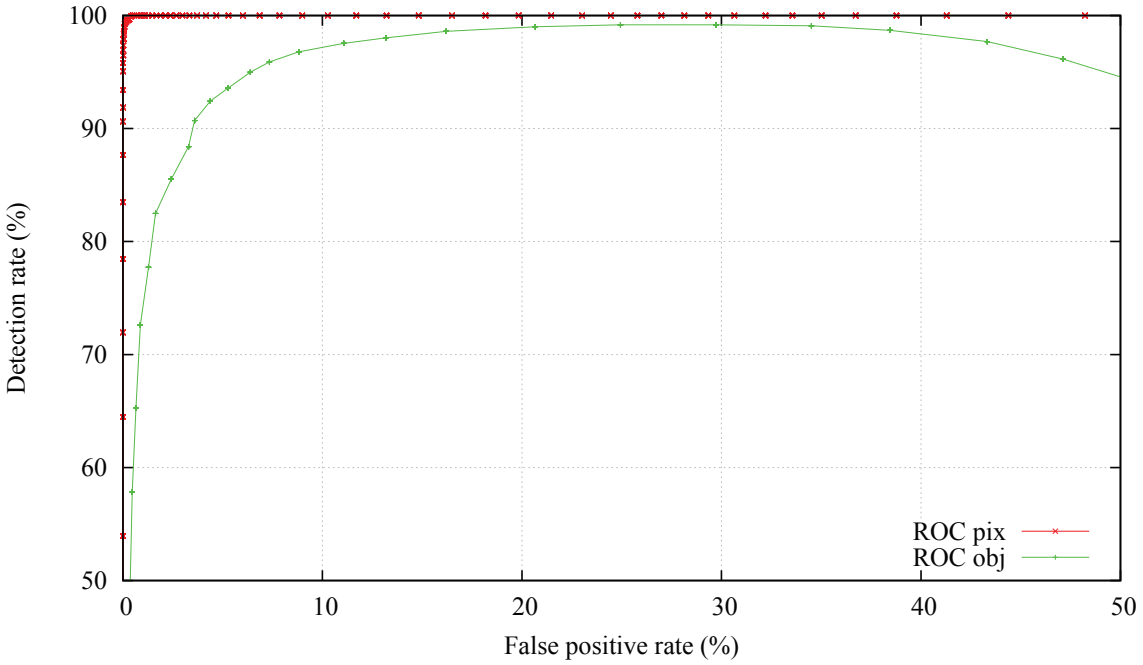


FIGURE B.8 – Illustration des performances de l'algorithme de détection aux niveaux pixel et objet. Le taux de fausse alarme au niveau objet peut être supérieure à 100% car plusieurs réponses positives peuvent être données par image. Le taux de détection finit également par décroître quand on augmente le seuil, car les taches autour du robot peuvent se mettre à déborder sur le décor et devenir trop grandes.

B.3 Affinement

Le but de l'étape d'affinement est à la fois de confirmer le résultat de la détection initiale par une analyse topologique de l'intérieur de la fenêtre, ainsi que d'estimer l'orientation et l'échelle de l'objet dans la fenêtre plus précisément.

B.3.1 Corrélation

La première version de l'algorithme d'affinement de la détection est basée sur la corrélation dans l'espace des teintes d'un template avec le candidat, en explorant toutes les rotations possibles dans le plan de l'image ainsi que de l'échelle dans une certaine mesure, par rapport à la taille de la fenêtre de détection. Cette exploration se fait dans un premier temps exhaustivement avec une résolution limitée (par exemple 12.5 ° pour les rotations), puis par dichotomie pour améliorer la résolution, afin de limiter le temps de calcul.

Les résultats ne sont cependant pas convaincants, tant du point du vue du taux de détection et d'une très grande sensibilité au seuil de décision rendant le réglage très difficile, que de la précision de l'orientation et échelle extraites, notamment à cause des changements d'apparences dus au changement de point de vue sur l'objet (figure B.9). Mais surtout cette méthode s'est révélée très lente du fait de l'exploration exhaustive sur l'ensemble des pixels nécessaire (environ 100 ms par candidat ; même si quelques optimisations étaient encore possibles, nous serions restés largement au dessus des 8 ms de la détection initiale).

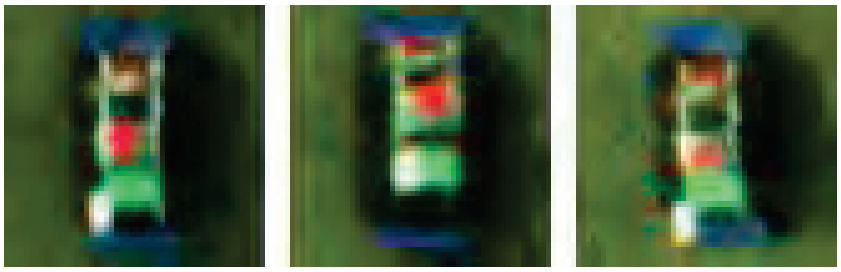


FIGURE B.9 – Exemple de changements d'apparence du robot Mana vu de différents points de vue depuis un hélicoptère (à gauche : à la verticale au centre de l'image ; au milieu : vu de l'arrière en haut de l'image ; à droite : vu de l'avant en bas de l'image). On remarquera en particulier le grand déplacement relatif de la tache rouge au centre située sur son capteur Vélodyne, plus grand que la taille de la tache elle-même, ou encore de l'antenne GPS en blanc à l'arrière gauche, situés plus en hauteur.

B.3.2 Taches de couleur

a Algorithme

Nous avons donc mis au point un second algorithme complètement différent, dont l'objectif était d'être plus rapide en ne réalisant qu'un seul traitement sur l'ensemble des pixels et la partie recherche)sur un modèle simplifié, et étant également plus souple pour faire face aux légers changements d'apparence constatés.

On procède aux étapes suivantes pour la partie confirmation de détection :

— On segmente l'image en taches de couleur similaires. Pour cela on applique un algorithme d'agglomération des pixels qui ont une saturation suffisante (en l'occurrence à nouveau l'algorithme des k-moyennes), en utilisant la distance L2 dans l'espace (u, v, hue) (où u et v sont les coordonnées en pixels dans l'image, et hue la teinte du pixel), afin d'obtenir des agglomérations à la fois compactes et de teinte proche (figure B.10(b)). La sensibilité au nombre de clusters choisis et faible car c'est le fond qui est le moins déterminé à cause de sa grande taille et de son uniformité, et qui s'adapte au nombre de clusters.

— On élimine les les clusters trop en contact avec le bord de la vignette [1], qui sont considérés comme faisant partie de l'arrière-plan (le sol) (figure B.10(c)).

1. Ceci est en fait une optimisation pour limiter le temps de calcul de la recherche qui suit, car cette dernière n'est pas du tout optimisée. Dans l'idéal comme expliqué section B.3.2.b page 210 cette étape ne devrait pas exister afin d'améliorer la précision.

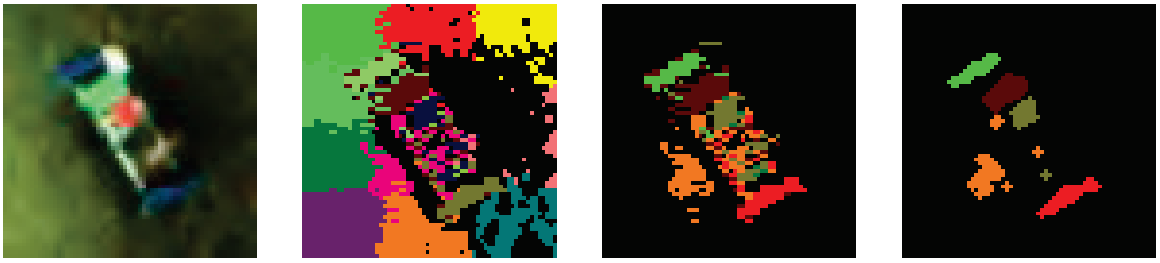
— On nettoie ensuite les clusters du bruit par l'application des opérateurs de morphologie mathématique d'érosion et de dilatation.

— On divise ensuite les différentes composantes connexes des clusters en différents clusters avec l'algorithme du *flood fill* (déjà décrit section B.2.2 page 206).

— On calcule ensuite un vecteur caractéristique constitué pour chaque cluster de sa teinte, sa taille, sa distance au centroïde, et son angle par rapport au cluster précédent.

— Lors de l'apprentissage, on définit le modèle avec la moyenne et l'écart-type des vecteurs caractéristiques obtenus sur les différents exemples, en imposant à l'avance quels clusters doivent être utilisés et dans quel ordre.

