Localisation d'une flotte de véhicules communicants par approche de type SLAM visuel décentralisé

Guillaume Bresson

N d'ordre : D.U 2439
EDSPIC : 645

Université Blaise Pascal - Clermont II
École doctorale des Sciences Pour l'Ingénieur

THÈSE

en vue de l'obtention du grade de

Docteur d'Université

Spécialité : Vision pour la Robotique

Présentée et soutenue par

Guillaume Bresson

Localisation d'une flotte de véhicules communicants par approche de type SLAM visuel décentralisé

Thèse dirigée par Roland Chapuis

préparée à l'Institut Pascal

soutenue le 21 février 2014

Jury :

Président :	François Charpillet	-	Directeur de recherche INRIA
Rapporteurs :	Fawzi Nashashibi	-	Directeur de recherche INRIA
	Juan D. Tardós	-	Professeur Université de Saragosse
Directeur :	Roland Chapuis	-	Professeur Université B. Pascal
Examinateurs :	Javier Ibañez-Guzmán	-	Ingénieur de recherche Renault
	Romuald Aufrère	-	Maître de conférences Université B. Pascal

Abstract

The localization of a vehicle with the use of SLAM techniques (Simultaneous Localization And Mapping) has been extensively studied during the last 20 years. However, only a few approaches have tried to extend these algorithms to a fleet of vehicles despite the many potential applications. It is the objective of this thesis.

First of all, a monocular SLAM for a single vehicle has been developed. This one proposes to pair an Extended Kalman Filter with a Cartesian representation for landmarks so as to produce accurate low density maps. Indeed, the extension of SLAM to several vehicles requires permanent communications inside the fleet. With only a few landmarks mapped, our approach scales nicely with the number of vehicles. Cheap sensors have been favored (a single camera and an odometer) in order to spread more easily the use of multi-vehicle applications. Correctives have been proposed in order to avoid the divergence problems induced by such a scheme. The experiments showed that our SLAM is able to furnish good localization results while being light and fast.

The drift affecting every SLAM algorithm has also been studied. Its integration inside the SLAM process, thanks to a dedicated architecture and a dynamic model, allows to ensure consistency even without an estimation of it. Loop closures or the integration of geo-referenced information becomes straightforward. They naturally correct all the past positions while still maintaining consistency. In a multi-vehicle scenario, it is a key aspect as each vehicle drifts differently from one another. It is consequently important to take it into account.

Our SLAM algorithm has then been extended to several vehicles. A generic structure has been used so as to allow any SLAM algorithm to replace our monocular SLAM. The multi-vehicle architecture avoids data incest (double-counting information) and handles network failures, be they communication breakdowns or latencies when receiving data. The static part of the drift model allows to take into account the fact that the initial positions of the different vehicles composing the fleet might be unknown. Consistency is thus permanently preserved. Our approach has been successfully tested using simulations and real experiments with various settings (row or column convoy with 2 or 3 vehicles) in a fully decentralized way.

Keywords : SLAM, decentralized, monocular, drift, EKF, multi-vehicle, localization, linearization, divergence.

Résumé

La localisation d'un véhicule via les techniques de SLAM (Simultaneous Localization And Mapping pour cartographie et localisation simultanées) a connu un essor important durant les 20 dernières années. Pourtant, peu d'approches ont tenté d'étendre ces algorithmes à une flotte de véhicules malgré les nombreuses applications potentielles. C'est ici l'objectif de cette thèse.

Pour ce faire, une approche de SLAM monoculaire pour un seul véhicule a d'abord été développée. Celle-ci propose de coupler un filtre de Kalman étendu avec une représentation cartésienne des amers afin de produire des cartes de faible densité mais de qualité. En effet, l'extension à plusieurs véhicules nécessite des échanges permanents par l'intermédiaire de communications sans fil. Avec peu d'amers dans les cartes, notre approche s'accommode bien du nombre de véhicules de la flotte. Des capteurs peu onéreux ont aussi été privilégiés (une unique caméra et un odomètre) afin de réduire le coût d'une extension multivéhicule. Des correctifs ont été proposés afin d'éviter les problèmes de divergence induits par les choix précédents. Des expérimentations ont montré que la solution de SLAM produite était légère et rapide tout en fournissant une localisation de qualité.

La dérive, inhérente à tout algorithme de SLAM, a également fait l'objet d'une analyse. Celle-ci a été intégrée au SLAM par l'intermédiaire d'une architecture dédiée et d'un modèle dynamique. Le but est de pouvoir rendre consistante la localisation fournie par le SLAM, même en l'absence d'estimation de la dérive. Cela permet d'effectuer des fermetures de boucle ou encore d'intégrer des informations géo-référencées de manière naturelle tout en conservant l'intégrité de la solution. En multivéhicule, cet aspect est un point clef puisque chaque véhicule dérive différemment des autres. Il est donc important de le prendre en compte.

Enfin, le SLAM a été étendu à plusieurs véhicules. Une structure générique a été prévue afin que notre approche monoculaire puisse être remplacée par n'importe quel algorithme de SLAM. Notre architecture décentralisée évite la consanguinité des données (le fait de compter deux fois une même information) et gère les défaillances réseau, que cela soit des ruptures de communication ou encore des latences dans la réception des données. La partie statique du modèle de dérive permet également de prendre en compte le fait que les positions initiales des véhicules d'une flotte puissent être inconnues. L'intégrité est ainsi maintenue en permanence. Enfin, notre approche étant entièrement décentralisée, elle a pu être testée et validée en simulation et avec des expérimentations réelles dans diverses configurations (convoi en colonne ou en ligne, avec 2 ou 3 véhicules).

Mots clefs : SLAM, décentralisé, monoculaire, dérive, EKF, multivéhicule, localisation, linéarisation, divergence.

Remerciements

Avant de rentrer dans des considérations techniques, je souhaite remercier toutes les personnes qui m'ont accompagné durant ces 3 années de thèse. Tout d'abord, je remercie mes encadrants, Romulus Rémus Rémoulade Aufrère et Roland Chapuis, pour leurs conseils avisés et leur bonne humeur (presque) permanente. J'en profite également pour remercier tous les gens de l'Institut Pascal ayant permis mon intégration dans leurs locaux. Je fais de même pour le personnel du département informatique de l'IUT de Clermont-Ferrand. Je tiens à remercier plus particulièrement Guénal, Sylvie et François pour m'avoir amené à cette thèse ainsi qu'à l'enseignement dans leurs locaux.

Je félicite mes collègues de bureau, passés et présents, pour avoir contribué au succès de cette thèse. Un remerciement spécial va au Dr Thomas Féraud avec qui j'ai travaillé en (très) étroite collaboration durant presque 3 ans. Je remercie également Florent et Pierre pour le support technique du simulateur et les passages réguliers dans le bureau.

D'un point de vue plus personnel, je fais des bisous à mes potos du midi qui m'ont permis de relâcher la pression. Je cite, par ordre de préférence (ça va faire des jaloux) : Clément, Nicolas, Nathalie, Pierre, Romain, Thomas (mais je l'avais déjà cité avant donc ça compte pas) et tous les autres.

En vrac, j'en profite pour citer tous les gens qui m'ont aidé, directement ou indirectement, à mener à bien cette thèse : Damien, Clément, Maxime, Paul, Christophe, Thierry, Vanessa, Corinne, Françoise, Guddy, Nicolas, Beberth, Fanny, Fanny, Marine, Pierre, Antoine, Romain, François, Christine, Florian, Jean-Baptiste, Sébastien et tous les autres que j'oublie certainement.

Évidemment, je termine par remercier ma famille pour l'accompagnement et le soutien psychologique. Merci à mes parents, beaux-parents, frères, belles-soeurs, beau-frère et tous les autres. Sur ces 3 années la personne la plus importante est sans l'ombre d'un doute Audrey à qui je dois d'en être arrivé là aujourd'hui. C'est grâce à son soutien et à ses encouragements que vous avez la chance (!) de lire ce bel ouvrage, alors merci à elle !

Puisqu'il me reste un petit peu de place, je me remercie d'être aller au bout de la thèse aussi, parce que dans le fond il n'y a pas de raisons de ne pas le faire. Je remercie également les arbres de me fournir les jolies feuilles utilisées pour ce mémoire. J'aimerais terminer par une citation de Winston Churchill qui prend tout son sens dans le cadre d'une thèse :

« Réussir, c'est aller d'échec en échec sans perdre l'enthousiasme. »

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

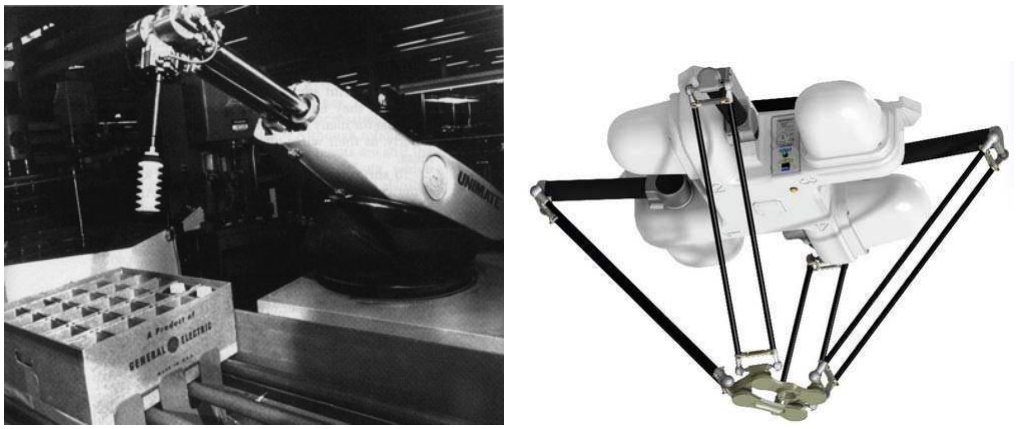
Introduction et contexte

Sommaire

1.1 Robotique et mobilité

La robotique est un domaine dont le nombre d'applications est en augmentation constante depuis son invention. Cet essor découle essentiellement du besoin d'automatiser des tâches répétitives et difficiles. En plus d'épargner des travaux physiquement épuisants à l'Homme, la robotique permet également d'effectuer de substantielles économies dans un monde de plus en plus compétitif et ouvert. Au fur et à mesure que des progrès ont été faits, la diversité des situations où la robotique était applicable a créé de nouvelles branches plus spécialisées, chacune avec ses objectifs et contraintes bien particuliers.

Le travail à la chaîne a, en grande partie, été supplanté par la robotique industrielle. Des bras manipulateurs sont maintenant utilisés afin de tordre, percer, peindre, assembler ou encore déplacer les matériaux nécessaires à l'élaboration d'un produit commercialisable (voir figure 1.1). La miniaturisation des composants ainsi que la croissante fiabilité des technologies employées ont largement contribué à la diffusion de ces robots, tout comme à leur acceptation auprès des professionnels puis du grand public.



(a) Robot Unimate, commercialisé en 1961 (b) Robot Adapt Quattro, conçu en 2007

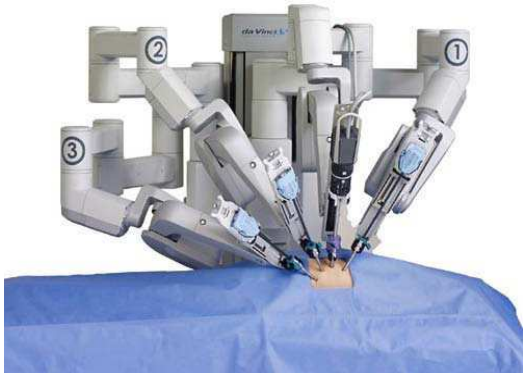
FIGURE 1.1 – Évolution des robots industriels en plus de 40 ans

Des machines intelligentes ont peu à peu été intégrées un peu partout. Le pan de la robotique domestique en est un bon exemple avec l'arrivée et la diffusion récente d'aspirateurs capables de nettoyer plusieurs pièces quotidiennement sans aucune intervention humaine (figure 1.2). Il s'agit certainement du type de robot le plus populaire actuellement. Il est aussi l'un des rares robots ménagers à exister. En effet, la limite de ces machines est souvent liée à la notion d'autonomie : elles doivent être suffisamment intelligentes pour ne pas nécessiter de présence humaine pour fonctionner. Cela est souvent difficile à accomplir avec des machines de petite taille qui ont des ressources limitées.

À l'inverse, les robots à visée médicale sont majoritairement conçus pour être

utilisés par des médecins et non en complète autonomie. Le but est souvent de faciliter le travail du chirurgien en fournissant des nouveaux moyens d'interagir avec les patients (figure 1.2).

(a) Roomba, vendu à plus de 5 millions d'exemplaires en 2009



(b) Da Vinci, 1750 exemplaires utilisés en 2011

FIGURE 1.2 – Robotique domestique et médicale

Globalement, il est possible de construire deux grandes catégories pour regrouper les robots : ceux mobiles, dont fait partie le Roomba par exemple, et ceux fixes (bras manipulateurs, machines médicales...). La mobilité peut être gérée de deux manières différentes : soit en téléopérant le robot, soit en le dotant d'une autonomie suffisante de manière à ce qu'il puisse se déplacer seul dans un environnement donné. Ce dernier cas implique que la machine soit suffisamment "intelligente" pour comprendre l'environnement qui l'entoure. De manière réductrice, un aspirateur autonome doit être capable d'éviter tous les objets d'une pièce, qu'ils soient fixes ou mobiles, de parcourir l'intégralité de la surface à nettoyer et de revenir à sa station de recharge. Chacune de ces actions nécessite de prendre des décisions par rapport aux informations disponibles à un moment donné.

Une grande variété est à noter dans la forme et la façon de se mouvoir de ces robots. Cela est fortement conditionné par l'environnement applicatif visé. De fait, la communauté scientifique sépare souvent les robots mobiles en trois grandes branches : les robots terrestres (d'intérieur ou d'extérieur), les robots aériens et enfin les robots sous-marins (voir figure 1.3). Cette catégorisation vient aussi du fait qu'il est extrêmement difficile de concevoir une intelligence qui puisse s'adapter à plusieurs milieux. Souvent, les équipes de recherche travaillant sur des environnements différents sont bien distinctes.

La catégorie "terrestre", dont ce manuscrit va traiter, se divise encore en fonction du moyen de déplacement du robot. Des chenilles, des pattes ou même des jambes peuvent être utilisées comme le montre la figure 1.4, même si le plus courant reste encore les roues. Le type de sol sur lequel le robot est amené à évoluer influe grande-

(a) Cybercar de l'INRIA

(b) Aeryon Scout de Aeryon Labs

(c) SeaExplorer par ACSA

FIGURE 1.3 – Des exemples de robots mobiles

ment sur la motorisation de celui-ci. Par ailleurs, les besoins sont très différents dans un milieu urbain et dans un contexte désertique par exemple. Les environnements ruraux sont également différenciés des autres puisque le manque de structure du sol poussent généralement à développer des véhicules bien plus volumineux que dans les autres milieux.

Les objectifs attribués à ces robots terrestres peuvent être variés. On peut noter le transport de marchandises, une visée éducative ou encore tout simplement le divertissement. Bien évidemment, ce sont loin d'être les seuls domaines où ils sont utilisés. Les systèmes de surveillance en font un usage croissant, tout comme le milieu agricole. De même, le transport de personnes (Intelligent Transportation Systems ou ITS en anglais) est souvent au coeur des problématiques associées à la robotique mobile.

La mobilité est un aspect essentiel de cette thèse. En effet, le but est ici de faire collaborer des véhicules afin qu'ils soient à même de savoir où ils se situent les uns par rapport aux autres. Le cadre est bien ici les robots terrestres. De manière plus précise, ces derniers peuvent être assimilés à des véhicules puisque l'on s'oriente vers des robots capables de transporter des personnes ou des marchandises. Cela implique des robots faisant une certaine taille mais ne définit pas pour autant la vitesse de ceux-ci. Dans le cadre du transport de personnes par exemple, la vitesse d'un véhicule peut facilement varier de quelques kilomètres par heure pour une

(a) NAO, petit robot humanoïde

(b) BigDog, robot quadrupède pour le transport de matériel

(c) Morph, robot à chenilles de LEGO Mindstorms

FIGURE 1.4 – Des robots terrestres avec des systèmes de mobilité différents

voiturette de golf à plus de 130 km/h pour une voiture plus classique. Cet écart a une grande influence sur la façon dont le système est conçu, tout comme sur le design de l'algorithmie destinée à l'animer.

Dans tous les cas, le système de perception du véhicule doit être capable de fournir des informations suffisamment détaillées pour permettre des actions complexes en toute sécurité.

1.2 Perception pour la robotique mobile

1.2.1 Place de la perception

Tout système de robotique mobile, voire de robotique, s'inscrit dans une logique perception - contrôle - action. Le schéma de la figure 1.5 montre les interactions entre ces différents modules. Son fonctionnement est simple : l'analyse de l'environnement ainsi que du véhicule lui-même constitue la partie perception. Ces informations vont

permettre au module de contrôle de décider de l'action à accomplir. Celle-ci sera, bien entendu, réalisée par la partie éponyme. Le retour de la dite action pourra ou non être communiqué à la perception afin d'orienter l'analyse. De manière similaire, le contrôle et la perception peuvent être amenés à échanger de l'information afin de valider ou de rejeter une première décision prise. L'ajout d'informations du côté de la perception peut en effet complètement changer le résultat de l'algorithme de contrôle.

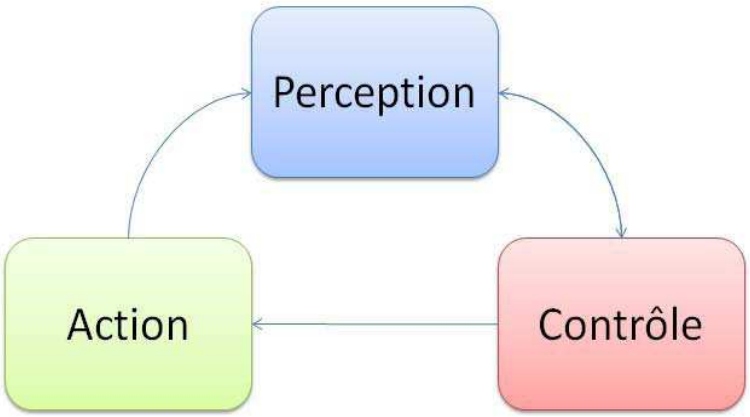


FIGURE 1.5 – Schéma général d'un système robotique

Considérons une application dont l'objectif est de rejoindre un point précis, tout en évitant d'écraser des piétons (ce qui est tout à fait louable). Le module de perception aura, dans ce cas, pour but principal de détecter les piétons à l'aide de capteurs (une caméra par exemple) tout en se localisant dans l'environnement. La partie contrôle quant à elle, devra décider du chemin à emprunter afin d'éviter les passants dont la détection aura été fournie par la perception. Enfin, la trajectoire validée sera effectuée au sein de la brique action qui devra traduire en commandes pour le véhicule le parcours donné par le module de contrôle. Le déplacement réellement effectué pourra être transféré à la perception afin de calculer la position des piétons précédemment détectés et ainsi vérifier leur présence pour la nouvelle position du robot. Le schéma général itérera jusqu'au point d'arrivée.

1.2.2 Système de localisation

Dans ce manuscrit, nous nous concentrerons majoritairement sur le module de perception. L'objectif étant de pouvoir localiser précisément les membres d'une flotte de véhicules, la fusion des données issues des capteurs est un point central. La partie guidage automatique ainsi que la prise de décision relative à des événements (évitement d'obstacles, détection de dangers...) sont des aspects qui ne seront pas abordés ici.

Le système de perception qui sera présenté tout au long de ce mémoire peut

être assimilé à un système de localisation, l'objectif étant de savoir où se situe chacun des véhicules d'une flotte. Pour ce faire, il est nécessaire d'équiper ces robots avec des capteurs de manière à ce qu'ils se repèrent dans l'environnement. Il en existe différentes sortes, à des prix très variables, chacun ayant ses avantages et ses défauts. Deux catégories sont généralement utilisées pour les regrouper : les capteurs dits proprioceptifs et extéroceptifs. Les premiers donnent des informations sur l'état même du véhicule. On retrouve, entre autres, les odomètres qui renseignent sur le déplacement effectué entre deux moments successifs (mesure relative) et les centrales inertielles qui elles permettent d'obtenir les vitesses linéaires et angulaires d'un robot mobile.

La seconde catégorie regroupe tous les capteurs permettant d'avoir des informations à l'aide de l'environnement. Les GPS en sont un bon exemple puisqu'ils utilisent des satellites afin d'obtenir une position absolue à un instant donné. Dans le cadre de la localisation d'un véhicule, les capteurs extéroceptifs (hormis les GPS) sont souvent utilisés pour reconnaître des portions de l'environnement déjà vues auparavant. La mesure du décalage d'une entité dans plusieurs données issues du capteur va permettre de déduire le déplacement du robot. Il est également possible d'identifier directement un véhicule distant et ainsi d'en inférer sa position relativement au capteur ayant servi à le détecter. Les plus communément employés sont les caméras, les télémètres et les radars.

Les intérêts de chacun de ces capteurs seront développés ultérieurement dans ce manuscrit et ce relativement à l'algorithme de localisation que l'on cherche à construire.

1.2.3 Extension au multivéhicule

Au cours des dernières décennies, la communauté autour de la robotique mobile s'est essentiellement concentrée sur la localisation d'un unique véhicule. L'extension de ces algorithmes vers des applications nécessitant plusieurs véhicules n'a été envisagée que très récemment. Pourtant, de nombreuses situations nécessitent, ou peuvent tirer avantage, d'un ensemble de robots fonctionnant de manière coopérative via des communications sans fil (voir figure 1.6).

Parmi les applications citées précédemment, les ITS tirent naturellement parti d'une extension à plusieurs véhicules. Un système de localisation efficace pourrait à terme conduire à des voitures entièrement autonomes. Le milieu agricole est également concerné par l'utilisation d'une flotte de robots. En effet, plutôt que d'utiliser une seule grosse machine, il est envisageable de se tourner vers une multitude de plus petits robots, chacun s'occupant, par exemple, d'un unique sillon. L'intérêt ici serait surtout d'éviter la compaction des sols causée par les machines volumineuses. De nombreuses autres possibilités pourraient être citées puisque le fait d'avoir plusieurs véhicules permet à la fois d'être plus rapide pour couvrir une zone et d'être plus précis dès lors qu'au moins deux véhicules passent au même endroit.

Néanmoins, le fait que l'émergence de ces systèmes soit très récente n'est pas anodin. D'importantes contraintes rendent la tâche beaucoup plus compliquée qu'avec

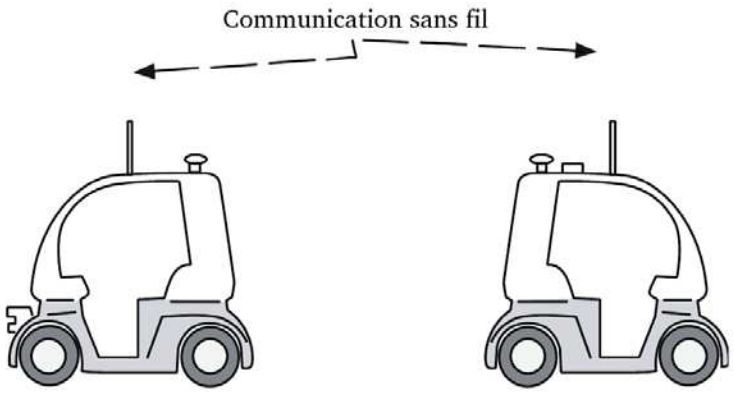


FIGURE 1.6 – Coopération dans le cadre d'un système multivéhicule

un seul véhicule. La présence d'une couche réseau est un premier point. Il faut concevoir l'application de telle sorte qu'elle soit capable de gérer les ruptures de communication qui peuvent intervenir. De même, les latences ainsi que les désynchronisations doivent être prises en compte. Il faut aussi décider de quelles informations envoyer et quand. En effet, la bande passante nécessaire à l'envoi et à la réception de données n'est pas infinie et dépend fortement du nombre de véhicules impliqués dans les échanges. La sélection des informations à envoyer joue également un rôle crucial dans la façon dont vont pouvoir être calculées les positions des robots. Sans a priori initial, un véhicule devra également être capable de situer chaque robot par rapport à lui-même, ce qui requiert des algorithmes robustes. De manière similaire à la bande passante, le coût calculatoire doit être au centre de la conception de l'application multivéhicule, la gestion des localisations des membres de la flotte induisant un surcoût non négligeable. Enfin, il faut aussi garder en tête le coût financier afin de rendre l'application commercialement viable. Chaque véhicule étant amené à embarquer des capteurs, il faudra être prudent lors du choix de ceux-ci car la solution globale verra son prix multiplié par le nombre de robots de la flotte. L'ensemble de ces points sera traité dans ce mémoire.

1.3 Organisation du mémoire

Avant même de développer un système multivéhicule, il est nécessaire de construire un algorithme où chaque robot est capable d'estimer sa propre position. Cela sera l'objet du chapitre 2. Bien évidemment, le design d'un tel processus sera fait en fonction des contraintes fixées par l'application multivéhicule (coût financier, temps de calcul, quantité d'informations à traiter...). Pour ce faire, nous présenterons d'abord les capteurs utilisés puis l'algorithme de localisation employé.

Les problématiques amenées par les choix faits seront étudiées lors du chapitre 3. Dans celui-ci, les améliorations apportées au système monovéhicule seront détaillées.

L'algorithme construit sera ensuite testé dans des situations variées afin de valider les différentes innovations présentées au fur et à mesure du chapitre.

Le chapitre suivant (chap. 4) se concentrera sur la dérive naturelle de ces algorithmes de localisation. Ce point est crucial car la divergence de ces systèmes se retrouve exacerbée lors de l'extension au multivéhicule. Un état de l'art sera dressé et une solution ainsi que son intégration seront proposées.

L'extension au cas multivéhicule sera décrite durant le chapitre 5. Après une analyse de la littérature dans le domaine, une architecture résolvant les différentes problématiques citées précédemment sera présentée. L'estimation de la localisation des robots d'une même flotte sera également largement abordée.

Bien que chaque contribution soit validée au fur et à mesure du mémoire (via simulation ou avec des expérimentations réelles) le chapitre 6 présentera les résultats globaux de l'algorithme multivéhicule développé. Divers exemples de trajectoires dans des situations variées seront exposés afin de démontrer l'efficacité de notre application.

Enfin, le chapitre 7 proposera une synthèse des résultats obtenus ainsi que des limites de l'application conçue. Les perspectives d'évolution de l'algorithme seront également abordées.

Chapitre 2

Localisation d'un véhicule

Sommaire

Construire un algorithme de localisation multirobot nécessite avant toute chose que chaque véhicule soit capable de se localiser. En effet, cette information de position sera partagée avec les autres membres de la flotte et devra être la plus juste possible. Au-delà même de ces considérations, façonner la partie monovéhicule pour qu'elle soit en adéquation avec son extension à plusieurs robots est essentiel. Chacun des choix concernant les aspects plus bas niveau va régir les performances globales de l'application multivéhicule.

Afin de prendre une décision éclairée, il convient de définir le cadre applicatif de nos développements. En effet, celui-ci va permettre de fixer les contraintes générales de notre système multivéhicule et donc orienter le choix des capteurs et de la méthode de localisation monovéhicule. L'objectif est de construire un système de localisation suffisamment général pour qu'il puisse ensuite être adapté à des applications bien précises. Cela implique que le nombre de véhicules peut être élevé et qu'il est donc important de limiter la quantité de données à échanger. Dans un contexte agricole par exemple, plus d'une dizaine de robots pourraient être utilisés en parallèle. On peut aussi imaginer un système de transport de personnes, dédié à une zone géographique précis, qui nécessiterait un volume de véhicules similaire.

Ces deux exemples sont intéressants puisqu'ils permettent aussi de constater que l'environnement peut être varié. Dans le contexte urbain, beaucoup d'informations peuvent être exploitées afin d'aider la localisation. Cela sera un peu moins le cas dans des environnements plus ouverts. Il faudra donc être vigilant à ce que la méthode et les capteurs choisis puissent opérer dans des contextes différents. Au niveau de la vitesse des véhicules, celle-ci sera peu élevée (autour de 2 mètres par seconde) afin de permettre une première validation dans des conditions maîtrisées.

Enfin du point de vue de la justesse de localisation souhaitée, il est difficile de définir un ordre de grandeur car celui-ci dépend de la tâche visée. Néanmoins, il est essentiel que le système développé puisse être spécialisé pour atteindre des précisions autorisant la conduite automatique (autour de la vingtaine de centimètres) lorsque cela est possible car de nombreuses applications en découlent.

Ce chapitre sera entièrement consacré à la conception de l'algorithme de localisation monovéhicule. Les différents choix technologiques seront toujours étudiés par rapport à la littérature ainsi qu'en relation avec l'impact qu'ils auront sur le multivéhicule. La section 2.1 présentera les différents capteurs disponibles pour la localisation. La section qui suit (section 2.2) se concentrera quant à elle sur les méthodes pour la localisation. La section 2.3 exposera ensuite le fonctionnement de la solution envisagée, tout d'abord de façon générale dans les sous-sections 2.3.1 et 2.3.2, puis par rapport aux capteurs sélectionnés (sous-section 2.3.3).

2.1 Analyse des capteurs pour la localisation

La localisation d'un unique véhicule est donc un domaine qui est aujourd'hui devenu un point central de la robotique mobile. De nombreux chercheurs travaillent sur cette thématique, que cela soit du côté de la conception des capteurs ou encore dans l'élaboration de méthodes pour la localisation. Afin de faciliter la compréhension, ces deux parties seront séparées mais traitées en relation l'une avec l'autre dans la suite de cette section.

Comme énoncé précédemment, on distingue généralement les capteurs proprioceptifs, renseignant sur l'état même du robot, de ceux dits extéroceptifs qui donnent des informations sur l'environnement. Nous allons commencer par étudier la première catégorie.

2.1.1 Capteurs proprioceptifs

2.1.1.1 Capteurs inertiels

Les capteurs inertiels exploitent des propriétés physiques pour extraire les accélérations (accéléromètres) ou les rotations (gyroscopes, gyromètres). Les centrales inertielles peuvent être considérées comme des capteurs, quand bien même il s'agit plus d'un regroupement de capteurs. En effet, celles-ci embarquent 3 gyromètres ainsi que 3 accéléromètres. Cela permet d'estimer les vitesses (linéaire et angulaire) ainsi que la pose complète d'un mobile, c'est-à-dire la position 3D et les 3 angles associés (roulis, tangage, lacet). L'intégration au cours du temps des mesures fournies par la centrale inertielle donne des résultats généralement satisfaisants. Bien évidemment, la qualité des capteurs qui la compose va fortement influencer la précision. Le prix est donc un aspect important lors du choix d'une centrale inertielle. Il est également possible d'utiliser uniquement des gyromètres sans centrale inertielle ce qui permet souvent d'avoir de meilleurs résultats avec un coût similaire.

Hormis le prix, les capteurs inertiels sont affectés par des erreurs cumulatives. Ceux-ci ne fournissent pas de valeurs absolues mais quantifient des mouvements. Ainsi, si une erreur de mesure est faite à un instant donné, elle sera conservée dans toutes les estimations suivantes. Afin de compenser ce problème, les capteurs inertiels sont souvent couplés avec d'autres informations.

2.1.1.2 Odomètre et angle volant

La mesure odométrique est à rapprocher des informations obtenues en sortie d'une centrale inertielle. Des capteurs, équipés sur les roues d'un véhicule, permettent de quantifier le déplacement accompli. Le principal avantage d'un odomètre est son coût ainsi que le fait qu'il est maintenant équipé en standard sur les véhicules. Il est généralement associé au capteur d'angle volant qui comme son nom l'indique renseigne sur l'angle de braquage des roues. Ces deux informations, couplées à un modèle d'évolution du robot, permettent d'obtenir des résultats de localisation convenable.

Néanmoins, tout comme les capteurs inertiels, les informations odométriques correspondent à des déplacements successifs rendant les erreurs cumulatives au fil du temps. Sur de longues distances, les écarts entre ce qui est estimé par odométrie et ce qui est réellement parcouru deviennent considérables. Les données odométriques (déplacement et angle de braquage) seront ainsi très souvent utilisées pour une première estimation qui sera ensuite affinée via un autre capteur. Contrairement à la centrale inertielle en revanche, uniquement le cap du robot (lacet) et la distance parcourue peuvent être estimés via l'odométrie.

2.1.2 Capteurs extéroceptifs

Les différents capteurs présentés jusqu'ici sont ceux principalement utilisés pour renseigner l'état du véhicule. Les suivants rentrent, quant à eux, dans la catégorie des extéroceptifs. Les utiliser induit bien souvent de suivre des objets de l'environnement afin d'obtenir une information de déplacement. Généralement, ces données sont stockées dans ce qui est appelé une carte. Celle-ci peut être construite et utilisée en temps réel ou être fournie au véhicule et exploitée par l'algorithme de localisation. Parmi ceux présentés, le GPS fait figure d'exception puisqu'il fournit des informations absolues et non relatives.

2.1.2.1 GPS

Le GPS (Global Positioning System) est très certainement le capteur de positionnement le plus populaire. Ce système de localisation par satellites permet d'obtenir une position absolue dont la précision varie en fonction du nombre de satellites disponibles. Le principe de fonctionnement est le suivant : des satellites parfaitement localisés transmettent en permanence leur position ainsi que la date associée à celle-ci. Un récepteur décode les signaux reçus et peut ainsi estimer, par triangulation, sa position sur la planète.

Créé à des fins militaires, le système GPS s'est ensuite ouvert aux civils avec des précisions volontairement dégradées. En 2000, cette contrainte a été levée permettant ainsi d'atteindre une précision d'une dizaine de mètres. Cela s'avère généralement suffisant comme aide à la navigation, d'autant plus que la position est généralement corrigée par la mise en correspondance avec la route. La dénomination de GPS est quelque peu trompeuse puisque plusieurs systèmes du même acabit existent. On peut citer GLONASS, Beidou et Galileo (qui n'est pas encore opérationnel) qui sont les équivalents russe, chinois et européen du système GPS américain. Ceux-ci offrent une précision similaire au GPS, voire même légèrement meilleure sous certaines conditions dans le cas de Galileo. Quelque soit le système de positionnement utilisé, il est important de noter que la qualité de la localisation fournie dépend du nombre de satellites disponibles.

Pour des applications visant à la conduite de manière autonome, cette précision est loin d'être suffisante. Des solutions, toujours basées sur les GPS, permettent d'améliorer ce point. Les GPS RTK (Real Time Kinematic) utilisent une station

terrestre dont la position est parfaitement connue. Celle-ci envoie les correctifs à appliquer au récepteur GPS afin d'améliorer la précision (voir figure 2.1). Cela permet d'obtenir des positions justes au centimètre près. Néanmoins, ces systèmes sont extrêmement onéreux. Ces capteurs sont souvent utilisés comme vérité terrain afin de quantifier la justesse de résultats de localisation.

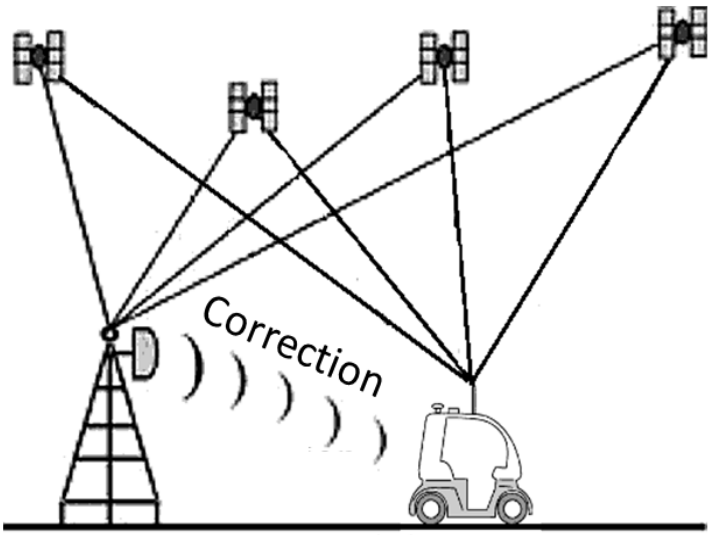
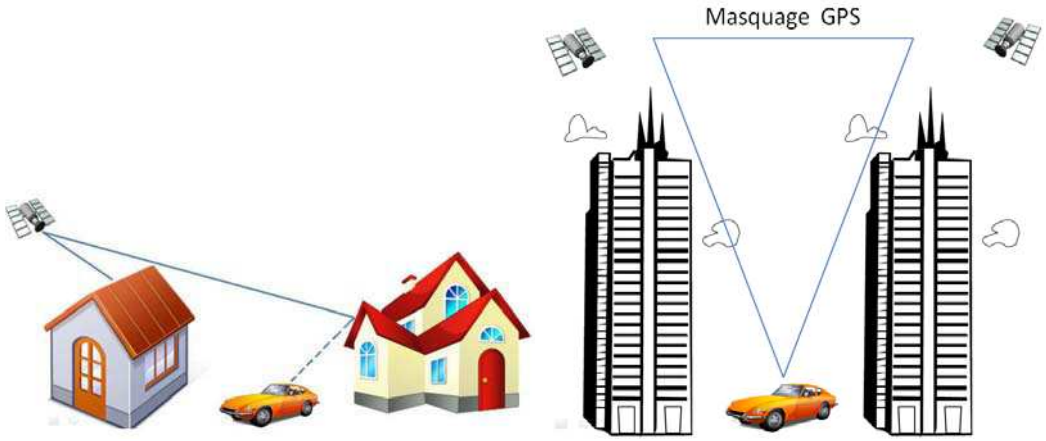


FIGURE 2.1 – Fonctionnement d'un GPS RTK avec la correction par la station

La localisation absolue fournie par des GPS bas coûts, bien qu'imprécise rend beaucoup plus facile le travail coopératif puisque toutes les positions sont données dans un même référentiel. Néanmoins, les GPS souffrent d'autres désavantages. Les deux principaux sont les multitrajets et les pertes du signal dans des environnements cloisonnés. Dans le premier cas, les signaux émis par les satellites sont réfléchis par des bâtiments au lieu d'être reçus directement par le GPS. Ainsi, le temps de vol est allongé et l'estimation de la position est biaisée. Ce phénomène est dangereux car le récepteur n'est pas conscient de cette erreur de positionnement : il y a perte de l'intégrité (la position réelle du véhicule n'est pas située dans l'incertitude de mesure). Des solutions palliatives existent néanmoins afin de contrer ce problème comme la technologie RAIM (Receiver Autonomous Integrity Monitoring) qui utilise la redondance d'information pour lever l'ambiguïté.

Le deuxième cas est différent. Il correspond à la perte du signal GPS à cause d'un environnement qui masque les satellites. Cela apparaît souvent en milieu urbain à cause des bâtiments. On parle de canyon urbain. Une vue schématique de ces deux problèmes est donnée en figure 2.2.



(a) Erreur de positionnement à cause de multitrajets (ligne réfléchie en pointillés)

(b) Perte du positionnement à cause d'un canyon urbain

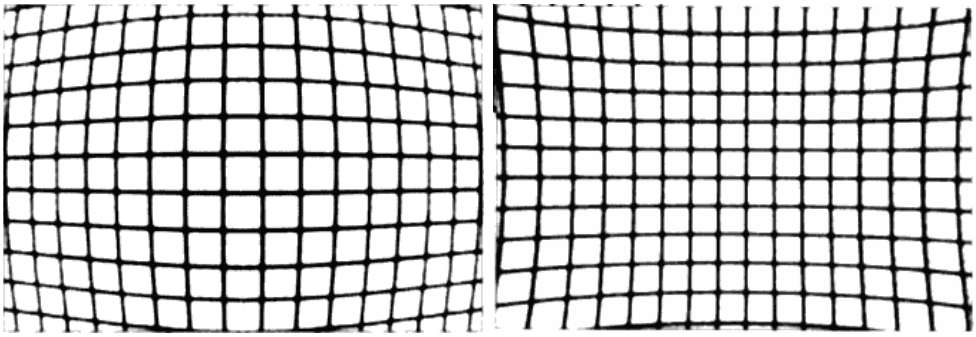
FIGURE 2.2 – Problèmes affectant les GPS

2.1.2.2 Caméra

Les caméras commencent à occuper l'habitacle de nos voitures depuis quelques années maintenant. Elles ont un faible coût et fournissent beaucoup de données. L'information visuelle étant celle sur laquelle l'humain se base le plus, il est naturel de s'orienter vers ces capteurs pour des applications de localisation. Le principe est souvent le suivant : extraire des objets statiques dans l'image qui soient facilement reconnaissables et les suivre. En mesurant les écarts dans l'image entre les positions successives des objets, il est possible de déduire le déplacement du véhicule. L'intérêt majeur des caméras est de fournir beaucoup d'informations qui, regroupées en sous-ensembles de pixels, sont assez facilement différentiables.

Parmi les contraintes de ces capteurs, il faut tout de même noter une phase d'étalonnage. Celle-ci permet de passer les informations de l'image, exprimées en pixels, dans un repère métrique plus classique. Cette étape est essentielle pour pouvoir mesurer les déplacements du véhicule. Les paramètres permettant d'inférer ce changement de repère sont appelés paramètres intrinsèques de la caméra. En plus de ceux-ci, on mesure généralement la position de la caméra par rapport au véhicule afin de pouvoir coupler les informations visuelles avec des données provenant d'un autre capteur. Ce nouveau changement de repère est régi par les paramètres extrinsèques de la caméra. Ceux-ci sont aussi estimés durant la phase d'étalonnage. Cette dernière sert aussi à estimer la distorsion qui affecte l'image (voir figure 2.3). Ce défaut optique, s'il n'est pas pris en compte, peut conduire à des problèmes de suivi.