— Lors de la détection, on teste toutes les associations possibles de clusters avec ceux du modèle, et on calcule à chaque fois le vecteur caractéristique et sa distance selon la norme L2 normalisée par les écarts-types, pour garder la meilleure association et comparer le score à un seuil.



(a) L'image originale.

(b) Les clusters bruts extraits, sans les pixels très peu saturés.

(c) Élimination des clusters touchant les bords.

(d) Débruitage par érosion et dilatation.

FIGURE B.10 – Les étapes d'extraction des taches colorées. Chaque couleur correspond à un cluster différent.

La partie estimation de l'orientation et de l'échelle est basée sur les mêmes clusters :

— Durant l'apprentissage, on enregistre les coefficients reliant en moyenne l'éloignement du cluster à leur barycentre à la taille de l'objet, et l'angle entre le cluster et leur barycentre à l'orientation de l'objet. Les écarts-types entre les exemples sont aussi enregistrés et servent à pondérer l'importance du cluster.

— Durant la détection, chaque cluster génère ainsi une hypothèse de taille et d'orientation, et on en fait la moyenne pondérée pour tous les clusters.

Affiner un candidat avec cette méthode prend moins de 10 millisecondes, sans aucune optimisation, et présente une robustesse bien supérieure à la méthode par corrélation, alors qu'il reste des voies d'amélioration. On peut ainsi voir dans les résultats présentés figure B.11 page suivante que l'on peut atteindre un taux de détection de 89.6% pour absolument aucun faux positif. Les améliorations possibles qui sont décrites dans la section suivante laissent de plus espérer de pouvoir améliorer encore significativement le taux de détection en conservant un taux de fausse alarme nul, de nombreux cas d'échec étant bien identifiés.

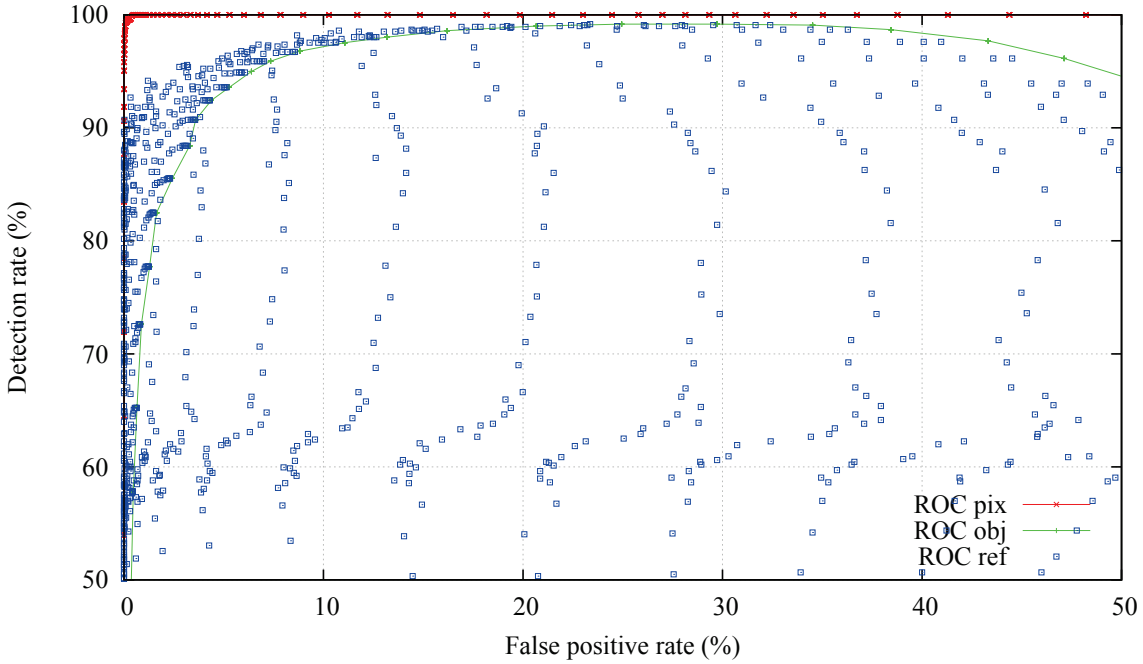


FIGURE B.11 – Illustration des performances de la chaîne complète de détection. La courbe après affinement (ref pour *refinement*) est en réalité un nuage de point car deux seuils, pour la détection initiale et l'affinement, sont explorés. Les points bleus sont le résultat de la projection vers la gauche et légèrement vers le bas de la courbe verte (l'affinement ne fait que supprimer des faux positifs, et peut un peu détériorer le taux de détection mais pas l'améliorer).

La précision concernant le calcul de la taille et de l'orientation n'est cependant toujours pas très bonne et devrait être améliorée.

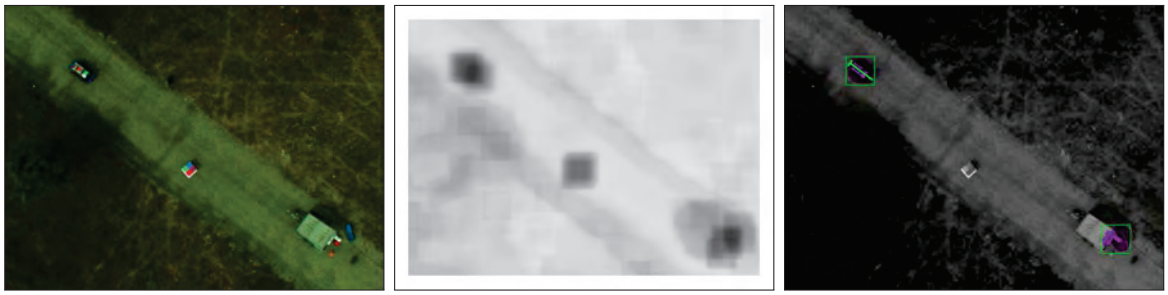
Les figures B.12 , B.13 page 212 et B.14 page 213 illustrent différents cas de réussite et d'échecs dans diverses situations.

b Voies d'amélioration

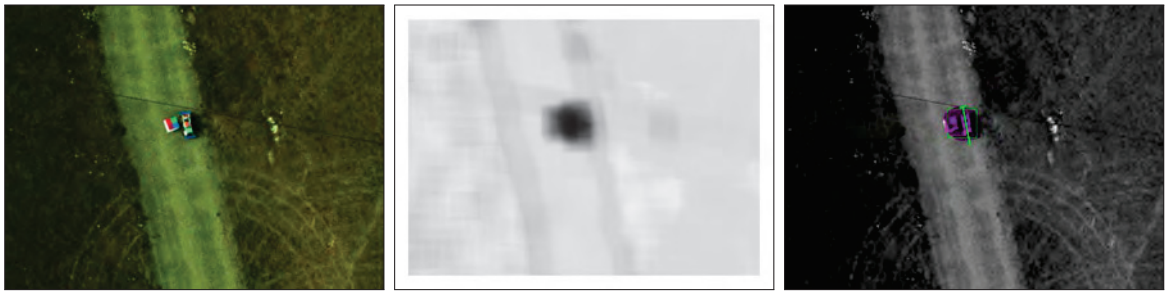
Si la dernière étape de recherche d'association des clusters semble très lourde (complexité factorielle), elle reste cependant tout à fait gérable avec un très faible nombre de taches (moins d'une dizaine, ce qui est en général suffisant), mais surtout on peut l'optimiser considérablement en élaguant l'arbre de recherche. Pour cela il suffit d'évaluer progressivement la distance, au fur et à mesure de la construction de l'hypothèse (à chaque ajout d'association d'une tache), et dès qu'elle a dépassé la valeur la plus faible déjà obtenue on arrête la recherche. Par exemple lors de l'association de la première tache, si on teste avec une tache qui a une teinte complètement différente, on aura directement une distance minimale qui sera très grande, et on pourra couper la branche en éliminant toute sa combinatoire.

Pour améliorer l'élagage, on peut veiller à explorer en premier les combinaisons les plus probables, avec une stratégie gloutonne, afin d'avoir rapidement une meilleure distance basse.