Les caméras sont grandement affectées par les conditions météorologiques. En effet, une forte pluie, un soleil éclatant ou encore des chutes de neige rendent le suivi d'objets plus difficile. Ce ne sont pas les seuls aspects ayant un impact sur la qualité



(a) Distorsion de type barillet (b) Distorsion de type croissant

FIGURE 2.3 – Schématisation des effets de la distorsion

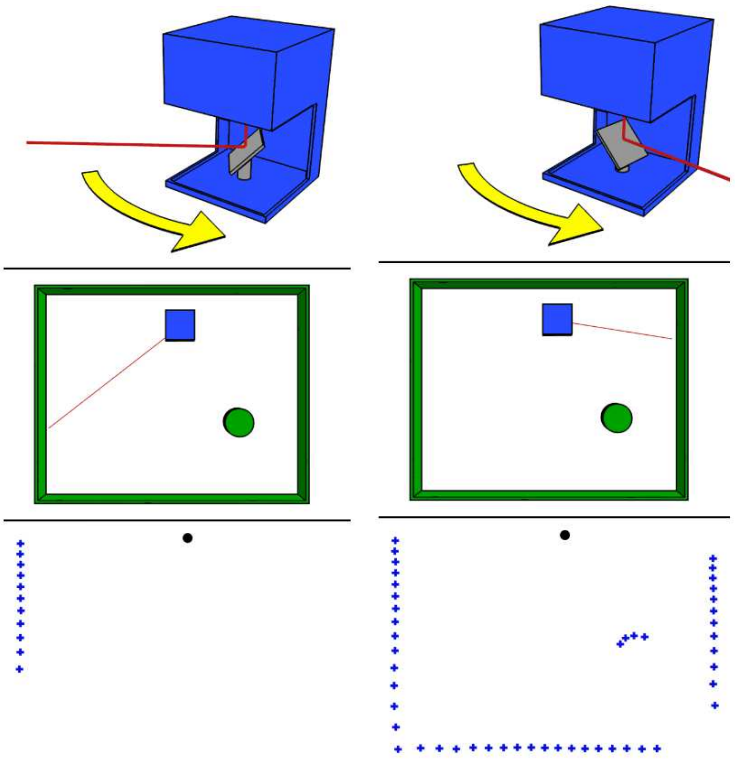
de l'image. On peut par exemple citer : le flou, la résolution du capteur, le temps de prise de vue et de saturation, etc. Les algorithmes construits doivent être robustes afin de résister à de telles conditions.

L'utilisation d'une unique caméra pose le problème de l'estimation de la distance des objets. À partir d'une seule image, la profondeur ne peut être déterminée. C'est pourquoi une paire stéréoscopique (deux caméras) est souvent utilisée. Celle-ci permet de trianguler la position d'un objet dès lors qu'il apparaît dans le champ de vue des deux caméras. Un compromis doit être fait sur l'écart entre les caméras : plus elles sont éloignées, plus l'estimation sera précise (parallaxe suffisante) et à l'inverse plus elles sont proches et plus le champ recouvrant nécessaire à l'estimation de la profondeur sera important. L'étalonnage joue un rôle clef pour une paire stéréoscopique car il permet de synchroniser les deux caméras afin qu'elles délivrent des images sur la même base de temps. Grâce à ce procédé, les informations extraites dans chaque caméra peuvent être comparées.

2.1.2.3 Télémètre laser

Les télémètres laser, souvent abrégés LRF pour Laser RangeFinders, sont une alternative intéressante aux caméras puisqu'ils fournissent directement la distance des objets par rapport au capteur. Le principe est simple : un faisceau laser est émis sur un miroir rotatif permettant ainsi de balayer l'environnement de 90 à 270 ° en fonction du télémètre. Le faisceau laser est réfléchi dès lors qu'un obstacle est percuté. Au retour, le temps de vol est mesuré et ainsi, la distance de l'obstacle peut être connue. La figure 2.4 propose un schéma de fonctionnement simplifié d'un LRF.

Le télémètre laser généralement employé dans la robotique mobile fournie une seule nappe et permet ainsi de n'avoir des informations que sur un seul plan. Ce capteur reste cependant assez cher même si la précision des données est très bonne. La résolution du télémètre laser peut également être problématique. Celle-ci est généralement de l'ordre de quelques points par degré. Reconnaître des objets préalablement



(a) (b)

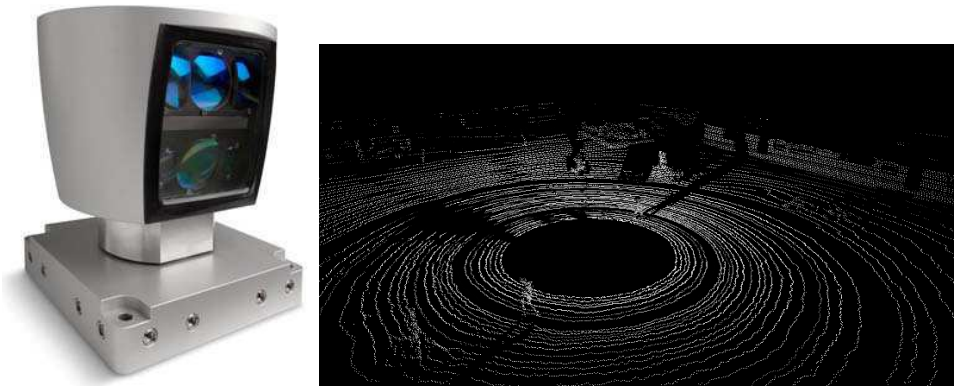
FIGURE 2.4 – Schéma de fonctionnement simplifié d'un télémètre laser

détectés devient alors plus difficile puisqu'ils seront uniquement identifiables grâce à faible nombre de points.

Des télémètres laser à plusieurs nappes existent, permettant ainsi d'avoir une information de hauteur, tout en augmenter la quantité de données. Plus récemment, de nouveaux LRF ont fait leur apparition. Leur intérêt réside dans la possibilité d'avoir des informations 3D sur 360 ° . Toutefois, la quantité de données à traiter est telle qu'il est difficile d'utiliser ce capteur en temps réel. Son prix est aussi très élevé limitant de fait les applications dans lesquelles il est employé. Enfin, la construction d'une carte nécessite que l'on puisse reconnaître des endroits déjà cartographiés. Avec les données fournies par ces capteurs, des plans et des droites 3D sont généralement utilisées, ce qui rend la tâche compliquée.

2.1.2.4 Radar

Le radar est très peu employé dans le domaine de la localisation. Pourtant, celui-ci n'est que très peu affecté par les conditions météorologiques ou les conditions de luminosité, ce qui est un avantage majeur par rapport à la caméra ou au télémètre laser.



(a) Laser 3D Velodyne (b) Exemple d'image laser 3D

FIGURE 2.5 – Laser 3D

Comme le télémètre laser, il s'agit d'un capteur de distance. Son principe est simple : une onde de haute énergie est émise et est réfléchie par les surfaces percutées. Le retour de l'écho va permettre de mesurer la distance à la surface et ainsi de cartographier l'environnement. Dans certains cas, ce capteur peut offrir une vue à 360 ° des alentours. En revanche, sa rotation est souvent lente, entraînant ainsi un important phénomène de distorsion dans les images radar [Vivet 2011].

Par ailleurs, le type de surface sur laquelle est réfléchie l'onde influe sur les résultats obtenus et peut ainsi créer des décalages importants avec la réalité. Il s'agit du speckle. Ce dernier peut générer des obstacles inexistants ou, au contraire, en faire disparaître. Ces capteurs demeurent assez coûteux, au même titre que les télémètres laser. Enfin, le même problème de reconnaissance dans la carte évoqué pour les capteurs télémétriques se pose ici aussi. En effet, le speckle et les effets de flou dans la réponse complexifient la tâche de mise en correspondance des informations.

2.1.3 Bilan des capteurs

Les capteurs présentés dans cette section sont à mettre en relation avec les contraintes établies par rapport au contexte de l'application. Il faut donc une solution peu onéreuse et qui à la fois fournisse suffisamment d'informations pour que différents véhicules puissent reconnaître un même endroit. Pour ces raisons, nous n'avons pas souhaité nous orienter vers des capteurs de distance qui, bien que commodes, sont à la fois chers et complexes à mettre en oeuvre pour la construction de cartes amenées à être utilisées par plusieurs véhicules.

Pour ces mêmes raisons de coût, nous avons décidé de privilégier le duo odomètre et angle volant plutôt qu'une centrale inertielle. L'avantage est que ces capteurs sont maintenant intégrés en standard sur la majorité des véhicules et offrent une première estimation satisfaisante même si elle est de moindre précision qu'avec une centrale inertielle plus coûteuse.

Bien entendu, les informations odométriques seules ne sont pas suffisantes. Ainsi,

nous avons décidé d'adjoindre une caméra. Ce choix n'est pas anodin. La caméra est devenue un standard dans le milieu automobile grâce à son faible prix et la quantité d'informations fournie. De plus, ce capteur est souvent utilisé quand il s'agit de reconnaître des endroits déjà traversés, ce qui correspond tout à fait à une approche multivéhicule. L'avantage de l'associer à l'odomètre est de pouvoir contraindre le facteur d'échelle. En effet, lorsqu'une caméra est utilisée pour de la localisation, tout est mesuré à un facteur d'échelle près qui ne peut être quantifié que dans certaines conditions. Cela ne serait pas gênant si celui-ci ne dérivait pas au fil du temps [Strasdat *et al.* 2010b], rendant ainsi la localisation délicate. Les données odométriques vont pouvoir prévenir ce comportement en imposant la position du véhicule dans un repère métrique.

L'ajout d'un GPS permettrait de lever bon nombre de contraintes pour l'extension à plusieurs véhicules. La localisation dans un repère absolu serait un avantage majeur puisque cela permettrait de connaître la position approximative de chaque membre de la flotte. Néanmoins, les problèmes de disponibilité évoqués ci-avant font qu'il est risqué de choisir un tel capteur pour résoudre le positionnement relatif des véhicules.

Nous avons choisi de développer une solution basée sur le couple odomètre et caméra. Ainsi, nous avons décidé de ne pas intégrer de GPS. Cependant, son utilisation pourrait être bénéfique. Il faudra donc que l'algorithme de localisation permette d'ajouter ce capteur facilement si l'on souhaite étendre notre application.

La solution de localisation de chaque véhicule sera donc locale, et non absolue, et basée sur l'utilisation conjointe d'une caméra avec des informations odométriques (déplacement et angle volant). Les différentes contraintes fixées en début de chapitre sont respectées par ce choix. Il faudra toutefois être vigilant aux faiblesses de ce duo. En effet, l'odométrie a tendance à accumuler les erreurs et une unique caméra ne permet pas une estimation immédiate de la distance. Ces points devront être pris en considération lors de la construction d'un algorithme de localisation. Ces inconvénients seront d'ailleurs largement évoqués dans le chapitre 3.

2.2 Méthodes de localisation pour le SLAM

Le choix de la méthode de localisation est primordial. C'est d'ailleurs une décision à prendre en adéquation avec les capteurs sélectionnés, à savoir dans notre cas une caméra, un odomètre et un capteur d'angle de braquage. Durant cette première partie du manuscrit, il a souvent été question de suivre des objets grâce aux capteurs afin de pouvoir estimer les déplacements d'un véhicule. Ces objets forment ce que l'on a défini comme étant une carte. Les applications visées par ces travaux de thèse ne prennent pas comme pré-requis l'existence d'une quelconque carte. Il est donc nécessaire que chaque véhicule construise la sienne au fur et à mesure, tout en se localisant dans celle-ci. Ce procédé est connu sous le nom de Simultaneous Localization And Mapping (SLAM), ou localisation et cartographie simultanées. Initié dès le début des années 90 [Smith *et al.* 1990][Leonard & Durrant-Whyte 1991],

voire même quelques années auparavant [Smith & Cheeseman 1986], le SLAM est un cadre regroupant toutes les méthodes de localisation basées sur des cartes construites de manière incrémentale. Les lecteurs intéressés pourront d'ailleurs se tourner vers [Durrant-Whyte & Bailey 2006] et [Bailey & Durrant-Whyte 2006] pour une introduction au domaine.

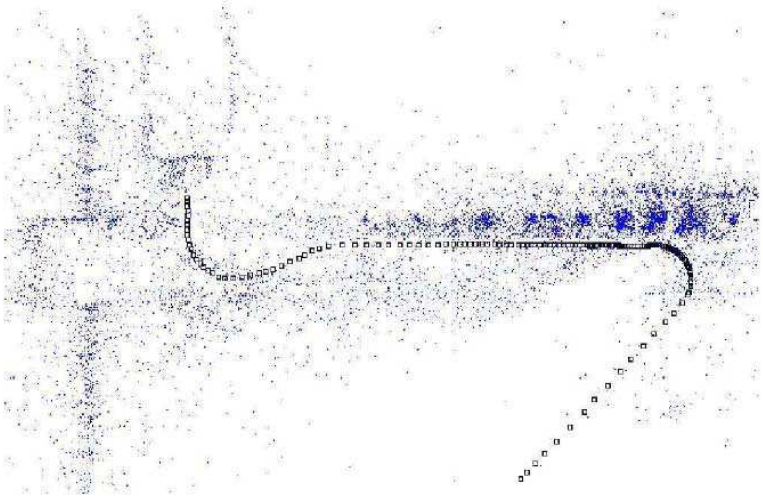
Bien évidemment, toutes les approches de localisation ne se basent pas sur du SLAM. Néanmoins, ces méthodes reposent essentiellement sur des capteurs fournissant une localisation absolue (souvent des GPS RTK), ce qui n'est pas notre cas. Quelques approches de localisation par caméra arrivent à s'extraire du cadre du SLAM. On peut par exemple citer [Lovregrove *et al.* 2011] qui travaille à partir des gradients de l'image ou [Aires *et al.* 2008] qui présente une odométrie via flot optique basée sur les couleurs. Ces méthodes sont en nombre restreint et peu adaptées au contexte multivéhicule. En effet, des informations sur l'environnement doivent être transmises entre les robots afin que ceux-ci puissent savoir où ils se situent les uns par rapport aux autres. Ainsi, toutes les techniques de localisation présentées ci-après s'inscrivent dans la thématique du SLAM en environnement statique appliqué à la vision. Des techniques spécifiques (SLAMMOT) existent néanmoins pour les environnements dynamiques mais ne seront pas abordées ici.

La littérature étant très vaste à ce sujet, cette section sera découpée en deux parties. Nous commencerons par une rapide analyse des méthodes de minimisation globale (sous-section 2.2.1) avant de détailler les différents filtres qui existent (sous-section 2.2.2). Ces deux courants s'opposent dans le principe : la minimisation globale est proche d'un estimateur de maximum de vraisemblance alors que les filtres s'orientent davantage vers la minimisation des erreurs quadratiques. Les techniques de localisation nécessitant la position exacte des amers observés pour fonctionner (triangulation, géométrie épipolaire) ne seront pas abordées ici puisque généralement difficiles à mettre en oeuvre sur des applications réelles. De plus, l'utilisation d'une unique caméra nécessite d'avoir une solution de localisation robuste aux approximations, éliminant de ce fait ces méthodes.

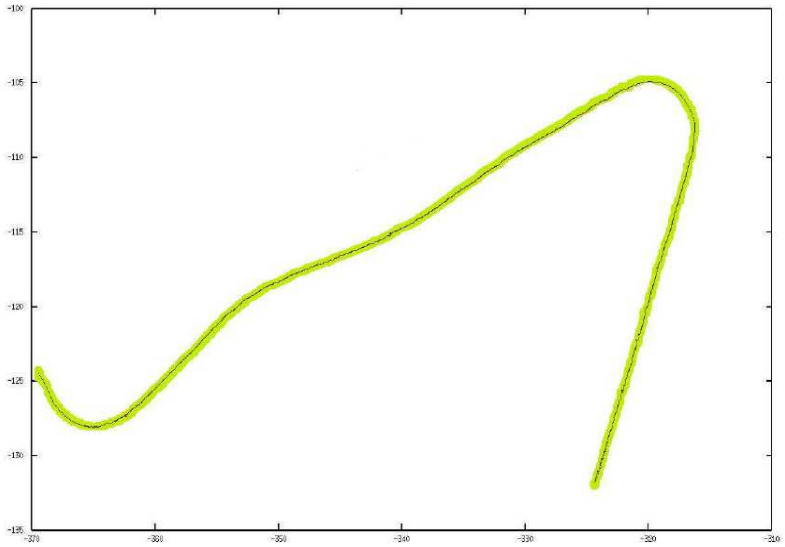
2.2.1 Minimisation globale par ajustement de faisceaux

L'ajustement de faisceaux est une méthode très répandue dans la communauté SLAM. Elle permet d'estimer la pose d'un véhicule compte tenu d'observations faites par les capteurs. La majorité du temps, cette approche repose sur l'utilisation de caméras puisqu'il est nécessaire d'avoir une quantité importante de points à suivre afin d'assurer la bonne estimation de la pose du véhicule. L'idée est simple : des points caractéristiques sont extraits dans l'image et forment la carte. Ils sont ensuite re-projetés dans les images suivantes et des correspondances sont cherchées. Une fois les associations entre les points de la carte et l'image courante trouvées, le but est d'estimer le déplacement ayant affecté le véhicule entre ces images afin de pouvoir minimiser les erreurs de projection des amers (points de la carte) par rapport aux associations trouvées. Les lecteurs intéressés pourront se référer à [Triggs *et al.* 2000] pour une revue détaillée des méthodes basées sur l'ajustement de faisceaux.

L'étape de minimisation globale est effectuée via un algorithme robuste, souvent de type Levenberg-Marquardt [Press *et al.* 1992]. L'inconvénient majeur des techniques d'ajustement de faisceaux est la lourdeur de cette minimisation. Une organisation hiérarchique des traitements a été proposée par [Hartley & Zisserman 2000] et ensuite appliquée à de la navigation autonome par [Royer *et al.* 2005] avec d'excellents résultats de localisation (voir figure 2.6).



(a) Reconstruction 3D (les points bleus sont ceux cartographiés par l'algorithme)



(b) Résultats de localisation (le trait épais est la vérité terrain et le trait fin ce qui est estimé par l'algorithme)

FIGURE 2.6 – Résultats de l'ajustement de faisceaux de [Royer *et al.* 2005] (approximativement 15 cm d'erreur)

Malheureusement, la construction de la carte et la localisation dans celle-ci doivent être séparées car trop coûteuses en temps. Nistér et al. [Nistér *et al.* 2006] se sont tournés vers une estimation de pose différente, faite sur des triplets d'images à chaque fois. L'algorithme repose sur la sélection de la meilleure solution de localisation via RANSAC [Fischler & Bolles 1981]. Néanmoins, bien que temps réel (autour de 13 Hz), l'approche fournie des résultats de localisation moins précis que [Royer *et al.* 2005], les erreurs s'accumulant au fil de la trajectoire.

Enfin, Mouragon et al. ont proposé un algorithme d'ajustement de faisceaux local [Mouragnon *et al.* 2006] permettant une localisation temps réel autour de 7,5 Hz avec de très bons résultats. Ces méthodes de minimisation globale sont donc tout à fait adaptées à la localisation temps réel. Néanmoins, 8000 points 3D ainsi que 240 images clefs doivent être conservés pour l'étape de localisation et ce pour une trajectoire d'environ 200 mètres. Cela constitue une importante masse de données. De plus, au moins 400 points doivent être associés par image pour que l'étape de minimisation se déroule bien. Cela signifie que l'environnement doit être suffisamment texturé pour pouvoir extraire assez de points caractéristiques et les reconnaître. Enfin, les incertitudes associées aux amers ainsi qu'à la pose du véhicule ne sont pas directement disponibles et doivent être calculées.

2.2.2 Filtrage

Les techniques de filtrage sont souvent opposées à celles d'ajustement de faisceaux. Les filtres ont longtemps été considérés comme meilleurs puisque plus facilement utilisables dans des applications temps réel. La sous-section précédente a montré que ce n'est maintenant plus vrai, des améliorations ayant permis d'accélérer les temps de calcul des approches par minimisation globale. Une grande partie de la communauté SLAM a d'ailleurs migré vers celles-ci. Néanmoins, les filtres sont toujours employés, certains cas étant encore propices à leur utilisation. La suite de cette sous-section va détailler les plus populaires dans le cadre du SLAM.

2.2.2.1 Filtre de Kalman

Le filtre de Kalman est l'estimateur le plus populaire encore à ce jour [Kalman 1960]. Celui-ci permet, à partir de mesures bruitées, d'estimer l'état d'un système comme par exemple la pose d'un véhicule ou la position d'amers (points de la carte). Le filtre de Kalman fournit, conjointement à l'estimation de l'état, l'incertitude de celui-ci sous la forme d'une matrice de variance-covariance. Il s'agit d'un estimateur récursif markovien d'ordre 1. Ainsi, uniquement l'état précédent du filtre et les nouvelles mesures sont nécessaires pour produire une nouvelle estimation de celui-ci. Le filtre de Kalman suppose néanmoins que toutes les mesures soient décorrélées et suivent une distribution gaussienne ce qui n'est pas forcément toujours le cas. Sa notoriété est également liée au fait qu'il est optimal dans le cas de fonctions linéaires et qu'il est très peu coûteux.

Son fonctionnement se découpe en 2 étapes : la prédiction et la mise à jour.

La phase de prédiction se charge de faire évoluer l'état depuis l'instant précédent jusqu'à l'instant courant. Les observations fournies par les capteurs viennent ensuite corriger l'état prédit : il s'agit de l'étape de mise à jour. À chaque fois, les incertitudes associées aux mesures, ainsi que celles affectant les différentes variables du vecteur d'état, sont prises en compte dans les calculs. Dans le cadre de la robotique mobile, la phase de prédiction consiste généralement à faire évoluer le véhicule suivant un modèle physique de celui-ci en fonction des données odométriques acquises. Pour l'étape de mise à jour, il s'agit d'observer les amers précédemment cartographiés. La différence entre ce qui est attendu et ce qui est observé va ainsi pouvoir corriger l'état prédit.

Ces 2 étapes sont généralement formalisées comme suit. L'estimation de l'état du système représentée par 2 variables : $\mathbf{x}_{k|k}$ qui est l'estimation de l'état à l'instant k considérant toutes les observations jusqu'à l'instant k, et $\mathbf{P}_{k|k}$ qui est la matrice de covariance (incertitude) associée à l'état $\mathbf{x}_{k|k}$. Soit $\mathbf{F}_k$ la forme matricielle du modèle linéaire d'évolution de l'état et $\mathbf{u}_k$ les commandes effectuées par le robot entre $k-1$ et k, l'étape de prédiction est alors :

$$\mathbf{x}_{k|k-1} = \mathbf{F}_k \mathbf{x}_{k-1|k-1} + \mathbf{B}_k \mathbf{u}_k \tag{2.1}$$

où $\mathbf{B}_k$ est le modèle liant les commandes reçues à l'état (cela est surtout valable dans un contexte de robotique mobile). La prédiction de la matrice de covariance se fait comme suit :

$$\mathbf{P}_{k|k-1} = \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^{\mathrm{T}} + \mathbf{Q}_k \tag{2.2}$$

où $\mathbf{Q}_k$ est le bruit du processus d'évolution.

Enfin, pour la mise à jour, soit $\mathbf{z}_k$ la variable regroupant les observations faites à l'instant k et $\mathbf{H}_k$ la forme matricielle du modèle d'observation liant $\mathbf{z}_k$ à l'état. On peut ainsi calculer l'innovation associée à la mise à jour de Kalman :

$$\boldsymbol{\Delta}_k = \mathbf{z}_k - \mathbf{H}_k \mathbf{x}_{k|k-1} \tag{2.3}$$

Le gain de Kalman est ensuite donné par :

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^{\mathrm{T}} (\mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k)^{-1} \tag{2.4}$$

où $\mathbf{R}_k$ est l'incertitude associée aux observations $\mathbf{z}_k$.

Une fois le gain de Kalman obtenu, la mise à jour peut être calculée et l'état (tout comme l'incertitude) corrigé :

$$\mathbf{x}_{k|k} = \mathbf{x}_{k|k-1} + \mathbf{K}_k \boldsymbol{\Delta}_k \tag{2.5}$$

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{K}_k \mathbf{H}_k \mathbf{P}_{k|k-1} \tag{2.6}$$

L'application du filtre de Kalman au SLAM propose d'excellentes propriétés en terme de convergence des données [Dissanayake *et al.* 2000][Dissanayake *et al.* 2001]. Malheureusement, sous cette

forme, ce filtre est rarement employé puisqu'il requiert des modèles linéaires ce qui est rarement le cas dans des applications de robotique mobile. Ainsi, une formulation adaptée aux systèmes non linéaires a été élaborée.

2.2.2.2 Filtre de Kalman étendu

Présenté dans [Kalman & Bucy 1961], le filtre de Kalman étendu ou EKF (Extended Kalman Filter) a pour but d'estimer l'état de systèmes régis par des modèles non linéaires. Son fonctionnement est très similaire au filtre de Kalman classique si ce n'est qu'une étape de linéarisation est nécessaire. Celle-ci est faite autour du point de fonctionnement, à savoir l'estimée courante. Pour la prédiction, la linéarisation du modèle d'évolution sera faite autour de $\mathbf{x}_{k-1|k-1}$. Pour l'étape de mise à jour, la linéarisation sera accomplie autour de $\mathbf{x}_{k|k-1}$. Dans les 2 cas, ces phases de linéarisation correspondent au développement de Taylor d'ordre 1, à savoir le calcul des matrices jacobiennes (matrices aux dérivées partielles) associées aux modèles. Plus de détails concernant l'EKF dans le cadre du SLAM seront donnés en section 2.3.

L'optimalité de l'EKF a été démontrée dès lors que la linéarisation se fait autour de la vraie valeur du vecteur d'état. Dans les faits, il s'agit de ce que l'on veut estimer et cette valeur n'est pas disponible. Plus l'estimée sera correcte, plus l'étape de linéarisation sera juste. En revanche, l'éloignement de la vraie valeur peut causer des problèmes d'inconsistance : la vérité ne serait alors plus comprise dans l'incertitude associée à l'estimation. Dans le cadre d'un SLAM, la divergence du vecteur d'état peut être conséquente [Julier & Uhlmann 2001b][Bar-Shalom *et al.* 2001].

Néanmoins, beaucoup d'approches se basent tout de même sur l'EKF. En effet, les estimées sont souvent suffisamment proches de la vérité pour éviter ces problèmes. On peut par exemple citer l'approche de Leonard et al. dans [Leonard & Feder 2000] qui utilise avec succès un EKF SLAM dans de grands environnements. L'utilisation d'un EKF pour le SLAM est souvent associée à l'emploi de capteurs de profondeur. Par exemple, dans [Guivant *et al.* 2000], un télémètre laser est utilisé. Cette approche a ensuite été étendue dans [Guivant & Nebot 2001] pour couvrir de grandes distances. Toujours avec des capteurs de distance, nous pouvons citer l'approche de [Newman & Leonard 2003] où un sonar permet une application sous-marine de l'EKF SLAM. Ces résultats suivent des travaux précédemment entrepris sur l'exploration, toujours par filtre de Kalman étendu [Newman *et al.* 2002]. Enfin, d'autres approches, couplant laser et vision avec un EKF, ont été expérimentées [Newman *et al.* 2006]. Bien que moins populaires, les processus utilisant une unique caméra pour l'EKF SLAM ont tout de même été largement étudiés [Davison 2003][Clemente *et al.* 2007].

Concernant les performances de l'EKF, celles-ci sont liées au nombre d'amers qui sont dans le vecteur d'état. Dans le cas de figure où la taille de la carte ne fait que croître (pas de suppression des anciens amers), cette charge ne fait qu'augmenter. En effet, la taille du vecteur d'état et de la matrice de covariance associée ont un gros impact sur le temps de calcul de la mise à jour. Il a été montré dans [Malartre 2011]

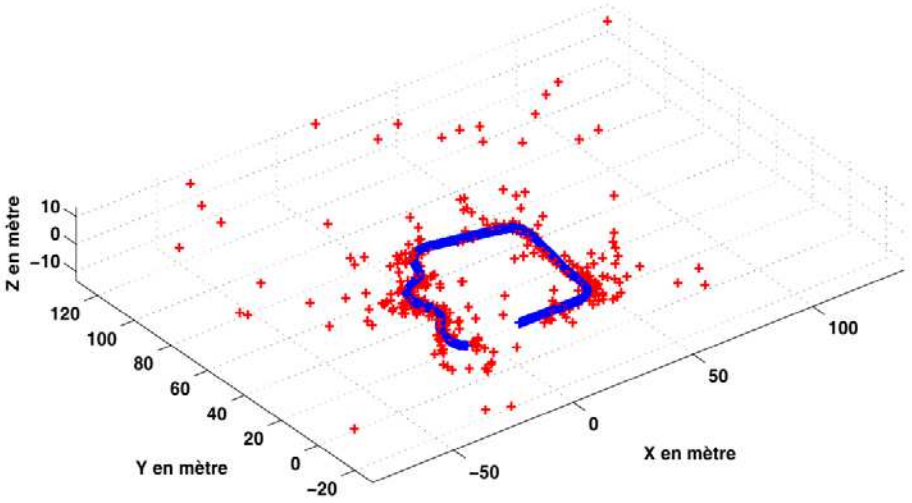
que traiter les observations une par une (c'est-à-dire que chaque observation entraîne une mise à jour) concède un gain de temps considérable. À l'inverse, prendre en compte toutes les observations d'un coup rend la mise à jour du filtre plus longue mais moins sensible aux fausses associations. Dans le premier cas (mise à jour itérative), si une observation est mauvaise, elle pourrait faire complètement diverger le filtre, empêchant ainsi le processus d'estimation de se dérouler correctement. Dans le second cas, celle-ci serait lissée et pondérée par le reste des observations qui tireront la mise à jour de l'état dans une autre direction. Ces quelques remarques sont valables pour l'ensemble des techniques de filtrage présenté ici.

Afin de conserver une mise à jour à temps presque constant, il est toujours possible de supprimer des amers dès lors qu'ils ne sont plus visibles. Le SLAM s'apparente dès lors davantage à de l'odométrie visuelle. En fonction des applications (pas de boucle dans la trajectoire par exemple), cela peut s'avérer intéressant. De plus, il est toujours possible d'extraire les amers qui ne sont plus visibles pour les stocker en dehors du vecteur d'état et ainsi alléger la charge calculatoire. Dans tous les cas, il a été constaté que les filtres de Kalman ne requièrent que peu d'amers pour atteindre une bonne précision [Strasdat *et al.* 2010a]. Bien sûr, cela est à nuancer en fonction de la qualité des capteurs et donc des amers. À titre d'exemple, nous pouvons citer [Féraud 2011] qui présente des résultats de localisation par EKF en se basant sur une carte de référence préalablement construite. Il s'agit du même contexte que dans [Royer *et al.* 2005]. Les résultats sont exposés sur la figure 2.7.

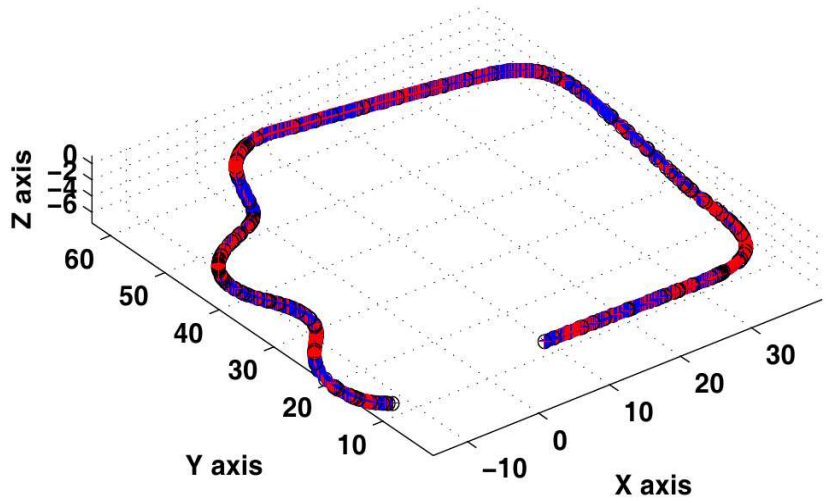
On peut constater que la précision atteinte est similaire à celle obtenue par Royer et al. dans [Royer *et al.* 2005]. Néanmoins, ici pour une trajectoire d'environ 170 mètres, uniquement 319 amers ont été cartographiés.

Il est également possible de réduire le temps des mises à jour sans avoir à supprimer d'amers. Il faut pour cela utiliser les méthodes de sous-cartes (submapping). Le principe est assez simple : à chaque fois qu'une carte devient trop lourde à compléter (d'autres critères sont également possibles), celle-ci est fermée et une nouvelle entièrement vide est ouverte. Une carte de plus haut niveau garde les liens entre les sous-cartes afin de ne pas perdre d'information. En plus d'alléger les temps de traitement, cela permet aussi de diminuer les erreurs de linéarisation de l'EKF. En effet, dans une nouvelle sous-carte, toutes les informations précédentes sont oubliées ce qui permet d'avoir une incertitude nulle au départ de celle-ci. En revanche, les erreurs de linéarisation perdurent dans la carte haut niveau.

Parmi les premières approches basées sous-cartes apparues, nous pouvons citer [Williams *et al.* 2002a] et le Constrained Relative Submap Filter où les sous-cartes sont décorrélées les unes des autres. Néanmoins, la structure de plus haut niveau en arbre rend l'intégration de fermetures de boucle (second passage à un endroit déjà exploré) difficile. Les approches de Leonard et al. dans [Leonard & Newman 2003] pour le SLAM à temps constant et de Bailey dans [Bailey 2002] avec les Network Coupled Features Maps sont assez similaires. Les amers communs entre les sous-cartes sont utilisés afin de garder les informations disponibles quand il y a une transition dans les sous-cartes. Malheureusement, ces approches ne prennent pas en compte le fait que les données soient corrélées créant ainsi des problèmes d'in-



(a) Reconstruction et trajectoire (les croix rouges sont les amers cartographiés par l'algorithme)



(b) Résultats de localisation (le trait bleu est la vérité terrain et le trait rouge ce qui est estimé par l'algorithme)

FIGURE 2.7 – Résultats de l'EKF de [Féraud 2011] (moins de 8 cm d'erreur)

consistance. Dans l'approche Atlas décrite dans [Bosse *et al.* 2003], un graphe est utilisé comme carte de haut niveau. Les noeuds représentent la position globale des sous-cartes et les arêtes, les transformations de l'une vers l'autre. La cohérence de l'ensemble est malheureusement sacrifiée puisque les mises à jour globales ne sont faites qu'avec un processus hors-ligne. Estrada et al. proposent une approche similaire [Estrada *et al.* 2005] où deux cartes globales sont maintenues : une intégrant les informations corrigeant l'emplacement des sous-cartes et l'autre non. Les corrections ne sont néanmoins pas appliquées aux sous-cartes rendant ainsi impossible de

se servir de ces informations dans la navigation locale. La représentation est également sous-optimale puisqu'un même amer peut être dans plusieurs sous-cartes. Une utilisation de sous-cartes conditionnellement indépendantes est proposée dans [Piniés & Tardós 2008]. Celle-ci permet de conserver les informations d'une sous-carte à l'autre et ainsi d'éviter de démarrer en perdant l'a priori disponible. Cela force cependant à faire attention aux données à marginaliser pour éviter de devenir inconsistant. L'approche décrite dans [Paz *et al.* 2008] propose de reconstituer la carte globale sans approximation. Pour ce faire, l'algorithme joint les sous-cartes deux à deux jusqu'à ce qu'il n'y ait plus qu'une carte globale. Le coût de la jointure est ainsi en partie lissé même si les dernières fusions restent coûteuses. Enfin, quelques séparations légèrement différentes ont été proposées. Dans [Blanco *et al.* 2009], les sous-cartes sont créées en fonction des amers qui sont observables simultanément. Le problème majeur est qu'uniquement les capteurs fournissant immédiatement la distance des amers sont utilisables. Enfin, un critère similaire pour trier les sous-cartes a été présenté dans [Chli & Davison 2009]. Celui-ci groupe ensemble les amers ayant de fortes corrélations. Cela n'empêche pas néanmoins les sous-cartes de ne pas intégrer toutes les corrections disponibles dans la carte haut niveau.

L'utilisation des sous-cartes pose d'autres problèmes que ceux évoqués ci-dessus. En effet, chaque sous-carte est construite avec un facteur d'échelle différent puisque les informations menant d'une sous-carte à une autre ne sont pas utilisées lors de leur construction. Les traitements hors-ligne, souvent obligatoires pour obtenir une carte optimale, sont aussi un point qui nuit à l'emploi de ces méthodes. Cependant, cela reste un bon moyen pour qu'un EKF SLAM cartographie de très longues distances.

Hormis le temps de calcul, il est toujours intéressant de regarder l'empreinte mémoire laissée par un algorithme. Dans le cas de l'EKF (et de ses dérivées), elle est faible car peu d'amers sont nécessaires pour couvrir de grandes distances. Néanmoins, pour atteindre une bonne précision de localisation, il faut veiller à avoir une bonne répartition des amers dans la carte (des points proches et éloignés) [Paz *et al.* 2008]. Dans les résultats présentés en figure 2.7, 319 points ont été utilisés pour une trajectoire de 170 mètres. Uniquement 20 points devaient être observables par image. Ces chiffres traduisent bien, dans ce cas précis, une faible occupation mémoire pour le filtre de Kalman étendu.

2.2.2.3 Filtre de Kalman sans parfum

Le filtre de Kalman sans parfum, ou Unscented Kalman Filter (UKF), a été introduit dès 1997 [Julier & Uhlmann 1997]. Son objectif est de compenser les faibles performances de l'EKF dans le cadre de systèmes hautement non linéaires. L'idée principale est d'éviter le calcul des jacobiennes qui tend à rendre les estimées inconsistantes. Pour cela, une méthode d'échantillonnage est introduite. Celle-ci sélectionne un nombre minimum de particules (appelées points sigma) qui sont ensuite pondérées autour de l'espérance via une fonction de vraisemblance. Ces points sigma passent ensuite par les fonctions non linéaires. Il est alors possible de ré-estimer l'état du système ainsi que sa covariance. Les différentes particules sont ré-échantillonnées

au fur et à mesure afin d'affiner les résultats. Les UKF sont plus précis que les EKF puisque l'étape de linéarisation est évitée.

Néanmoins, un important temps de calcul est requis par cette méthode, la rendant ainsi difficilement transposable à des applications réelles. L'UKF est donc logiquement très peu utilisé pour le SLAM. Les principaux travaux remontant au début des années 2000 [Wan & Merwe 2000][Julier & Uhlmann 2004]. Une application temps réel de l'UKF à un contexte monoculaire a tout de même été démontrée dans [Checklov *et al.* 2006].

2.2.2.4 Filtre d'information

Le filtre d'information [Maybeck 1982] est également souvent appelé la forme inverse du filtre de Kalman. En effet, sa particularité est de définir la matrice d'information comme étant l'inverse de la matrice de covariance :

$$\mathbf{Y}_{k|k} = \mathbf{P}_{k|k}^{-1} \tag{2.7}$$

Cela se traduit comme suit pour la définition du vecteur d'information :

$$\mathbf{y}_{k|k} = \mathbf{P}_{k|k}^{-1}\mathbf{x}_{k|k} \tag{2.8}$$

L'avantage de cette représentation est visible lors de l'étape de mise à jour. Cette dernière devient en effet additive et à l'intérêt de ne pas être dépendante de l'ordre dans lequel les observations sont intégrées [Liu & Thrun 2003]. Les mesures ainsi que les covariances associées sont exprimées sous forme linéaire de la façon suivante :

$$\mathbf{i}_k = \mathbf{H}_k^T\mathbf{R}_k^{-1}\mathbf{z}_k \tag{2.9}$$

$$\mathbf{I}_k = \mathbf{H}_k^T\mathbf{R}_k^{-1}\mathbf{H}_k \tag{2.10}$$

La mise à jour s'écrit alors comme ceci :

$$\mathbf{y}_{k|k} = \mathbf{y}_{k|k-1} + \mathbf{i}_k \tag{2.11}$$

$$\mathbf{Y}_{k|k} = \mathbf{Y}_{k|k-1} + \mathbf{I}_k \tag{2.12}$$

Il est également possible de rendre la matrice d'information plus éparse en cassant les liens faibles entre les données [Thrun *et al.* 2004]. L'avantage majeur est que la mise à jour s'effectue alors quasiment en temps constant [Eustice *et al.* 2005]. Cependant, uniquement quelques approches utilisent ce filtre dans le cadre du SLAM monovéhicule [Mahon *et al.* 2008][Walter *et al.* 2007][Gutmann *et al.* 2010]. Cela est principalement dû au fait qu'il est nécessaire de convertir chaque mesure, tout comme l'état, sous cette forme inverse. Cette étape peut être coûteuse car elle fait intervenir une inversion de matrice de complexité cubique.