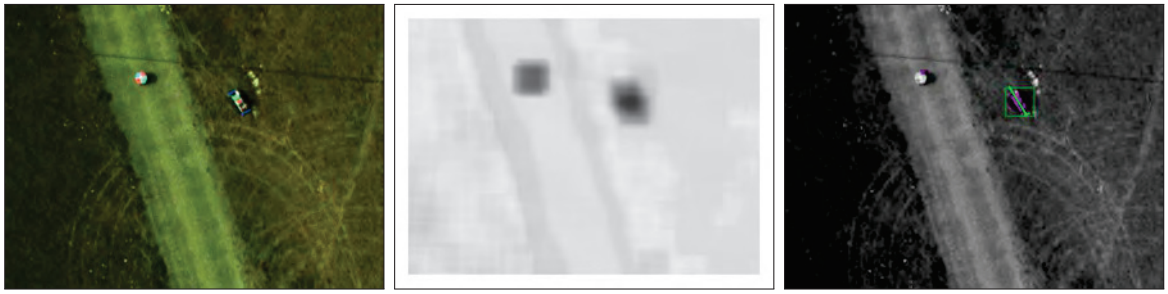
Cette amélioration du temps de calcul permettrait alors de conserver plus de taches potentielles, voire de considérer plusieurs hypothèses de taches (par exemple avec des fusions de



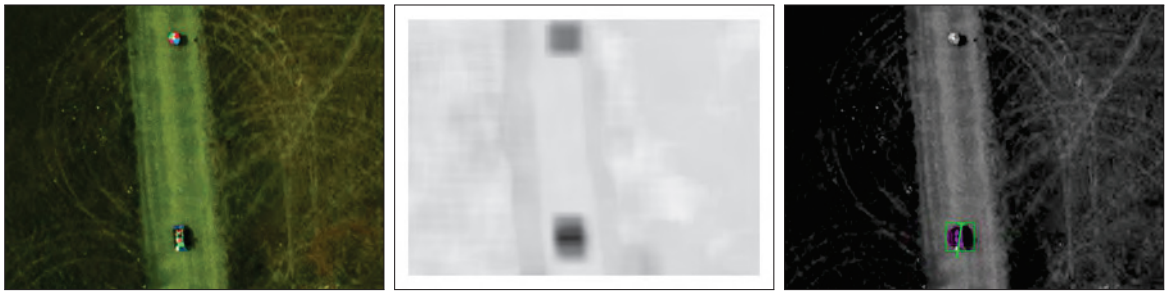
(a) Détection avec succès, un faux positif étant éliminé par l'étape d'affinement.



(b) Détection avec succès malgré la présence proche d'une cible aux couleurs proches : l'étape d'affinement extrait correctement l'objet recherché.



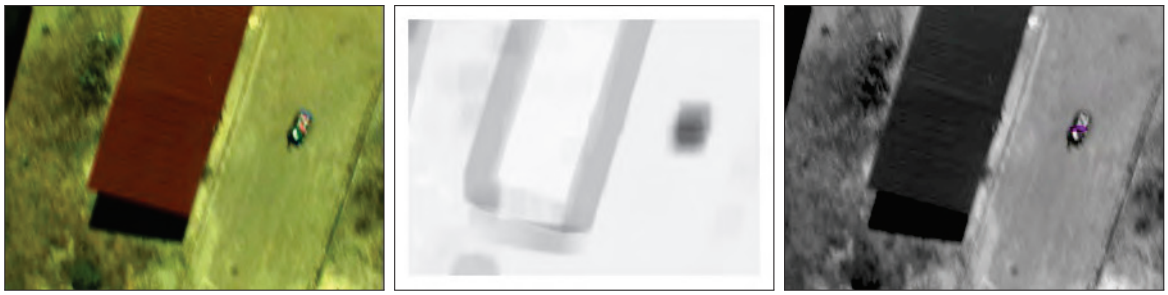
(c) Détection avec succès illustrant la robustesse au changement d'arrière-plan.



(d) Détection avec succès, mais estimation d'orientation imprécise.

FIGURE B.12 – Exemples de résultats de détection. La première image représente l'image originale, la deuxième le résultat de la détection par histogramme, enfin dans la troisième sont représentés les pixels où la détection par histogramme a dépassé le seuil en violet, les objets extraits par un carré vert, et les objets ayant passé avec succès l'étape d'affinement par une flèche verte représentant l'orientation et la taille de l'objet.

(a) Détection avec succès.

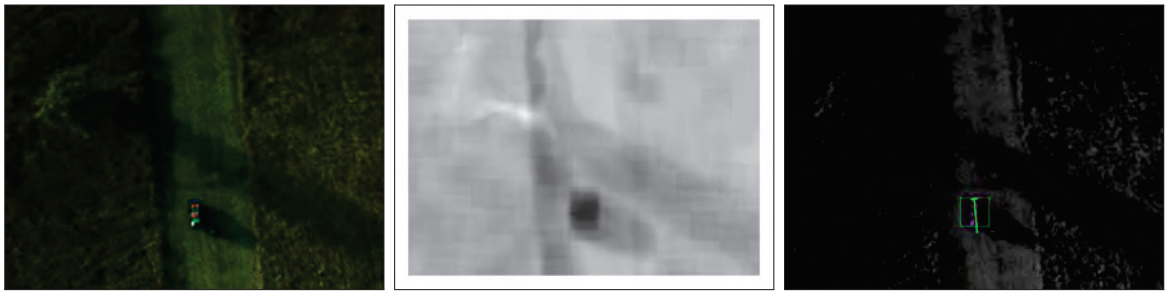


(b) Échec de la détection par histogramme (réponse insuffisante).

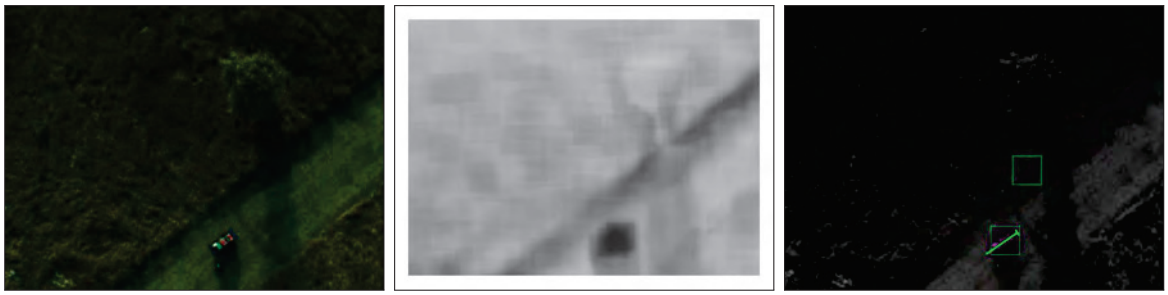
(c) Échec de l'affinement.

(d) Estimation de taille et orientation imprécise.

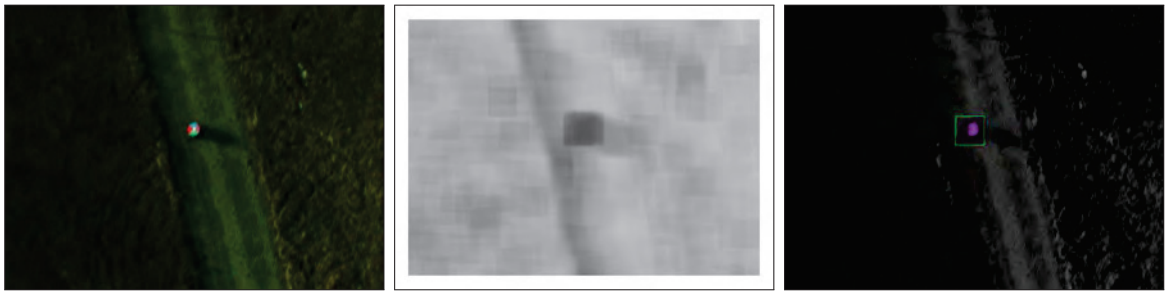
FIGURE B.13 – Exemples de résultats de détection avec différents cas d'échecs.



(a) Détection avec succès.



(b) Faux positif dans le décor éliminé par l'étape de raffinement.



(c) Faux positif sur une cible éliminé par l'étape d'affinement.

(d) Échec de l'étape d'affinement.

FIGURE B.14 – Exemples de résultats de détection avec différents cas d'échecs.

taches proches en teinte et distance), ce qui améliorerait donc le taux de détection.

La segmentation de l'image en taches de couleur peut également être largement améliorée. D'une part l'implémentation d'un algorithme d'agglomération permettrait de mieux optimiser son code, mais aussi d'utiliser une distance dédiée pour cette phase, ce que ne permet pas l'implémentation d'OpenCV. Par exemple la définition de la teinte est cyclique, et la distance utilisée devrait en tenir compte (pour le moment nous avons simplement décalé l'origine de la teinte dans les jaunes au lieu des rouges, car nous utilisons la couleur rouge mais pas la jaune). On pourrait également par le choix de la distance inciter l'algorithme à choisir des clusters plus connexes. Plus généralement d'autres algorithmes d'agglomération peuvent-être plus performants pour la tâche de segmentation, comme l'agglomération hiérarchique (*hierachical clustering*). Une étape initiale de pré-agglomération par canopées (*canopy clustering*) permettrait en outre de réduire significativement le temps de calcul.

Enfin la précision de la confirmation, mais surtout la précision de l'estimation de taille et d'orientation pourraient être améliorées en considérant un modèle 3D de l'objet à détecter pour considérer explicitement les changements d'apparence, en utilisant un algorithme d'apprentissage plus évolué. Cela permettrait également de tenir compte de masquages qui apparaissent, comme quand l'antenne GPS blanche vient masquer une partie du pare-chocs arrière dans les vues par l'avant.

Pour terminer, une voie pour rendre les systèmes de détection plus robustes se trouve dans l'approfondissement de l'analyse de l'incertitude. En général on ne considère que le score de l'algorithme de détection comme indication de certitude de la détection, et on utilise un seuil fixe dans un contexte donné pour prendre une décision. Mais il serait très utile de mettre en regard ce seuil avec la qualité des images, en analysant leur netteté, leur bruit, leur illumination, etc. C'est ce qu'un homme fera si on lui confie cette tâche : il prendra plus facilement la décision d'attribuer une détection avec un score de ressemblance donné sur une image de très mauvaise qualité que sur une image de bonne qualité, car il s'attend dans ce dernier cas à avoir une ressemblance plus forte, alors qu'il est normal que ce ne soit pas le cas avec une image de mauvaise qualité.

Enfin des aspects de cohérence temporelle dans les détections et le suivi sont également très efficaces pour améliorer le compromis détection / fausse alarme.

B.4 Multi-objets

Si plusieurs objets sont à détecter simultanément, comme c'est notre cas (le robot Mana ainsi que plusieurs cibles), il est possible d'être plus efficace en mettant en commun une partie des calculs :

- la conversion de l'image de pixels en image de numéros de case d'histogramme peut être partagée par la détection de tous les objets qui utilisent la même classification (et on utilise en général la même).
- la construction des histogrammes (qui prend un peu plus de la moitié du temps de détection initiale) peut être partagée par tous les objets qui ont une taille voisine.
- la comparaison des histogrammes peut être partagée par tous les objets qui ont le

même histogramme (par exemple nous avons équipé nos cibles de quatre taches de couleur différente, ce qui permet d'en détecter jusqu'à six cibles différentes avec le même histogramme, différentiables par l'ordre des couleurs (nombre de permutations divisé par le nombre de permutations circulaires).

— l'extraction des taches de couleur par l'algorithme d'affinement ne dépendant pas de l'objet recherché, cette étape peut en théorie être partagée par tous les objets, et en pratique par tous ceux qui arrivent à cette étape par le même processus, c'est-à-dire ceux qui ont le même histogramme.

B.5 Etiquetage

Afin de pouvoir fournir des données aux algorithmes d'apprentissage, mais surtout pouvoir évaluer automatiquement leurs performances sur des séquences entières, nous avons également développé un petit outil d'étiquetage des données. Cet étiquetage se fait en deux étapes, plus une étape supplémentaire pour la base d'apprentissage. Pour plus de rapidité, les exemples négatifs sont interpolés. La figure B.15 illustre l'interface de l'outil.

FIGURE B.15 – Illustration de l'outil d'étiquetage des données.

B.5.1 Position et taille approximative

Lors de la première étape, qui sert à l'algorithme de détection initiale par comparaison d'histogrammes, on relève simplement la position et la taille approximative de l'objet, sous forme d'un carré à placer sur l'objet à l'aide de la souris après avoir éventuellement ajusté sa taille.

Trois différentes qualités peuvent également être enregistrées pour chaque instance :

— *Bonne* : le détecteur doit réussir à détecter l'objet, il n'y a pas de raison légitime d'échec.

— *Suffisante* : le détecteur devrait pouvoir détecter l'objet, mais s'il n'y parvient pas ce ne sera pas catastrophique car il y des raisons qui peuvent l'expliquer (en général un flou excessif).

— *Insuffisante* : on ne cherche même pas à pouvoir détecter ces objets, car leur image est très dégradée, ou alors ils ne sont que partiellement visibles. Pour autant on les étiquette tout de même car on ne souhaite pas considérer qu'il y a eu fausse alarme si le détecteur parvient tout de même à les détecter !

B.5.2 Orientation et taille précise

La seconde étape est utilisée par l'algorithme d'affinement afin de pouvoir vérifier les estimations de taille et d'orientation. Elle consiste à relever deux points remarquables de l'objet, dont le lien avec la taille et l'orientation de l'objet son connus. Par exemple dans le cas de la figure B.10 page 209, on choisit le centre du pare-choc arrière et le centre du pare-choc avant.

B.5.3 Taches de couleur

Une troisième étape est également utilisée pour les exemples de la base d'apprentissage, afin d'identifier les taches de couleur pour l'algorithme les utilisant décrit section B.3.2 page 208. Il s'agit simplement de cliquer dans le bon ordre sur le centre de chacune des taches, afin que l'algorithme puisse y associer correctement les taches qu'il trouve pour l'apprentissage.

B.6 Bilan

Nous avons développé une chaîne de détection et d'identification visuelle d'objets équipés de façon simple de couleurs, qui est à la fois rapide et robuste aux différentes conditions qui peuvent être rencontrées sur le terrain, ce qui le rend utilisable de façon opérationnelle. Nous nous sommes basés sur un algorithme de détection existant, que nous avons adapté et optimisé, et avons introduit un algorithme original de reconnaissance. Les nombreuses améliorations proposées que nous n'avons pas eu le temps d'implémenter permettraient de plus d'améliorer encore les performances tout en réduisant encore le temps de calcul.

Présentation de petits travaux liés au contrôle du robot : un asservissement angulaire sur le gyroscope FOG pour faire respecter au robot les consignes de la couche de navigation, et des traitements sur l'odométrie 3D pour tenter de limiter les erreurs d'altitude dues au phénomène de tranfert de masse.

Annexe C

Contrôle

Sommaire

C.1 Asservissement angulaire

Comme on l'a vu, notre robot Mana possède quatre roues fixes et tourne donc par dérapage en appliquant un différentiel de vitesse sur les roues gauches et droites. Les capteurs odométriques sont donc de mauvaises indications de sa vitesse angulaire. Pourtant ce sont les seules données accessibles à la couche de commande bas niveau pour asservir la vitesse angulaire du robot sur la consigne [1]. En conséquence le robot échouait en pratique à appliquer la vitesse angulaire demandée, en tournant parfois trop vite ou parfois pas assez vite selon les conditions d'adhérence du terrain.

Notre module de navigation, basé sur un modèle numérique de terrain, n'est évalué qu'à la fréquence de 2 Hz, et il évalue des arcs de cercle qu'il donne ensuite en consigne au robot. Cette faible fréquence, ajoutée à une certaine latence de commande, et surtout à cette erreur d'asservissement qui faisait parfois sérieusement sous-virer le robot, provoquaient des échecs de l'évitement d'obstacle et des collisions.

Nous avons donc ajouté une seconde boucle d'asservissement par dessus celle du bas niveau, en utilisant le gyromètre à fibre optique comme mesure de la vitesse angulaire réelle. Cette architecture n'est bien sûr pas idéale, car elle ajoute un retard supplémentaire dans la boucle entre mesures et actionneurs, et le retard est l'ennemi principal des asservissements. Mais notre principal problème étant de réduire l'erreur statique, cela n'est pas critique et a permis d'améliorer drastiquement le comportement du robot lors de l'évitement des obstacles.

Cette asservissement dispose de deux modes :

— Lorsque la consigne haut niveau est d'aller parfaitement tout droit, on applique un asservissement en position, afin de rattraper d'éventuels défauts de navigation à cause du terrain et d'aller effectivement le plus droit possible. Ce mode est peu utilisé en autonome, mais peut être très utile manuellement, par exemple pour des calibrages. Nous adoptons dans ce cas un contrôleur proportionnel-intégral sur l'angle. L'action intégrale sert à supprimer l'erreur statique, mais surtout à compenser les erreurs commises dans le passé. L'action dérivée n'est pas nécessaire car les erreurs sont faibles tout comme les corrections.

$$\varepsilon = \theta_m - \theta_r \tag{C.1}$$
$$i = i + \varepsilon \tag{C.2}$$
$$\omega_c = k_p \cdot \varepsilon + k_i \cdot i \tag{C.3}$$

où θ_m et θ_r désignent respectivement le cap mesuré et de consigne (référence), et ω_c la vitesse angulaire de commande.