2.2.2.5 Filtre particulaire

Les filtres particulaires ou Particle Filters (PF) sont souvent utilisés quand l'état du système que l'on cherche à estimer ne suit pas une distribution gaussienne ou quand le caractère gaussien est cassé par la non linéarité des fonctions. Son principe est simple : l'état est échantillonné en un ensemble de particules suivant sa densité de probabilité. Ensuite, à l'instar du filtre de Kalman, une étape de prédiction est accomplie. Celle-ci applique à chaque particule le modèle d'évolution. De même, une phase de mise à jour suit. Dans celle-ci les particules sont pondérées suivant leur vraisemblance par rapport aux mesures. Les particules les plus probables sont conservées. Les autres sont éliminées et de nouvelles particules sont générées [Dellaert *et al.* 1999].

Cependant, l'application directe de cette méthode au SLAM est délicate. Cela nécessiterait d'avoir un ensemble de particules par amer de la carte, ce qui est, d'un point de vue calculatoire, extrêmement coûteux. Généralement, des hybrides EKF et PF sont utilisés pour le SLAM. L'algorithme le plus connu est le FastSLAM [Montemerlo *et al.* 2002] qui a lui-même été influencé par [Murphy 1999] et [Thrun *et al.* 2000]. Dans celui-ci, la carte est estimée via un EKF pendant que la localisation suit une approche particulaire. La qualité des résultats dépend fortement du nombre de particules utilisées. Ce choix est difficile à quantifier et une augmentation trop conséquente du nombre de particules pose le problème de la complexité calculatoire. Néanmoins, l'algorithme de FastSLAM a pu être appliqué en temps réel à un contexte monoculaire sur de petites distances (intérieur d'un bureau) dans [Eade & Drummond 2006].

Malgré le fait qu'il puisse représenter des distributions non gaussiennes, le FastSLAM souffre aussi d'inconsistance à long terme [Bailey *et al.* 2006b]. Pour éviter ce problème, cet algorithme a été combiné à un filtre d'information dans [Mohan & Madhava 2010]. Le problème majeur porte alors sur les complexités des deux méthodes qui se cumulent.

2.2.3 Bilan des méthodes de localisation

L'aperçu donné sur les méthodes de localisation dans le cadre d'un SLAM local permet de dresser un bilan et de faire un choix en ayant connaissance des forces et faiblesses de chacune. Les plus intéressantes, du point de vue du temps de calcul, semblent être l'ajustement de faisceaux et le filtre de Kalman étendu. Cet aspect est très important puisqu'une utilisation multivéhicule va forcément entraîner un surcoût calculatoire.

Un comparatif entre ces 2 approches a été dressé dans [Strasdat *et al.* 2010a] puis étendu dans [Strasdat *et al.* 2012] dans le cadre de l'utilisation de caméras. Il en ressort que les ajustements de faisceaux semblent plus précis que les EKF. Il est tout de même précisé qu'avec peu de ressources, il est plus intéressant de s'orienter vers un EKF puisqu'une minimisation globale est vite lourde. Un des problèmes de l'EKF reste l'augmentation constante du vecteur d'état mais des méthodes (citées

en 2.2.2.2) peuvent éviter cela. Un point important est le fait que l'EKF soit plus à même de traiter les données hautement incertaines ce qui est souvent le cas en utilisant une seule caméra (la profondeur des points étant plus qu'approximative lors des premières observations).

La revue de l'état de l'art faite dans cette section semble confirmer la tendance évoquée dans ces articles. La quantité d'informations utilisée dans l'ajustement de faisceaux indique que ces approches sont difficiles à adapter à un contexte multivéhicule. De son côté, l'EKF ne requiert que peu de points ce qui le désigne davantage pour des applications impliquant plusieurs véhicules.

Pour privilégier l'économie des ressources (processeur et mémoire), nous nous sommes naturellement tournés vers l'utilisation d'un filtre de Kalman étendu. L'emploi de sa version itérative nous permet en plus d'accomplir les mises à jour plus rapidement même s'il faudra être prudent sur la phase d'association des données. Au delà du temps de calcul et de la consommation mémoire, l'EKF fournit un accès immédiat à la covariance, ce qui est justement important pour l'association de données. Enfin, il offre la possibilité de fusionner d'autres capteurs très facilement. Cela permet de garder notre solution évolutive si d'aventure il s'avérait que l'on souhaitait ajouter un GPS par exemple.

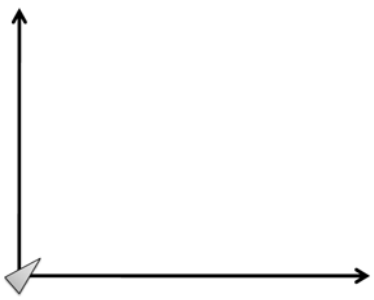
La méthode de localisation ainsi que les capteurs étant décidés, la section suivante va maintenant présenter l'utilisation que nous allons faire de l'ensemble en décrivant les différents modèles utilisés.

2.3 SLAM pour l'EKF dans un contexte de robotique mobile

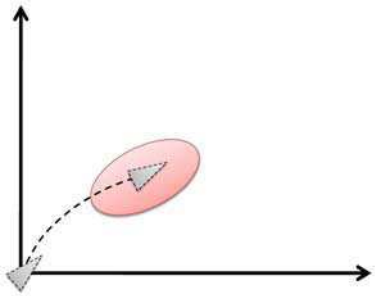
Avant de débuter la description des modèles utilisés, nous allons en premier lieu présenter le SLAM dans le cadre de l'utilisation d'un EKF. Pour ce faire, nous allons nous appuyer sur l'exemple présenté en figure 2.8.

Considérons l'initialisation d'un processus SLAM. Il convient tout d'abord de définir la position du véhicule en fonction d'un repère qui sera ensuite utilisé pour la navigation. Habituellement, et en l'absence d'informations absolues, le véhicule est très souvent initialisé à la position $(0, 0)$ d'un repère situé sur le point de départ de la trajectoire (figure 2.8(a)). En suivant son modèle d'évolution, qui peut être basé ou non sur des informations odométriques, la position du véhicule après déplacement est estimée. Le modèle ainsi que les données issues des capteurs étant imprécises, une incertitude sera affectée à la position via sa matrice de covariance. C'est l'étape de prédiction. Lors de celle-ci, l'incertitude associée à la position du véhicule augmente obligatoirement. La matrice de covariance se traduit graphiquement par une ellipse (quadrique en 3D) dans laquelle la vraie position du véhicule est située (figure 2.8(b)).

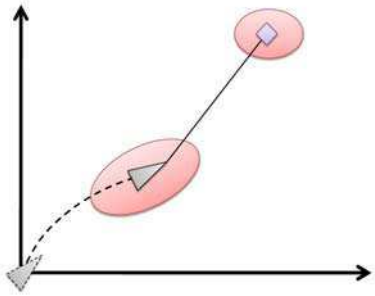
Puisque le véhicule vient seulement de commencer à bouger, il ne dispose pas encore d'une carte par rapport à laquelle il peut se localiser : il doit alors la construire. Pour cela, des points caractéristiques de l'environnement sont sélectionnés grâce à



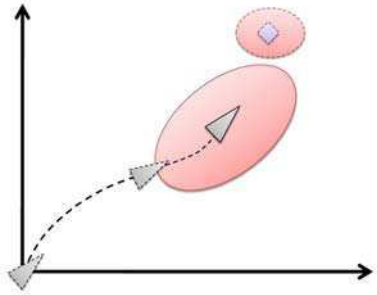
(a) Initialisation du véhicule (en gris)



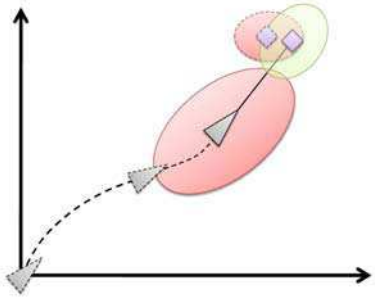
(b) Déplacement du véhicule (incertitude en rouge)



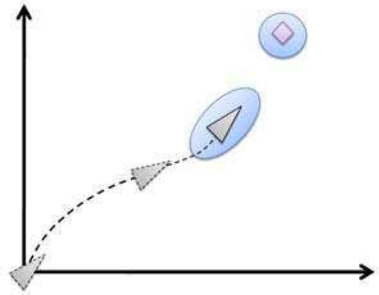
(c) Création d'un amer et de son incertitude (losange gris)



(d) Nouveau déplacement du véhicule



(e) Observation avec son incertitude (verte) du point de la carte



(f) Mise à jour via l'observation et amélioration des positions et incertitudes (bleues)

FIGURE 2.8 – Présentation d'un SLAM dans le contexte de l'EKF

la caméra (ou tout autre capteur extéroceptif). Ils servent de base pour initialiser des amers complets qui seront ensuite intégrés dans la carte. Chaque amer construit prend en compte l'incertitude associée au véhicule au moment de l'initialisation ainsi que l'incertitude quant à la mesure faite par la caméra. Cela permet d'obtenir un point statique de l'environnement avec sa propre incertitude (figure 2.8(c)). Une

fois la cartographie faite, le véhicule continue de se déplacer. L'intégration de son déplacement fait gonfler son incertitude. En revanche, l'amer n'est pas affecté car il est statique et ne dépend donc pas de la position du véhicule (figure 2.8(d)).

Maintenant qu'une carte est constituée (avec un seul amer), celle-ci va être utilisée afin d'améliorer la position du véhicule. Il y a deux façons de faire : la première consiste à utiliser le même détecteur de points caractéristiques que précédemment et d'essayer de trouver des correspondances avec la carte. Il est aussi possible de projeter les points de la carte dans l'espace capteur et de trouver une association à proximité. C'est généralement cette seconde méthode qui est employée. Ici, afin de faciliter la compréhension via les schémas, la première est illustrée. Un nouvel amer est construit avec sa propre incertitude (voir figure 2.8(e)). Grâce à un processus d'association de données, l'algorithme est capable de déterminer que ce nouvel amer correspond bien à celui déjà présent dans sa carte. Ce nouvel amer est alors considéré comme une observation de celui de la carte. Le modèle utilisé pour observer les points de la carte permet ensuite de calculer la mise à jour adéquate. L'utilisation du filtre de Kalman fait que non seulement l'amer sera précisé mais également la position du véhicule (figure 2.8(f)). Ce processus se répète ainsi jusqu'à la fin de la trajectoire.

La suite de cette section va détailler chaque étape en donnant les équations des différents modèles. La première sous-section traitera de la prédiction. Puis, la sous-section 2.3.2 exposera l'étape de mise à jour ainsi que le modèle d'observation utilisé. Enfin, la sous-section 2.3.3 détaillera les différents points sur lesquels l'aspect monoculaire de l'algorithme a une forte influence.

2.3.1 Prédiction

Soit $\mathbf{v}_k$ la pose du véhicule à l'instant k. Le contexte de vision nous permet de la définir suivant 6 degrés de liberté, à savoir la position 3D et les 3 angles associés :

$$\mathbf{v}_k = \begin{pmatrix} x_k \\ y_k \\ z_k \\ \Psi_k \\ \Phi_k \\ \Theta_k \end{pmatrix} \tag{2.13}$$

où $\begin{pmatrix} x_k & y_k & z_k \end{pmatrix}$ est la position 3D du véhicule et $\begin{pmatrix} \Psi_k & \Phi_k & \Theta_k \end{pmatrix}$ sont respectivement les angles de roulis, tangage et lacet.

Ces paramètres nous permettent de passer du repère monde $\Re_m$ au repère véhicule $\Re_v$ et inversement. En l'absence de référence absolue, le $(0, 0, 0)$ du repère monde sera fixé sur le début de la trajectoire. La figure 2.9 montre l'orientation de ces 2 repères. Dans cette figure, $\mathbf{t}_{vm}$ est la translation, issue de la position du véhicule $\begin{pmatrix} x_k & y_k & z_k \end{pmatrix}$, permettant de passer du repère monde au repère véhicule (ou inversement). De manière similaire, $\mathbf{R}_{vm}$ est la rotation, calculée à partir des

angles Ψ_k, Φ_k et Θ_k, qui complète le passage de $\Re_m$ à $\Re_v$ (ou encore une fois dans l'autre sens).

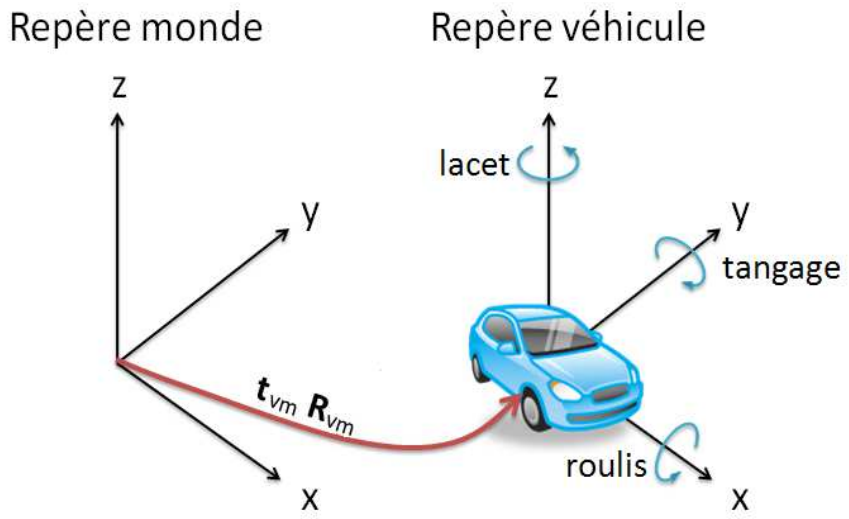


FIGURE 2.9 – Représentation des repères monde et véhicule et passage de l'un à l'autre

Dans le cadre du filtre de Kalman étendu, la prédiction de l'état du véhicule $\mathbf{v}_{k\ 1|k\ 1}$ de covariance $\mathbf{P}_{\mathbf{v}_{k\ 1|k\ 1}}$ s'exprime comme suit :

$$\mathbf{v}_{k|k\ 1} = f(\mathbf{v}_{k\ 1|k\ 1}, \mathbf{u}_k) \tag{2.14}$$

$$\mathbf{P}_{\mathbf{v}_{k|k\ 1}} = \mathbf{F}_{\mathbf{v}_{k\ 1|k\ 1}} \mathbf{P}_{\mathbf{v}_{k\ 1|k\ 1}} \mathbf{F}^{\mathrm{T}}_{\mathbf{v}_{k\ 1|k\ 1}} + \mathbf{F}_{\mathbf{u}_k} \mathbf{Q}_k \mathbf{F}^{\mathrm{T}}_{\mathbf{u}_k} \tag{2.15}$$

où f est une fonction non linéaire, $\mathbf{F}_{\mathbf{v}_{k\ 1|k\ 1}}$ est la jacobienne de f suivant $\mathbf{v}_{k\ 1|k\ 1}$ et $\mathbf{F}_{\mathbf{u}_k}$ la jacobienne de f suivant le vecteur de commande $\mathbf{u}_k$ affecté du bruit $\mathbf{Q}_k$.

Compte tenu du fait que le capteur odométrique nous fournit une distance de déplacement ds entre les instants $k \quad 1$ et k et l'angle volant δ entre les mêmes instants, nous pouvons définir le vecteur de commande $\mathbf{u}_k$ comme indiqué ci-après :

$$\mathbf{u}_k = \begin{pmatrix} ds \\ \delta \end{pmatrix} \tag{2.16}$$

La fonction non linéaire f se traduit par un modèle d'évolution prenant en compte l'état du véhicule $\mathbf{v}_{k\ 1|k\ 1}$ et les données odométriques $\mathbf{u}_k$.

Il s'agit d'un modèle bicyclette simplifié qui s'exprime de la manière suivante :

$$\begin{cases} x_{k|k-1} &= x_{k-1|k-1} + ds\cos(\Theta_{k-1|k-1} + \delta) \\ y_{k|k-1} &= y_{k-1|k-1} + ds\sin(\Theta_{k-1|k-1} + \delta) \\ z_{k|k-1} &= z_{k-1|k-1} + ds\sin(\Phi_{k-1|k-1}) \\ \Psi_{k|k-1} &= \Psi_{k-1|k-1} \\ \Phi_{k|k-1} &= \Phi_{k-1|k-1} \\ \Theta_{k|k-1} &= \Theta_{k-1|k-1} + \dfrac{ds\sin(\delta)}{L} \end{cases} \tag{2.17}$$

où L est l'entre-axe du véhicule.

La jacobienne $\mathbf{F}_{\mathbf{v}_{k-1|k-1}}$ peut ensuite être calculée :

$$\mathbf{F}_{\mathbf{v}_{k-1|k-1}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & -ds\sin(\Theta_{k-1|k-1} + \delta) \\ 0 & 1 & 0 & 0 & 0 & ds\cos(\Theta_{k-1|k-1} + \delta) \\ 0 & 0 & 1 & 0 & ds\cos(\Phi_{k-1|k-1}) & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.18}$$

Il en va de même pour $\mathbf{F}_{\mathbf{u}_k}$:

$$\mathbf{F}_{\mathbf{u}_k} = \begin{bmatrix} \cos(\Theta_{k-1|k-1} + \delta) & -ds\sin(\Theta_{k-1|k-1} + \delta) \\ \sin(\Theta_{k-1|k-1} + \delta) & ds\cos(\Theta_{k-1|k-1} + \delta) \\ \sin(\Phi_{k-1|k-1}) & 0 \\ 0 & 0 \\ 0 & 0 \\ \dfrac{\sin(\delta)}{L} & \dfrac{ds\cos(\delta)}{L} \end{bmatrix} \tag{2.19}$$

La matrice $\mathbf{Q}_k$ est généralement définie expérimentalement en fonction de la qualité des données odométriques.

Cette phase de prédiction n'est pas obligatoirement suivie par une étape de mise à jour. En effet, les capteurs fonctionnent à des fréquences différentes et il est courant que l'odomètre soit plus rapide que la caméra. Bien évidemment, rien n'empêche de faire plusieurs prédictions avant de faire une mise à jour. Seulement, il n'y a aucune garantie de l'ordre d'arrivée des mesures. Ainsi, dès qu'une image arrive, l'état du véhicule est extrapolé jusqu'à la date de l'image avec un modèle qui n'est pas basé sur l'odométrie (modèle à vitesse constante, etc.). Les états issus de l'odométrie et de la caméra sont ensuite fusionnés dès qu'une nouvelle donnée proprioceptive est disponible. Ce comportement est exposé dans la figure 2.10 au travers d'un exemple. Les lecteurs intéressés pourront se tourner vers [Tessier *et al.* 2006] pour plus de détails sur la gestion des données désynchronisées.

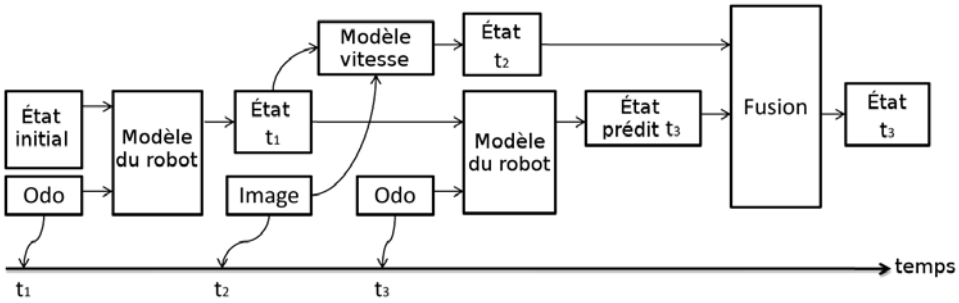


FIGURE 2.10 – Gestion des deux états en fonction des données fournies par les capteurs

Sur la figure 2.10, on peut voir qu'une donnée odométrique est reçue à la date t_1. Celle-ci permet, à l'aide du modèle du robot d'obtenir l'état t_1. Une image arrive à une date t_2 ultérieure à t_1. Il est donc nécessaire d'extrapoler l'état t_1 jusqu'à la date t_2 afin de pouvoir prendre en compte cette information. Pour ce faire, un modèle à vitesse constante est utilisé, ce qui génère l'état t_2. La réception d'une nouvelle donnée odométrique à t_3 va créer l'état prédit t_3 parallèlement à t_2. Ceux-ci sont ensuite fusionnés pour obtenir l'état t_3 qui intègre toutes les informations reçues jusqu'à la date t_3.

2.3.2 Mise à jour

La gestion des mises à jour est un peu plus compliquée puisqu'elle fait intervenir la carte. Les observations se faisant dans le repère image, il convient de bien définir les différents changements de repère qui peuvent intervenir. En plus des repères monde et véhicule préalablement définis, le repère caméra $\Re_c$ ainsi que le plan image $\Re_i$ doivent être pris en considération pour pouvoir passer d'amers 3D dans le monde à un point 2D dans le plan image.

Le passage de $\Re_v$ à $\Re_c$ est similaire à celui défini pour aller de $\Re_m$ à $\Re_v$. Une translation $\mathbf{t}_{cv}$ ainsi qu'une rotation $\mathbf{R}_{cv}$ sont nécessaires à sa réalisation. Les 6 paramètres de ce changement de repère sont donnés par les paramètres extrinsèques de la caméra qui correspondent à la mesure de la position de la caméra par rapport au repère véhicule. Considérons un point 3D $\mathbf{p}_m$ exprimé dans le repère monde. Son équivalent $\mathbf{p}_c$ dans $\Re_c$ peut être calculé de la façon suivante :

$$\mathbf{p}_c = \mathbf{R}_{cv}^{\mathrm{T}}(\mathbf{R}_{vm}^{\mathrm{T}}(\mathbf{p}_m \quad \mathbf{t}_{vm}) \quad \mathbf{t}_{cv}) \tag{2.20}$$

$\mathbf{R}_{vm}$ et $\mathbf{t}_{vm}$ sont calculés à partir de l'état du véhicule $\mathbf{v}_k$ comme suit :

$$\mathbf{t}_{vm} = \begin{bmatrix} x_k \\ y_k \\ z_k \end{bmatrix} \tag{2.21}$$

$$\begin{aligned}\mathbf{R}_{vm} &= \mathbf{R}_z(\Theta_k)\mathbf{R}_y(\Phi_k)\mathbf{R}_x(\Psi_k) \\ &= \begin{bmatrix}\cos(\Theta_k) & -\sin(\Theta_k) & 0\\ \sin(\Theta_k) & \cos(\Theta_k) & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\cos(\Phi_k) & 0 & \sin(\Phi_k)\\ 0 & 1 & 0\\ -\sin(\Phi_k & 0 & \cos(\Phi_k)\end{bmatrix}\begin{bmatrix}1 & 0 & 0\\ 0 & \cos(\Psi_k) & -\sin(\Psi_k)\\ 0 & \sin(\Psi_k) & \cos(\Psi_k)\end{bmatrix}\end{aligned} \tag{2.22}$$

Le calcul est similaire pour $\mathbf{R}_{cv}$ et $\mathbf{t}_{cv}$ mais à partir des paramètres extrinsèques mesurés.

Le passage de $\Re_c$ à $\Re_i$ est différent puisqu'il fait passer d'un point 3D à son équivalent 2D dans l'image. Pour effectuer cette opération, un modèle de la caméra est nécessaire. Le modèle sténopé est généralement utilisé de par sa simplicité. Celui-ci ne nécessite que quelques paramètres concernant la caméra qui peuvent être déterminés par un calibrage. Il s'agit des paramètres intrinsèques de la caméra. Ils peuvent être organisés sous forme matricielle afin de faciliter le changement de repère :

$$\mathbf{F} = \begin{bmatrix}c_u & f_u & 0\\ c_v & 0 & f_v\\ 1 & 0 & 0\end{bmatrix} \tag{2.23}$$

où $\begin{pmatrix}c_u & c_v\end{pmatrix}$ est le point principal de l'image (projection du centre optique) et $\begin{pmatrix}f_u & f_v\end{pmatrix}$ correspond à la distance focale suivant les 2 axes.

Le modèle sténopé nous donne ainsi la projection d'un point $\begin{pmatrix}x & y & z\end{pmatrix}^{\mathrm{T}}$ dans l'image :

$$\begin{bmatrix}su\\ sv\\ s\end{bmatrix} = \mathbf{F}\begin{bmatrix}x\\ y\\ z\end{bmatrix} \tag{2.24}$$

Cela nous donne le point $\begin{pmatrix}u & v\end{pmatrix}^{\mathrm{T}}$ dans l'image :

$$\begin{cases} u = \dfrac{c_u x + f_u y}{x} \\ \\ v = \dfrac{c_v x + f_v z}{x} \end{cases} \tag{2.25}$$

Nous pouvons maintenant compléter la figure 2.9 avec les nouveaux repères. Le résultat est visible en figure 2.11.

Les étapes présentées ici sont utilisées afin de projeter un point de la carte depuis le repère monde vers le repère image. Il s'agit de l'étape de prédiction des observations. Comme nous nous sommes placés dans le cadre d'un EKF itératif, nous définissions h comme la fonction non linéaire permettant de projeter un amer $\mathbf{a}_{i_{k-1}}$ depuis le repère monde dans le plan image. Il est ainsi possible de définir une observation $\mathbf{z}_k$ comme étant le point de l'image correspondant à la prédiction de $\mathbf{a}_{i_{k-1}}$ dans $\Re_i$. Ainsi, du point de vue de l'EKF, le calcul de l'innovation se traduit de la façon suivante :

$$\boldsymbol{\Delta}_k = \mathbf{z}_k - h(\mathbf{v}_{k|k-1}, \mathbf{a}_{i_{k-1}}) \tag{2.26}$$

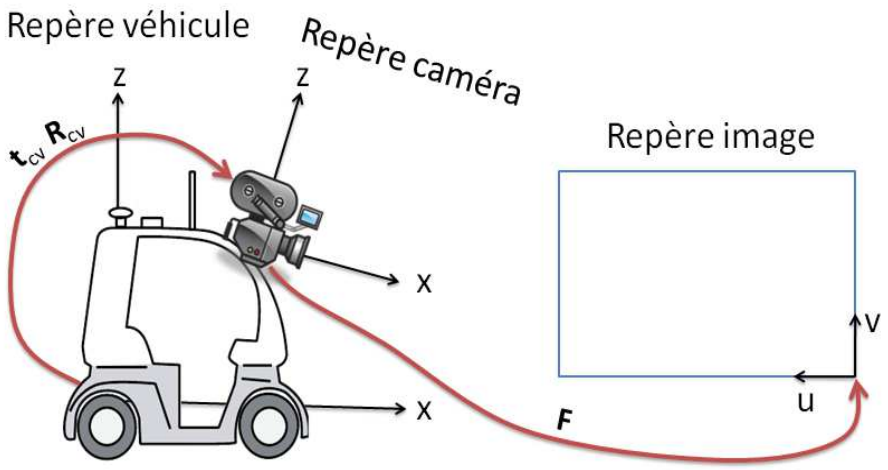


FIGURE 2.11 – Représentation des repères caméra et image et passage de l'un à l'autre

La fonction h correspond en fait à l'enchaînement des équations (2.20) et (2.25) pour l'amer $\mathbf{a}_{i_{k\ 1}}$ défini dans le repère $\Re_m$. Elle se traduit par le système suivant :

$$\begin{cases} u_{i_{k\ 1}} = \dfrac{\mathbf{F}_1 \mathbf{R}_{cv}^{\mathrm{T}}(\mathbf{R}_{vm}^{\mathrm{T}}(\mathbf{p}_m \quad \mathbf{t}_{vm}) \quad \mathbf{t}_{cv})}{\mathbf{F}_3 \mathbf{R}_{cv}^{\mathrm{T}}(\mathbf{R}_{vm}^{\mathrm{T}}(\mathbf{p}_m \quad \mathbf{t}_{vm}) \quad \mathbf{t}_{cv})} \\ v_{i_{k\ 1}} = \dfrac{\mathbf{F}_2 \mathbf{R}_{cv}^{\mathrm{T}}(\mathbf{R}_{vm}^{\mathrm{T}}(\mathbf{p}_m \quad \mathbf{t}_{vm}) \quad \mathbf{t}_{cv})}{\mathbf{F}_3 \mathbf{R}_{cv}^{\mathrm{T}}(\mathbf{R}_{vm}^{\mathrm{T}}(\mathbf{p}_m \quad \mathbf{t}_{vm}) \quad \mathbf{t}_{cv})} \end{cases} \tag{2.27}$$

où $\mathbf{F}_i$ représente la $i^{\text{ème}}$ ligne de la matrice de paramètres intrinsèques $\mathbf{F}$.

Nous pouvons maintenant calculer le gain de Kalman de manière similaire à l'équation (2.4). Ici, $\mathbf{R}_k$ caractérise le bruit dans l'image (matrice de taille 2×2) et $\mathbf{H}_k$ représente le calcul de la jacobienne de h suivant l'état $\mathbf{x}_{k|k\ 1}$ qui est ici limité à la pose du véhicule $\mathbf{v}_{k|k\ 1}$ et l'amer qui est couramment traité $\mathbf{a}_{i_{k\ 1}}$. Cela signifie que le reste des amers de la carte n'intervient pas dans h et par conséquent :

$$\frac{\partial h(\mathbf{v}_{k|k\ 1}, \mathbf{a}_{i_{k\ 1}})}{\partial \mathbf{a}_{j \neq i_{k\ 1}}} = 0 \tag{2.28}$$

Ainsi, la jacobienne $\mathbf{H}_k$ sera nulle, hormis sur les lignes correspondant à la pose du véhicule et à l'amer observé. Pour une carte comportant n amers numérotés de 0 à $n \quad 1$, $\mathbf{H}_k$ est de la forme suivante :

$$\mathbf{H}_k = \left[\frac{\partial h}{\partial \mathbf{v}_{k|k\ 1}} \quad \mathbf{0}_{2\times 3i} \quad \frac{\partial h}{\partial \mathbf{a}_{i_{k\ 1}}} \quad \mathbf{0}_{2\times 3n\ \ 3\ \ 3i} \right] \tag{2.29}$$

Les dérivées suivant les positions sont assez simples à calculer dès lors que l'on pose les bonnes variables. En partant de l'équation (2.27), on peut poser :

$$\begin{aligned} num_u &= \mathbf{F}_1\mathbf{R}_{cv}^{\mathrm{T}}(\mathbf{R}_{vm}^{\mathrm{T}}(\mathbf{p}_m - \mathbf{t}_{vm}) - \mathbf{t}_{cv}) \\ num_v &= \mathbf{F}_2\mathbf{R}_{cv}^{\mathrm{T}}(\mathbf{R}_{vm}^{\mathrm{T}}(\mathbf{p}_m - \mathbf{t}_{vm}) - \mathbf{t}_{cv}) \\ den &= \mathbf{F}_3\mathbf{R}_{cv}^{\mathrm{T}}(\mathbf{R}_{vm}^{\mathrm{T}}(\mathbf{p}_m - \mathbf{t}_{vm}) - \mathbf{t}_{cv}) \end{aligned} \tag{2.30}$$

Pour l'amer $\mathbf{a}_{i_{k-1}}$, on obtient :

$$\begin{aligned} \frac{\partial u}{\partial a_{x_{i_{k-1}}}} &= K_u\mathbf{R}_{cv}^{\mathrm{T}}\mathbf{R}_{vm}^{\mathrm{T}}\begin{pmatrix}1 & 0 & 0\end{pmatrix}^{\mathrm{T}} \\ \frac{\partial v}{\partial a_{x_{i_{k-1}}}} &= K_v\mathbf{R}_{cv}^{\mathrm{T}}\mathbf{R}_{vm}^{\mathrm{T}}\begin{pmatrix}1 & 0 & 0\end{pmatrix}^{\mathrm{T}} \\ \frac{\partial u}{\partial a_{y_{i_{k-1}}}} &= K_u\mathbf{R}_{cv}^{\mathrm{T}}\mathbf{R}_{vm}^{\mathrm{T}}\begin{pmatrix}0 & 1 & 0\end{pmatrix}^{\mathrm{T}} \\ \frac{\partial v}{\partial a_{y_{i_{k-1}}}} &= K_v\mathbf{R}_{cv}^{\mathrm{T}}\mathbf{R}_{vm}^{\mathrm{T}}\begin{pmatrix}0 & 1 & 0\end{pmatrix}^{\mathrm{T}} \\ \frac{\partial u}{\partial a_{z_{i_{k-1}}}} &= K_u\mathbf{R}_{cv}^{\mathrm{T}}\mathbf{R}_{vm}^{\mathrm{T}}\begin{pmatrix}0 & 0 & 1\end{pmatrix}^{\mathrm{T}} \\ \frac{\partial v}{\partial a_{z_{i_{k-1}}}} &= K_v\mathbf{R}_{cv}^{\mathrm{T}}\mathbf{R}_{vm}^{\mathrm{T}}\begin{pmatrix}0 & 0 & 1\end{pmatrix}^{\mathrm{T}} \end{aligned} \tag{2.31}$$

avec les variables K_u et K_v définies de la manière suivante :

$$\begin{aligned} K_u &= \frac{den\mathbf{F}_1 - num_u\mathbf{F}_3}{den^2} \\ K_v &= \frac{den\mathbf{F}_2 - num_v\mathbf{F}_3}{den^2} \end{aligned} \tag{2.32}$$

Le calcul des dérivées vis-à-vis de la position 3D du véhicule est très proche de ce qui a été présenté dans l'équation (2.31). En effet, la position du véhicule intervient au même niveau que la position du point.

En conséquence, nous pouvons en déduire :

$$\begin{aligned}\frac{\partial h}{\partial x_{k|k-1}} &= -\frac{\partial h}{\partial a_{x_{k-1}}} \\ \frac{\partial h}{\partial y_{k|k-1}} &= -\frac{\partial h}{\partial a_{y_{k-1}}} \\ \frac{\partial h}{\partial z_{k|k-1}} &= -\frac{\partial h}{\partial a_{z_{k-1}}}\end{aligned} \tag{2.33}$$

Le calcul des dérivées suivant l'orientation du véhicule donne une forme similaire :

$$\begin{aligned}\frac{\partial u}{\partial \Psi_{k|k-1}} &= K_u \mathbf{R}_{cv}^{\mathrm{T}} \frac{\partial \mathbf{R}_{vm}^{\mathrm{T}}}{\partial \Psi_{k|k-1}} (\mathbf{a}_{i_{k-1}} - \mathbf{t}_{vm}) \\ \frac{\partial v}{\partial \Psi_{k|k-1}} &= K_v \mathbf{R}_{cv}^{\mathrm{T}} \frac{\partial \mathbf{R}_{vm}^{\mathrm{T}}}{\partial \Psi_{k|k-1}} (\mathbf{a}_{i_{k-1}} - \mathbf{t}_{vm})\end{aligned} \tag{2.34}$$

Il est possible de développer davantage le terme à dériver :

$$\begin{aligned}\frac{\partial \mathbf{R}_{vm}^{\mathrm{T}}}{\partial \Psi_{k|k-1}} &= \left(\frac{\partial \mathbf{R}_{vm}}{\partial \Psi_{k|k-1}} \right)^{\mathrm{T}} \\ &= \left(\mathbf{R}_z(\Theta_{k|k-1}) \mathbf{R}_y(\Phi_{k|k-1}) \frac{\partial \mathbf{R}_x(\Psi_{k|k-1})}{\partial \Psi_{k|k-1}} \right)^{\mathrm{T}}\end{aligned} \tag{2.35}$$

La dérivée de la matrice de rotation étant simple, elle ne sera pas détaillée ici. Les équations (2.34) et (2.35) peuvent être directement adaptées aux angles $\Phi_{k|k-1}$ et $\Theta_{k|k-1}$ sans aucun problème. Une fois le gain de Kalman calculé à partir de $\mathbf{H}_k$, la mise à jour de l'état et de la covariance suit les mêmes équations que pour la forme linéaire du Kalman (voir équations (2.5) et (2.6)).

Les modèles d'évolution et d'observation ont maintenant été présentés. Ils ont à chaque fois été introduit en fonction des données capteurs attendues. Néanmoins, certaines spécificités liées à l'utilisation d'une caméra ont été volontairement omises. Elles feront l'objet de la prochaine sous-section.

2.3.3 Spécificité de la vision

L'initialisation des amers est la première étape de la constitution d'une carte. Il faut pour cela être capable de détecter des points dans l'image qui soient pertinents. En effet, ces points vont être suivis dans de nombreuses images. Pour cela, il faut s'assurer qu'ils soient suffisamment distincts pour pouvoir les reconnaître sans faire d'erreur. De plus, il faut que cette étape soit rapide afin de ne pas pénaliser l'algorithme de SLAM.

Les points distinctifs d'une image correspondent à des endroits marquant un fort changement par rapport au voisinage. Il s'agit souvent de coins d'objets de l'environnement (coins d'immeubles, de lignes blanches ou de fenêtres par exemple en contexte urbain). Les extracteurs de coins sont de loin les plus connus et les plus employés. Les plus célèbres sont le détecteur d'Harris [Harris & Stephens 1988] et sa variante Shi-Tomasi [Shi & Tomasi 1994]. Ils sont rapides et fortement parallélisables. De plus, l'utilisation du gradient de l'image pour le calcul des points caractéristiques rend leur sélection robuste aux changements d'illumination dans l'image. Il a d'ailleurs été montré dans [Schmid *et al.* 2000] que le détecteur d'Harris est l'un des plus performants de sa catégorie.

Malheureusement, sous cette forme, ces détecteurs ne sont pas robustes au changement de point de vue ou d'échelle. En revanche, les détecteurs de type Difference of Gaussians (DoG) [Lowe 2004] ou Laplacian of Gaussian (LoG) [Lindeberg 1998] le sont. Ils souffrent néanmoins d'autres défauts comme une sensibilité aux changements de luminosité.

Bien qu'il existe une version de Harris capable de gérer différentes échelles [Mikolajczyk & Schmid 2004], cela induit un surcoût calculatoire qui n'est pas négligeable. Il est important de garder à l'esprit que les résultats d'un détecteur sont fortement liés au contexte (conditions climatiques par exemple) ainsi qu'à l'environnement [Mikolajczyk *et al.* 2005]. Il est donc difficile de choisir le meilleur détecteur, il s'agira le plus souvent d'un compromis entre le coût de calcul et la robustesse de celui-ci.

Parmi les approches robustes, nous pouvons citer celles qui intègrent à la fois la détection et l'extraction (algorithme d'association) de caractéristiques. Les plus célèbres sont le SIFT [Lowe 1999] et le SURF [Bay *et al.* 2008]. Ceux-ci sont néanmoins coûteux et sont donc difficiles à intégrer dans des approches temps réel. Cependant, les résultats obtenus sont impressionnants comme cela a été montré pour le SIFT dans [Mikolajczyk & Schmid 2005].

Il est bon de rappeler que l'utilisation d'un détecteur dans notre application sert uniquement à initialiser les points. Ceux-ci sont ensuite suivis grâce à leur prédiction (comme vu en sous-section 2.3.2) ainsi qu'un opérateur de corrélation qui sera présenté plus loin dans cette sous-section. Nous nous sommes donc orientés vers un détecteur de Harris (ou de Shi-Tomasi parfois mais avec des résultats similaires) car il donne des points suffisamment robustes pour le suivi.

Un travail a tout de même été fait sur le détecteur afin de l'adapter à notre contexte. En effet, en se contentant de ne garder que les n meilleurs points fournis par le détecteur de Harris, il est plus que probable qu'ils soient quasiment tous groupés autour du même objet. Or, pour que l'estimation de la pose du véhicule via l'algorithme de SLAM soit correcte, il est important d'avoir des points proches et des points éloignés. Les amers proches aident à estimer la translation alors que ceux situés loin fixent davantage l'orientation du robot [Paz *et al.* 2008]. Une bonne répartition des amers s'impose donc. Pour ce faire, et également afin de paralléliser le traitement, l'image a été découpée en 4 zones. Dans chacune d'entre elles, les meilleurs points de Harris sont gardés (supérieurs à un seuil). De plus, lorsqu'un

point est sélectionné, il empêche tous ceux situés dans son voisinage proche d'être conservés. Cela permet de garder une certaine diversité dans la position des points. Enfin, lors de l'initialisation de nouveaux amers, une priorité sera donnée aux parties de l'image où le nombre de points suivis est le plus faible.

Affiner la position des amers nécessite de pouvoir les observer. Pour ce faire, la projection de l'amer dans l'image (étape décrite dans la sous-section 2.3.2) donne une première idée d'où chercher. Généralement, un descripteur est extrait autour du point 2D lors de son initialisation. Il s'agit d'une zone de pixels centrée sur le point. Ensuite, lors du suivi, une fenêtre rectangulaire autour de la projection de l'amer est définie afin de donner un espace de recherche. La correspondance entre le descripteur extrait à l'initialisation et la fenêtre de recherche se fait par corrélation normalisée ou Zero mean Normalized Cross-Correlation (ZNCC) [Lewis 1995]. Cette mesure de similarité va tester le descripteur sur chaque pixel de la fenêtre. Le meilleur score obtenu sur la zone de recherche est ensuite comparé à un seuil correspondant au minimum de ressemblance souhaité pour valider une association. Si la valeur obtenue est bien supérieure au seuil, alors le point est sélectionné comme observation et utilisé pour la mise à jour. Sinon, l'amer est conservé tant qu'il n'a pas été inobservable pendant plusieurs images consécutives (une occultation par un autre objet est possible) et son suivi recommence sur l'image suivante.