— Sinon on applique un asservissement en vitesse, afin de rester au plus proche de la consigne. Nous adoptons dans ce cas un contrôleur proportionnel sur la vitesse angulaire. L'action intégrale n'est pas nécessaire car la commande et la consigne étant de même

1. En réalité la plateforme Segway RMP400 étant constituée de deux Segway Patroller accolés, elle dispose de gyromètres de lacet. Ils se sont cependant révélés inexploitables car fournissent des données corrompues par des variations basses fréquences. Nous ne savons donc pas si l'asservissement en vitesse angulaire du fabricant fonctionne mal parce qu'il est basé sur l'odométrie peu fiable, ou parce qu'il est basé sur ces gyromètres défectueux.

nature (des vitesses angulaires) il n'y a pas d'erreur statique, mais on pourrait tout de même l'envisager pour compenser les erreurs commises dans le passé si le module de navigation n'a pas modifié la consigne entre temps. L'action dérivée ne s'est pas révélée nécessaire non plus car on dispose d'un bon a priori lors des grands changements de consigne et les erreurs à corriger demeurent faibles.

$$\varepsilon = \omega_m - \omega_r \tag{C.4}$$
$$\dot{\omega_c} = k_p \cdot \varepsilon \tag{C.5}$$
$$\omega_c \mathrel{+}= \dot{\omega_c} \tag{C.6}$$

où ω_m et ω_r désignent respectivement la vitesse angulaire mesurée et de consigne (référence), et ω_c la vitesse angulaire de commande.

C.2 Odométrie 3D

Avant l'utilisation de RT-SLAM comme solution de localisation sur le robot, une solution utilisant l'odométrie pour la vitesse linéaire, un gyromètre à faible dérive pour le cap, et une INS pour les angles d'attitude était utilisée. L'odométrie 3D consiste à considérer que le vecteur vitesse du robot est aligné avec son axe longitudinal, dont l'inclinaison est donné par la mesure de l'angle d'assiette du robot, afin d'estimer l'altitude du robot.

L'inconvénient de cette approche, comme souligné section 6.1 page 136, est que sur un robot avec des amortisseurs ou des roues gonflables, le transfert de masse lors des accélérations ou décélérations fait dévier le vecteur vitesse de l'axe longitudinal (similaire à l'angle d'incidence des avions). La conséquence est que l'estimation de l'altitude augmente de façon erronée lors des accélérations, et diminue de façon erronée lors des décélérations. De plus plusieurs raisons se cumulant font que le phénomène est nettement plus important à l'accélération qu'à la décélération :

— Les accélérations sont en général plus longues que les décélérations, car elles nécessitent plus d'énergie. La déformation des pneus n'étant pas linéaire avec l'accélération, l'altitude accumulée est plus importante avec une accélération plus faible et plus longue que plus forte et plus courte.

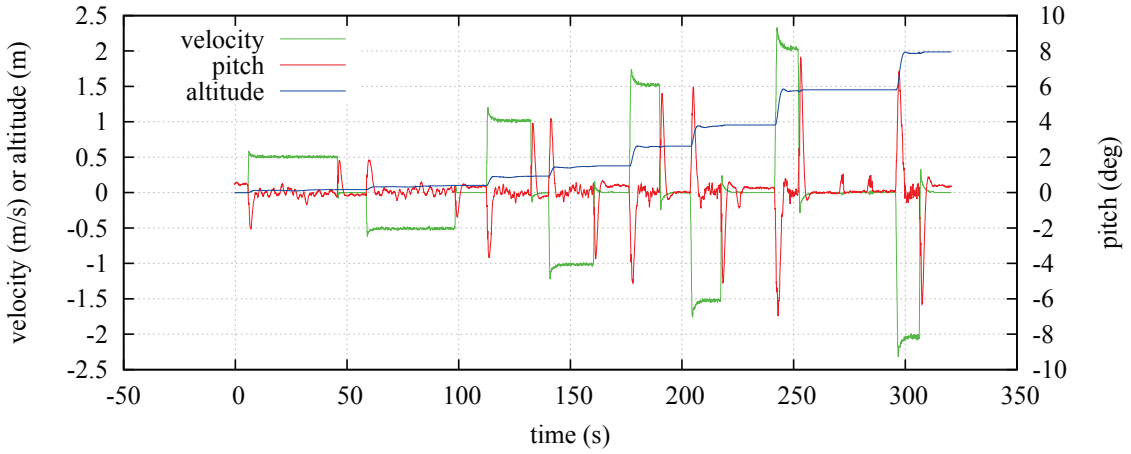
— Le moment inertiel du robot retarde la prise d'incidence, aussi l'incidence la plus forte coïncide avec les vitesses les plus élevées lors des accélérations, et avec les vitesses les plus faibles lors des décélérations, causant à nouveau une accumlation d'altitude plus élevée lors des accélération que lors des décélérations.

— Même à vitesse constante, le robot fournit la propulsion au niveau des roues, tandis que les frottements de l'air le freinent sur l'ensemble de sa surface avant, causant une légère incidence dans le sens d'une accélération.

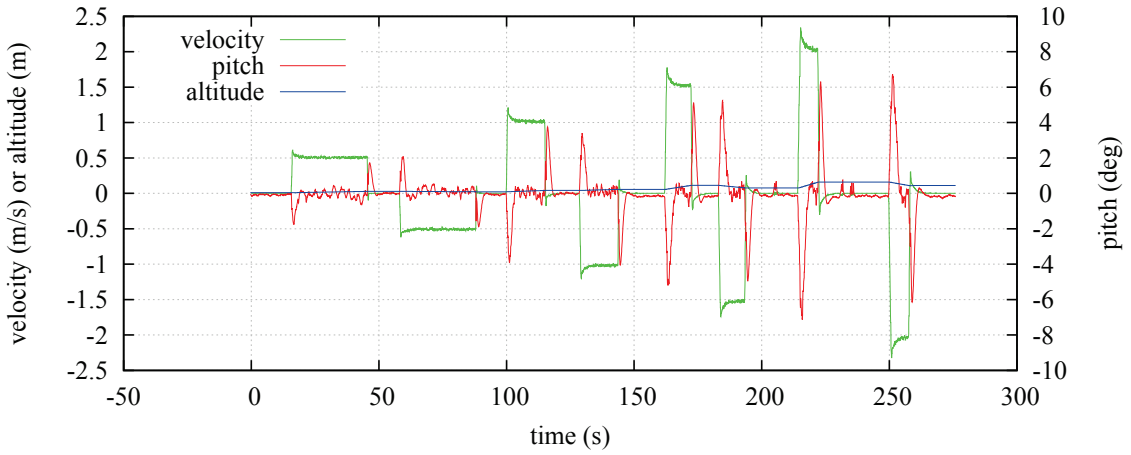
Ainsi la trajectoire du robot a tendance a constamment prendre de l'altitude, avec des paliers d'élévation francs lors des accélérations, des paliers d'abaissement beaucoup plus réduits lors des décélérations, et un léger bais d'élévation lors des régimes à vitesse constante.

Par exemple lors d'une accélération de 0 à 2 m/s en 1 s, l'incidence dépasse 6 °, et le gain en altitude est d'environ 50 cm. Lors de la décélération de 2 m/s à 0 en 0.5 s, l'incidence dépasse

également -6 °, mais la perte d'altitude est inférieure à 2 cm. Ceci est plus largement illustré figure C.1(a).



(a) Avec utilisation de l'angle d'assiette réel.



(b) Avec figeage de l'angle d'assiette pendant les accélérations.

FIGURE C.1 – Illustration des variations d'angle d'assiette et d'altitude z lors d'allers-retours d'avant en arrière sur un sol plat à des vitesses de plus en plus importantes, avec des accélérations de $2\,\mathrm{m/s^2}$ et des décélérations de $4\,\mathrm{m/s^2}$.

Ce phénomène est donc préjudiciable pour la localisation absolue, mais également pour la construction des modèles numériques de terrain.

Nous avons donc dans un premier temps limité les accélérations à $2\,\mathrm{m/s^2}$ en imposant une rampe sur la consigne de vitesse, car le phénomène était encore plus important avec l'accélération maximale du robot, mais comme on l'a vu cela n'est pas suffisant.

Dans un second temps nous avons donc essayé de désactiver la mise à jour de l'assiette pour le calcul de l'odométrie 3D pendant les phases d'accélération, ce qui a permis d'atténuer largement le problème sur un exemple typique, comme illustré figure C.1(b), mais s'est montré beaucoup moins efficace sur des séquences réelles sur un terrain fortement 3D.