La combinaison d'un détecteur de Harris avec un opérateur de ZNCC est assez courante. Elle a été employée avec succès dans les travaux de Davison et al. [Davison 2003][Davison *et al.* 2007] mais également dans ceux de Clemente et al. [Clemente *et al.* 2007].

Cette section a présenté les différentes étapes intervenant dans la réalisation d'un algorithme de SLAM par un filtre de Kalman étendu. L'ensemble des modèles a été développé dans le cadre d'une application par vision. Néanmoins, l'approche telle que décrite jusqu'ici n'est pas viable. De nombreuses erreurs de linéarisation affectent le processus de localisation et le font rapidement diverger. Le chapitre suivant abordera cette problématique en l'illustrant par des exemples. Ensuite, une étude de la littérature ainsi que des solutions seront proposées.

Chapitre 3

Gestion de la non linéarité dans le SLAM monoculaire

Sommaire

La solution de localisation proposée dans le chapitre précédent est en l'état difficilement utilisable. Bien que conçue pour des systèmes non linéaires, celle-ci nécessite des aménagements afin de pouvoir fournir des résultats de localisation convaincants lorsqu'une seule caméra est employée. En effet, en l'absence d'une information précise quant à la profondeur des points cartographiés, l'algorithme peut rapidement diverger. Il est toujours possible d'utiliser les solutions mentionnées précédemment (UKF, filtre particulaire) mais l'EKF offre des avantages intéressants (intégration facile des capteurs, quantité de données utilisées, ...). Nous avons ainsi choisi d'analyser le filtre de Kalman étendu dans le contexte de la localisation monoculaire.

La section 3.1 fera le point sur les erreurs de linéarisation pouvant affecter notre solution. Celles-ci seront, dans un premier temps, illustrées via des exemples dans la sous-section 3.1.1, puis une étude de la littérature avec les solutions généralement employées sera fournie (sous-section 3.1.2). La section 3.2 détaillera la solution développée pour pallier les problèmes relevés et des analyses seront données pour chaque point amélioré. La section 3.3 présentera les simulations et expérimentations réalisées pour démontrer l'efficacité de notre solution. Une discussion générale en section 3.4 conclura sur l'approche développée.

3.1 Erreurs de linéarisation

Les erreurs de linéarisation pouvant affecter le processus d'estimation interviennent à différents niveaux. L'utilisation des méthodes habituelles peuvent les favoriser dans le contexte monoculaire de notre application. Cette section a pour vocation, dans un premier temps, de montrer les différents moments où la linéarisation peut échouer et les conséquences sur le système (sous-section 3.1.1). La sous-section 3.1.2 fournira une étude de la littérature sur des palliatifs spécifiquement conçus pour le SLAM monoculaire. Comme pour les sections précédentes, les avantages et inconvénients de chaque correctif seront analysés.

3.1.1 Constatation du problème

L'étape de linéarisation, qui intervient dans le filtre de Kalman dès lors que des modèles non linéaires sont impliqués, peut facilement échouer si le point de linéarisation choisi est éloigné de la vraie valeur que l'on cherche à estimer. Dans le cas d'un SLAM monoculaire, cette situation est omniprésente puisque la distance des points n'est pas immédiatement disponible. Ainsi, la méthode choisie pour initialiser les points 3D à partir de données 2D de l'image va fortement conditionner le bon déroulement des mises à jour qui vont suivre. Au-delà du point lui-même, la création de son incertitude 3D de manière cohérente avec celle définie dans l'image est aussi un facteur important pour la bonne réalisation d'un processus EKF SLAM. Il est ainsi possible d'identifier trois aspects majeurs qui peuvent directement ou indirectement engendrer des erreurs de linéarisation : l'initialisation du point 3D et de son incertitude à partir d'une information 2D, le suivi des points dans l'image qui dépend de la linéarisation de la fonction permettant de projeter les points depuis le

monde vers le plan image et enfin la mise à jour des points qui s'appuie elle aussi sur une étape de linéarisation de la fonction d'observation. Ces trois aspects vont maintenant être détaillés afin d'offrir une vue claire de l'implication de chacun sur la bonne ou mauvaise réussite du processus de SLAM.

3.1.1.1 Initialisation d'un amer à distance inconnue

Considérons une fonction h_{ci} qui, à partir d'un point 3D $\mathbf{p}_c$ dans le repère caméra $\Re_c$, permet d'obtenir un point $\mathbf{p}_i$ dans le plan image $\Re_i$. L'initialisation de l'amer suit l'ordre inverse de ce modèle. Le but est de remonter, à partir d'un point caractéristique et du bruit l'affublant dans l'image, une position 3D ainsi que l'incertitude associée. Le point caractéristique est défini ainsi : $\mathbf{p}_i = \begin{pmatrix} u & v \end{pmatrix}^{\mathrm{T}}$ avec son bruit d'observation $\mathbf{P}_{\mathbf{p}_i}$ qui se traduit par une matrice 2×2 supposée diagonale composée de σ_u^2 et σ_v^2, respectivement les variances suivant les axes $\vec{u}$ et $\vec{v}$ de l'image. L'initialisation devant se faire sur le rayon de vue du point caractéristique, il est alors possible de fixer une profondeur initiale. Ainsi, une distance fixe x_d est utilisée pour contraindre l'initialisation. À partir du modèle sténopé présenté en sous-section 2.3.2, il est possible de déduire la position initiale de $\mathbf{p}_c = \begin{pmatrix} x & y & z \end{pmatrix}^{\mathrm{T}}$:

$$\begin{cases} x = x_d \\ y = \dfrac{ux - c_u x}{f_u} \\ z = \dfrac{vx - c_v x}{f_v} \end{cases} \tag{3.1}$$

Le choix de x_d peut se faire en fonction de l'environnement dans lequel est destiné à évoluer le véhicule. Dans un environnement cloisonné, une faible valeur pour x_d sera mieux indiquée qu'une distance importante, qui elle sera plus favorable à des milieux ouverts. Dans tous les cas, cela signifie que le point 3D initialisé sera certainement éloigné de sa position réelle, induisant ainsi des erreurs de linéarisation.

Si l'initialisation du point 3D est simple, convertir une incertitude 2D en 3D est plus délicat. La méthode couramment employée consiste à laisser le filtre de Kalman s'en charger. L'incertitude du point 3D est initialisée avec des valeurs quasi-infinies et une mise à jour de la covariance est réalisée via le filtre de Kalman en considérant comme bruit d'observation celui associé à la mesure dans l'espace capteur. Afin d'illustrer la covariance construite, un exemple d'initialisation a été conduit avec un bruit dans l'image de 4 pixels. Le point est initialisé au centre d'une image fictive large de 480 pixels. Une caméra fictive, avec une focale de 720 pixels, est ensuite utilisée pour construire l'ellipsoïde associé à l'incertitude dans l'image. Le résultat est visible en figure 3.1. Le point 2D mène à un point 3D situé à 500 mètres sur l'axe des $\vec{x}$ (distance d'initialisation) et 0 sur $\vec{y}$ (croix rouge). Le vrai point (étoile verte) est situé à 20 mètres devant la caméra (triangle noir). La figure 3.1 présente une vue en coupe (suivant le plan Oxy) de l'ellipsoïde généré.

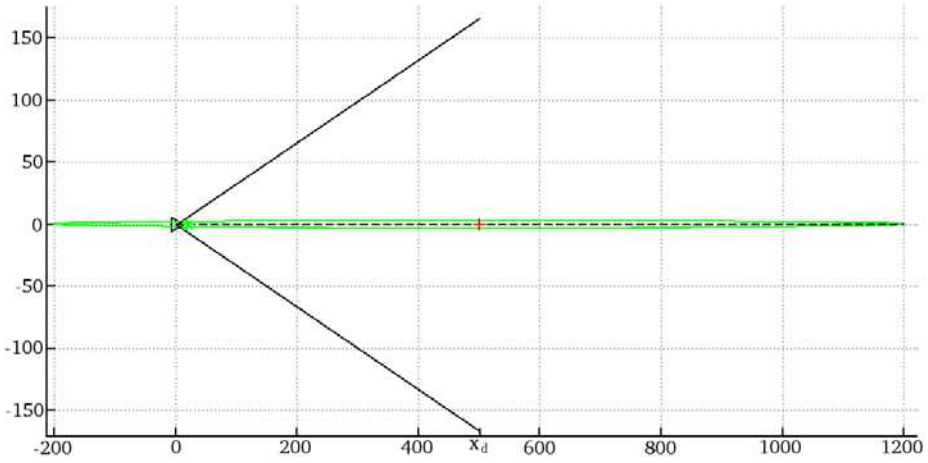


FIGURE 3.1 – Initialisation de l'incertitude par une mise à jour de l'EKF. Le point initialisé est en rouge. Le vrai point est l'étoile verte. Le triangle noir représente la caméra et le cône son champ de vue. La ligne pointillée noire est la ligne de vue sur laquelle le point a été observé. L'ellipse verte est la représentation graphique à 1σ de l'incertitude créée. Ici, le point est initialisé à 500 mètres. L'incertitude dans l'image σ_u est fixée à 4 pixels et la focale f_u vaut 720 pixels.

On peut constater que la vraie position du point est bien englobée par l'incertitude construite. Celle-ci n'est donc pas erronée. Elle ne tient cependant pas compte de l'ensemble des informations à disposition. En effet, le point vient juste d'être observé, cela signifie donc qu'il est situé face à la caméra. Ainsi, la covariance ne devrait pas inclure des possibilités derrière la caméra. On s'expose à ce qu'un point tout juste initialisé puisse être mis à jour derrière la caméra. Si cela arrive, celui-ci ne serait plus ré-observé puisque considéré comme derrière le véhicule. Ce comportement est entièrement lié aux valeurs initiales données à la covariance du point avant la mise à jour. Cette dernière permet de contraindre suivant l'axe $\vec{y}$ mais pas suivant $\vec{x}$, nécessitant ainsi de correctement fixer l'incertitude suivant cet axe avant la mise à jour (dans cet exemple, $\sigma_x = 700$).

L'autre possibilité pour générer l'incertitude 3D consiste à utiliser les jacobiennes directement à la création du point. Pour ce faire, nous cherchons à trouver l'incertitude du point 3D dans $\Re_c$ afin que sa projection via les jacobiennes soit cohérente avec le bruit d'observation défini dans l'image. Pour ce faire, nous nous appuyons sur la fonction h_{ci} définie précédemment. L'incertitude 2D $\mathbf{P}_{\mathbf{p}_i}$ du point $\mathbf{p}_i$ se calcule ainsi de la façon suivante à partir de l'incertitude $\mathbf{P}_{\mathbf{p}_c}$ du point 3D $\mathbf{p}_c$ exprimé dans $\Re_c$:

$$\mathbf{P}_{\mathbf{p}_i} = \mathbf{H}_{ci}\mathbf{P}_{\mathbf{p}_c}\mathbf{H}_{ci}^{\mathrm{T}} \tag{3.2}$$

où $\mathbf{H}_{ci}$ est la jacobienne associée à la fonction h_{ci}.

À partir de l'équation (3.1), il est possible de calculer la jacobienne $\mathbf{H}_{ci}$ ainsi :

$$\mathbf{H}_{ci} = \begin{pmatrix} -\frac{f_u y}{x^2} & \frac{f_u}{x} & 0 \\ -\frac{f_v z}{x^2} & 0 & \frac{f_v}{x} \end{pmatrix} \tag{3.3}$$

On peut ensuite déduire l'expression littérale de la covariance $\mathbf{P}_{\mathbf{p}_i}$:

$$\mathbf{P}_{\mathbf{p}_i} = \begin{pmatrix} \frac{\sigma_x^2 f_u^2 y^2}{x^4} + \frac{\sigma_y^2 f_u^2}{x^2} & \frac{\sigma_x^2 f_u f_v y z}{x^4} \\ \frac{\sigma_x^2 f_u f_v y z}{x^4} & \frac{\sigma_x^2 f_v^2 z^2}{x^4} + \frac{\sigma_z^2 f_v^2}{x^2} \end{pmatrix} \tag{3.4}$$

où σ_x, σ_y et σ_z sont les écarts types du point 3D dans $\Re_c$ suivant les axes $\vec{x}$, $\vec{y}$ et $\vec{z}$.

L'incertitude doit être calculée suivant l'axe $\vec{x}$ afin de réduire le nombre de variables du système à résoudre. Celle-ci sera ensuite tournée afin d'être orientée sur le rayon de vue de l'observation initiale. Ainsi, nous calculons la projection du point défini par l'équation (3.1) sur l'axe $\vec{x}$:

$$\begin{pmatrix} \rho & 0 & 0 \end{pmatrix}^{\mathrm{T}} = \begin{pmatrix} \sqrt{x^2 + y^2 + z^2} & 0 & 0 \end{pmatrix}^{\mathrm{T}} \tag{3.5}$$

Nous obtenons la matrice de covariance $\mathbf{P}_{\mathbf{p}_i}$ suivante :

$$\mathbf{P}_{\mathbf{p}_i} = \begin{pmatrix} \frac{\sigma_y^2 f_u^2}{\rho^2} & 0 \\ 0 & \frac{\sigma_z^2 f_v^2}{\rho^2} \end{pmatrix} \tag{3.6}$$

Il devient alors possible, par identification, de remonter aux valeurs de σ_y et σ_z. Afin de traduire au mieux la réalité physique de cette incertitude dans le repère caméra $\Re_c$, la variable d_{min} est introduite. Celle-ci renseigne la distance à partir de laquelle la caméra peut voir. En fonction de la position de celle-ci sur le véhicule et de son orientation, il est possible de définir une zone dans laquelle la caméra ne peut rien percevoir. La variable d_{min} permet d'éviter d'intégrer cette zone dans l'incertitude 3D et donc d'affiner la valeur de σ_x. On obtient ainsi le système suivant :

$$\begin{cases} \sigma_x &= \rho - d_{min} \\ \sigma_y &= \frac{\rho \sigma_u}{f_u} \\ \sigma_z &= \frac{\rho \sigma_v}{f_v} \end{cases} \tag{3.7}$$

La figure 3.2 reprend les mêmes paramètres que la 3.1 mais avec le résultat d'une initialisation par les jacobiennes. On peut constater dans cette vue de dessus que

l'ellipse ne dépasse pas l'observateur, ce qui correspond bien à la réalité physique du point.

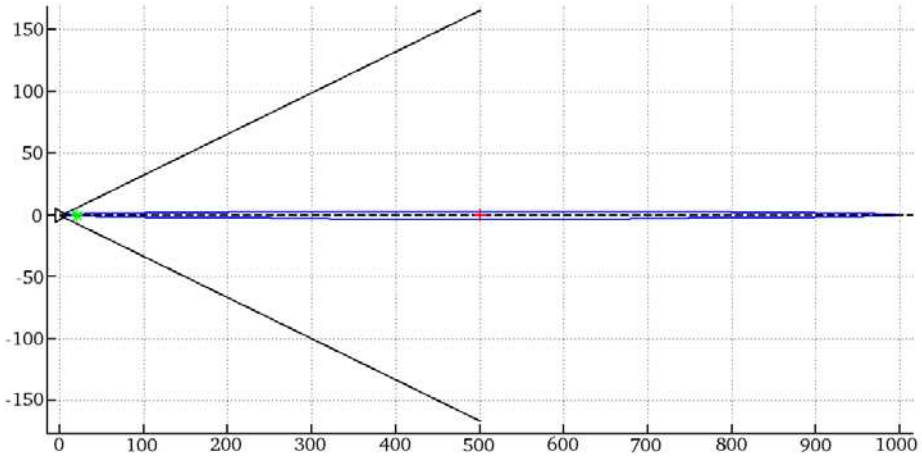


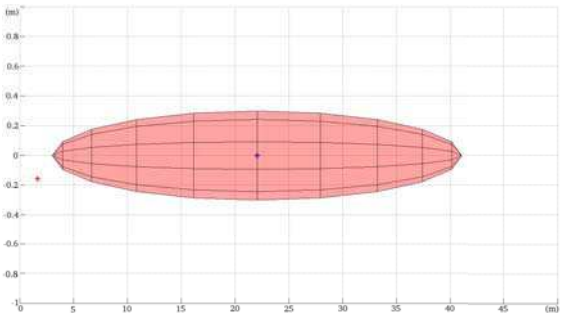
FIGURE 3.2 – Initialisation de l'incertitude par les jacobiennes. Le code couleur et le contexte sont les mêmes que précédemment. L'ellipse bleue correspond à la représentation graphique de l'incertitude créée par les jacobiennes.

Les méthodes d'initialisation présentées au cours des derniers paragraphes donnent des résultats intéressants. Néanmoins, le fait d'utiliser une représentation cartésienne classique rend les linéarisations qui suivront délicates. En effet, le point créé peut être situé loin de sa vraie valeur et donc faire échouer l'étape de linéarisation induite par l'utilisation des jacobiennes. Ces problèmes apparaissent lors du suivi des amers et de l'étape de mise à jour, ces deux cas impliquant le calcul de la jacobienne associée au modèle utilisé. Il s'agit des deux problèmatiques abordées ci-après.

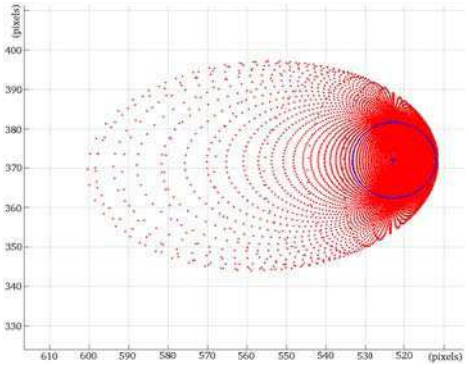
3.1.1.2 Suivi des points via une fenêtre de recherche

Les jacobiennes interviennent à deux moments : lors du suivi des amers et lors de leur mise à jour. Dans le premier cas, elles servent à projeter l'incertitude depuis le repère monde $\Re_m$ dans $\Re_i$. Leur calcul se base sur la fonction d'observation définie à l'équation (2.27) qui permet de passer les points 3D de $\Re_m$ à $\Re_i$. Les différentes rotations, tout comme le modèle sténopé, rendent cette fonction hautement non linéaire. Ainsi, le calcul des jacobiennes peut échouer. Cette méthode de suivi des amers, bien que standard, peut donc donner des résultats aberrants. Cela est d'ailleurs exacerbé par le contexte monoculaire où les estimations des points 3D sont initialement loin des vraies positions. De plus, la correspondance entre une incertitude dans l'image et une incertitude dans le monde est difficile à maintenir puisque la projection d'une incertitude depuis le plan image vers le repère caméra n'est pas un ellipsoïde mais un cône. Les passages d'un repère à l'autre via les jacobiennes peuvent donc ne pas être cohérents et ainsi rendre plus difficile le suivi des points.

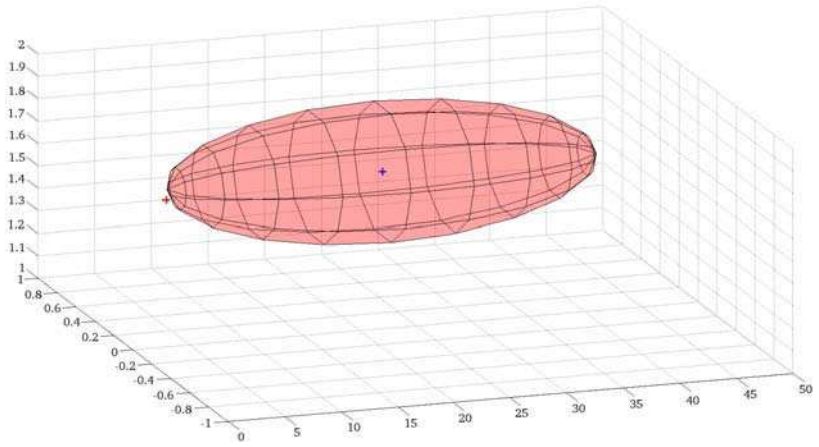
Dans le cas d'une projection via les jacobiennes, le calcul de la covariance 2D se fait via $\mathbf{H}$, la jacobienne associée à la fonction d'observation h (voir équation (2.27)). Malheureusement, la projection de l'incertitude est inexacte et sous-estime souvent la taille réelle de l'ellipse dans l'image. L'exemple de simulation en figure 3.3 montre clairement que l'ellipse estimée via la jacobienne du modèle d'observation est erronée. La conséquence directe est que le suivi du point est difficile à maintenir sur de grandes distances, ce qui rend plus délicate la convergence dudit point vers sa vraie valeur.



(a) Ellipsoïde dans le monde. Vue suivant le plan Oxy. La croix rouge est l'emplacement de la caméra et la croix bleue le centre de l'ellipse (estimée du point 3D).



(b) Projection dans l'image. Les points rouges sont les projections des points composant l'enveloppe de l'ellipsoïde correspondant à l'incertitude 3D de l'amer. La croix bleue est la projection du point 3D dans le plan image. L'ellipse bleue est l'incertitude associée au point et projetée via les jacobiennes.

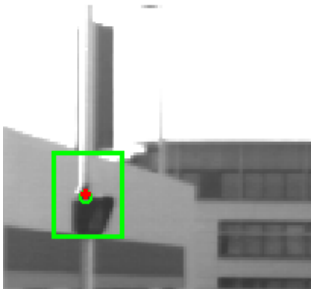


(c) Ellipsoïde dans le monde.

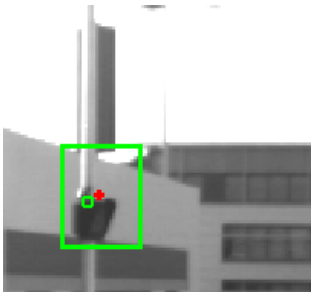
FIGURE 3.3 – Exemple de simulation d'une projection d'un ellipsoïde dans l'image. Celui-ci est projeté depuis l'origine vers le repère caméra, puis dans l'image. Cette dernière mesure 1024 par 768 pixels.

Afin de montrer l'implication que peuvent avoir les erreurs de linéarisation sur le processus de suivi des points, nous avons conduit une expérimentation. Celle-ci est très simple : il s'agit d'un véhicule évoluant sur une ligne droite et suivant un seul et unique point. Pour éviter de mêler les différentes sources d'erreurs, uniquement l'étape de prédiction du filtre de Kalman est accomplie via l'odométrie. Aucune mise à jour n'est faite. La caméra utilisée sert uniquement à suivre un point initialisé dans l'image. Le but est de montrer que la fenêtre calculée par les jacobiennes à partir de l'incertitude 3D d'un point ne permet pas de suivre celui-ci très longtemps. Dans cet exemple, et en l'absence de mise à jour, la taille de la fenêtre de recherche ne fait que grossir du fait de l'incertitude grandissante du véhicule. Le résultat de ce test est disponible en figure 3.4.

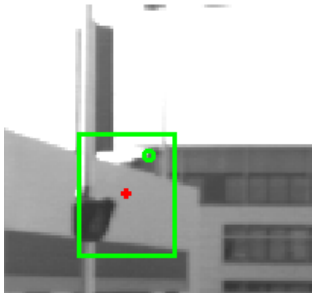
(a) Initialisation du point



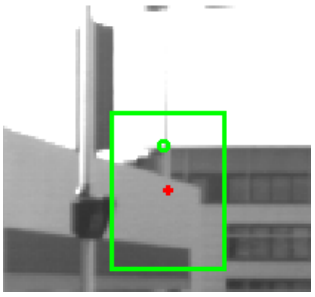
(b) Image 1 : observation correcte



(c) Image 2 : observation correcte



(d) Image 3 : observation incorrecte



(e) Image 7 : observation incorrecte

FIGURE 3.4 – Zone de recherche et suivi par les jacobiennes. Le rectangle vert est la fenêtre calculée à partir de la projection de la covariance par les jacobiennes. La croix rouge est la projection de la position estimée de l'amer (sauf sur la première image où là elle correspond au point initialisé). Le cercle vert est l'observation trouvée par ZNCC.

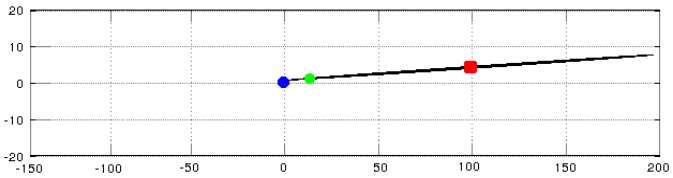
Dès la troisième image, on constate que la fenêtre de recherche ne permet pas de trouver le vrai point. L'incertitude dérive rapidement sans que sa taille augmente suffisamment pour pouvoir intégrer la position réelle du point dans l'image. Ce problème est critique car il a d'importantes répercussions. La perte du point dès la troisième image signifie qu'un nouveau point va devoir être initialisé pour compenser. Il y a donc un surcoût calculatoire non négligeable. De plus, le nombre réduit de points précis, car difficiles à suivre longtemps, rend la localisation du véhicule très approximative. Le fait de s'éloigner de la vraie position du point est aussi problématique car cela provoque des comportements non souhaités, comme des associations avec des observations fausses par exemple. Les risques de divergence du système tout entier sont donc importants et tout spécialement en cas de longs déplacements entre les images.

3.1.1.3 Mise à jour de la position d'un amer

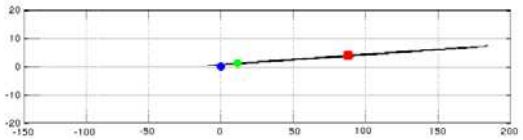
Le deuxième aspect concerné par les erreurs de linéarisation est la mise à jour par le filtre de Kalman. La jacobienne du modèle d'observation est également impliquée dans le calcul du gain. L'effet de cette jacobienne sur le système peut être particulièrement pénalisant car le gain de Kalman fait alors évoluer l'estimation de la position des amers de manière incohérente. Cela peut donner par exemple des points qui sont mis à jour derrière le véhicule alors qu'ils viennent d'être observés ou, plus généralement, des corrections trop grandes. La conséquence directe est que l'amer mis à jour sort de son incertitude initiale, ce qui est normalement extrêmement improbable. De même que pour le suivi, plus de points doivent être initialisés puisque ceux-ci deviennent régulièrement inobservables obligeant ainsi le système à sélectionner de nouveaux points caractéristiques dans l'image. Il y a donc encore une fois une charge calculatoire supplémentaire à cause de mauvaises linéarisations. Le plus critique reste le fait que le véhicule, tout comme les amers, peut avoir son estimation qui diverge. Les erreurs de linéarisation impliquées par l'étape de mise à jour sont exposées en figure 3.5 au travers d'un exemple.

Les conditions de cet exemple sont similaires à celles décrites pour la figure 3.4. Il s'agit de la même séquence. Le point mis à jour est le même que celui précédemment suivi pour démontrer l'échec du calcul de la fenêtre englobante par la méthode des jacobiennes. Cette fois, l'étape de mise à jour est active. Les observations utilisées pour les mises à jour sont correctes afin d'éviter d'avoir à considérer plusieurs problèmes à la fois. Ici, uniquement les faiblesses de la mise à jour sont visibles. Dès la deuxième mise à jour, la position du point devient incohérente et passe derrière le véhicule qui vient juste de l'observer. Plus le point est loin de sa vraie valeur, plus ce phénomène a de l'importance.

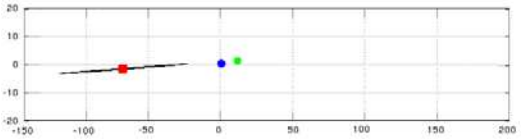
Les problèmes constatés ci-dessus existent dès lors que des modèles non linéaires sont impliqués. Cependant, l'utilisation de capteurs fournissant la profondeur des amers diminuerait grandement les chances de voir ce phénomène se déclarer puisque les linéarisations seraient quasiment exactes et que les incertitudes des points seraient faibles. Le contexte monoculaire et l'incertitude que cela génère quant à la position



(a) Initialisation de l'amer à 100 mètres sur le rayon de vue



(b) Première mise à jour correcte



(c) Seconde mise à jour erronée

FIGURE 3.5 – Mise à jour avec erreur de linéarisation. Vue de dessus des mises à jour d'un amer. Le cercle bleu est la position du véhicule. Le carré rouge est la position estimée de l'amer. Le cercle vert est la vraie position de cet amer. L'ellipse noire est l'incertitude associée à l'amer rouge.

des amers sont extrêmement favorables à ces erreurs. Ainsi, la communauté de la robotique mobile travaillant sur le SLAM monoculaire a rapidement proposé des solutions pour contourner ces problèmes. Celles-ci seront présentées dans la sous-section suivante.

3.1.2 Étude de la littérature

Les problèmes de linéarisation évoqués dans la sous-section précédente sont généralement traités dans la littérature via une méthode d'initialisation particulière. Cette dernière, couplée à une représentation plus adaptée des amers peut grandement réduire les erreurs de linéarisation. On distingue deux courants principaux : les initialisations retardées et les initialisations immédiates.

Dans la première catégorie, il est question de retarder l'initialisation jusqu'à ce que l'amer soit assez précis pour avoir sa profondeur bien estimée et ainsi éviter les erreurs de linéarisation. Dans [Bailey 2003] par exemple, Bailey propose d'attendre au moins 2 observations d'un même point pour avoir une estimation de sa profondeur. Il est précisé qu'avec seulement deux points l'initialisation peut être mal conditionnée (pas assez précise). Trois facteurs permettent de décider si une initialisation est correcte ou pas : l'incertitude quant à la pose du véhicule, l'incertitude des mesures et enfin l'écart entre les 2 poses véhicule à partir desquelles l'amer a été observé. Un indice cumulant ces informations permet de décider si les 2 mesures de l'amer sont suffisantes ou s'il est nécessaire d'attendre d'autres observations. Dans [Davison 2003], un filtre particulaire est utilisé afin de pouvoir tester simultanément plusieurs hypothèses de profondeur. La caméra évoluant en intérieur, 100 particules sont étalées sur 5 mètres (distance maximale des points). Chaque particule projetée

dans l'image donne une ellipse dans laquelle une observation est cherchée. En fonction des associations faites, des poids sont distribués aux particules. Dès lors que l'approximation de la distance devient meilleure, l'amer est initialisé dans un EKF et l'incertitude est approximée par une gaussienne.

Ces méthodes sont efficaces et évitent les erreurs de linéarisation. Cependant, retarder l'intégration d'un amer pose le problème de la perte de données entre l'instant où il est vu la première fois et le moment où il est réellement inséré dans le filtre. Pendant ce laps de temps, le suivi d'un amer peut apporter des informations précieuses quant à l'orientation du véhicule étant donné que l'amer est plutôt bien localisé de ce point de vue. De plus, cela signifie qu'au cours de virages serrés quasiment aucune autre information que l'odométrie ne sera disponible. En effet, les amers n'auront pas le temps de converger et ne seront donc jamais intégrés au filtre, laissant ainsi l'estimation du véhicule se reposer uniquement sur des informations proprioceptives.

Les approches se reposant sur des initialisations immédiates proposent en revanche d'intégrer les amers dès leur première observation. Afin d'éviter les problèmes de linéarisation, les approches [Kwok & Dissanayake 2004] et [Solà *et al.* 2005] insèrent plusieurs estimations par amer dans le vecteur d'état. Au fur et à mesure que de nouvelles observations sont disponibles, les hypothèses qui s'avèrent fausses sont supprimées. Des poids sont également associés aux hypothèses afin d'éviter que les corrélations dans les données ne viennent perturber le bon fonctionnement du filtre. Ces travaux sont assez similaires à ceux de Davison et al. [Davison 2003], seulement les amers sont directement intégrés dans le vecteur d'état. Il est donc nécessaire de stocker plusieurs hypothèses par point ce qui induit un surcoût mémoire important. Le plus handicapant reste que chaque point devra être traité dans le cadre du filtre, impliquant ainsi des étapes de prédiction et de mise à jour supplémentaires en plus de la gestion des poids. La charge calculatoire induite par ces initialisations multi-hypothèses est un inconvénient puisque cela limite le nombre d'amers qui peuvent être suivis simultanément.

Toujours dans les initialisations immédiates, la méthode la plus couramment employée est l'utilisation de la paramétrisation en profondeur inverse. Cette représentation a été introduite dans [Montiel *et al.* 2006], puis reprise dans [Civera *et al.* 2008] et appliquée au SLAM monoculaire avec succès. La représentation en profondeur inverse offre l'avantage de mieux représenter les grandes incertitudes suivant un axe. Les amers 3D exprimés en profondeur inverse requièrent 6 paramètres pour estimer leur position. Un amer 3D $\mathbf{a}_i$ est défini comme suit :

$$\mathbf{a}_i = \begin{pmatrix} x_i & y_i & z_i & \theta_i & \phi_i & \rho_i \end{pmatrix}^{\mathrm{T}} \tag{3.8}$$

où x_i, y_i et z_i représentent le centre optique de la caméra où le point 3D a été observé pour la première fois (ancre). θ_i et ϕ_i sont l'azimut et l'élévation du rayon d'observation $\mathbf{m}(\theta_i, \phi_i)$. La profondeur du point d_i est codée par sa valeur inverse $\rho_i = 1/d_i$.

Ces 6 paramètres modélisent un point 3D situé à :

$$\begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} + \frac{1}{\rho_i}\mathbf{m}(\theta_i, \phi_i) \tag{3.9}$$

La figure 3.6, issue de [Montiel *et al.* 2006], montre les différents repères pour deux observations d'un point et la place de la représentation inverse dans ceux-ci.

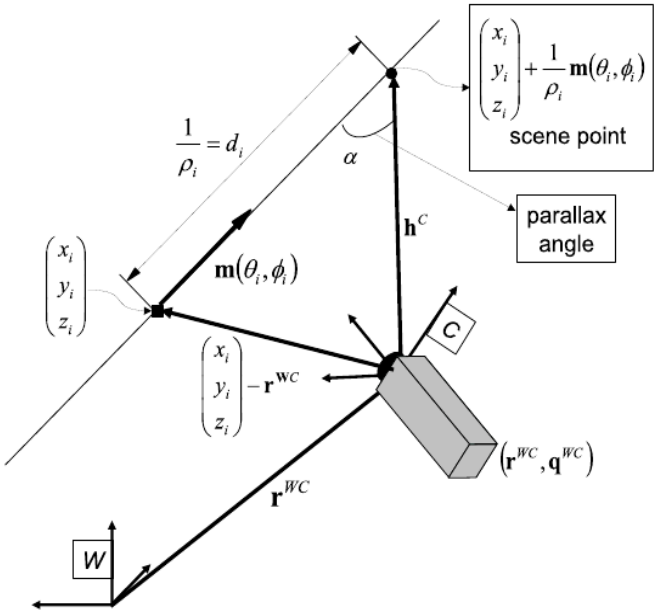


FIGURE 3.6 – Paramétrisation inverse et représentation de [Montiel *et al.* 2006]

Bien que cette représentation soit intéressante, elle demande à ce que chaque point de la carte soit stocké avec 6 paramètres au lieu de 3 pour une représentation cartésienne classique. Cela signifie que la taille de la carte est doublée en mémoire pour le vecteur d'état et quadruplée pour la covariance associée. Souvent, les points sont d'ailleurs convertis depuis la profondeur inverse vers la forme cartésienne classique [Civera *et al.* 2007]. L'état intermédiaire des points doit alors être géré et des indices doivent être calculés pour savoir quand passer d'un point en profondeur inverse à un amer classique. Cela induit aussi qu'il faut être prudent quant à la manière dont sont traités les points puisque les représentations sont différentes et n'ont pas les mêmes implications sur l'étape de mise à jour. Enfin, et c'est certainement le plus important, la représentation en profondeur inverse n'empêche pas les erreurs de linéarisation de se produire. Bien que le risque soit réduit, l'éventualité est toujours présente [Bekris *et al.* 2006]. Même avec cette représentation, une solution aux erreurs de linéarisation doit être envisagée.

La communauté s'est, en grande partie, concentrée sur le problème de l'initialisation et de la représentation des amers qui sont souvent les principaux générateurs

d'erreurs de linéarisation. Néanmoins, même avec une représentation adaptée, ces problèmes sont toujours présents et des solutions doivent être trouvées pour le suivi des points ainsi que leurs mises à jour.

Dans le cas du suivi, la méthode la plus employée reste l'utilisation des jacobiennes. Dans [Davison *et al.* 2007], celle-ci est appelée recherche active et est identique à la méthode décrite précédemment. Son utilisation est fréquente dans la littérature. D'autres techniques ont misé sur des fenêtres de recherche dont la taille est fixe. C'est le cas de [Mouragnon *et al.* 2006] qui propose de calculer la taille maximale de celles-ci en fonction du déplacement maximum possible du véhicule entre deux images successives. Malheureusement, les boîtes englobantes obtenues dans l'image sont souvent trop grandes par rapport à l'incertitude réelle du point. Enfin, d'autres méthodes, ne dépendant pas d'une fenêtre de recherche, existent. On peut par exemple citer [Petrovskaya & Thrun 2009] qui propose d'utiliser la méthode de Laplace pour positionner correctement une gaussienne au maximum global de la fonction d'observation. Néanmoins, cette méthode nécessite une étape de minimisation qui peut être coûteuse.

Pour la mise à jour via le filtre de Kalman, nous pouvons également citer les approches de Davison [Davison 2003] qui ne font que de petites mises à jour pour éviter des corrections de Kalman fausses. Pour ce faire, il est nécessaire d'utiliser des caméras très rapides ou d'avoir un véhicule qui ne se déplace que très lentement. Ces deux contraintes sont difficilement envisageables dans notre contexte. Enfin, nous pouvons noter les travaux de Parsley et al. [Parsley & Julier 2008] qui a été le premier à constater la possibilité pour des amers mis à jour de passer derrière le véhicule, et ce même en utilisant la paramétrisation en profondeur inverse. La solution proposée dans cet article consiste simplement à annuler la mise à jour dès lors que le point est mis à jour avec une profondeur négative.

En conclusion, l'état de l'art sur les problèmes de linéarisation dans le cas monoculaire donne quelques clefs pour comprendre comment envisager le problème. Néanmoins, aucune solution ne semble optimale pour construire une approche qui soit à la fois économe (temps de calcul et consommation mémoire) et robuste. En effet, la localisation peut très rapidement diverger et rendre le processus de SLAM complètement faux. Dans la section suivante, nous proposerons de traiter chaque point où les linéarisations interviennent afin d'avoir une approche capable de parer à toutes les éventualités.

3.2 Solution proposée

La section précédente a montré l'impact que peuvent avoir les erreurs de linéarisation. L'étude de la littérature a aussi mis en avant que, souvent, une fréquence de caméra élevée couplée à une représentation des amers en profondeur inverse est suffisante pour des applications de SLAM monoculaire. Néanmoins, la majorité de ces approches ne considèrent pas le surcoût induit par cette représentation. Dans le cadre des applications à plusieurs véhicules que nous visons, cette contrainte est

d'autant plus handicapante qu'elle va être amplifiée par le nombre de véhicules, que cela soit sur le réseau, en terme de consommation mémoire ou dans les traitements.

Ce contexte particulier nous oriente donc davantage vers la conservation d'une représentation cartésienne classique. Il est donc nécessaire de traiter les différentes étapes où la linéarisation intervient afin d'en limiter l'impact. Tout d'abord, nous commencerons par proposer un calcul plus optimal de la fenêtre de recherche en sous-section 3.2.1. Enfin, la sous-section 3.2.2 introduira un correctif pour le gain de Kalman évitant les problèmes de divergence lors de la mise à jour. Chaque contribution sera étayée par une mise en application concrète. La solution monoculaire présentée dans cette section est basée sur des premiers travaux initiés au sein du laboratoire (voir [Féraud 2011] pour plus de détails).

3.2.1 Suivi des amers

La projection dans l'image de l'incertitude associée à l'estimation d'un point pose le problème de la linéarisation. Comme vu en section 3.1.1, les jacobiennes ne garantissent pas un résultat correct dès lors qu'elles sont calculées loin de la vraie valeur. Afin d'éviter ces problèmes puisque les estimées des amers peuvent être très approximatives, nous avons choisi de développer une approche géométrique. Cette méthode a été publiée pour la première fois dans [Bresson *et al.* 2011] puis étendue ensuite dans [Bresson *et al.* 2012b].

Ici, l'objectif est de réussir à obtenir une incertitude dans l'image correspondant à un ellipsoïde dans le repère monde. Considérons un amer 3D $\mathbf{a}_c = \begin{pmatrix} x_0 & y_0 & z_0 \end{pmatrix}^{\mathrm{T}}$ défini dans le repère caméra et sa covariance $\mathbf{P}_{\mathbf{a}_c}$. Cette dernière peut s'écrire :

$$\mathbf{P}_{\mathbf{a}_c} = \begin{bmatrix} a & b & c \\ b & d & e \\ c & e & f \end{bmatrix} \tag{3.10}$$

Pour plus de commodité dans la suite de l'explication, nous définissions son inverse comme suit :

$$\begin{aligned} \mathbf{P}_{\mathbf{a}_c}^{-1} &= \frac{1}{det\ \mathbf{P}_{\mathbf{a}_c}} \begin{bmatrix} df - e^2 & ce - bf & be - cd \\ ce - bf & af - c^2 & cb - ae \\ be - cd & cb - ae & ad - b^2 \end{bmatrix} \\ &= \begin{bmatrix} A & B & C \\ B & D & E \\ C & E & F \end{bmatrix} \end{aligned} \tag{3.11}$$

où $det\ \mathbf{P}_{\mathbf{a}_c}$ est le déterminant de la matrice $\mathbf{P}_{\mathbf{a}_c}$.