Des petits travaux d'électronique analogique et numérique qui se sont révélés nécessaires pour synchroniser les caméras : une adaptation d'impédance entre la centrale inertielle et les caméras, ainsi qu'un déphaseur de synchronisations pour faire du bicam alterné.

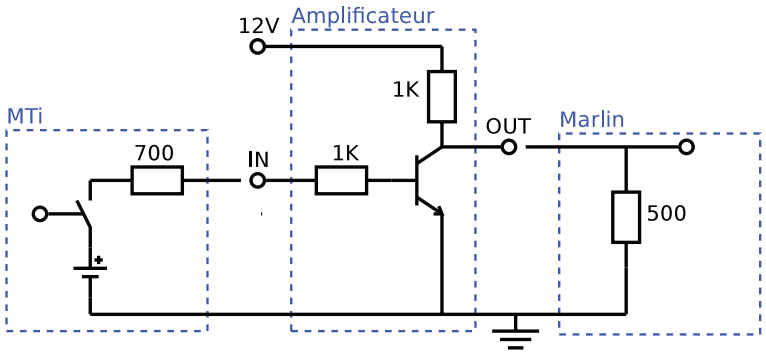
Annexe D

Électronique de synchronisation des capteurs

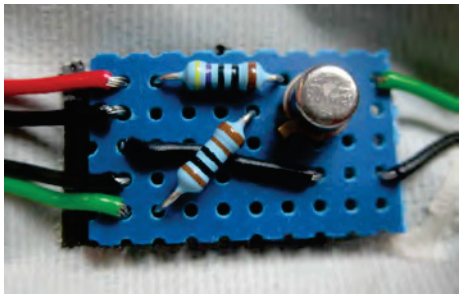
Sommaire

D.1 Synchronisation des caméras AVT Marlin avec une IMU XSensMTi

La synchronisation d'une centrale inertielle XSens MTi avec une ou des caméras AVT Marlin pose un problème d'impédance. En effet la sortie de synchronisation de la centrale MTi a une impédance relativement forte (environ 700 Ω au lieu d'idéalement nulle) sous une tension faible (3 V). L'entrée de synchronisation de la caméra Marlin a quant à elle une impédance relativement faible (500 Ω au lieu d'idéalement infinie), et fonctionne en logique TTL, c'est-à-dire que la tension présente doit être supérieure à 2.2 V pour que l'entrée soit considérée haute. Une application rapide de la loi d'Ohm montre que si on branche directement la sortie de la centrale MTi à l'entrée de la caméra Marlin on n'aura une tension que de 1.25 V, qui ne sera donc jamais interprétée comme haute. Il est donc nécessaire de réaliser un étage d'adaptation d'impédance avec un amplificateur, tel qu'un transistor comme présenté figure D.1. Notons que ce montage a pour effet d'inverser le signal, mais ce n'est pas important car la caméra est configurable pour l'interpréter de façon inversée.



(a) Schéma électronique



(b) Réalisation

FIGURE D.1 – L'étage amplificateur pour adapter l'impédance de l'IMU XSens MTi à la caméra AVT Marlin.

D.2 Synchronisation des caméras pour le bicam alterné

Pour faire du bicam alterné comme vu section 6.5.2 page 155, il est nécessaire de contrôler le déclenchement des deux caméras pour déphaser leur prise d'image d'une demi-période. Il est

donc indispensable d'utiliser un signal de déclenchement externe (*trigger*). Cette fonctionnalité de générer deux signaux déphasés d'une demi-période n'est cependant pas standard. On pourrait à l'aide d'un microcontrôleur générer ou transformer de tels signaux, mais il est plus simple et plus élégant de concevoir un convertisseur à partir de quelques composants d'électronique logique. On pourrait également acquérir les images des deux caméras à fréquence double, et sélectionner logiciellement une image sur deux de façon décalée (ce que nous avons en réalité fait pour pouvoir comparer l'approche alternée à l'approche synchronisée), mais cela représente un gaspillage de ressources, et peut être tout simplement impossible pour des raisons de bande passante (ou comme dans notre cas générer des corruptions pour cause de bus presque saturé).

a Cahier des charges

On souhaite donc avoir un montage qui pour une source quelconque de signal de déclenchement (oscillateur, sortie de synchronisation d'un autre capteur, logiciel) fournisse deux lignes de *trigger* de fréquence moitié et déphasées d'une demi-période. On souhaite également que la durée des impulsions soit conservée car elle peut être interprétée par les caméras comme la durée d'exposition.

b Conception

On veut ainsi un circuit logique séquentiel à une entrée I, un état sur un bit Q, et deux sorties O1 et O2, avec la table de vérité présentée figure D.2.

I	**Q**	**Q+**	**O1**	**O2**
0	0	1	0	0
0	1	0	0	0
1	0	0	1	0
1	1	1	0	1

FIGURE D.2 – Table de vérité du circuit logique séquentiel

On peut interpréter ce circuit logique comme un compteur 1 bit qui commande un démultiplexeur avec deux sorties. Le compteur 1 bit peut être réalisé avec une bascule en mode toggle, et le démultiplexeur avec deux simples portes logiques.

On parvient à un résultat similaire avec une conception classique de circuit de logique séquentielle : une bascule stocke l'état, la transition d'état se fait avec l'opération combinatoire *Q+ = XOR(I,Q)* qui est ce que fait une bascule en mode toggle, et les sorties s'obtiennent avec des opérations combinatoires NOR.

Comme on contrôle le signal et la commande de voie du démultiplexeur avec la même entrée, il faut absolument maitriser les délais de propagation dans les circuits pour s'assurer que l'on ne va pas générer de pics parasites lors des transitions. Il faut soit basculer sur front montant et être sur que la commande de voie changera avant le signal, soit basculer sur front descendant et être sur que le signal changera avant la commande de voie. Comme le signal est transmis

directement depuis l'entrée, et que la commande de voie doit passer par une bascule, on doit choisir la seconde solution.

Le schéma présenté figure D.3 est une solution avec une bascule et trois portes NOR.

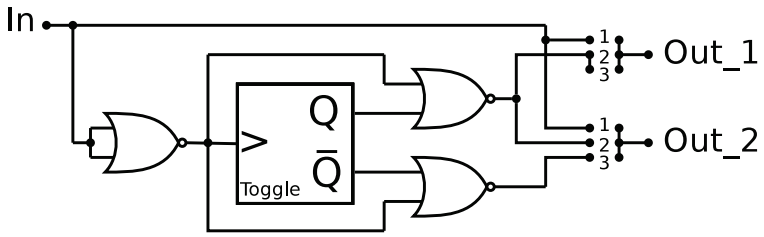


FIGURE D.3 – Schéma du circuit logique séquentiel

c Réalisation

Le schéma D.3 suppose que la bascule fonctionne sur front montant de l'horloge. Si ce n'est pas le cas, il ne faudra pas brancher directement l'horloge de la bascule sur l'entrée, mais au contraire ajouter une autre porte NOT sur l'horloge uniquement afin de toujours s'assurer que la commande de voie n'arrivera pas avant le signal (on dispose de toute façon d'une porte non utilisée sur le circuit intégré, qui fournissent les portes logiques par 2 ou 4).

On peut réaliser la bascule soit avec une bascule T (en reliant T à Vcc), soit avec une bascule D (en reliant /Q et D), soit avec une bascule JK (en reliant J et K à Vcc). Il suffit donc d'un circuit intégré double bascule D ou JK, et d'un circuit quadruple NAND. On s'orientera plutôt vers la technologie CMOS pour accepter une plage de tensions plus large.

On pourra sur les sorties positionner des cavaliers afin de choisir si l'on envoie sur les sorties :

1. le signal d'origine à haute fréquence,
2. un signal à fréquence moitié synchronisé pour les deux caméras (bicam synchronisé ou stéréovision),
3. un signal à fréquence moitié déphasé pour les deux caméras (bicam alterné).

Notons que ce montage règle également le problème de l'adaptation d'impédance entre les caméras Marlin et centrales inertielles MTi étudié à la section précédente.

Nous n'avons cependant pas réalisé ce montage, les ingénieurs électroniciens du laboratoire préférant s'orienter sur une solution avec microcontrôleur.