L'idée principale de cette approche géométrique est de trouver les plans tangents à l'ellipsoïde qui permettent de délimiter l'ellipse dans l'image. Afin de clarifier la méthode, un exemple 2D est présenté en figure 3.7. Celui-ci pourra être suivi en parallèle avec les équations (données pour une incertitude 3D) afin de comprendre

l'implication de chaque contrainte sur la construction de la solution. La figure 3.7(a) montre la vue 2D du problème à résoudre.

Afin de trouver les plans tangents à l'ellipsoïde, il faut tout d'abord identifier les points composant la surface de l'ellipsoïde. Soit $\mathscr{E}$, l'ellipsoïde généré par la matrice de covariance $\mathbf{P}_{\mathbf{a}_c}$. Les points $\begin{pmatrix} x & y & z \end{pmatrix}^{\mathrm{T}}$ sur la surface de $\mathscr{E}$ vérifient la relation de Mahalanobis suivante :

$$\begin{pmatrix} x-x_0 \\ y-y_0 \\ z-z_0 \end{pmatrix}^T \mathbf{P}_{\mathbf{a}_c}^{-1} \begin{pmatrix} x-x_0 \\ y-y_0 \\ z-z_0 \end{pmatrix} = 1 \tag{3.12}$$

où $\begin{pmatrix} x_0 & y_0 & z_0 \end{pmatrix}^{\mathrm{T}} = \mathbf{a}_c$ est le centre de $\mathscr{E}$ (voir figure 3.7(b)).

Les plans que l'on cherche, en plus d'être tangents à l'ellipsoïde, sont orthogonaux aux normales des points formant la surface de $\mathscr{E}$ (voir figure 3.7(c)). Cette contrainte supplémentaire sur les plans peut s'exprimer grâce au gradient en ces points :

$$\begin{aligned} \vec{n} &= \nabla\mathscr{E}(x,y,z) = \begin{pmatrix} \frac{\partial\mathscr{E}}{\partial x} & \frac{\partial\mathscr{E}}{\partial y} & \frac{\partial\mathscr{E}}{\partial z} \end{pmatrix}^T \\ &= \begin{pmatrix} 2(x-x_0)A + 2(y-y_0)B + 2(z-z_0)C \\ 2(x-x_0)B + 2(y-y_0)D + 2(z-z_0)E \\ 2(x-x_0)C + 2(y-y_0)E + 2(z-z_0)F \end{pmatrix} \end{aligned} \tag{3.13}$$

Ainsi, les points faisant partie d'un plan P tel que défini précédemment, c'est-à-dire orthogonal à la normale, vérifient la relation suivante :

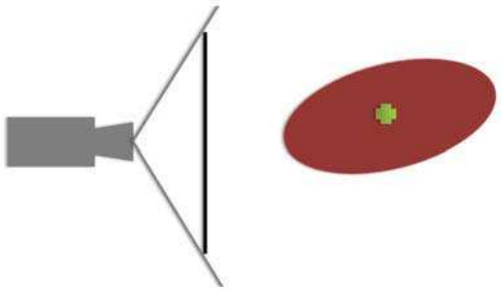
$$\begin{pmatrix} x_p \\ y_p \\ z_p \end{pmatrix} \in P \Leftrightarrow \begin{pmatrix} x_p - x \\ y_p - y \\ z_p - z \end{pmatrix}^T \cdot \vec{n} = 0 \tag{3.14}$$

Quelques plans tangents exprimés grâce à ces équations sont visibles sur la figure 3.7(d). Uniquement les plans passant par l'origine (la caméra) sont intéressants pour trouver l'ellipse dans l'image. Nous cherchons donc les plans dont le point $\begin{pmatrix} 0 & 0 & 0 \end{pmatrix}^{\mathrm{T}}$ fait partie. Cela donne comme contrainte :

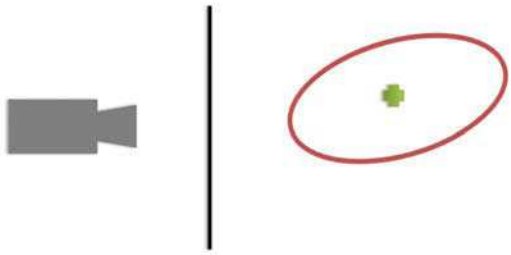
$$\begin{pmatrix} x & y & z \end{pmatrix} \cdot \vec{n} = 0 \tag{3.15}$$

La sélection de plans obtenus est illustrée sur la figure 3.7(e). Sur cette vue de dessus, il ne reste plus qu'à projeter dans l'image les points de l'ellipse où les droites sont tangentes. Néanmoins, en repassant en 3D, on se rend compte qu'il reste encore une infinité de points satisfaisant les contraintes précédemment définies. En projetant tous ces points, il serait possible d'obtenir le contour exact de l'ellipse dans l'image. Cela est bien sûr impossible. Il faut alors ajouter une nouvelle contrainte afin de réduire le nombre de points à projeter.

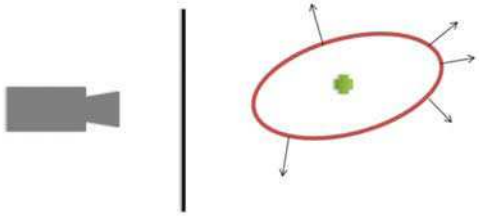
Afin de comprendre quels points garder, la figure 3.8 montre comment les plans tangents à l'ellipsoïde et passant par l'origine intersectent l'image.



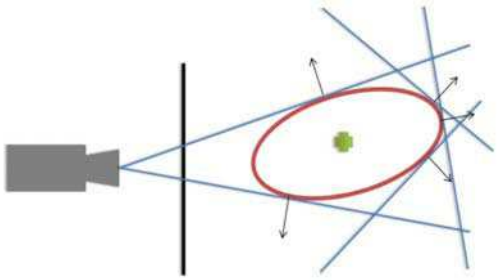
(a) Caméra avec son plan image (qui sur cette vue est une droite). La croix verte est un amer. L'ellipse rouge est l'incertitude que l'on cherche à projeter dans le plan image.



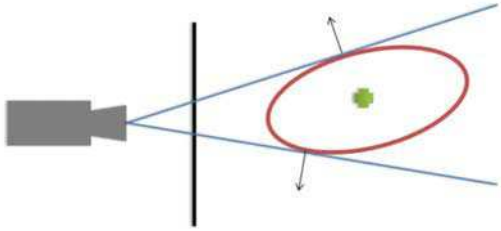
(b) Sélection des points composant le contour de l'ellipse (en rouge).



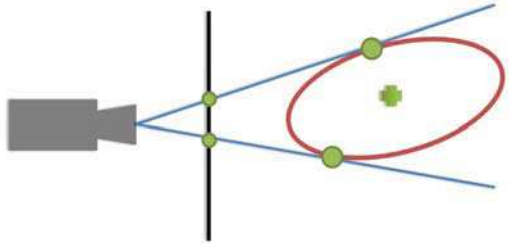
(c) Exemples de normales à ces points.



(d) Droites tangentes correspondantes (en bleues). Dans le cas 3D, il s'agit de plans.



(e) Droites tangentes passant par l'origine. En 3D, ce sont des plans.



(f) Solution sur cette vue 2D. Les points verts sur l'ellipse peuvent être projetés dans l'image pour obtenir la zone de recherche (entre les 2 points verts sur l'image).

FIGURE 3.7 – Exemple 2D pour le calcul de la fenêtre englobante

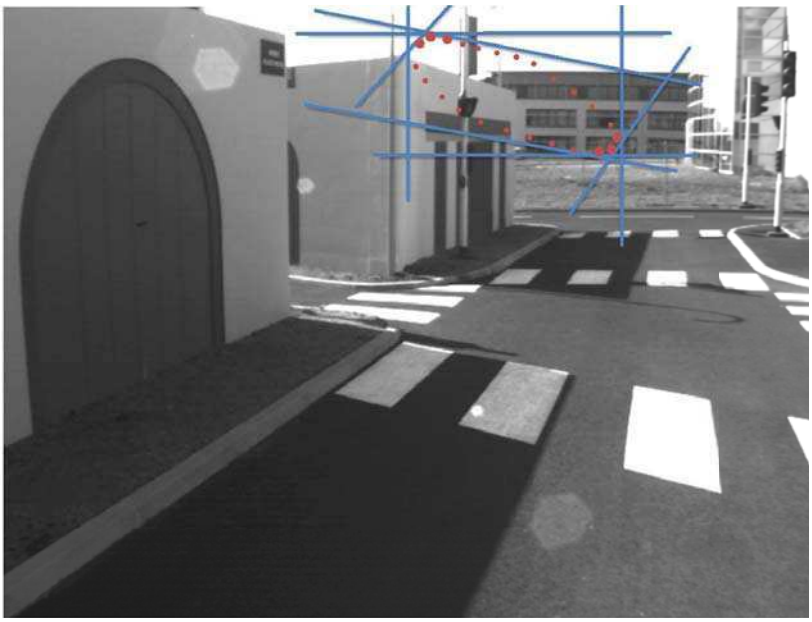


FIGURE 3.8 – Exemples de plans intersectant l'image. Les droites bleues sont les projections des plans dans l'image. Les gros points rouges sont les points sur $\mathscr{E}$ associés aux plans bleus. Les petits points rouges sont la projection d'autres points validant les contraintes mais dont les plans n'ont pas été représentés pour des raisons de lisibilité.

L'ellipse est facilement identifiable dans cette image. L'objectif est maintenant de trouver une fenêtre englobante (zone rectangulaire) autour de cette ellipse. Cela s'explique par le fait que les algorithmes de corrélation utilisés pour la mise en correspondance entre prédiction et observation dans l'image (et dont fait partie le processus de ZNCC) sont plus rapides sur des fenêtres rectangulaires que sur des formes elliptiques. Ainsi, nous cherchons à garder uniquement les points donnant les plans intersectant horizontalement ou verticalement l'image et donc la caméra. Cela permet de ne garder que 4 points dont les 4 plans tangents forment une zone rectangulaire dans l'image. Ceux-ci incluent l'axe des $\vec{y}$ ou des $\vec{z}$ suivant s'ils intersectent l'image horizontalement ou verticalement :

$$\vec{y} \cdot \vec{n} = 0 \Leftrightarrow (x \quad x_0)B + (y \quad y_0)D + (z \quad z_0)E = 0 \tag{3.16}$$

$$\vec{z} \cdot \vec{n} = 0 \Leftrightarrow (x \quad x_0)C + (y \quad y_0)E + (z \quad z_0)F = 0 \tag{3.17}$$

Ces contraintes peuvent être regroupées afin de définir les systèmes à résoudre. Dans le cas des deux points donnant les parties haute et basse de la fenêtre (plans horizontaux), on peut poser le système suivant :

$$\begin{cases} \begin{pmatrix} x-x_0 \\ y-y_0 \\ z-z_0 \end{pmatrix}^T \mathbf{P}_{\mathbf{a}_c}^{-1} \begin{pmatrix} x-x_0 \\ y-y_0 \\ z-z_0 \end{pmatrix} = 1 \\ \\ \begin{pmatrix} x & y & z \end{pmatrix} \cdot \vec{n} = 0 \\ \\ \vec{y} \cdot \vec{n} = 0 \end{cases} \tag{3.18}$$

La résolution du système précédent donnera deux points $\mathbf{p}_{h_i}$ avec $i \in \{1, 2\}$ dont les tangentes intersectent horizontalement l'image. Pour les parties gauche et droite de la fenêtre (plans verticaux), il faudra résoudre le système suivant :

$$\begin{cases} \begin{pmatrix} x-x_0 \\ y-y_0 \\ z-z_0 \end{pmatrix}^T \mathbf{P}_{\mathbf{a}_c}^{-1} \begin{pmatrix} x-x_0 \\ y-y_0 \\ z-z_0 \end{pmatrix} = 1 \\ \\ \begin{pmatrix} x & y & z \end{pmatrix} \cdot \vec{n} = 0 \\ \\ \vec{z} \cdot \vec{n} = 0 \end{cases} \tag{3.19}$$

De même, deux points $\mathbf{p}_{v_i}$ avec $i \in \{1, 2\}$ ont des tangentes qui passent par l'image verticalement.

La résolution du système (3.18) donne les deux points suivants :

$$\begin{cases} \Delta_h = \beta_h^2 - 4\alpha_h\gamma_h \\ \\ x_{h_1} = \dfrac{-\beta_h + \sqrt{\Delta_h}}{2\alpha_h} \quad \Big| \quad x_{h_2} = \dfrac{-\beta_h - \sqrt{\Delta_h}}{2\alpha_h} \\ \\ z_{h_1} = -\dfrac{m_{h_0}}{m_{h_1}} x_{h_1} - \dfrac{n_h}{m_{h_1}} \quad \Big| \quad z_{h_2} = -\dfrac{m_{h_0}}{m_{h_1}} x_{h_2} - \dfrac{n_h}{m_{h_1}} \\ \\ y_{h_1} = y_0 - \dfrac{(x_{h_1} - x_0)B + (z_{h_1} - z_0)E}{D} \quad \Big| \quad y_{h_2} = y_0 - \dfrac{(x_{h_2} - x_0)B + (z_{h_2} - z_0)E}{D} \end{cases} \tag{3.20}$$

avec :

$$\begin{cases} \mathbf{M}_h = \begin{pmatrix} A - \dfrac{B^2}{D} & C - \dfrac{BE}{D} \\ C - \dfrac{BE}{D} & F - \dfrac{E^2}{D} \end{pmatrix} \\ m_{h_0} = -\begin{pmatrix} x_0 & y_0 \end{pmatrix} \begin{pmatrix} A - \dfrac{B^2}{D} \\ C - \dfrac{BE}{D} \end{pmatrix} \quad \Big| \quad m_{h_1} = -\begin{pmatrix} x_0 & y_0 \end{pmatrix} \begin{pmatrix} C - \dfrac{BE}{D} \\ F - \dfrac{E^2}{D} \end{pmatrix} \\ n_h = \begin{pmatrix} x_0 & y_0 \end{pmatrix} \mathbf{M}_h \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} - 1 \\ \alpha_h = \begin{pmatrix} 1 \\ -\dfrac{m_{h_0}}{m_{h_1}} \end{pmatrix}^T \mathbf{M}_h \begin{pmatrix} 1 \\ -\dfrac{m_{h_0}}{m_{h_1}} \end{pmatrix} \quad \Big| \quad \gamma_h = \left(F - \dfrac{E^2}{D} \right) \left(\dfrac{n_h}{m_{h_1}} \right)^2 - n_h \\ \beta_h = 2 \left[\left(F - \dfrac{E^2}{D} \right) \dfrac{m_{h_0} n_h}{m_{h_1}^2} - \left(C - \dfrac{BE}{F} \right) \dfrac{n_h}{m_{h_1}} \right] \end{cases} \tag{3.21}$$

Et de façon similaire pour les points dont les tangentes passent par l'image verticalement, on peut résoudre le système (3.19) de la façon suivante :

$$\begin{cases} \Delta_v = \beta_v^2 - 4\alpha_v \gamma_v \\ x_{v_1} = \dfrac{-\beta_h + \sqrt{\Delta_h}}{2\alpha_h} \quad \Big| \quad x_{v_2} = \dfrac{-\beta_h - \sqrt{\Delta_h}}{2\alpha_h} \\ y_{v_1} = -\dfrac{m_{v_0}}{m_{v_1}} x_{v_1} - \dfrac{n_v}{m_{v_1}} \quad \Big| \quad y_{v_2} = -\dfrac{m_{v_0}}{m_{v_1}} x_{v_2} - \dfrac{n_v}{m_{v_1}} \\ z_{v_1} = z_0 - \dfrac{(x_{v_1} - x_0)C + (y_{v_1} - y_0)E}{F} \quad \Big| \quad z_{v_2} = z_0 - \dfrac{(x_{z_2} - x_0)C + (y_{v_2} - y_0)E}{F} \end{cases} \tag{3.22}$$

avec :

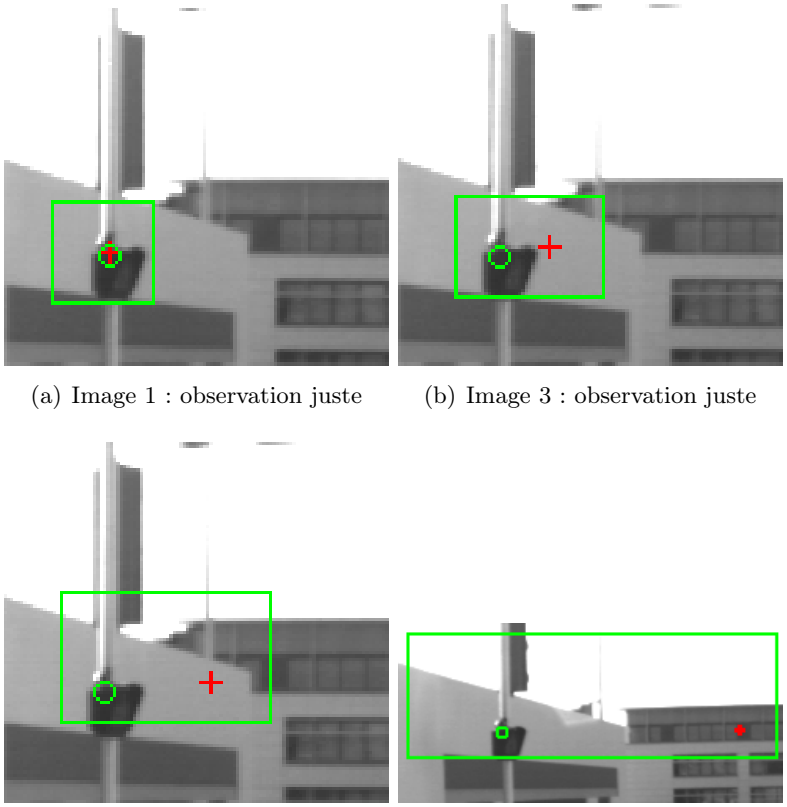
$$\begin{cases} \mathbf{M}_v = \begin{pmatrix} A - \dfrac{C^2}{F} & B - \dfrac{CE}{F} \\ B - \dfrac{CE}{F} & D - \dfrac{E^2}{F} \end{pmatrix} \\ m_{v_0} = -\begin{pmatrix} x_0 & y_0 \end{pmatrix} \begin{pmatrix} A - \dfrac{C^2}{F} \\ B - \dfrac{CE}{F} \end{pmatrix} \quad \Big| \quad m_{v_1} = -\begin{pmatrix} x_0 & y_0 \end{pmatrix} \begin{pmatrix} B - \dfrac{CE}{F} \\ D - \dfrac{E^2}{F} \end{pmatrix} \\ n_v = \begin{pmatrix} x_0 & y_0 \end{pmatrix} \mathbf{M}_v \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} - 1 \\ \alpha_v = \begin{pmatrix} 1 \\ -\dfrac{m_{v_0}}{m_{v_1}} \end{pmatrix}^T \mathbf{M}_v \begin{pmatrix} 1 \\ -\dfrac{m_{v_0}}{m_{v_1}} \end{pmatrix} \quad \Big| \quad \gamma_v = \left(D - \dfrac{E^2}{F} \right) \left(\dfrac{n_v}{m_{v_1}} \right)^2 - n_v \\ \beta_v = 2 \left[\left(D - \dfrac{E^2}{F} \right) \dfrac{m_{v_0} n_v}{m_{v_1}^2} - \left(B - \dfrac{EC}{F} \right) \dfrac{n_v}{m_{v_1}} \right] \end{cases} \tag{3.23}$$

Les points obtenus peuvent ensuite être projetés dans l'image afin de pouvoir calculer la fenêtre englobante. En reprenant l'exemple de suivi présenté dans la figure 3.4 et en appliquant la méthode exposée ici, les résultats sont bien meilleurs et peuvent être consultés en figure 3.9. Le contexte de l'expérimentation est le même que précédemment, à savoir uniquement du suivi et aucune mise à jour.

Cette fois-ci le suivi dure plus longtemps grâce à la fenêtre qui grandit en même temps que la pose du véhicule devient incertaine. Même après 18 images, l'algorithme est toujours en mesure d'associer correctement l'estimée avec l'observation. Il est aussi important de constater que la projection de l'amer dans l'image n'est pas forcément centrée dans la zone de recherche. En effet, en fonction de l'orientation de l'ellipsoïde, il est logique que le point soit décalé dans le plan image.

Afin d'illustrer l'intérêt de cette fenêtre de suivi, nous avons réalisé une seconde expérimentation. Pour cela, nous avons utilisé un véhicule se déplaçant dans un environnement urbain statique à 2 mètres par seconde sur une trajectoire de 170 mètres. La caméra équipée fonctionne à 7,5 Hz. Dans chaque image, notre algorithme suit 20 amers. Dès lors qu'un point sort de l'image ou ne peut plus être suivi, un nouveau point est initialisé. La trajectoire accomplie est visible en figure 3.10.

Cette même trajectoire a été rejouée deux fois. La première fois, un EKF SLAM utilisant la méthode de projection par les jacobiennes pour le calcul de la boîte englobante est employé. La deuxième fois, le même algorithme mais avec le calcul



(a) Image 1 : observation juste (b) Image 3 : observation juste

(c) Image 7 : observation juste (d) Image 18 : observation juste

FIGURE 3.9 – Zone de recherche et suivi par l'approche géométrique. Le rectangle vert est la fenêtre calculée à partir de la projection de la covariance par l'approche géométrique présentée ici. La croix rouge est la projection de la position estimée de l'amer. Le cercle vert est l'observation trouvée par ZNCC.

de fenêtre présenté dans cette section a été utilisé. L'EKF SLAM est strictement identique dans les deux cas, seul le calcul géométrique de la fenêtre diffère d'une exécution à l'autre. L'étape de mise à jour est active dans les deux cas. Celle-ci peut provoquer des erreurs de linéarisation qui ne sont pas corrigées ici. La convergence des amers n'est donc pas garantie. Afin de comparer ces approches, nous définissons deux métriques qui seront les mêmes pour toutes les comparaisons de l'algorithme monovéhicule : le nombre d'amers initialisés et le nombre d'amers ayant convergé.

De nouveaux amers sont initialisés dès lors que ceux précédemment suivis ne sont plus visibles. Le véhicule peut tout simplement les avoir dépassés ou ils peuvent être occultés. Néanmoins, dans le cas de l'approche par les jacobiennes, le fait que la fenêtre de suivi ne soit pas correcte peut empêcher le suivi d'un amer et ainsi déclencher de nouvelles initialisations. Cet indicateur le notifiera. Le nombre d'amers conservés est également intéressant. Il est tout d'abord bon de rappeler qu'un amer est conservé dès lors qu'il a convergé. La convergence est atteinte quand l'incertitude

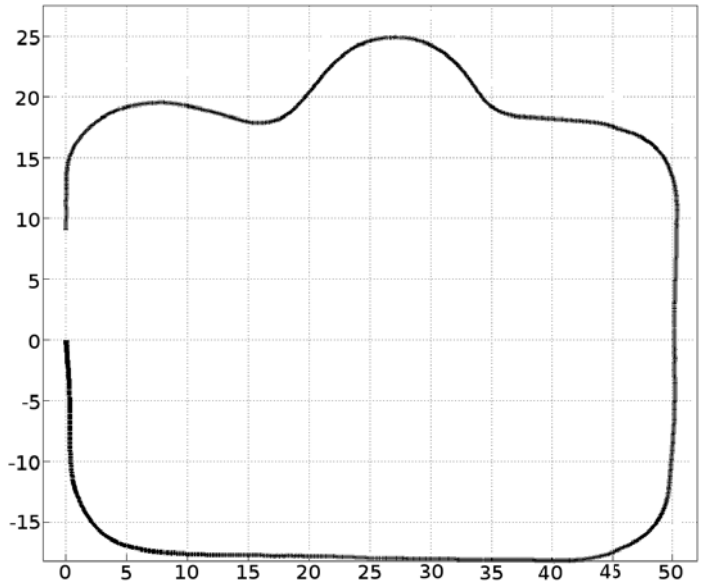


FIGURE 3.10 – Trajectoire utilisée pour la validation du suivi

initiale du point est réduite à une valeur inférieure à un seuil. Dans notre cas, nous utilisons les incertitudes de l'amer suivant les 3 axes $\vec{x}$, $\vec{y}$ et $\vec{z}$. Si la somme de ces 3 incertitudes est inférieure à 50 centimètres alors l'amer est considéré comme ayant convergé puisque précis. Cette métrique permet de vérifier que des erreurs de linéarisation ne sont pas venues affecter le suivi ou les mises à jour de l'amer. Les deux méthodes étant affectées par les problèmes de mise à jour, on pourra en déduire l'impact qu'a le suivi sur la convergence des amers. Les points sont initialisés à 500 mètres afin d'assurer que leur vraie position est bien incluse dans l'ellipsoïde initial. Néanmoins, à cette distance, parvenir à faire converger les points est difficile et nécessite de nombreuses observations pour ce faire. Ainsi, le calcul géométrique de la projection des amers évitant l'implication des jacobiennes est favorisé par ce contexte. De plus, la faible vitesse de la caméra tend également à mettre en difficulté le calcul par les jacobiennes puisque les changements qui interviennent entre deux images sont plus conséquents qu'à haute fréquence, et nécessite donc une fenêtre adéquate. Les résultats pour cette expérimentation sont fournis dans la table 3.1.

	EKF SLAM avec suivi par les jacobiennes	EKF SLAM avec la méthode géométrique
Amers initialisés	1275	973
Amers conservés	172	273

TABLE 3.1 – Comparatif des deux méthodes de suivi

On peut tout d'abord noter que le nombre d'initialisations est plus important sans la méthode de suivi géométrique. Cela paraît logique puisque beaucoup d'amers n'ont pas pu être suivis correctement, forçant l'algorithme à en initialiser d'autres. Plus de 25% d'amers en plus ont été initialisés avec les jacobiennes par rapport à notre approche géométrique. Il s'agit là d'un gain de temps considérable puisque le détecteur de Harris est en conséquence moins sollicité avec notre approche géométrique. Le nombre d'amers conservés confirme cette tendance avec 101 amers gardés en plus avec l'approche géométrique par rapport à la méthode des jacobiennes. Ces chiffres illustrent bien la responsabilité des erreurs de linéarisation dans le suivi. Le peu d'amers en sortie du SLAM avec les jacobiennes indique qu'une localisation de qualité sera difficile à atteindre. Environ un amer sur 13 a convergé avec cette technique alors que plus d'un amer sur 4 a été gardé grâce au calcul exact de la boîte englobante. Cependant, ces résultats sont à nuancer de par la difficulté à faire converger les points. L'approche par les jacobiennes est souvent capable de suivre les amers durant quelques itérations sans pour autant avoir assez d'informations pour préciser leur position. Aussi, comme cela a été indiqué précédemment le contexte (distance d'initialisation et vitesse de la caméra) fait que le calcul des boîtes englobantes est crucial, ce qui n'est pas forcément toujours le cas, comme cela sera montré dans la section 3.3 consacrée aux expérimentations.

La sous-section suivante va se concentrer sur les problèmes de linéarisation intervenant lors de l'étape de mise à jour.

3.2.2 Mise à jour via le filtre de Kalman

Les conséquences que peuvent avoir une mauvaise linéarisation lors du calcul de la mise à jour via le filtre de Kalman ont montré qu'il s'agit d'un point crucial à traiter. Bien que l'utilisation de la représentation cartésienne classique tend à exacerber son impact, la paramétrisation en profondeur inverse n'est pas exempte de ce problème [Parsley & Julier 2008]. Durant cette sous-section, nous nous pencherons sur comment réduire au maximum les erreurs de linéarisation lors de la phase de mise à jour. Ces travaux ont d'abord été publiés avec une méthode générale dans [Féraud *et al.* 2011a] avant d'être appliqués au SLAM monoculaire dans [Bresson *et al.* 2012b].

Les perturbations générées par la linéarisation d'une fonction peuvent être détectées dès lors que celle-ci peut s'exprimer sous la forme d'un rapport entre deux fonctions linéaires [Féraud *et al.* 2011b]. C'est bien le cas du modèle permettant la projection des amers dans le plan image. Ainsi, dès lors qu'une erreur de linéarisation est détectée, un correctif est appliqué afin de réduire l'influence de la linéarisation. Celui-ci intervient comme un facteur multiplicatif pour le gain de Kalman. C'est ce dernier qui quantifie l'impact que va avoir une observation sur l'estimée courante. En le modulant, il devient possible de réduire l'importance de la mise à jour et ainsi les erreurs de linéarisation.

Afin de savoir quand la mise à jour doit être corrigée, il est nécessaire de comprendre quand la linéarisation du système a échoué. Pour ce faire, il faut étudier le

résultat attendu en sortie de mise à jour. Nous avons déjà énoncé que les erreurs de linéarisation apparaissent plus fréquemment quand l'incertitude est grande, c'est-à-dire juste après l'initialisation d'un amer. Considérons un monde simplifié plan, sorte de vue de dessus. Dans ce monde, un amer est initialisé et affublé d'une incertitude conséquente sur le rayon de vue de l'observation. Le véhicule se déplace ensuite et une nouvelle observation arrive. Analysons les deux cas extrêmes, à savoir : une observation parfaite et une observation avec une incertitude infinie. L'impact de chacune de ces observations sur le point est différente. Une vue schématisée du problème est disponible en figure 3.11. Avec une observation parfaite, la mise à jour amènera l'amer directement sur celle-ci. En revanche, une observation avec une incertitude infinie n'aura pas d'impact sur la position de l'amer.

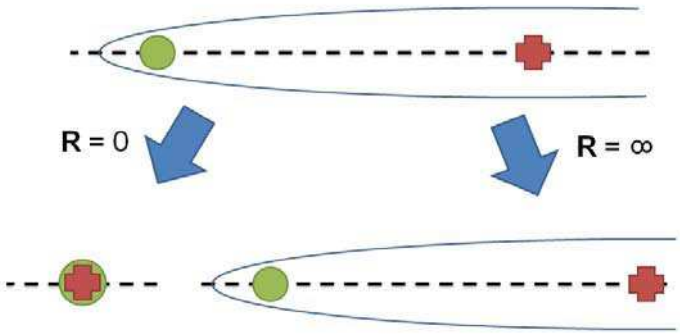


FIGURE 3.11 – Mise à jour d'un amer suivant différentes observations. La croix rouge représente l'estimation de l'amer. L'ellipse bleue est son incertitude. Le rond vert est l'observation. La matrice $\mathbf{R}$ est le bruit associé à l'observation.

À partir de ces constatations, nous pouvons définir un intervalle dans l'image où les amers mis à jour doivent être situés. En l'absence de nouvelles informations (observation avec incertitude infinie), la projection de l'amer doit rester à sa position. Avec la meilleure correction possible (observation avec incertitude nulle), la projection de l'amer mis à jour doit être située sur l'observation. Ainsi, il est possible de considérer qu'une erreur de linéarisation est survenue si la projection du point mis à jour n'est pas située entre la projection du point avant mise à jour et l'observation. Les possibilités de mise à jour sont montrées dans la figure 3.12.

Une fois l'échec de la linéarisation détecté, le rôle du facteur correctif du gain de Kalman est d'amener la mise à jour de la projection de l'amer sur l'observation. Le bruit d'observation étant relativement faible, l'observation est une information fiable dès lors qu'une erreur de linéarisation est constatée. Cela se traduit par la relation suivante :

$$\mathbf{z}_k = h(\hat{\mathbf{x}}_{k|k}) \tag{3.24}$$

avec $\mathbf{z}_k = \begin{pmatrix} z_{u_k} & z_{v_k} \end{pmatrix}^{\mathrm{T}}$.

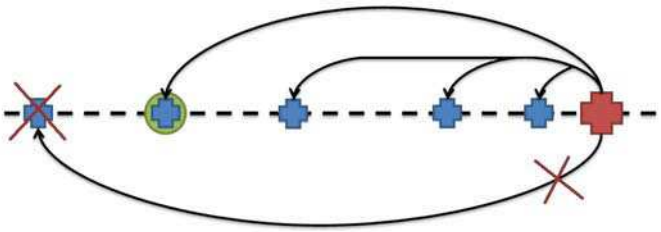


FIGURE 3.12 – Les différentes possibilités de mise à jour. La croix rouge représente l'estimation de l'amer. Le rond vert est l'observation. Les croix bleues montrent différentes possibilités de mise à jour et lesquelles sont rejetées.

Afin de rendre cette équation vraie, nous définissions $\mathbf{\Omega}_k$ le gain de Kalman corrigé par le facteur correctif r :

$$\mathbf{\Omega}_k = r \cdot \mathbf{K}_k \tag{3.25}$$

avec $\mathbf{K}_k$ le gain de Kalman.

Il est bon de montrer que l'introduction de ce correctif ne brise pas le rapport entre l'incertitude de l'observation et celle affectant la projection de l'estimée dans l'image :

$$\begin{aligned} \mathbf{\Omega}_k &= r \cdot \mathbf{K}_k \\ &= r \cdot \mathbf{P}_k \mathbf{H}_k^{\mathrm{T}} \ \ \mathbf{H}_k \mathbf{P}_k \mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k \big)^{\ 1} \\ &= \mathbf{P}_k \mathbf{H}_k^{\mathrm{T}} \left(\mathbf{H}_k \frac{\mathbf{P}_k}{r} \mathbf{H}_k^{\mathrm{T}} + \frac{\mathbf{R}_k}{r} \right)^{\ 1} \end{aligned} \tag{3.26}$$

Les deux incertitudes sont donc affectées de la même façon. De plus, il est possible de déduire l'intervalle de valeurs de r. En effet, $\mathbf{R}_k$ est le meilleur a priori disponible pour le bruit d'observation. Cela signifie que celui-ci ne peut pas être plus petit que la valeur renseignée. Ce bruit, après application du correctif r, ne peut donc pas être réduit. On peut donc en déduire que r doit être inférieur ou égal à 1 pour ne pas altérer $\mathbf{R}_k$ en ce sens. Une correction supérieure à 1 signifierait que la projection de l'amer mis à jour est déjà dans l'intervalle autorisé de mise à jour et donc que la correction ne devrait pas être considérée. L'influence de r sur $\mathbf{P}_k$ permet de comprendre que r doit être supérieur à 0. $\mathbf{P}_k$ est une matrice de covariance, elle est donc semi-définie positive par nature. Une valeur négative de r changerait cela et empêcherait son utilisation dans le filtre de Kalman. r est donc appliqué uniquement s'il appartient à l'intervalle]0 1].

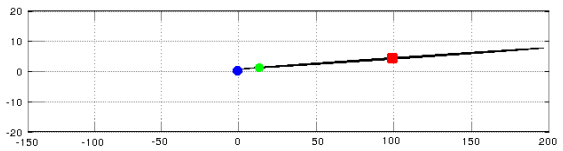
En considérant l'équation de mise à jour de Kalman pour l'état (2.26) ainsi que la fonction d'observation (2.27), formuler l'observation comme la projection de l'amer mis à jour (voir équation (3.24)) se fait comme suit :

$$\begin{cases} z_{u_k} &= \dfrac{\mathbf{F}_1\mathbf{R}_{cv}^{\mathrm{T}}(\mathbf{R}_{vm}^{\mathrm{T}}(\mathbf{p}_m + \mathbf{\Omega}_k\mathbf{\Delta}_k \quad \mathbf{t}_{vm}) \quad \mathbf{t}_{cv})}{\mathbf{F}_3\mathbf{R}_{cv}^{\mathrm{T}}(\mathbf{R}_{vm}^{\mathrm{T}}(\mathbf{p}_m + \mathbf{\Omega}_k\mathbf{\Delta}_k \quad \mathbf{t}_{vm}) \quad \mathbf{t}_{cv})} \\ z_{v_k} &= \dfrac{\mathbf{F}_2\mathbf{R}_{cv}^{\mathrm{T}}(\mathbf{R}_{vm}^{\mathrm{T}}(\mathbf{p}_m + \mathbf{\Omega}_k\mathbf{\Delta}_k \quad \mathbf{t}_{vm}) \quad \mathbf{t}_{cv})}{\mathbf{F}_3\mathbf{R}_{cv}^{\mathrm{T}}(\mathbf{R}_{vm}^{\mathrm{T}}(\mathbf{p}_m + \mathbf{\Omega}_k\mathbf{\Delta}_k \quad \mathbf{t}_{vm}) \quad \mathbf{t}_{cv})} \end{cases} \tag{3.27}$$

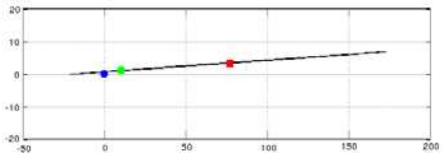
De là, il est possible d'extraire le calcul du correctif en sortant $\mathbf{\Omega}_k$ de l'équation :

$$\begin{cases} r_u &= \dfrac{(z_{u_k} \quad u_{i_{k\ 1}})\mathbf{F}_3\mathbf{R}_{cv}^{\mathrm{T}}(\mathbf{R}_{vm}^{\mathrm{T}}(\mathbf{p}_m \quad \mathbf{t}_{vm}) \quad \mathbf{t}_{cv})}{(\mathbf{F}_1 \quad z_{u_k}\mathbf{F}_3)\mathbf{R}_{cv}^{\mathrm{T}}\mathbf{R}_{vm}^{\mathrm{T}}\mathbf{K}_k\mathbf{\Delta}_k} \\ r_v &= \dfrac{(z_{v_k} \quad v_{i_{k\ 1}})\mathbf{F}_3\mathbf{R}_{cv}^{\mathrm{T}}(\mathbf{R}_{vm}^{\mathrm{T}}(\mathbf{p}_m \quad \mathbf{t}_{vm}) \quad \mathbf{t}_{cv})}{(\mathbf{F}_2 \quad z_{u_k}\mathbf{F}_3)\mathbf{R}_{cv}^{\mathrm{T}}\mathbf{R}_{vm}^{\mathrm{T}}\mathbf{K}_k\mathbf{\Delta}_k} \end{cases} \tag{3.28}$$

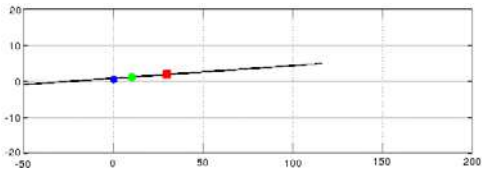
Le facteur correctif le plus faible entre r_u et r_v sera conservé. Cela évite de conduire à des estimations trop confiantes en sur-estimant le correctif à appliquer. Grâce à ce dernier, l'impact des erreurs de linéarisation sera grandement réduit. L'exemple utilisé précédemment (figure 3.5), montrait que le point mis à jour pouvait passer derrière le véhicule. Reprenons ce même exemple, mais cette fois-ci avec l'application de la correction présentée dans cette sous-section. Les résultats des mises à jour sont exposés en figure 3.13.



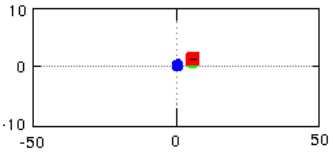
(a) Initialisation de l'amer à 100 mètres sur le rayon de vue



(b) 1ère mise à jour correcte



(c) 2ème mise à jour correcte

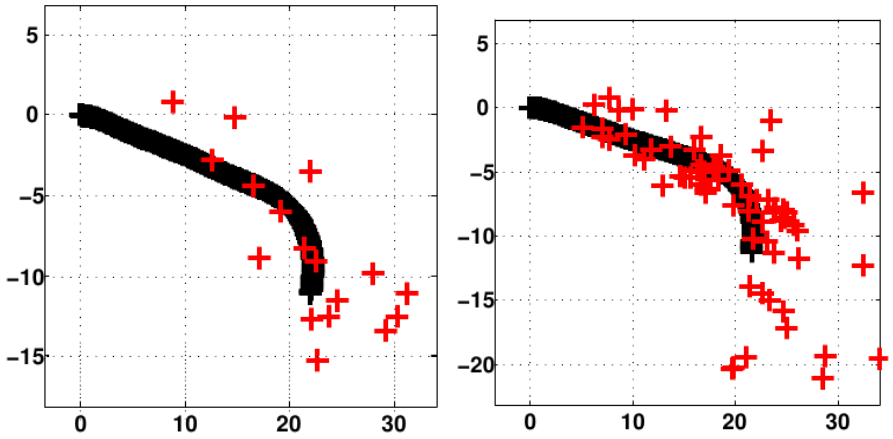


(d) 10ème mise à jour correcte

FIGURE 3.13 – Mise à jour avec application du facteur correctif. Vue de dessus des mises à jour d'un amer. Le cercle bleu est la position du véhicule. Le carré rouge est la position estimée de l'amer. Le cercle vert est la vraie position de cet amer. L'ellipse noire est l'incertitude associée à l'amer rouge.

Le comportement incohérent de la mise à jour a ici complètement disparu. Le facteur correctif permet d'éviter le passage de l'amer mis à jour à des positions normalement impossibles. On peut remarquer sur cet exemple que l'amer converge progressivement vers sa vraie valeur. Celle-ci est atteinte après 10 mises à jour sans erreur de linéarisation.

Comme précédemment, et afin de quantifier l'apport du correctif du gain de Kalman introduit ici, une expérimentation a été réalisée. Il s'agit d'une courte trajectoire de 30 mètres en environnement urbain statique. Celle-ci a été parcourue par un véhicule, équipé d'une caméra fonctionnant à 10 Hz et roulant à une vitesse de 1 mètre par seconde. La trajectoire a été jouée deux fois : la première fois par un SLAM monoculaire sans le correctif introduit ici et la deuxième fois avec le même processus SLAM mais avec l'intégration de la correction du gain de Kalman. Les localisations calculées ainsi que les amers conservés dans la carte à la fin sont visibles en figure 3.14. Les deux cas testés (application ou pas du correctif) sont exposés.



(a) Trajectoire sans correctif (b) Trajectoire avec correctif

FIGURE 3.14 – Trajectoire suivie avec application ou non du correctif. Les croix rouges sont les amers conservés (incertitude inférieure à un seuil).

Beaucoup plus d'amers ont été conservés avec l'application du correctif. Les deux indices utilisés dans l'expérimentation précédente sont toujours de bons indicateurs de la qualité de notre solution. En plus du nombre d'amers initialisés et conservés, nous quantifions les divergences observées. Nous caractérisons une divergence comme le fait qu'un amer mis à jour n'intègre pas l'observation dans son incertitude. Ces résultats sont disponibles dans la table 3.2.

Quasiment 3 fois plus d'amers ont dû être initialisés en l'absence du correctif du gain. Cela s'explique par le simple fait que les amers sont plus difficiles à observer dès lors que des mises à jour incorrectes ont lieu. Comme pour le suivi, le correctif fait gagner un temps considérable en sollicitant moins le détecteur de points caractéristiques qui est coûteux. On peut noter que seulement 32 amers sur 384 (moins de 9%) ont convergé à cause des erreurs de linéarisation. Avec le gain de Kalman cor-

	EKF SLAM sans correctif	EKF SLAM avec correctif
Amers initialisés	384	134
Amers conservés	32	75
Divergences	439	0

TABLE 3.2 – Comparatif des deux méthodes pour les mises à jour

rigé, 75 amers sur 134 (plus de 55%) ont été conservés, ce qui est encourageant pour la qualité globale de la localisation. Du côté des divergences, 439 ont été constatées dans l'EKF de base. Le fait qu'il y ait plus de divergences que de points initialisés est normal : un point qui diverge est gardé tant que des associations sont trouvées. Le correctif permet d'éviter ces divergences.

Différentes innovations, comblant les faiblesses initiales du filtre de Kalman étendu, ont été proposées tout au long de cette section. En choisissant d'utiliser une représentation optimale, nous évitons les surcharges en temps de calcul et en mémoire. Les résultats préliminaires exposés pour chaque aspect amélioré du filtre semblent confirmer que nos palliatifs sont efficaces pour contrer les erreurs de linéarisation. Des expérimentations de plus grande envergure sont néanmoins nécessaires pour pleinement valider notre SLAM monoculaire avec l'ensemble des améliorations afin de pouvoir ensuite l'utiliser dans le cadre d'applications à plusieurs véhicules.

3.3 Expérimentations réelles et simulées

Afin de donner une vue globale des résultats de notre algorithme, des simulations et expérimentations ont été conduites. L'objectif est tout d'abord de tester la robustesse de notre algorithme en le confrontant à des conditions extérieures défavorables ou à des capteurs fonctionnant à faible cadence. Ensuite, la qualité de la localisation calculée doit être éprouvée afin de garantir que notre méthode puisse offrir des cartes suffisamment précises malgré leur faible densité. Enfin, il est intéressant de pouvoir comparer les résultats de localisation obtenus avec une approche EKF SLAM monoculaire faisant référence dans la communauté, à savoir celle de Civera et al. dans [Civera *et al.* 2010]. Pour ce faire, le même jeu de données utilisé dans [Civera *et al.* 2010] sera appliqué à notre algorithme. Il sera ainsi possible de comparer la qualité de la localisation ainsi que le temps de calcul nécessaire.

3.3.1 Trajectoires à faible cadence caméra

Les expérimentations réelles qui vont être présentées dans cette sous-section ont toutes été réalisées sur la plate-forme expérimentale urbaine PAVIN. Il s'agit d'un site spécialement construit pour tester les algorithmes de robotique mobile dans des conditions réelles mais contrôlées. Ce site est composé de passages piétons, trottoirs, routes, ronds-points, immeubles, feux, panneaux... Une vue de PAVIN est disponible en figure 3.15 et un aperçu de l'environnement est donné en figure 3.16.

Deux expérimentations réelles ont été menées pour tester la robustesse de l'algorithme développé. Pour cela, dans ces deux expérimentations, une caméra fonctionnant à une fréquence peu élevée (moins de 4 Hz) a été utilisée. Avec une vitesse d'acquisition des images faible, les erreurs de linéarisation sont plus importantes. En effet, chaque nouvelle observation est située loin de la précédente et rend donc encore plus crucial le calcul de la fenêtre englobante. De plus, la mise à jour de Kalman est sujette à des erreurs de linéarisation plus importante à cause de l'écart entre les observations et les estimations. Afin de pousser l'algorithme dans ces derniers retranchements, uniquement 10 amers par image sont suivis (de nouveaux sont initialisés dès lors que moins de 5 sont visibles). Les conditions extérieures affectent également les images rendant ainsi le suivi plus difficile. Un fort ensoleillement quasiment permanent est à noter. En plus de la gêne occasionnée en étant face au soleil, un artefact permanent est visible sur toutes les images. D'importantes distorsions sont également visibles sur les images. Celles-ci n'ont pas été prises en compte dans le SLAM monoculaire construit afin de tester la robustesse de l'algorithme de suivi. Quelques unes de ces conditions sont montrées en figure 3.16. En revanche, les points ont été initialisés à 100 mètres afin d'être plus réaliste par rapport au contexte de l'expérimentation. Cette configuration permet de favoriser la convergence des amers.

FIGURE 3.15 – Plate-forme expérimentale PAVIN

Toutes les expérimentations de ce manuscrit ont été conduites avec des VIPALab. Il s'agit de véhicules électriques prévus pour faire du transport de personnes. Une photographie d'un de ces véhicules est disponible en figure 3.17. Celui-ci était ici conduit manuellement dans PAVIN à environ 2 mètres par seconde. Du côté du matériel, un ordinateur portable embarquant un processeur Intel Core i5 cadencé à 2,40 GHz a été utilisé.

La première trajectoire est longue de 170 mètres. Cela correspond à un tour de PAVIN. Il y a donc de grands virages ainsi que d'importantes lignes droites. Les capteurs sont toujours les mêmes : une caméra Marlin F-1318, un odomètre et un capteur d'angle de braquage. La particularité de cette séquence est que la caméra

(a) (b)

(c) (d)

FIGURE 3.16 – Aperçu de quelques images de l'environnement PAVIN. Celles-ci sont issues de la trajectoire présentée ici.

FIGURE 3.17 – Véhicule électrique VIPALab

opère à faible fréquence, à savoir 3,75 Hz. Un GPS RTK, permettant d'obtenir une vérité terrain avec une précision centimétrique, est également embarqué sur le véhicule. Celui-ci est uniquement utilisé afin de pouvoir caractériser nos résultats de localisation par rapport à une vérité terrain. À titre de comparaison, le même algorithme de SLAM monoculaire mais sans le correctif du gain de Kalman et le calcul géométrique de la fenêtre a été appliqué à cette trajectoire. Les résultats obtenus sont visibles en figure 3.18.

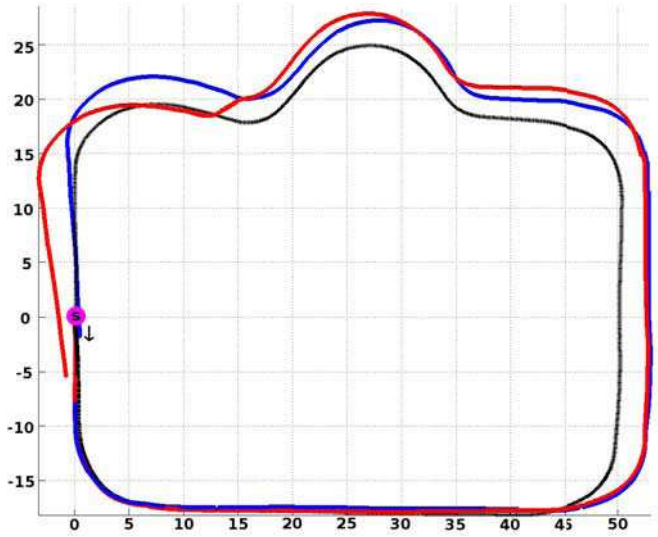


FIGURE 3.18 – Trajectoires accomplies avec une caméra à 3,75 Hz. En noir, la vérité terrain (GPS RTK). En rouge, l'EKF SLAM sans les correctifs. En bleu, la trajectoire calculée par notre algorithme. La pastille rose indique le point de départ de la trajectoire.

On peut constater que la trajectoire calculée par notre algorithme est plus juste que celle de l'approche sans correctifs. Bien que la divergence du SLAM sans correctifs ne soit pas flagrante, on peut voir que l'orientation du véhicule n'est pas correctement estimée à la fin de la trajectoire et conduirait certainement à des dérives plus importantes si la trajectoire continuait. Ces bons résultats du SLAM sans correctifs s'expliquent par le fait que notre modèle d'évolution (présenté en sous-section 2.3.1), alimenté par des données odométriques de qualité, permet déjà une localisation assez précise, limitant ainsi l'effet que peut avoir le suivi des amers via la caméra. De plus, les vérifications appliquées pour détecter les erreurs de linéarisation empêchent la plupart des divergences. Dans l'ensemble, les trajectoires obtenues sont proches de la vérité. Bien entendu, les résultats issus du SLAM sont difficilement comparables à la vérité terrain puisque notre algorithme se localise par rapport à une carte relative. Celle-ci tend en plus à diverger (tout comme la localisation) et cela ne peut être résolu sauf dans des conditions particulières. Néanmoins, le nombre d'amers initialisés et gardés est toujours intéressant car il renseigne sur la qualité

de la carte construite. Ces résultats sont disponibles dans la table 3.3.

	SLAM sans correctifs	SLAM avec correctifs
Amers initialisés	472	408
Amers conservés	267	354

TABLE 3.3 – Amers utilisés sur une trajectoire réelle avec une caméra à 3,75 Hz

Ces résultats confirment ceux obtenus précédemment, même si les ratios entre initialisés et conservés sont plus meilleurs pour les deux approches. La distance d'initialisation plus appropriée a joué un rôle important dans l'amélioration globale de ces rations. Le SLAM sans correctifs a dû initialiser plus d'amers durant la trajectoire. En effet, les conditions étant difficiles, il n'était pas en mesure de les suivre sur de longues périodes de temps. De manière similaire, presque 100 amers de plus ont convergé avec l'EKF SLAM utilisant les correctifs. Le taux de convergence est de moins de 57% pour l'algorithme sans correctifs contre presque 87% pour le SLAM présenté ici.

La taille de la carte après ces 170 mètres est un point important. Dans le cadre d'applications multivéhicules, ces cartes vont être régulièrement échangées. En regardant la place que prend une carte, il est possible de se rendre compte si notre approche est adaptée à un contexte multirobot. En considérant que les points sont encodés en double précision et que l'on échange également les matrices de covariance de ceux-ci ainsi que leur patch, on peut calculer que pour 170 mètres, la carte ne nécessite que 77 ko. Cela rend notre approche très attractive pour le multivéhicule car une telle carte peut facilement être stockée ou échangée au vu des technologies de communication actuelles. En comparaison, l'utilisation de la paramétrisation en profondeur inverse [Montiel *et al.* 2006][Civera *et al.* 2010] fait monter la taille de la carte à 162 ko.

Pour cette deuxième expérimentation, nous avons utilisé la même trajectoire que précédemment mais avec une caméra ralentie à moins de 2 Hz. Avec cette configuration, les erreurs de linéarisation sont exacerbées. Encore une fois, une approche identique mais sans les correctifs sera comparée à notre EKF SLAM. Les trajectoires accomplies sont visibles en figure 3.19.

La trajectoire obtenue par le SLAM avec correctifs est très proche de la vérité terrain et également de celle calculée avec la caméra à 3,75 Hz. Cela confirme bien la robustesse de notre approche puisque même avec une vitesse d'acquisition plus faible, l'algorithme est capable d'atteindre des précisions similaires. En revanche, cette fois-ci l'approche sans correctifs a rapidement divergé à cause d'erreurs de linéarisation. Juste après le premier virage (voir zone cerclée en vert), l'orientation est faussée et un saut dans la position peut être noté. Néanmoins, après cela, la plupart des mauvaises linéarisations sont évitées bien que la localisation ait été irrémédiablement affectée au début de l'expérimentation.

Les conditions extérieures ont bien affecté l'algorithme et rendu le suivi plus compliqué. En effet, avec d'importantes réflexions sur les images (causées par un

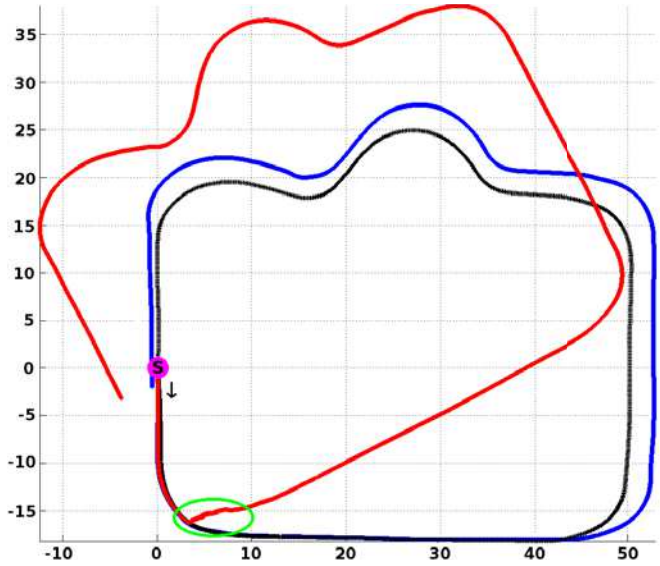


FIGURE 3.19 – Trajectoires accomplies avec une caméra à moins de 2 Hz. En noir, la vérité terrain (GPS RTK). En rouge, l'EKF SLAM sans correctifs. En bleu, la trajectoire calculée par notre algorithme. La pastille rose indique le point de départ de la trajectoire.

fort ensoleillement), il était courant qu'aucune observation pertinente ne soit trouvée pour certains amers. Dans ce cas, aucune mise à jour n'était accomplie et une nouvelle tentative de suivi pour ces amers était faite sur l'image suivante. La taille de la fenêtre de recherche augmentait et le suivi devenait seulement possible grâce à l'approche géométrique introduite ici.

Néanmoins, il est important de vérifier que la carte de faible densité construite est suffisante pour aider à la localisation d'autres véhicules. Afin de démontrer ce point, la prochaine expérimentation consistera en la localisation d'un véhicule à l'aide d'une carte prélablement construire par un autre véhicule.

3.3.2 Utilisation d'une carte préalablement construite

Dans l'expérimentation présentée dans cette sous-section un premier véhicule est chargé de construire une carte qui servira de base à un deuxième véhicule qui suivra ensuite à peu près la même trajectoire. Ce deuxième robot, conduit manuellement, devra se localiser à l'intérieur de la carte précédemment construite. Un simulateur a été utilisé afin de faciliter le fait de pouvoir effectuer des trajectoires similaires. La mesure de la distance entre les chemins réellement suivis par les deux véhicules par rapport à celle estimée va donner une indication de la qualité de localisation qui est envisageable avec une carte peu fournie en amers comme la nôtre.

Le simulateur utilisé ici (Cobaye) possède une physique et des capteurs virtuels réalistes, retranscrivant le comportement réel des véhicules électriques destinés au

transport de personnes. Les environnements intégrés à Cobaye ont été cartographiés à partir des lieux réels par une société extérieure spécialisée dans le domaine. Dans le cadre des résultats exposés ici, l'environnement utilisé est le centre-ville de Clermont-Ferrand : la place de Jaude. Cette place est composée de divers immeubles, monuments et routes. Des plots, des lampadaires ou encore des bancs sont également présents. L'avantage principal de cet environnement est le fait qu'il soit suffisamment ouvert pour permettre des trajectoires différentes de l'environnement PAVIN. Quelques images de la place de Jaude dans le simulateur sont disponibles en figure 3.20.

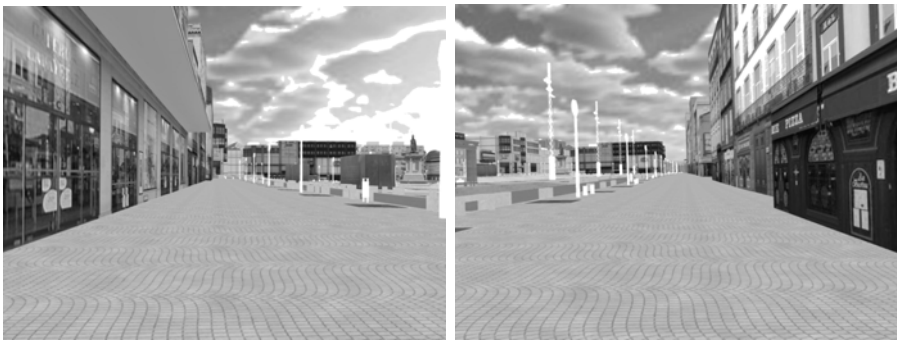
(a) (b)

(c) (d)

FIGURE 3.20 – Environnement place de Jaude sous le simulateur

Le simulateur sert uniquement à générer des données capteur. Celui-ci n'est en charge d'aucun traitement et se contente de transmettre les trames contenant les données au calculateur prévu pour les traiter comme le ferait un capteur réel. Un ordinateur indépendant de la génération des données est utilisé afin d'exécuter l'algorithme de SLAM monoculaire, comme c'est le cas avec de vrais véhicules. Ici, les interactions avec le simulateur se limitent à la réception des données issues des capteurs virtuels.

Le véhicule employé pour tester le SLAM monoculaire est équipé de manière similaire aux expérimentations précédentes, à savoir avec une caméra et un odomètre. La caméra opère à 10 images par seconde. Quelques images en sortie du capteur peuvent être vues en figure 3.21. Enfin, le véhicule se déplace à environ 2 mètres par seconde sur l'ensemble de ces deux trajectoires d'approximativement 170



(a) (b)

FIGURE 3.21 – Quelques exemples d'images issues de la caméra virtuelle

mètres chacune.

La méthode de localisation du second véhicule diffère quelque peu du SLAM présenté jusque là. Ici, il n'est pas question de cartographie mais uniquement de localisation. Ainsi, l'algorithme se contente de projeter les amers proches de la carte fournie dans l'image courante et de les associer avec des observations qui serviront à mettre à jour la pose du véhicule. Les amers ne sont pas améliorés. La localisation s'appuie également sur un filtre de Kalman étendu et le calcul de la fenêtre englobante est toujours fait avec l'approche décrite dans ce chapitre.

Lors de son passage, le premier véhicule a cartographié environ 600 amers. Ce sont ces points qui sont ensuite utilisés par le deuxième véhicule pour se localiser. Nous avons initialisé plus de points par image que dans l'expérimentation précédente afin de pouvoir assurer que le deuxième véhicule puisse retrouver des amers facilement. Les trajectoires suivies sont visibles sur la figure 3.22.

La trajectoire calculée par notre approche demeure assez proche de la vérité donnée par des GPS RTK. La dérive est donc majoritairement évitée même si la distance parcourue reste faible. Les deux trajectoires accomplies sont quasiment les mêmes. On peut noter que la localisation du deuxième véhicule dans la carte du premier est assez précise. Quelques écarts peuvent être observés dans les virages serrés là où moins de points sont disponibles. La différence entre les écarts de position réels et les écarts mesurés entre le SLAM et le rejeu est disponible en figure 3.23.

On peut observer que l'erreur moyenne de localisation se situe autour de 10 centimètres. Cela est tout à fait acceptable pour des applications impliquant une conduite autonome. Néanmoins, quelques pics sont visibles vers la fin de la trajectoire. Il s'agit de la même erreur de localisation constatée précédemment. L'absence de points durant les virages est critique puisqu'elle conditionne la bonne estimation du cap. On peut voir qu'avec une légère erreur d'orientation, la position du véhicule peut être mal estimée. Intégrer plus de points durant les virages pourrait être un moyen de résoudre ce problème. Cependant, ces résultats sont encourageants pour des applications avec plusieurs véhicules.

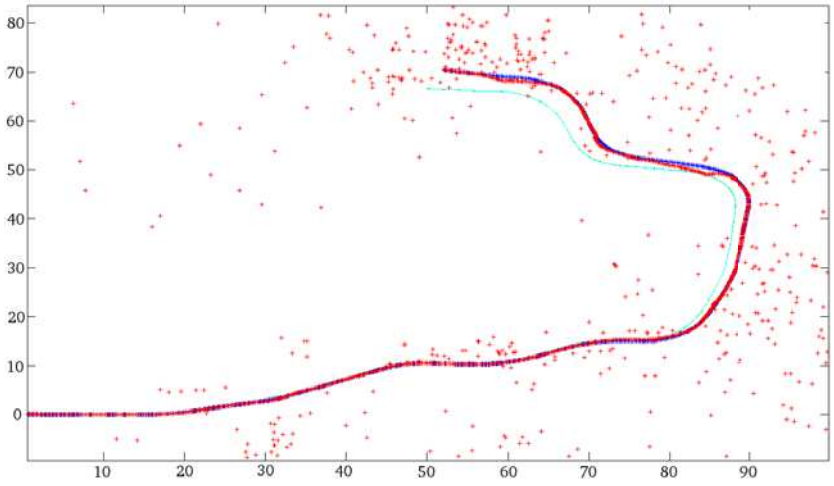


FIGURE 3.22 – Trajectoires suivies par les deux véhicules. En vert, la vérité terrain du premier véhicule. En cyan (quasiment superposé à la courbe verte), la vérité terrain du second véhicule. En bleu, la trajectoire calculée par le premier véhicule. En rouge, celle du deuxième véhicule utilisant la carte du premier. Les croix rouges sont les amers cartographiés par le premier véhicule et suivis par le second.

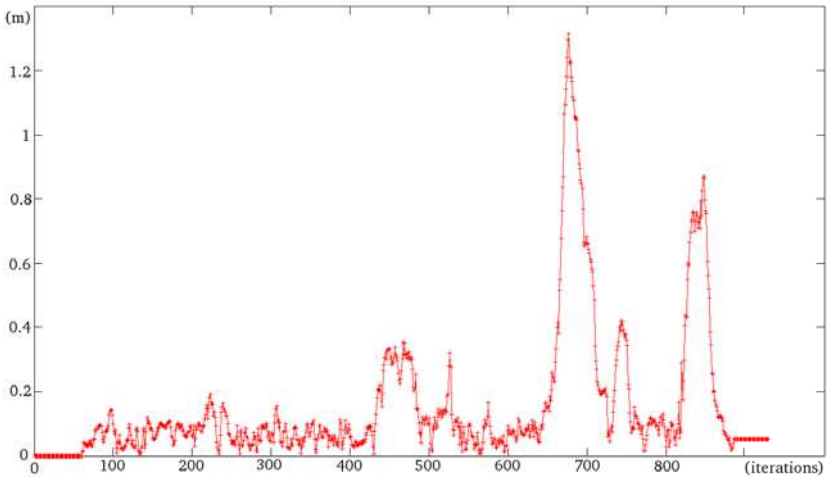


FIGURE 3.23 – Erreur de localisation du second véhicule dans la carte du premier

3.3.3 Comparaison avec une approche existante

L'objectif de cette sous-section est de comparer notre algorithme à celui proposé par Civera et al. dans [Civera *et al.* 2010] et utilisant un EKF SLAM monoculaire avec la paramétrisation en profondeur inverse. Cette approche est considérée comme la référence de l'état de l'art car la représentation en profondeur inverse permet d'éviter la majorité des erreurs de linéarisation. En plus de la répresentation en pro-

fondeur inverse,une méthode d'association de données plus complexe est utilisée : le 1-point RANSAC. Cet algorithme permet de guider le processus d'association de données en contraignant le filtre à faire une première mise à jour avec l'observation permettant de garder le plus de correspondances après coup. Afin de pouvoir comparer de manière adéquate notre algorithme, nous avons utilisé le jeu de données provenant de la base de données RAWSEEDS (http ://www.rawseeds.org) qui est également employé dans [Civera *et al.* 2010].

La trajectoire concernée a été enregistrée sur le campus de Milan. De nombreux bâtiments, rues, bancs, grillages, etc. sont présents ainsi que des passages dans des zones plus ouvertes avec des arbres, de l'herbe ou encore du gravier. Le robot employé (Robocom) est de taille réduite et a été développé afin de pouvoir à la fois effectuer des trajectoires en intérieur et en extérieur. Une photographie de ce dernier est disponible en figure 3.24.

FIGURE 3.24 – Robot utilisé dans cette expérimentation

Le Robocom évolue à faible vitesse (moins de 1 mètre par seconde). Celui-ci est équipé d'une caméra fournissant des images 320x240 à 30 Hz ainsi que d'un odomètre. Les conditions de l'expérimentation sont illustrées en figure 3.25 avec quelques images de l'environnement. Afin de rester comparable avec [Civera *et al.* 2010], le même nombre de points est suivi par image, à savoir 25.

Cette expérimentation est aussi l'occasion de confronter notre algorithme à une trajectoire plus longue. Celle-ci mesure 1360 mètres environ. La figure 3.26 montre la trajectoire estimée lorsqu'uniquement les données proprioceptives sont utilisées (en rouge). La courbe bleue correspond à la trajectoire donnée par le GPS, qui sert ici uniquement de vérité terrain.

(a) (b)

(c) (d)

FIGURE 3.25 – Quelques images tirées du jeu de données de Milan

On peut constater une importante dérive odométrique tout au long de la trajectoire. Il est important de noter que ces résultats sont identiques à ceux obtenus par Civera et al. et que donc uniquement l'implication de la vision va être mesurée. La figure 3.27 montre la trajectoire calculée par notre algorithme de SLAM par rapport à la vérité terrain ainsi que celle obtenue par Civera et al. et publiée dans [Civera *et al.* 2010].

On peut noter que la dérive odométrique est en grande partie corrigée. Une partie de l'erreur angulaire n'a cependant pas pu être corrigée dans la seconde moitié de la trajectoire créant ainsi ce décalage par rapport au GPS. La mesure de l'écart entre la vérité et la position calculée par notre algorithme de SLAM donne une bonne indication de la qualité de la localisation. Sur l'intégralité de la trajectoire, une dérive de 0,84% en moyenne a été mesurée. En comparaison, l'approche de Civera et al. atteint une dérive de 0,7%. Les résultats sont donc relativement proches. L'utilisation d'un algorithme d'association de données (le 1-point RANSAC) plus performant que celui que mettons en oeuvre peut sans doute expliquer la légère différence dans les résultats. En effet, une technique permettant de trouver des associations globalement cohérentes est utilisée par Civera et al.. Celle-ci permet d'éviter les fausses associations qui peuvent affecter la qualité de la position du robot. L'évolution de l'écart entre la vérité terrain et la position calculée par notre EKF SLAM est visible en figure 3.28.

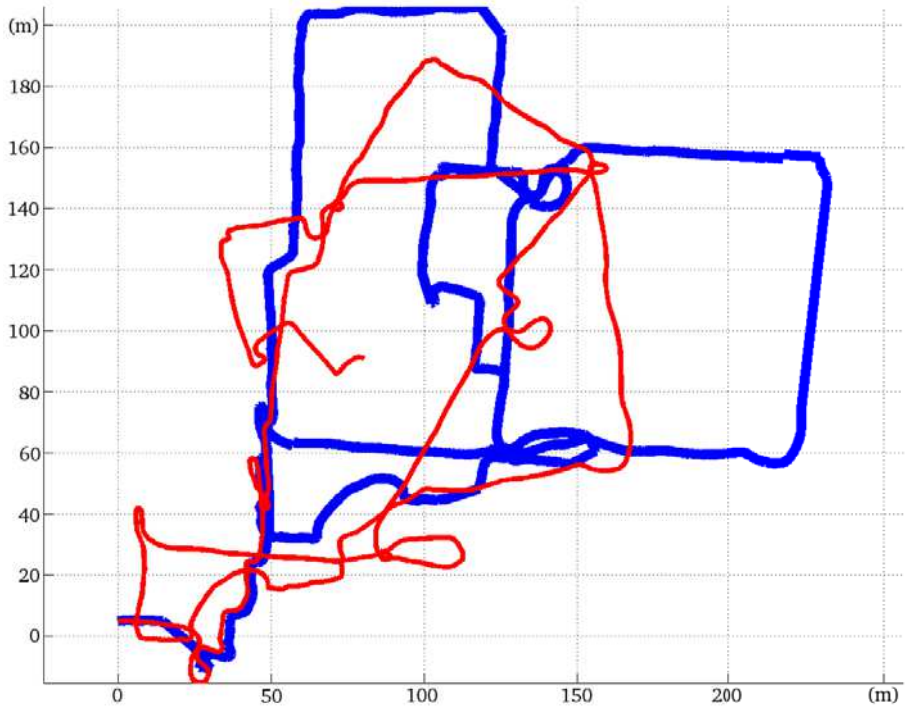
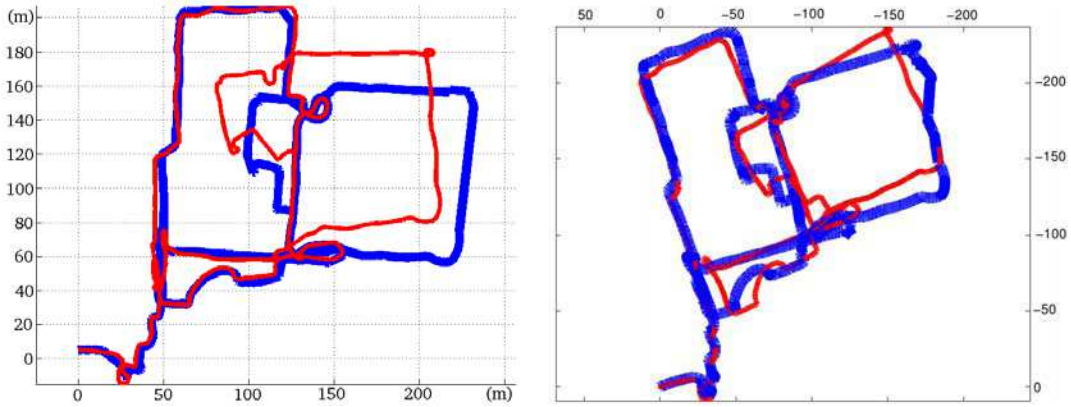


FIGURE 3.26 – Odométrie et GPS de la trajectoire de Milan. La courbe bleue est le GPS (vérité terrain) et la rouge le résultat de la navigation à l'estime sans la caméra.



(a) Résultats de localisation de notre algorithme

(b) Résultat de localisation de [Civera *et al.* 2010]

FIGURE 3.27 – Résultats de localisation avec le SLAM monoculaire. Dans les deux cas, la courbe bleue est la vérité terrain et la courbe rouge, la localisation calculée par l'algorithme de SLAM.

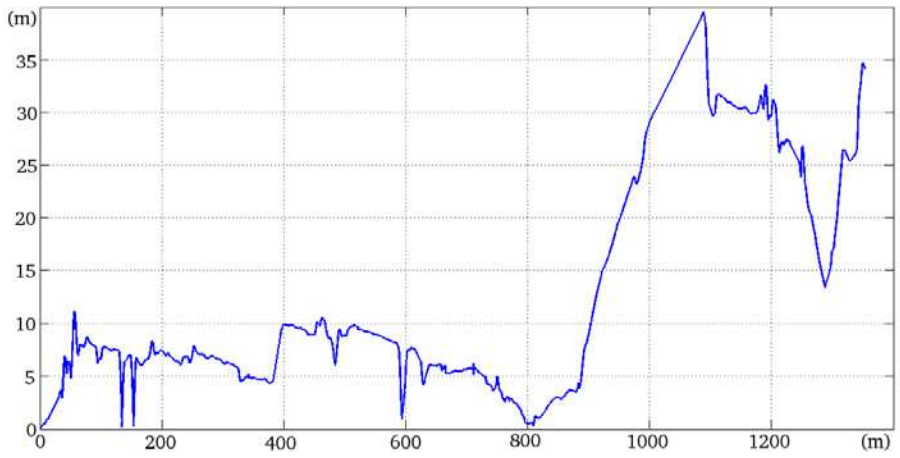
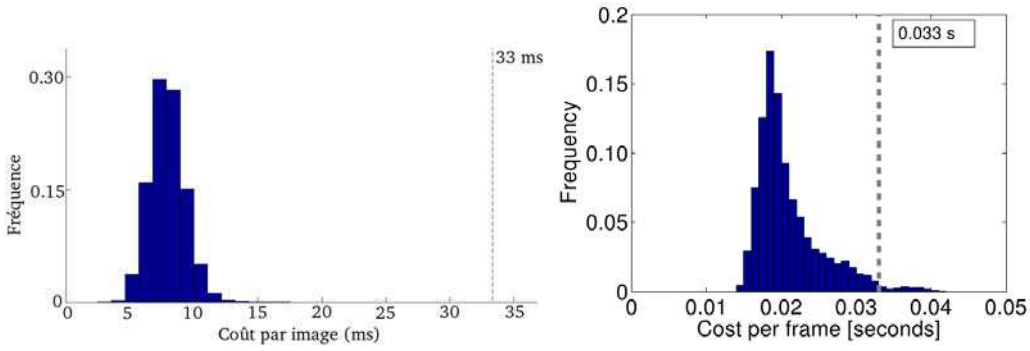


FIGURE 3.28 – Écart entre la position calculée et la vérité terrain en fonction de la distance parcourue

Concernant le temps de calcul, nous avons mesuré le temps de traitement par image. Les résultats sont disponibles en figure 3.29. Ils montrent le temps pris par notre algorithme et par celui de Civera et al.. La barre verticale indique le temps réel pour la caméra opérant à 30 Hz.



(a) Temps de calcul de notre algorithme (b) Temps de calcul de [Civera *et al.* 2010]

FIGURE 3.29 – Temps de calcul pris par image, pour chaque algorithme, représenté sous la forme de la fréquence du temps de traitement. La barre verticale indique la fréquence caméra, à savoir la référence à ne pas dépasser pour être temps réel.

On peut s'apercevoir que notre algorithme prend en moyenne 8 ms par image. Celui de Civera et al. s'approche davantage des 17 ms par image. Cela s'explique très simplement par le fait que nous n'avons pas de processus d'association de données haut niveau contrairement à leur approche. Néanmoins, au vu des résultats, cela pourrait être un ajout intéressant afin de rendre notre algorithme plus ro-

buste. Néanmoins, les résultats obtenus sont très proches de ceux exposés dans [Civera *et al.* 2010], ce qui constitue une validation importante pour notre algorithme.

3.4 Discussion

Au cours de ce chapitre, une solution pour le SLAM monoculaire a été proposée. Celle-ci repose sur une représentation cartésienne classique de la carte et de la pose du véhicule. Bien qu'elle soit optimale car n'induisant pas de surcoût calculatoire ou de mémoire, la représentation cartésienne est assez sensible à la linéarisation des modèles. Dans le cas d'un système monoculaire, cet aspect est exacerbé par l'absence d'information de profondeur pour les amers. Néanmoins, en privilégiant des approches géométriques traduisant la réalité physique de l'état de la carte, la majorité de ces linéarisations sont évitées. Quand ce n'est pas le cas, des palliatifs permettant de réduire l'effet de celles-ci sont employés. Les résultats obtenus par diverses simulations ou expérimentations montrent la viabilité ainsi que la robustesse de notre approche.

L'occupation mémoire de l'algorithme est très faible. La combinaison entre la représentation optimale et le filtre de Kalman étendu est idéale pour produire des cartes peu denses. Ces dernières restent cependant suffisantes pour des applications de guidage automatique. Concernant le temps de calcul, tous les résultats présentés ici ont été obtenus en temps réel avec un ordinateur portable équipé d'un processeur Intel Core i5 cadencé à 2,40 GHz. Le temps de traitement des données odométriques est extrêmement faible de l'ordre de quelques microsecondes. La partie vision est en revanche plus consommatrice de ressources. En effet, 25 millisecondes sont en moyenne nécessaires pour des images de 1024x768 et 8 millisecondes pour une résolution de 320x240. Cela inclut le suivi des points, leur mise à jour ainsi que l'initialisation de nouveaux amers via un détecteur de Harris si cela est requis. La majorité du temps de calcul est occupée par le détecteur de Harris ou par les associations de points via ZNCC. Il s'agit en effet de traitements coûteux. Cependant, ceux-ci peuvent facilement être parallélisés afin d'accélérer l'algorithme. Cela n'a pas été nécessaire ici car le temps de calcul par image était largement inférieur à la fréquence caméra. Les correctifs développés dans ce chapitre ne ralentissent pas l'algorithme. L'approche géométrique pour la projection de l'ellipsoïde dans l'image prend moins d'une microseconde à calculer. De même, le correctif du gain de Kalman est également rapide à calculer avec moins d'une microseconde aussi en moyenne.

Bien que l'approche soit déjà robuste, certains points pourraient être améliorés afin de renforcer encore davantage la fiabilité de l'algorithme. Le point crucial semble être la reconnaissance des amers. Comme cela a été expliqué précédemment, une erreur d'association peut conduire à la divergence du système. L'utilisation d'un filtre de Kalman récursif rend cet aspect encore plus dangereux. Les méthodes développées ici permettent d'éviter la majorité des écueils dans le suivi des amers. Néanmoins, en cas d'erreur, aucune procédure n'est en place afin de corriger une

mauvaise association. Il pourrait être intéressant de faire une sauvegarde de l'état avant mise à jour afin de pouvoir retourner en arrière si d'aventure une association s'avérait fausse. De telles approches existent dans la littérature. Nous pouvons citer [Neira & Tardós 2002] où les auteurs proposent l'algorithme de Joint Compatibility Branch and Bound (JCBB). Dans celui-ci, le but est de sélectionner l'hypothèse qui intègre le plus grand nombre d'appariements compatibles entre eux. Le principe est que chaque nouvelle correspondance ajoutée à l'hypothèse d'association globale ne doit pas produire une mise à jour empêchant les autres de se réaliser. Dès que c'est le cas, l'algorithme annule la mise à jour et remonte dans un arbre de possibilités jusqu'à trouver les amers qui vont conduire à la meilleure solution d'association. Le problème du JCBB est avant tout son coût qui le rend difficilement intégrable dans une application avec des visées temps réel. Dans [Civera *et al.* 2010], le 1-point RANSAC est présenté. Le but est de fournir un algorithme plus rapide que le JCBB en intégrant des informations a priori. La meilleure observation de chaque amer est testée avec une mise à jour de l'état (et pas de la covariance). Ensuite, l'association permettant de conserver le maximum de correspondances après mise à jour est utilisée comme point de départ afin de guider le processus. Les autres associations sont enfin intégrées des plus fiables aux plus incertaines. Le surcoût induit par cet algorithme est moins important que pour le JCBB. Le 1-point RANSAC pourrait être intégré à notre approche. On peut enfin citer [Stewenius *et al.* 2006] où la matrice essentielle entre 2 vues est calculée. Celle-ci sert à fournir un seuil qui permettra d'identifier les observations fausses et à les rejeter via un RANSAC.

En plus d'une méthode globale permettant de valider les mises à jour, il serait intéressant de rendre plus robuste le simple suivi des points en utilisant des descripteurs plus discriminatoires. L'approche développée dans [Molton *et al.* 2004] est une possibilité. Celle-ci consiste à calculer la normale des descripteurs sélectionnés afin de pouvoir ensuite déformer le patch en fonction de la prédiction de l'amer. Ainsi, les associations ne sont plus affectées par un changement de point de vue. Une approche similaire a été appliquée avec succès à du SLAM visuel dans [Charmette *et al.* 2010]. Concernant la méthode d'association, à savoir la ZNCC, celle-ci pourrait être renforcée en utilisant une variante ayant une précision sous-pixellique [Baker & Matthews 2004]. Ces quelques méthodes permettraient très certainement d'améliorer la robustesse globale de l'application bien qu'il faudrait être prudent quant à la surcharge calculatoire engendrée.

Le problème le plus important reste la dérive naturelle du SLAM qui cause l'inconsistance de la localisation. Cette dérive, déjà évoquée lors des résultats, est d'autant plus handicapante qu'elle va affecter le SLAM à plusieurs véhicules. La localisation de chaque robot divergera différemment de celles des autres rendant le positionnement global dans un repère commun difficile à établir. Il est donc important de traiter ce problème dans un cadre monovéhicule afin que cela ne perturbe pas la localisation de l'ensemble d'une flotte. Le problème sera étudié dans le prochain chapitre et une solution sera proposée.

Chapitre 4

Dérive naturelle du SLAM et inconsistance

Sommaire

Les résultats exposés lors du chapitre précédent montrent que le SLAM tend à dériver avec le temps. Dans le cadre d'une application avec un seul véhicule, cette dérive pose le problème de l'inconsistance de la localisation calculée. En revanche, la localisation dans une carte préalablement construite est toujours possible, comme cela a été illustré précédemment, dès lors que la trajectoire suivie est similaire. La dérive sera alors contrainte grâce aux associations faites avec la carte de référence. Néanmoins, cela n'empêche pas les positions des amers la composant d'être également inconsistantes. Ce point s'avère gênant lorsque l'on souhaite intégrer des informations absolues qui ne sont, par nature, pas affectées par la divergence du SLAM. De même, reconnaître un endroit dans lequel le véhicule est déjà passé devient difficile sans un a priori sur la dérive. Cet aspect se retrouve amplifié avec une flotte de robots puisqu'afin de mettre les cartes de chaque véhicule dans un repère commun, il faut pouvoir trouver des associations entre elles. Avec des dérives différentes, et sans la prise en compte de celles-ci, il est presque impossible de retrouver des portions de trajectoire communes.