Bibliographie

[Angeli *et al.*, 2008,] Angeli, A., Filliat, D., Doncieux, S. et Meyer, J.-A. (2008). A fast and incremental method for loop-closure detection using bags of visual words. *IEEE Transactions On Robotics, Special Issue on Visual SLAM*, 24(5). 20

[Ayache et Faugeras, 1988,] Ayache, N. et Faugeras, O. (1988). Building, registrating, and fusing noisy visual maps. *International Journal of Robotics Research*, 7(6). 19

[Bailey et Durrant-Whyte, 2006,] Bailey, T. et Durrant-Whyte, H. (2006). Simultaneous Localisation and Mapping (SLAM) : Part I The Essential Algorithms. *IEEE Robotics and Automation Magazine*, 13(2):99–110. 19, 20

[Bailey *et al.*, 2006a,] Bailey, T., Nieto, J., Guivant, J., Stevens, M. et Nebot, E. (2006a). Consistency of the ekf-slam algorithm. *In Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 3562–3568. IEEE. 23, 47, 184

[Bailey *et al.*, 2006b,] Bailey, T., Nieto, J. I. et Nebot, E. M. (2006b). Consistency of the fastslam algorithm. *In ICRA*, pages 424–429. IEEE. 24

[Barreto, 2003,] Barreto, J. P. A. (2003). *General central projection systems, modeling, calibration and visual servoing.* Thèse de doctorat, Fac. de Ciências e Tecnologia de Coimbra. 56

[Berger et Lacroix, 2008,] Berger, C. et Lacroix, S. (2008). Using planar facets for stereovision slam. *In IEEE/RSJ International Conference on Intelligent Robots and Systems, Nice (France).* 75

[Berger et Lacroix, 2010,] Berger, C. et Lacroix, S. (2010). DSeg : Détection directe de segments dans une image. *In 17ème congrès francophone AFRIF-AFIA Reconnaissance des Formes et Intelligence Artificielle (RFIA).* 64

[Besl et McKay, 1992,] Besl, P. et McKay, N. (1992). A method for registration of 3-d shapes. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 14(2):239–256. 17

[Borenstein *et al.*, 1995,] Borenstein, J., Everett, H. R. et Feng, L. (1995). *Navigating Mobile Robots : Sensors and Techniques.* A. K. Peters. 12

[Borenstein et Feng, 1996,] Borenstein, J. et Feng, L. (1996). Gyrodometry : A new method for combining data from gyros and odometry in mobile robots. *In IEEE International Conference on Robotics and Automation, Minneapolis, Minnesota (USA)*, pages 423–428. 15

[Botterill *et al.*, 2010,] Botterill, T., Mills, S. et Green, R. (2010). Bag-of-words-driven single camera simultaneous localisation and mapping. *Journal of Field Robotics*, 28(2). 20

[Bouguet, 2013,] Bouguet, J.-Y. (2013). Camera calibration toolbox for matlab. `http://www.vision.caltech.edu/bouguetj/calib_doc/`. 57

[Castellanos *et al.*, 2007,] Castellanos, J. A., Martínez-cantín, R., Tardós, J. D. et Neira, J. (2007). Robocentric map joining : Improving the consistency of ekf-slam. *Robotics and Autonomous Systems*, 55:21–29. 183

[Chatila et Laumond, 1985,] Chatila, R. et Laumond, J.-P. (1985). Position referencing and consistent world modeling for mobile robots. *In IEEE International Conference on Robotics and Automation, St Louis (USA)*, pages 138–145. 18

[Civera *et al.*, 2009,] Civera, J., Grasa, O., Davison, A. et Montiel, J. (2009). 1-point ransac for ekf-based structure from motion. *In Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3498 –3504. 68

[Codol, 2013,] Codol, J.-M. (2013). *Hybridation GPS/Vision monoculaire pour la navigation d'un robot en milieu extérieur*. Thèse de doctorat, University of Toulouse. 48

[Cummins et Newman, 2008,] Cummins, M. et Newman, P. (2008). FAB-MAP : Probabilistic Localization and Mapping in the Space of Appearance. *The International Journal of Robotics Research*, 27(6):647–665. 190

[Cummins et Newman, 2011,] Cummins, M. et Newman, P. (2011). Appearance-only SLAM at large scale with FAB-MAP 2.0. *The International Journal of Robotics Research*, 30(9). 20

[Davison, 2003,] Davison, A. (2003). Real-time simultaneous localisation and mapping with a single camera. *In Proc. of IEEE International Conference on Computer Vision (ICCV)*, pages 1403–1410. 19, 63

[Davison *et al.*, 2007,] Davison, A., Reid, I., Molton, N. et Stasse, O. (2007). Monoslam : Real-time single camera slam. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29:1052–1067. 66, 73

[Dellaert et Kaess, 2006,] Dellaert, F. et Kaess, M. (2006). Square Root SAM : Simultaneous location and mapping via square root information smoothing. *IJRR*, 25(12):1181. Special issue on RSS 2006. 24

[Dorveaux, 2011,] Dorveaux, E. (2011). *Navigation Magnéto-Inertielle Principes et application à un système podométrique indoor*. Thèse de doctorat, École Nationale Supérieure des Mines de Paris. 11

[Durrant-Whyte, 1988,] Durrant-Whyte, H. (1988). Sensors Models and Multisensor Integration. *International Journal on Robotics Research*, 7:97–113. 18

[Durrant-Whyte et Bailey, 2006,] Durrant-Whyte, H. et Bailey, T. (2006). Simultaneous Localisation and Mapping (SLAM) : Part II State of the Art. *IEEE Robotics and Automation Magazine*, 13(3):108–117. 20

[Estrada *et al.*, 2005,] Estrada, C., Neira, J. et Tardos, J. (2005). Hierarchical slam : Real-time accurate mapping of large environments. *Robotics, IEEE Transactions on*, 21(4): 588–596. 25, 184, 189

[Eudes, 2011,] Eudes, A. (2011). *Localisation et cartographie simultanées par ajustement de faisceaux local : propagation d'erreurs et réduction de la dérive à l'aide d'un odomètre*. Thèse de doctorat, Université Blaise Pascal - Clermont-Ferrand II. 25

[Fox *et al.*, 1998,] Fox, D., Burgard, W. et Thrun, S. (1998). Active markov localization for mobile robots. *Robotics and Autonomous Systems*, 25(3/4):195–207. 18

[Fraundorfer et Scaramuzza, 2012,] Fraundorfer, F. et Scaramuzza, D. (2012). Visual odometry : Part ii - matching, robustness, and applications. *IEEE Robotics and Automation Magazine*, 19(2). 16

[Gebre-Egziabher, 2004,] Gebre-Egziabher, D. (2004). *Design and Performance Analysis of a Low-Cost Aided Dead Reckoning Navigator*. Thèse de doctorat, Stanford University. 93

[Gonzalez, 2013,] Gonzalez, A. (2013). *Localisation par vision multi-spectrale – Application aux systèmes embarqués*. Thèse de doctorat, University of Toulouse. 48

[Huang *et al.*, 2008,] Huang, G. P., Mourikis, A. I. et Roumeliotis, S. I. (2008). Analysis and improvement of the consistency of extended kalman filter based slam. *In In IEEE Int. Conf. on Robotics and Automation (ICRA)*. 23, 75, 180

[Ila *et al.*, 2010,] Ila, V., Porta, J. M. et Andrade-Cetto, J. (2010). Information-Based Compact Pose SLAM. *IEEE Transactions on Robotics*, 26(1):78–93. 24

[J. A. Castellanos et Tardós, 2004,] J. A. Castellanos, J. N. et Tardós, J. D. (2004). Limits to the consistency of ekf-based slam. *In 5th IFAC Symp. on Intelligent Autonomous Vehicles, IAV'04*, Lisbon, Portugal. 184

[Julier et Uhlmann, 2001,] Julier, S. J. et Uhlmann, J. K. (2001). A counter example to the theory of simultaneous localization and map building. *In In IEEE Intl. Conf. on Robotics and Automation (ICRA*, pages 4238–4243. 75

[Jung et Lacroix, 2003,] Jung, I.-K. et Lacroix, S. (2003). Simultaneous localization and mapping with stereovision. *In International Symposium on Robotics Research, Siena (Italy))*. 19

[Kaess *et al.*, 2008,] Kaess, M., Ranganathan, A. et Dellaert, F. (2008). iSAM : Incremental Smoothing and Mapping. *IEEE Transactions on Robotics*, 24(8):1365–1378. 20, 24

[Klein et Murray, 2007,] Klein, G. et 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. 25

[Konolige, 2010,] Konolige, K. (2010). Sparse sparse bundle adjustment. *In Proc. BMVC*, pages 102.1–11. doi :10.5244/C.24.102. 24

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

[Lemaire, 2006,] Lemaire, T. (2006). *Localisation Et Cartographie Simultanées Avec Vision Monoculaire*. Thèse de doctorat, University of Toulouse. 22

[Lemaire et Lacroix, 2007,] Lemaire, T. et Lacroix, S. (2007). Long term SLAM with panoramic vision. *Journal of Field Robotics*, 24(1-2):91–111. 19