Afin de présenter les tenants et aboutissants de cette problématique, nous commencerons par donner un état de l'art quant à l'inconsistance du SLAM en section 4.1. Celui-ci aura pour but d'expliquer l'origine du phénomène en plus d'en comprendre la nature. Lors de la section 4.2, nous introduirons le modèle de dérive développé ainsi que son intégration dans une architecture dédiée. Des résultats viendront ensuite valider l'ensemble. Les expérimentations montreront des cas où la dérive a un impact majeur. Celles-ci seront détaillées en section 4.3. Afin de conclure ce chapitre sur l'inconsistance du SLAM, quelques remarques seront faites dans la section 4.4 sur les résultats obtenus. Les travaux présentés ici ont fait l'objet des publications [Bresson *et al.* 2013b] et [Bresson *et al.* 2013c].

4.1 État de l'art

4.1.1 Constatation et origine

L'étude de la linéarisation proposée dans le chapitre précédent montre que l'utilisation de modèles non linéaires dans le cadre du SLAM provoque des divergences importantes. Des propriétés de convergence ont été démontrées dans le cas linéaire [Dissanayake *et al.* 2001] faisant du filtre de Kalman classique un outil incontournable. Néanmoins, pour des applications réelles, des modèles de systèmes non linéaires sont souvent obligatoires. C'est le cas du SLAM monoculaire décrit dans le chapitre précédent. Bien que l'utilisation d'une unique caméra tende à amplifier cette divergence, celle-ci intervient dans tous les algorithmes de SLAM.

Le problème de divergence de l'EKF SLAM a été maintes fois constaté. Dans [Bar-Shalom *et al.* 2001], les auteurs montrent que le filtre tend à produire des estimées optimistes dès lors que la linéarisation est faite loin de la vérité. Cela signifie que même avec une idée correcte de la profondeur du point, le filtre produit toujours des estimations trop sûres. Cela se traduit très simplement : l'incertitude accompagnant l'état (que cela soit pour le véhicule ou les amers) est trop faible. En

conséquence, l'estimation de l'état est dite inconsistante, la vraie valeur n'étant pas incluse dans l'incertitude estimée. Ainsi, la localisation fournie par les méthodes de SLAM dérive au cours du temps [Huang & Dissanayake 2007]. Le filtre calcule une pose pour le véhicule qui est dite biaisée et qui se traduit par une dérive. Il en va bien évidemment de même pour les amers.

La dérive est très fortement liée à la distance parcourue. Plus une trajectoire est longue, plus la localisation est biaisée. Ce phénomène est illustré en figure 4.1.

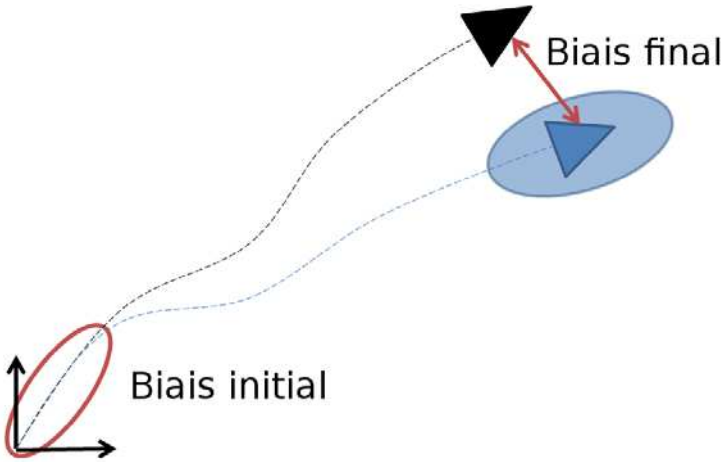


FIGURE 4.1 – Exemple de dérive en fonction de la distance parcourue. Le triangle noir représente la vérité et le bleu l'estimation. L'ellipse bleue est l'incertitude associée au véhicule bleu. Le biais est quasiment nul au début de la trajectoire et augmente progressivement avec la distance parcourue.

En plus des modèles non linéaires, Julier et al. dans [Julier & Uhlmann 2001b] ajoutent que la corrélation entre les données est aussi une source de sur-convergence. En effet, le filtre de Kalman produit des résultats optimistes, c'est-à-dire qu'il suppose que les données issues d'un capteur sont entièrement décorrélées temporellement (non blanches). Ce n'est malheureusement pas le cas et la non prise en compte de cet état de fait conduit également à l'inconsistance. Si chaque nouvelle donnée capteur était totalement décorrélée de la précédente, cela signifierait que toutes les informations portées par celle-ci seraient neuves. C'est ce que considère le filtre de Kalman. Ainsi, en fusionnant une nouvelle observation, les informations qui ne sont pas nouvelles sont de nouveaux intégrées, réduisant l'incertitude plus que cela ne devrait.

En tentant de caractériser cette dérive, de nombreux auteurs ont pu constater qu'une approximation dans l'orientation a plus d'impact qu'une erreur de translation. C'est le cas de Bailey et al. dans [Bailey *et al.* 2006a] qui indiquent qu'avec quelques degrés d'erreur, il devient évident que l'orientation tient une place prépondérante dans la dérive du SLAM. Cela a également été montré expérimentalement dans [Frese 2006]. Néanmoins, il reste difficile de caractériser l'impact de l'erreur

angulaire car elle n'est pas directement comparable à un écart métrique. Quand bien même, le cap du véhicule est impliqué dans l'estimation de la position, ce qui rend son estimation cruciale et donc son implication dans la dérive évidente.

4.1.2 Solutions pour éviter l'inconsistance

La plupart des travaux traitant directement ce problème d'inconsistance ont tendance à chercher des façons d'éviter localement la divergence. Néanmoins, certaines solutions proposent des manières de corriger la dérive par des moyens externes, indépendants du filtrage. Au cours de cette sous-section, nous aborderons tout d'abord les approches cherchant à contourner les problèmes d'inconsistance avant de traiter des techniques permettant de corriger la divergence.

4.1.2.1 Approches évitant la divergence

La première possibilité pour éviter la divergence est l'utilisation de sous-cartes [Estrada *et al.* 2005]. Ces approches, déjà présentées en 2.2.2.2, permettent de séparer la carte globale en sous-cartes. Dans chacune d'entre elles, le repère de référence est ré-initialisé. Ainsi, il est possible d'éviter la dérive du filtre à un niveau local en ne permettant que de courtes trajectoires par sous-carte. Cela pose néanmoins le problème de l'étalonnage de la longueur maximale de chaque sous-carte. Initialiser des sous-cartes trop régulièrement conduit à une perte d'informations conséquente. Des sous-cartes trop grandes empêchent de s'abstraire de la divergence. Cependant, même en fixant celle-ci de manière adéquate, il n'y a aucune garantie concernant l'intégrité de la localisation. En effet, chaque mesure contribue à rendre le système inconsistant et donc ce phénomène peut intervenir même avec peu de mesures. De plus, si la divergence est évitée dans la sous-carte, ce n'est pas le cas dans la carte globale qui garde les traces de ces sous-cartes. Bien que cette solution réduise localement les problèmes liés à la linéarisation, cela ne résout pas le problème de la divergence. Dans un système multivéhicule, un repère commun est nécessaire afin de pouvoir estimer des distances relatives entre les robots. Pour exprimer les positions des véhicules dans celui-ci, il faudrait donc utiliser la localisation haut niveau, qui elle est affectée par la dérive.

De manière similaire, les approches robot-centrées permettent de s'abstraire en partie de la divergence [Castellanos *et al.* 2004]. Au lieu d'avoir un repère monde fixe, les estimations sont toujours données par rapport à la position du robot. Ainsi, les problèmes d'inconsistance sont moins fréquents puisqu'ils ne se cumulent pas comme c'est habituellement le cas. Dans [Castellanos *et al.* 2007], les auteurs affirment que les EKF robot-centrés offrent une meilleure consistance. Néanmoins, comme pour les sous-cartes, la divergence n'est pas résolue dès que l'on sort du repère robot-centré. De plus, l'intégrité des estimations ne peut être garantie, même localement, car les amers peuvent diverger avec seulement quelques mesures. Enfin, le passage en multivéhicule, qui est l'objectif final visé dans nos travaux, est compliqué à effectuer pour des approches robot-centrées. Les raisons sont identiques

à celles évoquées pour les sous-cartes : il est nécessaire de passer par un repère commun qui lui sera affecté par les divergences. Quelques approches ont néanmoins utilisé une technique robot-centrée pour du SLAM visuel monovéhicule avec succès [Civera *et al.* 2009][Williams & Reid 2010]. Dans celles-ci, il s'agit plus d'odométrie visuelle que de SLAM : les points sont oubliés au fur et à mesure qu'ils ne sont plus visibles. Dans ce cas, la méthode robot-centrée est tout à fait appropriée. Pour une application impliquant plusieurs véhicules, cela est en revanche très peu indiqué.

4.1.2.2 Techniques corrigeant la divergence

Les méthodes évitant la divergence ne sont pas satisfaisantes dans un contexte multirobot. Souvent, la communauté résout la dérive de manière plus naturelle en intégrant des informations absolues. L'utilisation d'un GPS, même bas coût, peut contraindre la dérive à l'incertitude associée aux mesures de ce capteur. Une donnée absolue, même ponctuelle, est intéressante car elle ne souffre d'aucune divergence puisqu'elle n'est pas dépendante de l'état précédent du véhicule comme le sont les mesures relatives. L'utilisation de l'infrastructure est également une possibilité. Avec l'émergence du standard 802.11p, les communications véhicule à infrastructure (Vehicle To Infrastructure : V2I) ou véhicule à véhicule (Vehicle To Vehicle : V2V) peuvent être considérées comme une solution viable pour les approches collaboratives dans les années à venir. En effet, ce standard réserve une bande passante élevée tout en étant compatible avec des véhicules en mouvement. Ainsi, il devient possible d'utiliser une infrastructure parfaitement localisée afin de réduire la divergence. De même, chaque véhicule étant affecté par une dérive différente, le partage d'informations peut aider à corriger celle-ci.

Dans [Seo *et al.* 2005], une infrastructure de capteurs RFID est utilisée. L'emplacement de chacun étant parfaitement connu, déduire la position d'un véhicule évoluant dans ce milieu est aisé. La trajectoire accomplie par le véhicule n'est plus affectée par la dérive car celle-ci est constamment corrigée par les capteurs RFID. Dans [Conte & Doherty 2009], un véhicule aérien utilise des images satellites pour obtenir une localisation absolue. Pour ce faire, elles sont comparées à des images en sortie d'une caméra classique. Cela permet de déduire la position du véhicule de manière absolue et donc sans dérive. En dehors de ces méthodes, le véhicule peut toujours utiliser une simple base de données de points géo-référencés. La reconnaissance de certains de ces points permet de supprimer la dérive et ainsi de retrouver la consistance de la localisation.

La correction de la divergence par des informations absolues est un bon moyen de forcer la consistance ponctuellement. L'intégration d'un GPS reste problématique à cause des défauts inhérents à ce capteur (multitrajet, masquage...). Dans tous les cas, il faut pouvoir représenter la dérive entre deux données absolues afin de garantir la consistance à tout moment. Hormis cet aspect, la fusion de données absolues dans un filtre de Kalman reste compliquée car la localisation peut avoir perdue son intégrité par rapport au repère absolu. En conséquence, rien ne peut garantir le comportement de l'EKF dans ce cas.

La deuxième possibilité pour corriger la dérive est la fermeture de boucle. Cela correspond à reconnaître un endroit où le véhicule est déjà passé afin de pouvoir relier la fin de la boucle avec son début. Cette action corrige la dérive puisque les amers cartographiés au premier passage sont forcément affectés par un biais de position plus faible que lors du second passage. Dans [Martinelli *et al.* 2005], il est dit que fermer la boucle permet de contrer la dérive à l'intérieur de celle-ci. Néanmoins, il est également précisé qu'après fermeture, il y a toujours sur-convergence. Les auteurs expliquent que la covariance entre deux amers est surestimée et amplifie donc l'impact que peut avoir la fermeture de boucle. En effet, si l'estimation des liens entre les amers est plus forte que ce qu'elle ne devrait, il est normal que la correction de l'un d'entre eux affecte énormément les autres. Même si la dérive est corrigée autour du point de fermeture, cela ne sera pas le cas à d'autres endroits de la trajectoire. Avec la sur-convergence, l'intégrité de la localisation n'est pas garantie.

Une autre approche a été proposée dans [Cole & Newman 2006]. Dans celle-ci, une fois la fermeture de boucle identifiée, les erreurs sont redistribuées de manière probabiliste autour de la trajectoire passée. Cela permet d'éviter le phénomène de sur-convergence constaté précédemment. Néanmoins, rien ne permet de garantir la consistance de la localisation obtenue. De même que pour les approches basées sur des données absolues, la trajectoire dérive et n'est pas intègre avant la fermeture de boucle sans que cela ne soit pris en compte. Il s'agit encore une fois d'une information ponctuelle permettant de corriger la dérive pour un instant donné. Enfin, et toujours de façon similaire aux approches précédentes, le résultat de localisation après fermeture de boucle ne peut être assuré par le filtre de Kalman puisque celui-ci se base sur des données non intègres.

Les solutions présentées ici sont toutes envisageables afin de pouvoir estimer la dérive à un moment donné. Cependant, il est essentiel que celle-ci puisse être représentée à tout instant, même en l'absence de ces informations. C'est pourquoi il est nécessaire de modéliser la dérive et de l'intégrer dans un filtre de Kalman afin de pouvoir l'estimer conjointement avec l'état du véhicule et de la carte.

4.2 Intégration de la dérive dans le SLAM

L'état de l'art a montré que la dérive du SLAM pouvait être gérée de différentes manières. Ici, nous avons décidé de la modéliser par un processus dynamique. La prise en compte de ce modèle permettra d'exprimer la carte et la pose d'un véhicule de façon non biaisée.

Le modèle développé, qui intègre la notion de biais dans la localisation, sera introduit dans la sous-section 4.2.1. Néanmoins, même avec ce modèle, la conception d'une architecture dédiée est une étape indispensable. Les raisons à cela, ainsi que la dite architecture seront présentées en sous-section 4.2.2.

4.2.1 Modèle de dérive

4.2.1.1 Définition du modèle

L'inconsistance des algorithmes de SLAM se traduit par un écart entre l'estimation de la pose du véhicule et où celui-ci est réellement. Cet écart n'est d'ailleurs pas couvert par l'incertitude de l'estimation (matrice de covariance). Il peut être vu comme un biais dans la localisation du robot. Le biais affectant un véhicule dépend de la longueur du chemin parcouru. Plus une trajectoire sera longue, plus le biais de localisation sera important. La valeur du biais est ainsi étroitement liée à l'abscisse curviligne. Cela signifie que le biais affectant la localisation à un moment n'est pas le même que celui qui l'affecte l'instant juste après (bien qu'il en soit très proche). La conséquence directe est que les amers cartographiés à des dates distinctes seront sujets à des valeurs de biais différentes.

Nous définissons le biais comme les 3 paramètres (translation et rotation) exprimant la différence entre la pose réelle et la pose estimée du robot :

$$\mathbf{b}_s = \begin{pmatrix} b_{x_s} \\ b_{y_s} \\ b_{\Theta_s} \end{pmatrix} \tag{4.1}$$

où $\begin{pmatrix} b_{x_s} & b_{y_s} \end{pmatrix}$ est le biais de position et b_{Θ_s} est le biais angulaire. Le choix de la représentation avec uniquement 3 paramètres, au lieu des 6 de la pose véhicule, est principalement motivé par le fait que la dérive concerne quasiment exclusivement ces 3 paramètres dans le cas d'une caméra fixe dans un véhicule. Bien entendu, les équations qui vont suivre pourraient être étendues à un modèle à 6 paramètres.

Cette estimation du biais est indexée suivant l'abscisse curviligne courante s. $\mathbf{b}_s$ est associé à sa matrice de covariance $\mathbf{P}_{\mathbf{b}_s}$ qui permet de renseigner l'incertitude que l'on peut avoir quant à la divergence de la localisation.

Soit s l'abscisse curviligne associée au moment k. Ainsi, la pose estimée et biaisée du véhicule $\mathbf{v}_k = \begin{pmatrix} x_k & y_k & \theta_k \end{pmatrix}^{\mathrm{T}}$ peut être corrigée par le biais $\mathbf{b}_s$ afin d'intégrer la dérive. La pose du véhicule non biaisée $\mathbf{v}_{u_k}$ (u pour unbiased) peut alors être calculée ainsi :

$$\mathbf{v}_{u_k} = \mathbf{R}_z(b_{\Theta_s})\mathbf{v}_k + \mathbf{b}_s \tag{4.2}$$

L'intégration du biais peut bien évidemment aussi être faite pour les amers afin d'en corriger la divergence. Pour un amer 3D $\mathbf{a}_{i_k}$ affecté par un biais $\mathbf{b}_s$, sa version non biaisée se calcule de la façon suivante :

$$\mathbf{a}_{u_{i_k}} = \mathbf{R}_z(b_{\Theta_s})\mathbf{a}_{i_k} + \begin{bmatrix} b_{x_s} \\ b_{y_s} \\ 0 \end{bmatrix} \tag{4.3}$$

Nous avons choisi d'intégrer le biais et de le définir dans le repère monde de manière additive mais il est tout de même nécessaire de prendre en compte la rotation induite par la dérive angulaire. L'intégration de la rotation au niveau du repère monde $\Re_m$ a l'avantage de permettre de complètement corriger une trajectoire dont l'angle serait biaisé. La place de la rotation s'explique également par la vision que nous avons de la place du biais dans le SLAM. En effet, ce biais peut-être vu comme un changement de repère permettant de passer d'une localisation dans le monde biaisé à un positionnement prenant en compte la dérive. L'inconvénient d'une telle représentation tient surtout au fait que la rotation, et donc le biais angulaire, est valable autour de l'origine. Dès lors que la distance au repère d'origine devient grande, la place de la rotation devient prépondérante. Une même correction angulaire n'aura pas le même impact si elle est appliquée loin ou proche de l'origine. La trajectoire aura tendance a être beaucoup plus décalée à grande distance. Dans ce cas là, le biais de position vient compenser cet écart, ce qui rend sa paramétrisation plus compliquée à établir.

Une autre possibilité serait de considérer la correction angulaire au niveau du repère véhicule $\Re_v$, puis, une fois estimée, de venir l'appliquer globalement dans $\Re_m$. Néanmoins, la difficulté repose sur le fait que les amers sont exprimés dans $\Re_m$ et doivent nécessairement l'être afin de pouvoir être associés au sein d'une flotte. Ainsi, l'intégration du biais, pour un amer 3D $\mathbf{a}_{i_k}$ dans ce cas-ci pourrait se faire de la manière suivante :

$$\mathbf{a}_{u_{i_k}} = \mathbf{R}_z(\Theta_k)\left(\mathbf{R}_z(b_{\Theta_s})\left(\mathbf{R}_z(\Theta_k)^{\mathrm{T}}\left(\mathbf{a}_{i_k} - \begin{bmatrix} x_k \\ y_k \\ z_k \end{bmatrix}\right)\right) + \begin{bmatrix} b_{x_s} \\ b_{y_s} \\ 0 \end{bmatrix}\right) + \begin{bmatrix} x_k \\ y_k \\ z_k \end{bmatrix} \tag{4.4}$$

Le problème de cette intégration vient du fait que dès lors qu'une estimation de $\mathbf{b}_s$ sera disponible, celle-ci sera dépendante d'une pose véhicule bien précise. En effet, étant calculée dans $\Re_v$, la correction devra être propagée en considérant la position courante du véhicule. Pour cela, il serait alors nécessaire de travailler non plus sur un biais $\mathbf{b}_s$ tel que défini mais sur des écarts de biais :

$$\Delta\mathbf{b}_s = \mathbf{b}_s - \mathbf{b}_{s-\Delta s} \tag{4.5}$$

où $\mathbf{b}_{s-\Delta s}$ est l'estimation de biais précédant $\mathbf{b}_s$ et Δs la distance parcourue entre les deux estimations du biais.

Au sein d'un seul véhicule, cette intégration est envisageable puisqu'il n'y a qu'un seul repère véhicule. Dans une application multivéhicule, il faudrait alors gérer les valeurs de $\Delta\mathbf{b}_s$ suivant le véhicule concerné, ce qui rend cette solution beaucoup plus complexe à mettre en œuvre.

Ce problème peut également être résolu en translatant la carte, ainsi que la pose du véhicule, de la distance séparant le véhicule à l'origine au moment de l'estimation du biais. Puis, après coup, d'appliquer la translation inverse afin de correctement repositionner la carte. Ainsi, la correction est valide, puisque faite autour de l'origine,

sans pour autant nécessiter de changer la structure de la carte. Nous verrons plus tard dans cette sous-section (4.2.1.3), via des exemples de simulation, quel impact a l'estimation du biais loin de l'origine avec et sans la solution proposée ici. Les équations (4.2) et (4.3) peuvent alors s'écrire de la façon suivante :

$$\mathbf{v}_{u_k} = \mathbf{R}_z(b_{\Theta_s})(\mathbf{v}_k - \mathbf{t}_{mv}) + \mathbf{b}_s + \mathbf{t}_{mv} \tag{4.6}$$

$$\mathbf{a}_{u_{i_k}} = \mathbf{R}_z(b_{\Theta_s})(\mathbf{a}_{i_k} - \mathbf{t}_{mv}) + \begin{bmatrix} b_{x_s} \\ b_{y_s} \\ 0 \end{bmatrix} + \mathbf{t}_{mv} \tag{4.7}$$

où $\mathbf{t}_{mv}$ est la translation permettant de passer de $\Re_m$ à $\Re_v$. Nous considérerons à partir de maintenant que la pose du véhicule (ou de l'amer) est celle translatée de la distance du véhicule à l'origine. De même, nous considérerons que celle-ci est de nouveau translatée dans l'autre sens après l'intégration du biais.

L'évolution typique de la dérive durant les trajectoires suggère qu'un modèle dynamique du biais est requis. Les modèles auto-régressifs sont pertinents car ils lient chaque estimée à celle la précédant temporellement. Néanmoins, nous avons déjà énoncé que l'évolution de la divergence ne dépendait pas du temps mais de la distance parcourue. Dans le cas contraire, cela signifierait qu'un véhicule statique pourrait dériver, ce qui n'est pas le cas. L'équation d'évolution d'un système linéaire stochastique continu est [Thrun *et al.* 2005] :

$$\dot{\mathbf{b}}(s) = \mathbf{A}_c\mathbf{b}(s) + \mathbf{B}_c\mathbf{u}(s) + \mathbf{M}_c\varepsilon(s) \tag{4.8}$$

où $\mathbf{b}$ est le vecteur d'état, $\mathbf{A}_c$ la matrice d'état, $\mathbf{u}$ le vecteur d'entrée, $\mathbf{B}_c$ la matrice liant $\mathbf{u}$ à $\mathbf{b}$, ε le bruit d'évolution supposé blanc et $\mathbf{M}_c$ la matrice liant le bruit d'évolution à l'état. s représente ici l'abscisse curviligne.

Dans le cas de l'évolution du biais, nous considérerons un modèle approximatif basé sur une simple marche aléatoire. En effet, l'impact que peut avoir le biais de rotation sur la position $\begin{pmatrix} x_k & y_k \end{pmatrix}^{\mathrm{T}}$ est intégré au sein du calcul de la pose du véhicule et n'est donc pas directement impliqué dans l'évolution de la dérive, bien qu'il ait un impact. L'implication du biais de rotation sera étudiée plus loin dans cette sous-section. Ainsi, dans le cas de l'évolution de la dérive, $\mathbf{A}_c$ et $\mathbf{B}_c$ sont nulles et on obtient :

$$\dot{\mathbf{b}}(s) = \varepsilon(s) \tag{4.9}$$

La discrétisation du système permet d'obtenir l'évolution suivante du modèle d'état :

$$\begin{aligned} \mathbf{b}_s &= \mathbf{A}\mathbf{b}_{s-\Delta s} + \varepsilon_{\Delta s} \\ &= \mathbf{b}_{s-\Delta s} + \varepsilon_{\Delta s} \end{aligned} \tag{4.10}$$

où $\mathbf{b}_{s-\Delta s}$ est l'estimée de biais antérieure à $\mathbf{b}_s$ et $\varepsilon_{\Delta s}$ est un bruit blanc représentant la dérive qui est intervenue entre $s - \Delta s$ et s (Δs étant la distance parcourue par le

véhicule entre $k-1$ et k). La matrice d'état $\mathbf{A}$ du modèle discret est définie comme suit :

$$\begin{aligned} \mathbf{A} &= \exp(\mathbf{A}_c \Delta s) \\ &= 1 \end{aligned} \tag{4.11}$$

Comme $\varepsilon_{\Delta s}$ ne peut être directement estimé, il est nécessaire de caractériser sa variance $\mathbf{P}_{\varepsilon_{\Delta s}}$. La variance de $\varepsilon_{\Delta s}$ est donnée par :

$$\mathbf{P}_{\varepsilon_{\Delta s}} = \int_0^{\Delta s} \exp(\mathbf{A}_c s)\mathbf{M}_c\mathbf{P}_\varepsilon\mathbf{M}_c^T \exp(\mathbf{A}_c^T s)ds \tag{4.12}$$

où $\mathbf{P}_\varepsilon$ est la matrice de covariance du bruit blanc $\varepsilon(s)$ et $\mathbf{M}_c = \mathbf{I}$.

Par rapport aux équations (4.10) et (4.12), nous pouvons en déduire l'incertitude du biais courant comme suit :

$$\mathbf{P}_{b_s} = \mathbf{P}_{b_{s-\Delta s}} + \Delta s \mathbf{P}_\varepsilon \tag{4.13}$$

Nous souhaitons déterminer $\mathbf{P}_\varepsilon$ et pour cela il est possible de caractériser la dérive du véhicule expérimentalement sur une certaine distance. On se place ici dans un cadre stationnaire : l'évolution du biais est identique où que l'on se trouve. Cette hypothèse permet de quantifier de manière empirique le pas d'évolution de l'incertitude du biais :

$$\mathbf{P}_\varepsilon = \frac{\mathbf{D}^2}{s} \tag{4.14}$$

où $\mathbf{D}^2$ représente la dérive quadratique observée sur une distance s donnée. Par exemple, à la fin d'une trajectoire de 100 mètres, si une dérive d'un mètre est à noter sur l'axe $\vec{x}$, alors il faudra fixer $\mathbf{P}_\varepsilon$ à 0,01 pour ce même axe.

À présent, nous allons mesurer l'impact que peut avoir l'intégration du biais de rotation sur la position du véhicule. Pour ce faire, considérons un cas simple où un véhicule évolue en ligne droite selon l'axe $\vec{x}$. Au cours de ce déplacement, nous considérerons que le véhicule ne dérivera que d'un point de vue angulaire afin de pouvoir identifier quel poids a le paramètre b_{Θ_s} sur b_{x_s} ou b_{y_s}. Néanmoins, b_{x_s} et b_{y_s} étant purement additifs sur x_k et y_k, l'ajout d'une dérive sur ces paramètres n'influe en rien sur l'évolution de b_{Θ_s}.

Ainsi, après avoir parcouru s mètres, la dérive sur y_k (la dérive sur x_k étant négligeable) peut être exprimée de la façon suivante à partir de l'intégration présentée en équation (4.2) :

$$y_{u_k} = s \ \sin(b_{\Theta_s}) \tag{4.15}$$

Cette dérive peut être considérée comme faible et nous pouvons ainsi simplifier l'équation (4.15) :

$$\begin{aligned} y_{u_k} &\approx sb_{\Theta_s} \\ &\approx s\sum_{i=0}^{s}\Delta b_{\Theta_i} \end{aligned} \tag{4.16}$$

Vérifions maintenant quel impact à cette dérive sur l'incertitude :

$$\sigma^2_{y_{u_k}} = s^2 var[\textstyle\sum_{i=0}^{s}\Delta b_{\Theta_i}] \tag{4.17}$$

Comme il s'agit d'un bruit supposé blanc, la variance de la somme est la somme des variances :

$$\sigma^2_{y_{u_k}} = s^3\sigma^2_{b_\Theta} \tag{4.18}$$

On constate bien que le biais angulaire a une place importante dans la dérive de position. Ce comportement est bien traduit dans la variance du biais de position. L'impact est d'autant plus important que la distance parcourue est grande.

Le modèle, tel que défini à l'équation (4.10), montre que chaque estimation du biais est liée à la précédente. Il est possible de montrer l'implication que cela a sur la covariance. Considérons $cov(\mathbf{b}_{s-\Delta s}, \mathbf{b}_s)$, le lien entre les incertitudes associées aux biais $\mathbf{b}_{s-\Delta s}$ et $\mathbf{b}_s$. On peut écrire cette relation sous la forme :

$$\begin{aligned} cov(\mathbf{b}_{s-\Delta s}, \mathbf{b}_s) &= E[\mathbf{b}_{s-\Delta s}\mathbf{b}_s] - E[\mathbf{b}_{s-\Delta s}]E[\mathbf{b}_s] \\ &= E[\mathbf{b}^2_{s-\Delta s} + \mathbf{b}_{s-\Delta s}\varepsilon] - E[\mathbf{b}_{s-\Delta s}]E[\mathbf{b}_{s-\Delta s} + \varepsilon] \\ &= E[\mathbf{b}^2_{s-\Delta s}] + E[\mathbf{b}_{s-\Delta s}\varepsilon] - E[\mathbf{b}_{s-\Delta s}](E[\mathbf{b}_{s-\Delta s}] + E[\varepsilon]) \end{aligned} \tag{4.19}$$

Le bruit ε est blanc donc $E[\varepsilon] = 0$. Celui-ci étant indépendant de $\mathbf{b}_{s-\Delta s}$, alors $E[\mathbf{b}_{s-\Delta s}\varepsilon] = 0$ et donc :

$$\begin{aligned} cov(\mathbf{b}_{s-\Delta s}, \mathbf{b}_s) &= E[\mathbf{b}^2_{s-\Delta s}] - E[\mathbf{b}_{s-\Delta s}]^2 \\ &= \mathbf{P}_{b_{s-\Delta s}} \end{aligned} \tag{4.20}$$

On peut ainsi en déduire que les incertitudes de biais sont toujours liées par la variance antérieure. L'avantage majeur est que si la valeur du biais pour une abscisse curviligne donnée peut être estimée, alors l'information se propagera sur l'ensemble des biais. Cette rétro-action respectera néanmoins le modèle et évitera ainsi les sur-convergences puisque les covariances entre biais sont correctement estimées.

4.2.1.2 Mise à jour du biais dans le cas unidimensionnel

Afin d'illustrer la mise à jour du biais, nous commencerons par un cas de biais unidimensionnel comme illustré en figure 4.2.

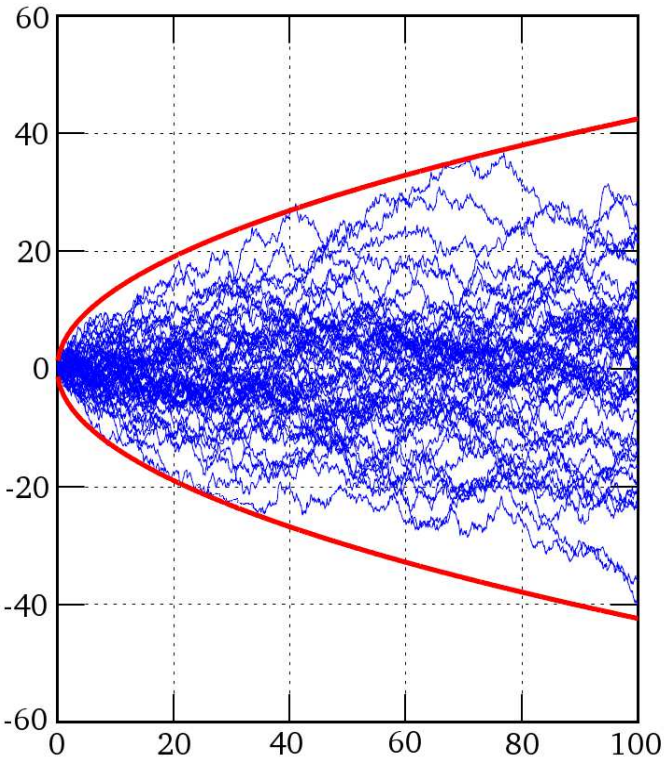


FIGURE 4.2 – Exemple de différentes réalisations d'un biais unidimensionnel. Ici, l'évolution de la valeur du biais en fonction de l'abscisse curviligne est montrée. Les courbes bleues sont 50 réalisations d'un biais 1D. En rouge, l'incertitude associée à ces réalisations (représentée à 3σ).

Cet exemple est une simulation de 50 différentes réalisations d'un biais 1D. Dans celle-ci ε est calculé par un générateur de nombres aléatoires. La divergence quadratique maximum a été fixée à 20 unités par mètre. Un nouvel ε est calculé à chaque pas ($\Delta s = 0,1$ unité). Les courbes rouges représentent l'écart type du biais à chaque pas de temps. En bornant les valeurs du biais avec une telle enveloppe, nous pouvons assurer que la consistance de la localisation calculée est maintenue. On peut montrer que la même manière qu'au 4.2.1.1 que la variance évolue linéairement en fonction de s (l'écart type évoluera donc en fonction de la racine carrée de s).

Celle-ci diminuera uniquement lorsqu'une information capable d'estimer la valeur du biais sera disponible. Comme énoncé précédemment, cela peut être une information absolue, une fermeture de boucle, ou encore dans un contexte multivéhicule, des associations d'amers entre différents robots.

Afin de comprendre comment fonctionne le biais et comment il se met à jour,

on considère un véhicule qui évolue le long d'une ligne en avançant d'une unité par itération. Cette mesure est bruitée afin de simuler un biais de localisation. L'estimation de ce dernier ainsi que la pose du véhicule sont intégrées dans un filtre de Kalman. Bien entendu, le biais n'est pas directement estimé et sa covariance évolue en fonction de la dérive maximale donnée. Les valeurs de dérive sont générées aléatoirement. La figure 4.3 montre l'évolution de ce véhicule sur une dimension. L'incertitude calculée à partir de l'intégration du biais est également affichée.

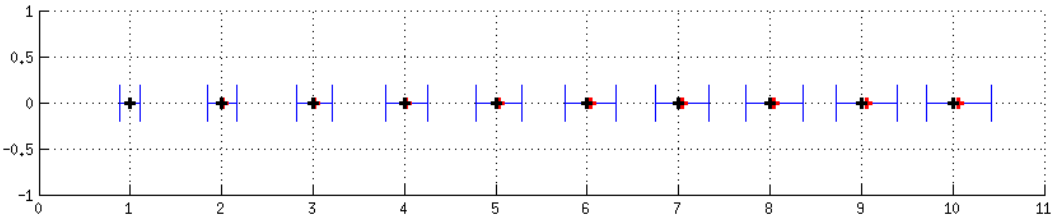


FIGURE 4.3 – Exemple de localisation 1D d'un véhicule avec intégration du biais. Les croix rouges représentent l'estimation de la position du véhicule. Les croix noires sont la vérité. Les intervalles bleus sont les incertitudes 1D calculées à partir de l'intégration du biais.

On peut remarquer que la position dérive tout doucement au fur et à mesure de la trajectoire mais que l'évolution de l'incertitude, en intégrant le biais, couvre bien cette dérive. La consistance est assurée tout au long de cette simulation. La figure 4.4 montre l'évolution de l'incertitude du biais pour les différentes itérations.

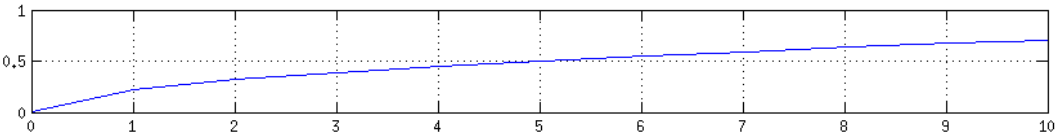


FIGURE 4.4 – Exemple d'évolution de l'incertitude d'un biais 1D

Cette évolution est similaire à celle montrée dans la simulation précédente. Le but de cette simulation est avant tout de montrer comment le biais est affecté dès lors qu'une estimation de celui-ci peut se dégager. Pour ce faire, nous avons considéré qu'après 5 unités de déplacement, un amer observé à la première unité était réobservé afin de montrer l'impact d'un processus SLAM sur le biais. L'observation est alors fusionnée avec l'estimation courante de l'amer et nous pouvons ainsi constater son impact sur le biais. La même trajectoire avec cette fusion est visible en figure 4.5.

La position du véhicule est corrigée à la cinquième unité et diverge ensuite de nouveau progressivement en l'absence de nouvelles informations sur la dérive. La fusion de l'amer permet aussi de faire décroître l'incertitude du biais et donc du véhicule. Celle-ci est réduite à la valeur qui affublait le véhicule à l'unité 1, endroit où

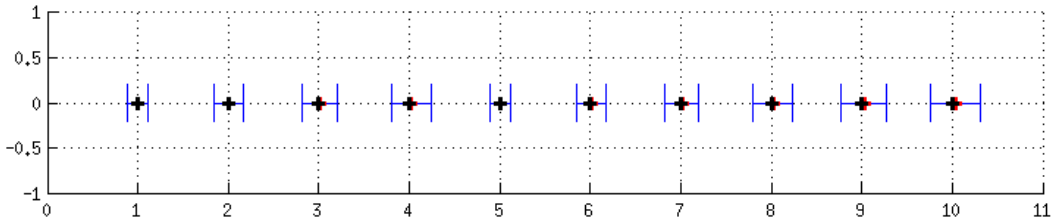


FIGURE 4.5 – Exemple de localisation 1D d'un véhicule avec correction du biais. Les croix rouges représentent l'estimation de la position du véhicule. Les croix noires sont la vérité. Les intervalles bleus sont les incertitudes 1D calculées à partir de l'intégration du biais. L'amer vu à l'unité 1 et ré-observé et fusionné à l'unité 5. Ici, les valeurs affichées sont celles calculées à chaque itération. La rétro-action sur la position et l'incertitude pour les unités 1 à 4 n'est donc pas montrée.

l'amer a été vu la première fois. Bien entendu, l'observation à l'unité 5 est affectée par une dérive supérieure à celle de l'unité 1. Ainsi, la réduction de la variance du biais est bornée par sa valeur lors de la première estimation de l'amer. L'incertitude ré-augmente après en suivant son évolution naturelle. Le profil d'évolution de l'incertitude du biais avec la fusion de l'amer est disponible en figure 4.6.

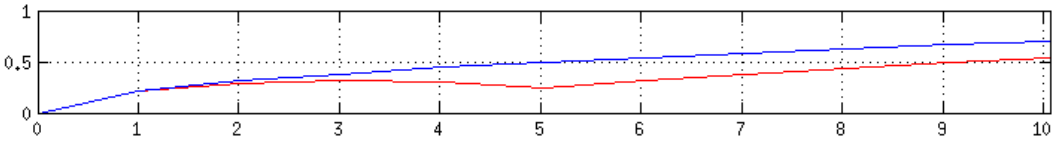


FIGURE 4.6 – Exemple d'évolution et de correction (en rouge) de l'incertitude d'un biais 1D. L'incertitude sans correction est en bleu.

L'évolution de l'incertitude montre bien qu'il y a une réduction de celle-ci dès que l'amer est fusionné, ce qui est normal. On peut aussi remarquer que l'incertitude est également réduite pour les biais précédents. Elle suit une évolution en cloche entre la première observation de l'amer à l'unité 1 et sa fusion à l'unité 5. La position du véhicule la plus éloignée de ces amers (unité 3) est donc la moins corrigée par cette fusion. L'évolution de l'incertitude de biais après l'unité 5 est la même que précédemment mais part cette fois d'une incertitude plus faible.

L'enchaînement d'opérations similaires de fusion aurait pour conséquence de progressivement lisser l'incertitude vers le bas en réduisant peu à peu ce phénomène de cloche. Dans cet exemple, nous avons présenté un cas particulier de fermeture de boucle. Il serait également possible de fusionner des informations absolues.

Dans un cas unidimensionnel, l'évolution du biais et de son incertitude sont prévisibles. Néanmoins, avec 3 degrés de liberté, son comportement est plus difficile à déterminer puisque les différents paramètres du biais sont liés (le cap du véhicule est lié à sa position 2D).

4.2.1.3 Évolution du biais dans le cas 2,5D

Intégrité dans le cas 2,5D

L'objectif de la simulation qui va suivre est de vérifier qu'un véhicule affecté par une dérive angulaire possède une incertitude qui reste intègre dès lors que cette dérive est prise en compte à l'aide des équations décrites précédemment. Considérons une trajectoire de 100 mètres où le véhicule peut dériver de 0,001 radian par mètre parcouru. Afin d'illustrer l'impact de la dérive angulaire, uniquement une très faible dérive de position se manifeste dans cette simulation.

La figure 4.7 présente la trajectoire accomplie en considérant une réalisation aléatoire de la dérive précédemment définie.