[Lemaire *et al.*, 2005,] Lemaire, T., Lacroix, S. et Sola, J. (2005). A practical 3d bearing-only slam algorithm. *In Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2449 – 2454. 63

[Leonard et Durrant-Whyte, 1991,] Leonard, J. et Durrant-Whyte, H. (1991). Simultaneous map building and localisation for an autonomous mobile robot. *In IEEE International Workshop on Intelligent Robots and Systems*, pages 1442–1447. 19

[Levinson *et al.*, 2007,] Levinson, J., Montemerlo, M. et Thrun, S. (2007). Map-based precision vehicle localization in urban environments. *In Proceedings of Robotics : Science and Systems*, Atlanta, GA, USA. 18

[Lu et Milios, 1997,] Lu, F. et Milios, E. (1997). Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4:333–349. 17

[Maimone *et al.*, 2007,] Maimone, M., Cheng, Y. et Matthies, L. (2007). Two years of visual odometry on the mars exploration rovers. *Journal of Field Robotics*, 24(3):169–186. 16

[Mallet, 2001,] Mallet, A. (2001). *Localisation d'un robot mobile autonome en environnements naturels*. Thèse de doctorat, Institut National Polytechnique de Toulouse. 15

[Mallet *et al.*, 2000,] Mallet, A., Lacroix, S. et Gallo, L. (2000). Position estimation in outdoor environments using pixel tracking and stereovision. *In IEEE International Conference on Robotics and Automation, San Francisco, Ca (USA)*, pages 3519–3524. 16

[Marquez, 2012,] Marquez, D. (2012). *Towards Visual Navigation in Dynamic and Unknown Environment : Trajectory Learning and Following, with Detection and Tracking of Moving Objects*. Thèse de doctorat, University of Toulouse. 48

[Matsumoto *et al.*, 1998,] Matsumoto, Y., Inaba, M. et Inoue, H. (1998). Memory-based navigation using omni-view sequence. *In* Zelinsky, A., éditeur : *International Conference on Field and Service Robotics, Canberra (Australia)*. Springer-Verlag. 20

[Mei, 2013,] Mei, C. (2013). Omnidirectional calibration toolbox for matlab. `http://www.robots.ox.ac.uk/~cmei/Toolbox.html`. 57

[Molton *et al.*, 2004,] Molton, N. D., Davison, A. J. et Reid, I. D. (2004). Locally planar patch features for real-time structure from motion. *In Proc. British Machine Vision Conference*. BMVC. 75

[Montemerlo et Thrun, 2003,] Montemerlo, M. et Thrun, S. (2003). Simultaneous localization and mapping with unknown data association using fastslam. *In IEEE International Conference on Robotics and Automation*, pages 1985 – 1991. 19, 23

[Montiel, 2006,] Montiel, J. M. M. (2006). Unified inverse depth parametrization for monocular slam. *In Proc. of Robotics : Science and Systems (RSS)*, pages 16–19. 63, 73

[Mouragnon *et al.*, 2009,] Mouragnon, E., Lhuillier, M., Dhome, M., Dekeyser, F. et Sayd, P. (2009). Generic and real-time structure from motion using local bundle adjustment. *Image and Vision Computing*, 27(8):1178–1193. 25

[Moutarlier et Chatila, 1991,] Moutarlier, P. et Chatila, R. (1991). Incremental free-space modelling from uncertain data by an autonomous mobile robot. *In International Workshop on Intelligent Robots and Systems, Osaka (Japon)*, pages 1052–1058. 19

[Muhammad et Lacroix, 2011,] Muhammad, N. et Lacroix, S. (2011). Loop closure detection using small-sized signatures from 3d lidar data. *In IEEE International Symposium on Safety, Security, and Rescue Robotics, Tokyo (Japan)*. 190

[Olson, 2010,] Olson, E. (2010). A passive solution to the sensor synchronization problem. *In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 117

[Roussillon et Lacroix, 2010,] Roussillon, C. et Lacroix, S. (2010). Contrat tarot, tranche conditionnelle, thème b3 "slam à long terme". Rapport technique, LAAS/CNRS. 185, 189

[Scaramuzza, 2008,] Scaramuzza, D. (2008). *Omnidirectional Vision : from Calibration to Root Motion Estimation*. Thèse de doctorat, Swiss Federal Institute of Technology Zurich (ETHZ). 56

[Scaramuzza et Fraundorfer, 2011,] Scaramuzza, D. et Fraundorfer, F. (2011). Visual odometry : Part i - the first 30 years and fundamentals. *IEEE Robotics and Automation Magazine*, 18(4):80–92. 16

[Se *et al.*, 2005,] Se, S., Lowe, D. et Little, J. (2005). Vision based global localization and mapping for mobile robots. *IEEE Transactions on Robotics*, 21(3):364–375. 19

[Sizintsev *et al.*, 2008,] Sizintsev, M., Derpanis, K. G. et Hogue, A. (2008). Histogram-based search : a comparative study. *In IN CVPR*. 199, 200

[Smith *et al.*, 1987,] Smith, R., Self, M. et Cheeseman, P. (1987). A stochastic map for uncertain spatial relationships. *In Robotics Research : The Fourth International Symposium, Santa Cruz (USA)*, pages 468–474. 18

[Solà, 2005,] Solà, J. (2005). Delayed vs undelayed landmark initialization for bearing only SLAM. *In Extended abstract for the SLAM Workshop of the IEEE Int. Conf. on Robotics and Automation*, Barcelona. 63

[Solà, 2007,] Solà, J. (2007). *Towards Visual Localization, Mapping and Moving Objects Tracking by a Mobile Robot : a Geometric and Probabilistic Approach.* Thèse de doctorat, Institut National Polytechnique de Toulouse. 19, 59

[Solà *et al.*, 2007,] Solà, J., Monin, A. et Devy, M. (2007). BiCamSLAM : Two times mono is more than stereo. *In IEEE Int. Conf. on Robotics and Automation*, pages 4795–4800, Rome, Italy. IEEE. 37, 155

[Solà *et al.*, 2012,] Solà, J., Vidal-Calleja, T., Civera, J. et Montiel, J. M. M. (2012). Impact of landmark parametrization on monocular EKF-SLAM with points and lines. *International Journal of Computer Vision*, 97:339–368. 65

[Solà *et al.*, 2009,] Solà, J., Vidal-Calleja, T. et Devy, M. (2009). Undelayed initialization of line segments in monocular SLAM. *In IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 1553–1558, Saint Louis, USA. IEEE. 64

[Stefano *et al.*, 2005,] Stefano, L. D., Mattoccia, S. et Tombari, F. (2005). Zncc-based template matching using bounded partial correlation. *Pattern Recognition Letters*, 26(14): 2129 – 2134. 72

[Strasdat *et al.*, 2010,] Strasdat, H., Montiel, J. M. M. et Davison, A. (2010). Real-time monocular slam : Why filter ? *In Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2657–2664. 21, 25

[Thrun *et al.*, 2005,] Thrun, S., Burgard, W. et Fox, D. (2005). *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press. 23, 24

[Thrun *et al.*, 1998,] Thrun, S., Fox, D. et Burgard, W. (1998). A probabilistic approach to concurrent mapping and localization for mobile robots. *Autonomous Robots*, 5:253–271. 19

[Thrun *et al.*, 2004,] Thrun, S., Liu, Y., Koller, D., Ng, A. et Durrant-Whyte, H. (2004). Simultaneous localisation and mapping with sparse extended information filters. *International Journal of Robotics Research*, 23(7–8):693–716. 19, 23

[Triggs *et al.*, 2000,] Triggs, B., Mclauchlan, P., Hartley, R. et Fitzgibbon, A. (2000). Bundle adjustment – a modern synthesis. *In Vision Algorithms : Theory and Practice.* , Springer-Verlag, Berlin, Germany. 19

[Vidal-Calleja *et al.*, 2011,] Vidal-Calleja, T., Berger, C., Solà, J. et Lacroix, S. (2011). Large scale multiple robot visual mapping with heterogeneous landmarks in semi-structured terrain. *Robotics and Autonomous Systems*, 59(9):654–674. 185, 186, 187

[Viola et Jones, 2001,] Viola, P. et Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *In Proc. IEEE Conf. Comput. Vis. Pattern Recog*, pages 511–518. 72

[Williams, 1992,] Williams, J. (1992). *From Sails to Satellites (The Origin and Development of Navigational Science)*. Oxford University Press. 10

[Wolfram, 2013,] Wolfram, R. (2013). Wolfram alpha. `http://www.wolframalpha.com/`. 101