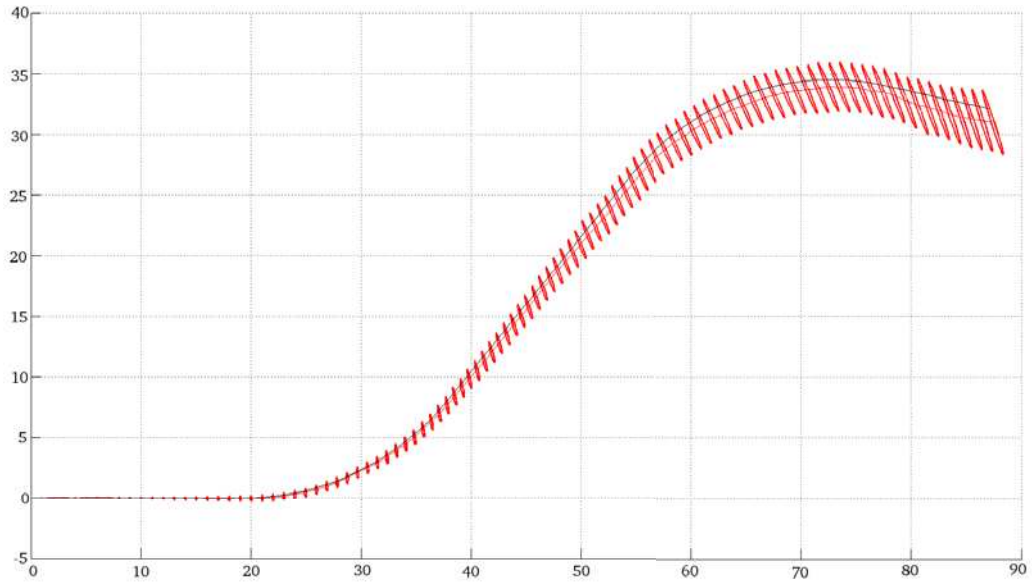


FIGURE 4.7 – Trajectoire de simulation. La courbe noire est la vérité terrain. La courbe rouge est celle obtenue en présence d'une dérive angulaire maximale de 0,001 radian par mètre parcouru. Les ellipses rouges sont les incertitudes à 3σ calculées en prenant en compte la dérive angulaire.

On peut constater que malgré la dérive angulaire, l'intégrité est maintenue tout au long de la trajectoire. L'évolution de la dérive et son intégration telles que définies précédemment permettent, dans cet exemple, de conserver la consistance de l'estimation au cours de la trajectoire.

Montrons maintenant que, quelle que soit la réalisation de la dérive angulaire, l'incertitude maximale de position, compte tenu de cette dérive, intègre bien les différentes possibilités de trajectoires. Pour ce faire, une nouvelle simulation, dont le contexte est similaire à la précédente, a été menée. Au cours de celle-ci, 1000 réalisations d'un biais angulaire ont été générées à partir d'une dérive maximale de 0,001 radian par mètre. Les résultats sont présentés en figure 4.8.

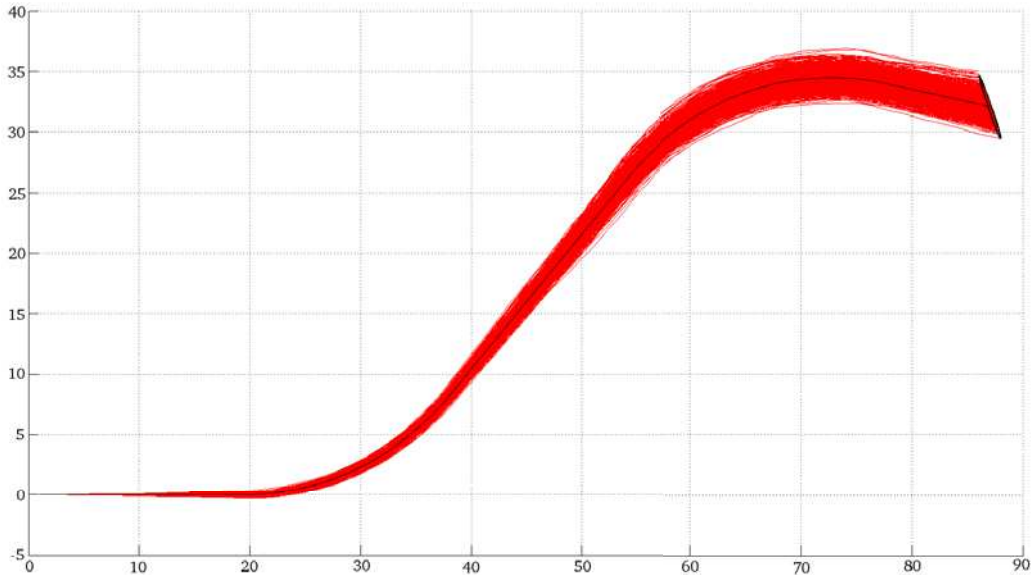


FIGURE 4.8 – Différentes réalisations de trajectoires. La courbe noire est la vérité terrain. Chaque courbe rouge est obtenue à partir d'une réalisation. Ici, 1000 réalisations sont présentées. L'ellipse noire est l'incertitude à 3σ calculée à partir de l'évolution de biais définie ici.

On peut noter que ces 1000 réalisations sont presque toutes englobées par l'incertitude de position à 3σ. Cela permet de montrer que le modèle de dérive que nous avons défini est bien cohérent et capable de correctement modéliser une dérive angulaire.

Impact de la rotation sur le modèle de biais

Le but des simulations qui vont suivre est de montrer comment la rotation autour de l'origine peut influencer la valeur du biais. Les quelques résultats qui vont suivre permettront de pointer la nécessité d'effectuer les estimations du biais à proximité de l'origine.

Pour ce faire, des données odométriques ont été générées. Celles-ci ont été utilisées de façon bruitée afin de simuler le comportement réel d'un véhicule. Une version exacte a également été employée afin d'avoir une vérité qui servira de référence. Les deux trajectoires accomplies (exacte et bruitée) sont disponibles en figure 4.9.

Des amers simulés ont été créés pour pouvoir simuler un processus SLAM. L'objectif est ici d'étudier le comportement du biais lors d'une fermeture de boucle. Ainsi, un ensemble de points a été défini en début de trajectoire (8). Au moment du passage au point de bouclage, c'est-à-dire au point de départ de la trajectoire, ces mêmes points sont utilisés à partir de la position du véhicule ayant dérivé. En associant les points du premier passage avec ceux du second, la correction à appliquer à

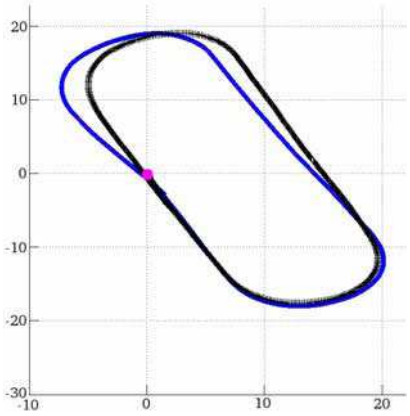


FIGURE 4.9 – Trajectoire de simulation utilisée. La courbe noire représente la vérité et la bleue celle estimée à partir d'une odométrie bruitée. La pastille rose est le point de départ.

la pose du véhicule peut-être calculée et la rétro-action appliquée à l'intégralité de la trajectoire.

L'association de données est fournie afin d'éviter des erreurs et ainsi se concentrer uniquement sur l'impact du biais sur la trajectoire. En revanche, la gestion du biais, des amers et de la pose véhicule est faite de manière similaire à ce qui a été présenté jusqu'ici. Deux cas vont être exposés : la mise à jour du biais autour de l'origine $(0, 0)$ et la même chose mais pour la trajectoire translatée de $(1000, 1000)$. Dans ce second cas, uniquement le point de départ du véhicule est changé, le reste des données odométriques est identique. Afin de garder un point de comparaison, la trajectoire parfaite est également décalée d'un kilomètre.

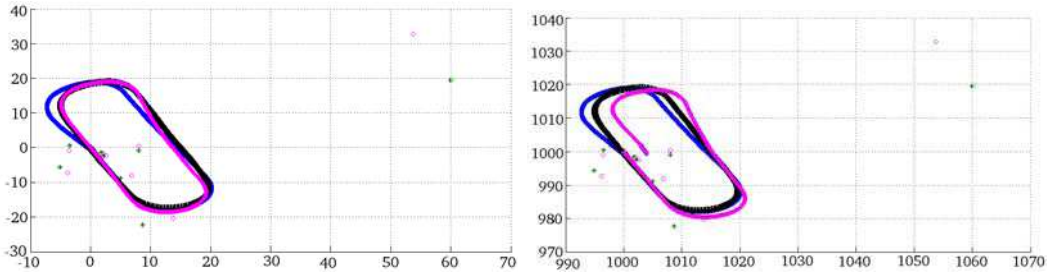
La figure 4.10 montre le résultat de l'estimation du biais et de sa rétro-action lorsque la carte n'est pas ramenée à l'origine. Les amers lors du premier et second passage sont également affichés, ainsi que leur position après la mise à jour du biais.

La table 4.1 donne la valeur du biais calculée au point de fermeture de la boucle pour les deux configurations testées ci-avant.

	Début en $(0, 0)$	Début en $(1000, 1000)$
b_x	0,176	-219,382
b_y	0,570	226,684
b_θ	-0,222	-0,221

TABLE 4.1 – Estimation du biais au point de bouclage (sans translation à l'origine)

On peut constater que les valeurs sont bien différentes en fonction de l'endroit où a lieu l'estimation du biais par rapport à l'origine. Lorsque cela est fait autour du $(0, 0)$, on peut noter que les résultats sont justes et que la rétro-action appliquée à la trajectoire est correcte. En revanche, à un kilomètre, la valeur du biais de position



(a) Trajectoire débutée en $(0, 0)$ (b) Trajectoire débutée en $(1000, 1000)$

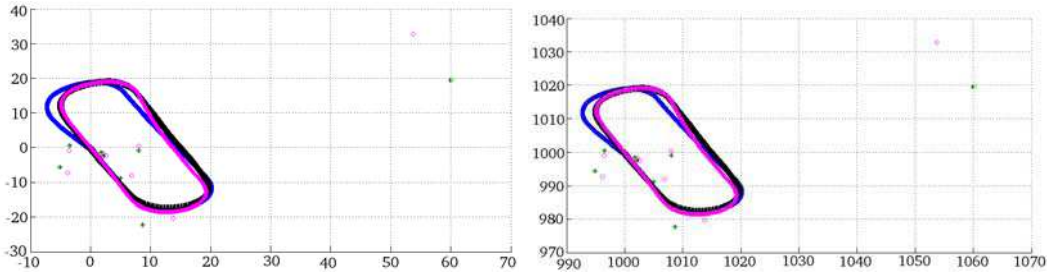
FIGURE 4.10 – Trajectoire calculée sans translation à l'origine. La courbe magenta est la trajectoire calculée avec l'estimation du biais faite lors du bouclage. Les cercles verts sont les amers cartographiés au premier passage et les magentas lors du second. Les croix noirs sont les positions des amers après application du biais calculé.

est bien plus importante. L'angle de biais est, quant à lui, correctement estimé. La conséquence à un kilomètre est que l'application de la rotation correspondante va énormément décaler la trajectoire et ainsi demander une plus grande correction en translation afin de ramener la trajectoire autour du point de bouclage. À cette distance, la justesse de l'estimation laisse à désirer puisqu'il devient difficile de trouver une valeur de translation qui corrige l'intégralité de la trajectoire de façon adéquate. De plus, cela signifie qu'il faudrait calibrer le bruit du biais de position en fonction de la distance à l'origine au moment de la mise à jour de celui-ci. En pratique, cette dernière contrainte peut être difficile à tenir.

C'est pour ces raisons que nous avons proposé précédemment une solution permettant de s'abstraire de ces problèmes d'estimation du biais à grand distance de l'origine. L'idée est très simple, il s'agit de translater de la distance séparant le véhicule à l'origine la pose véhicule et les amers concernés par une prise en compte du biais. Une fois l'intégration de la dérive faite, les estimées subissent la transformation inverse, à savoir la translation inverse vers la pose véhicule. La conséquence est que le calcul du biais est toujours fait autour de l'origine et est donc valide quel que soit la pose du véhicule. Le surcoût est négligeable puisqu'il ne s'agit que de deux translations (vers l'origine et son inverse) dès lors qu'une pose prenant en compte le biais est souhaitée. La figure 4.11 montre, pour les mêmes deux cas que précédemment, le résultat obtenu dès lors que cette translation est intégrée.

De façon similaire, la table 4.2 donne la valeur du biais au niveau du passage par le début de la trajectoire.

Grâce à ce simple correctif, la trajectoire est identique quelle que soit la distance à l'origine. Il est tout de même bon de garder à l'esprit qu'un tel écart dans l'estimation du biais comme cela a été montré en figure 4.10 n'existe que lorsque la distance à l'origine est très importante. À quelques dizaines de mètres, les résultats obtenus sont plus satisfaisants même sans décaler la carte autour de l'origine. Néanmoins, la mise en place de la solution présentée ici est triviale et permet d'éviter ce comportement.



(a) Trajectoire débutée en $(0, 0)$ (b) Trajectoire débutée en $(1000, 1000)$

FIGURE 4.11 – Trajectoire calculée avec translation à l'origine. Le code couleur est le même que précédemment.

	Début en $(0, 0)$	Début en $(1000, 1000)$
b_x	0,176	0,176
b_y	0,570	0,570
b_θ	-0,222	-0,222

TABLE 4.2 – Estimation du biais au point de bouclage (avec translation à l'origine)

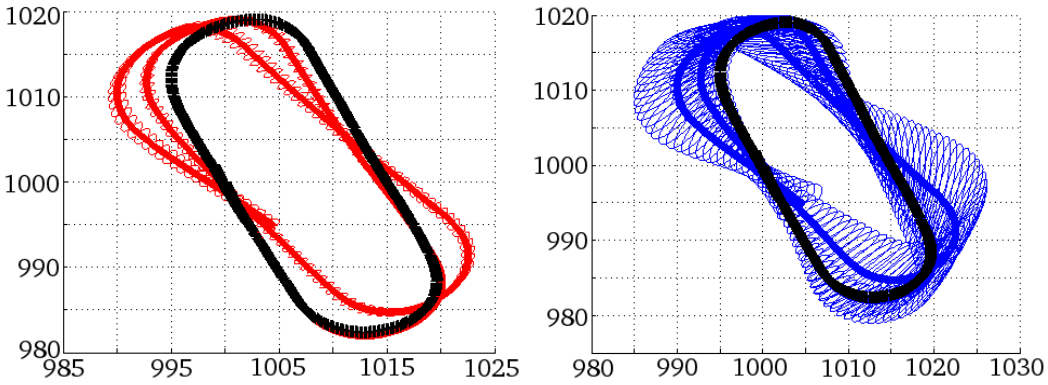
Exemple de deux fermetures de boucle

Reprenons maintenant la simulation précédente en faisant continuer le véhicule sur une deuxième boucle. Le but est ici de montrer dans le détail les incertitudes calculées ainsi que l'impact du biais angulaire sur la localisation à grande distance et ce pour deux bouclages. Un ensemble d'amers est toujours défini autour de l'origine et l'association de données est connue. Le bouclage se fera donc sur les deux passages successifs autour du point de départ de la trajectoire. Comme précédemment, nous considérerons que le véhicule début sa trajectoire en $(1000, 1000)$.

La figure 4.12(a) montre les résultats fournis par un EKF classique qui ne prendrait pas en compte la dérive ou les associations de données. La figure 4.12(b) montre, quant à elle, la trajectoire calculée dès lors que le biais est intégré mais non estimé (aucune association de données).

Il est possible de constater qu'un EKF classique sous-estime l'incertitude de position alors que l'intégration du biais permet de conserver l'intégrité (pose véhicule réelle incluse dans l'ellipse estimée) tout au long de la trajectoire. Bien évidemment, le fait de faire deux tours au lieu d'un accentue encore l'impact de la dérive angulaire sur la trajectoire. L'intégration de la dérive pour le calcul de l'incertitude nécessite de translater la pose du véhicule à l'origine afin d'être correct. Une fois fait, l'ellipse obtenue est ramenée autour de la pose du véhicule.

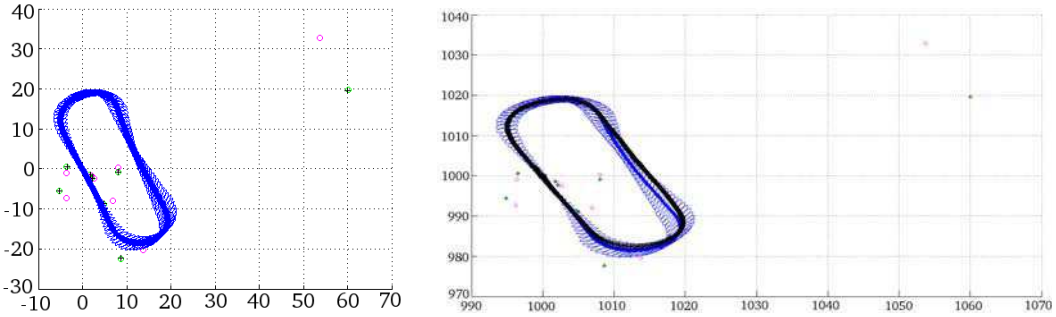
En prenant en compte les associations de données, il est possible de corriger la trajectoire comme cela a été montré précédemment. La figure 4.13(a) montre la correction calculée autour de l'origine et comment celle-ci se répercute sur l'ensemble



(a) Trajectoire sans prise en compte du biais (b) Trajectoire avec prise en compte du biais

FIGURE 4.12 – Trajectoires sans associations de donnée. La courbe noire est la vérité. La trajectoire rouge est celle donnée par un SLAM classique et les ellipses de la même couleur correspondent à l'incertitude de positionnement. La trajectoire bleue représente la même chose mais avec la prise en compte du biais (toujours sans association).

de la trajectoire au niveau de l'incertitude. Ici, nous montrons le premier tour de boucle corrigé. Néanmoins, en pratique, la correction se calcule à un instant donné et donc pour le vecteur d'état à ce même moment qui ne contient qu'une unique pose véhicule et la carte des amers à translater et pas l'intégralité des précédentes poses du véhicule. La figure 4.13(b) montre, une fois la fermeture de boucle faite, la trajectoire estimée autour de la pose réelle du véhicule à 1 kilomètre.



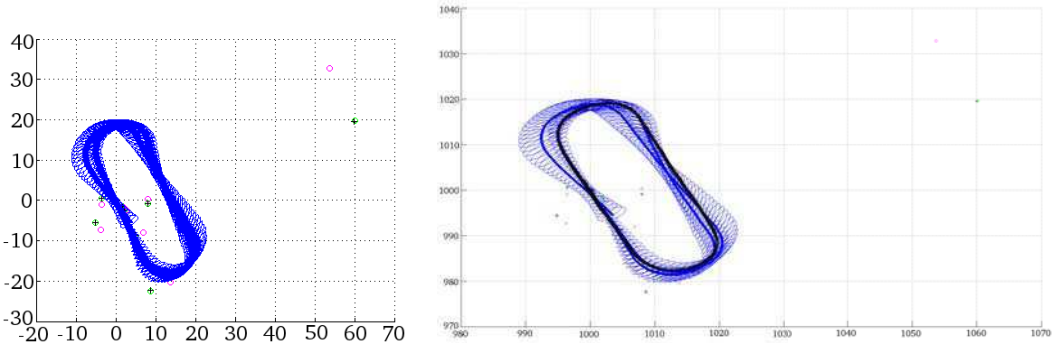
(a) Trajectoire estimée autour de l'origine (b) Trajectoire estimée après translation

FIGURE 4.13 – Trajectoires après la première boucle. Le code couleur est le même que précédemment (de même pour les amers). Ici, uniquement le premier bouclage est effectué et pas le second.

La trajectoire corrigée est bien évidemment la même que celle obtenue précédemment. On peut constater ici que les incertitudes calculées intègrent toujours

la vérité terrain. À partir de cette première estimation du biais, toutes les poses véhicules, afin de pouvoir prendre en compte la dérive, devront tout d'abord être ramenée autour de l'origine. Le biais pourra ainsi être pris en compte de manière correcte, après quoi la nouvelle pose non biaisée subit la translation inverse.

La figure 4.14(a) présente la trajectoire complète avec uniquement le premier bouclage prenant en compte le biais mais autour de l'origine. La figure 4.14(b) montre quant à elle, la trajectoire complète pour un bouclage calculée par l'algorithme, c'est-à-dire après application de la translation inverse. En pratique, la pose courante du véhicule est corrigée de sa dérive dès qu'elle est disponible ce quiImpact de la rotation sur le modèle de biais n'ajoute que les deux translations afin d'avoir la pose non biaisée courante.



(a) Trajectoire complète estimée autour de l'origine

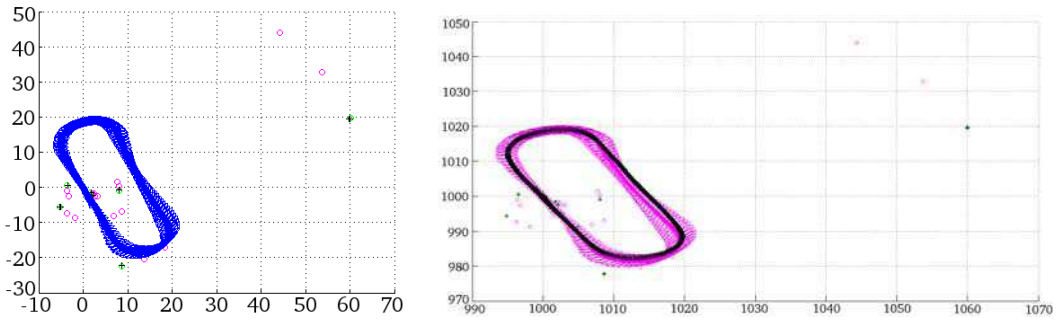
(b) Trajectoire complète estimée après translation

FIGURE 4.14 – Trajectoires après la première boucle. Le code couleur est le même que précédemment (de même pour les amers). Ici, uniquement le premier bouclage est effectué et pas le second.

On peut tout d'abord constater que l'on maintient bien l'intégration au cours du deuxième tour. Sans nouvelle information de bouclage, l'incertitude se contente de grossir et on peut observer un comportement de dérive similaire à la première boucle. L'estimation non biaisée du véhicule est correcte bien qu'éloignée de l'origine.

Enfin, la figure 4.15(a) montre le résultat en fin de trajectoire après le second bouclage une fois le biais pris en compte autour de l'origine. La figure 4.15(b) montre la trajectoire également après le nouveau bouclage mais correctement ramenée autour de la pose réelle du véhicule.

On peut noter que les trajectoires calculées autour de l'origine et autour de la vraie pose du véhicule sont identiques. La méthode utilisée afin de ne pas être dépendant de la distance à l'origine est donc efficace. Au cours du deuxième bouclage, la pose du véhicule a été corrigée et la dérive est correctement estimée. Les incertitudes calculées après les bouclages permettent toujours de conserver l'intégrité de la localisation.



(a) Trajectoire estimée après les deux bouclages autour de l'origine

(b) Trajectoire estimée après les deux bouclages et translation

FIGURE 4.15 – Trajectoires après les deux boucles. Le code couleur est le même que précédemment (de même pour les amers).

Dans tous les cas, le biais ne peut pas être intégré directement dans le filtre de Kalman car il ne permettrait pas de représenter la dérive telle qu'elle a été définie ici. Ce point sera discuté dans la prochaine sous-section.

4.2.2 Architecture dédiée

Grâce au modèle présenté ci-avant, il devient possible d'intégrer le biais dans le processus d'estimation afin de prendre en compte la dérive induite par le SLAM. Chaque pose de véhicule (ou amer) pourrait être directement connectée à l'estimation de biais correspondante dans le filtre de Kalman. Néanmoins, procéder d'une telle façon ne provoquerait pas le comportement attendu. En effet, l'incertitude provenant du biais ainsi que celles issues des mesures seraient mélangées dans la matrice de covariance associée au vecteur d'état. Comme cela a été plusieurs fois énoncé, l'intégration d'observations via le filtre de Kalman tend à faire sur-converger l'estimation de la pose du véhicule. En mixant les incertitudes, puis en intégrant de nouvelles observations, l'incertitude provenant du biais serait alors globalement réduite par ce problème de sur-convergence. Cette méthode n'est donc pas viable pour la prise en compte de la dérive.

Plutôt que de l'intégrer dans le SLAM dit bas niveau, nous avons décidé de sortir l'estimation du biais dans un deuxième filtre. Ce dernier se contente de recevoir les amers et la pose du véhicule en provenance du SLAM classique présenté jusqu'ici. Les nouvelles observations sont uniquement intégrées dans le bas niveau et n'affectent donc pas l'estimation du biais. En plus d'éviter ce phénomène, cette configuration permet de rendre le SLAM bas niveau totalement indépendant de la gestion du biais. Par conséquent, il est possible d'utiliser n'importe quel algorithme de SLAM à la place de l'EKF monoculaire présenté ici. Le SLAM doit juste fournir la pose du véhicule, sa covariance et les positions et incertitudes des amers. Les capteurs utilisés importent peu et une approche employant plus qu'une simple caméra peut remplacer notre solution monoculaire. De même, la méthode de localisation n'a pas

d'importance dans ce cas et ouvre ainsi la porte à la minimisation globale ou à d'autres filtres.

Le deuxième processus d'estimation mis en place est également basé sur un filtre de Kalman étendu afin de pouvoir intégrer facilement les informations permettant d'estimer la dérive. L'organisation globale du système est montrée en figure 4.16.

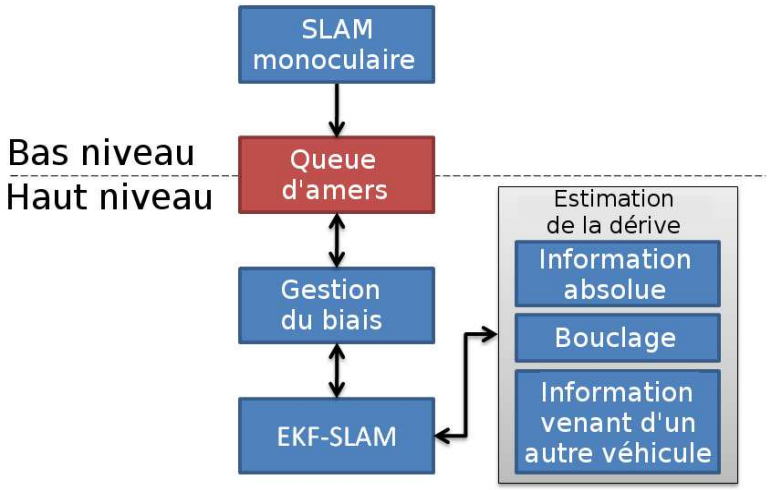


FIGURE 4.16 – Organisation globale du système

Les deux parties de ce système interagissent uniquement par l'intermédiaire d'une queue dans laquelle les amers du bas niveau (et la pose du véhicule) sont déposés. Le haut niveau se charge ensuite d'intégrer ces amers ainsi que la dérive dans l'EKF. Les informations permettant de faire une estimation du biais sont directement gérées dans le haut niveau. Il est ainsi facile de prendre en compte une information absolue, une fermeture de boucle ou encore des amers provenant d'autres véhicules.

Afin de ne pas surcharger la mémoire en menant deux processus SLAM de front, le bas niveau peut être spécialisé vers une odométrie visuelle. L'intérêt est que dès que les amers ne sont plus visibles, ceux-ci sont supprimés du bas niveau et uniquement conservés dans le SLAM intégrant la dérive. Ainsi, il est toujours possible d'identifier des fermetures de boucle et le coût du bas niveau reste faible en plus d'être approximativement constant. Côté charge processeur, celle-ci n'évolue que très peu car le haut niveau n'a pas pour but de faire directement le suivi des amers. Celui-ci cherche seulement à estimer la dérive et n'effectue pas les mêmes actions que le SLAM bas niveau.

Il est bon de noter que les amers sont transmis au haut niveau seulement lorsqu'ils sont précis (incertitude inférieure à moins de 50 centimètres cumulés sur les 3 axes). En effet, envoyer régulièrement des estimations d'un même amer nécessiterait la prise en compte du biais à chaque fois ce qui induit un surcoût calculatoire non négligeable. De plus, des amers imprécis ne sont pas utiles à l'estimation de la dérive.

L'algorithme d'association de données cherchant des fermetures de boucle (ou des informations similaires) requiert des amers précis afin d'être robuste. Une fois envoyé, les amers peuvent toujours être améliorés localement dans le bas niveau. Néanmoins, le gain d'information est faible par rapport à l'incertitude de biais, rendant cette perte à haut niveau quasiment nulle.

Le fait de transformer le SLAM bas niveau en une odométrie visuelle pose la question de quand envoyer les amers vers le SLAM haut niveau. Deux possibilités sont envisageables : soit dès qu'ils ne peuvent plus être observés, soit dès qu'ils sont suffisamment précis pour être utile au haut niveau. Nous privilégierons la seconde méthodes car ces amers peuvent ainsi aider l'estimation de la dérive plus rapidement que si l'on doit attendre qu'ils ne soient plus observables. La figure 4.17 montre les interactions entre le SLAM bas niveau et le filtre prenant en compte la dérive.

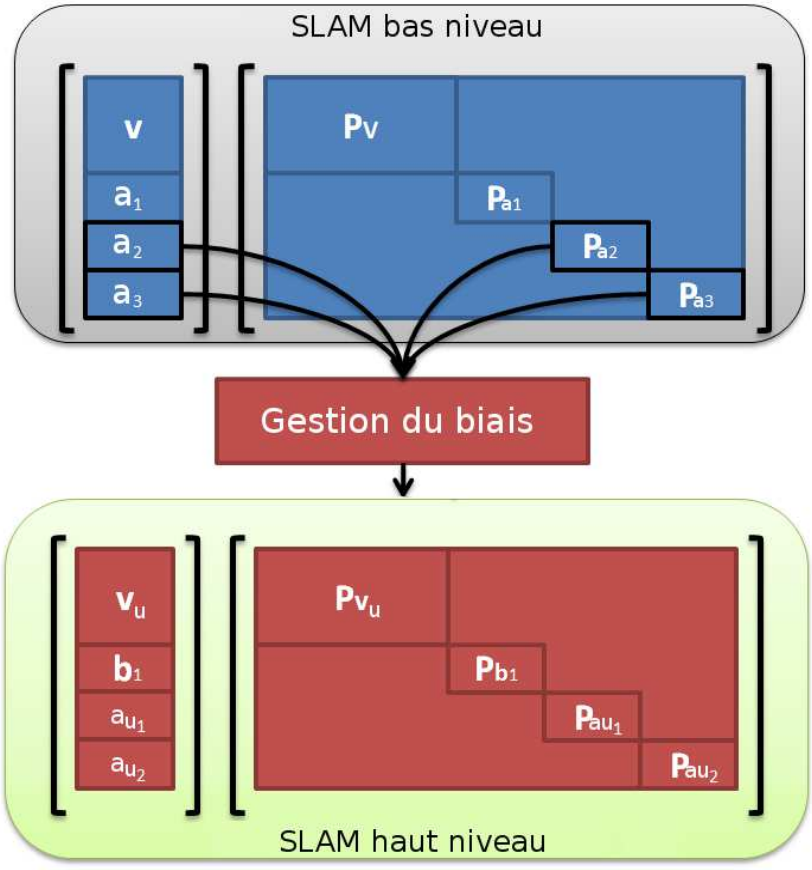


FIGURE 4.17 – Interactions entre les deux niveaux de l'architecture. $\mathbf{v}$ est la pose du véhicule et $\mathbf{P}_v$ la covariance associée. $\mathbf{a}_i$ représente l'amer i et $\mathbf{P}_{a_i}$ l'incertitude correspondance. $\mathbf{b}_j$ est une estimation de biais et $\mathbf{P}_{b_j}$ sa covariance. Les versions des amers et de la pose du véhicule prenant en compte le biais sont indexées avec un u.

Dans cet exemple, 2 amers sont envoyés vers le SLAM haut niveau. On peut remarquer qu'uniquement les variances sont envoyées et pas les covariances. Cela s'explique par le fait que comme nous avons décidé d'échanger uniquement des amers précis, leurs connexions via les covariances sont très faibles dans le bas niveau par rapport à leurs liens au travers du biais dans le haut niveau. Ainsi, elles sont négligeables et peuvent être omises. Dès que les amers sont reçus, l'estimation de biais courante ($\mathbf{b}_1$ ici) leur est appliquée. Des liens sont ainsi re-créés dans la matrice de covariance par l'intermédiaire de la dérive. Les amers se retrouvent naturellement liés au biais. Si des informations permettent de mettre à jour un amer ou le biais, l'ensemble du vecteur d'état sera corrigé.

Les deux amers précédemment envoyés seront supprimés du bas niveau dès qu'ils ne seront plus utiles. Considérons le système précédent après la suppression de ces 2 points et la convergence du dernier amer dans le SLAM local. L'état de ce système est montré en figure 4.18.

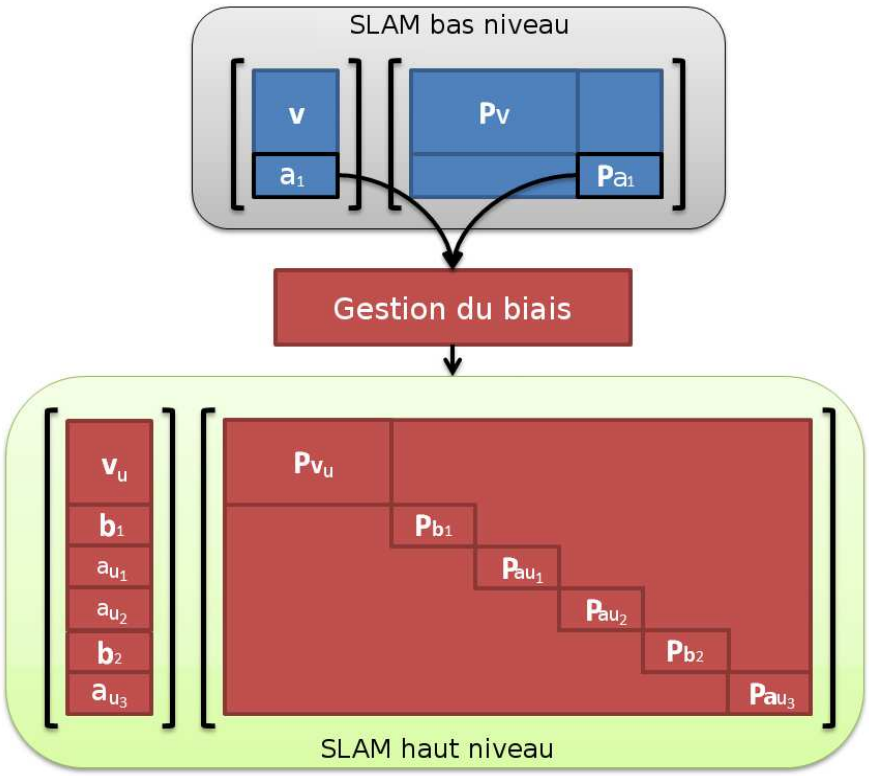


FIGURE 4.18 – État du système après une nouvelle convergence. L'amer encadré dans le SLAM bas niveau est copié dans la partie décentralisée. Le module de gestion du biais lui permet d'intégrer sa dérive $\mathbf{b}_2$ et ainsi d'être lié au reste du vecteur d'état haut niveau.

La carte du SLAM prenant en compte le biais a été augmentée d'une nouvelle estimation du biais et de l'amer reçu. Ce dernier a été connecté au reste de l'état grâce

au biais $\mathbf{b}_2$. On peut voir que seulement deux estimations du biais sont intégrées au vecteur d'état pour 3 amers et la pose du véhicule. Cela ne devrait normalement pas être le cas. Chaque amer dépend d'une abscisse curviligne précise et devrait avoir son propre biais. Néanmoins, le problème est qu'ajouter un biais par amer augmenterait la taille de la carte très rapidement. De plus, sur des abscisses curvilignes proches, l'augmentation de l'incertitude de biais est faible. Par conséquent, nous avons choisi d'intégrer de nouvelles estimations du biais régulièrement, en fonction de la distance parcourue (et non pour chaque amer). Cette contrainte est pertinente dès lors que cette distance n'est pas trop importante (un biais tous les 5 mètres est acceptable par exemple) et que l'incertitude du biais est majorée par la distance maximale que couvrira celui-ci. Sur de courtes distances, l'évolution du biais est donc négligeable.

Afin de ne pas perturber le fonctionnement du SLAM local, les actions du haut niveau sont déclenchées périodiquement pour vérifier si de nouveaux amers sont disponibles. Le fonctionnement global autour de la gestion du biais est expliqué en figure 4.19.

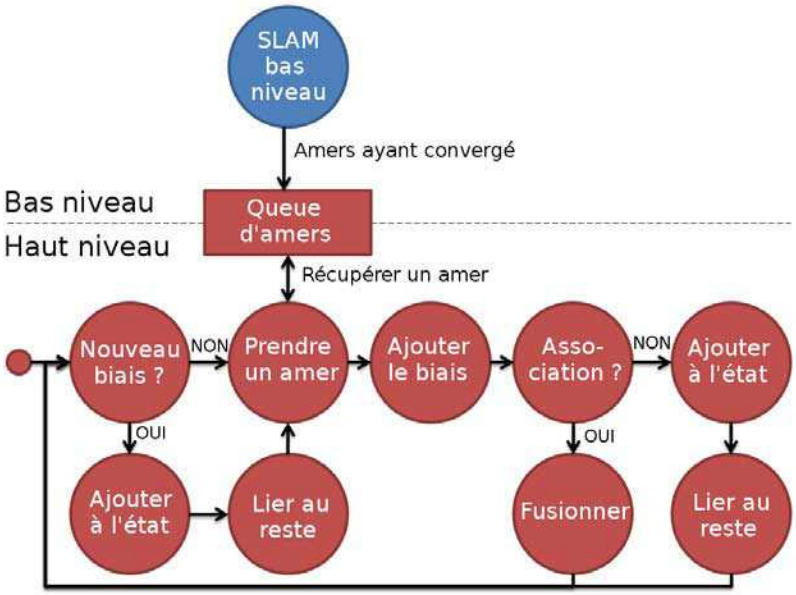


FIGURE 4.19 – Fonctionnement de l'algorithme haut niveau

Les nouveaux amers sont d'abord testés pour des associations avant d'être intégrés dans le filtre. Si la recherche de correspondances est positive alors une fusion (association point 3D - point 3D) est déclenchée. Sinon, les amers sont connectés au reste du vecteur d'état. Avant d'ajouter un nouveau biais, l'abscisse curviligne est vérifiée. Si une nouvelle estimée doit être créée alors elle augmente le vecteur d'état et doit être connectée au reste de la carte.

L'architecture développée permet de correctement prendre en compte la dérive naturelle des algorithmes de SLAM. Celle-ci a été conçue afin d'éviter des surcoûts

trop importants. De même, sa conception est favorable à son extension multivéhicule comme cela sera expliqué en section 5.2. Il nous reste maintenant à étudier comment connecter le vecteur d'état au biais via le filtre de Kalman. De même, l'intégration de la dérive pour les amers et la pose du véhicule doit être analysée.

4.2.3 Intégration de la dérive

Le filtre de Kalman étendu est un outil intéressant quand il s'agit d'initialiser des estimations. En effet, la liaison d'un biais aux amers concernés ou d'un biais au précédent peut facilement être faite par ce filtre. Dans notre cas, la première étape est la connexion de différents biais entre eux. Comme la dérive ne peut être directement estimée, l'initialisation du biais par le filtre de Kalman est simple :

$$\mathbf{b}_s = \mathbf{b}_{s-1} \tag{4.21}$$

La dérive entre $s-1$ et s (appelée précédemment ε) est fixée à 0 et le manque de connaissance au sujet de sa valeur est intégrée dans sa covariance. Pour ce faire, nous initialisons celle-ci avec des valeurs infinies. L'idée est d'utiliser l'EKF pour l'affiner. Nous définissons une fonction d'observation $h_b(\mathbf{b}_{s-1}, \mathbf{b}_o)$ opérant la différence entre deux estimées de biais consécutives $\mathbf{b}_{s-1}$ et une observation de $\mathbf{b}_s$: $\mathbf{b}_o$. Cette différence produit un résultat égal à zéro ce qui permet aussi à l'innovation de Kalman d'être nulle. Cela signifie que seule la matrice de covariance $\mathbf{P}_{\mathbf{b}_s}$ associée au biais $\mathbf{b}_s$, ainsi que ces liens avec $\mathbf{P}_{\mathbf{b}_{s-1}}$, sont affectés. On pose $\mathbf{P}_{\mathbf{b}_o} = \mathbf{P}_{\mathbf{b}_{s-1}} + \Delta s \mathbf{P}_\varepsilon$. Ainsi, l'erreur d'observation est définie comme suit :

$$\mathbf{R}_b = \mathbf{P}_{\mathbf{b}_o} - \mathbf{P}_{\mathbf{b}_{s-1}} \tag{4.22}$$

Il est ensuite possible de calculer la matrice jacobienne associée à la fonction d'observation h_b. Celle-ci est quasiment nulle partout sauf pour les dérivées des estimées de biais concernées :

$$\mathbf{H}_b = \begin{bmatrix} \mathbf{0}_{3\times\dots} & -\mathbf{I}_{3\times3} & \mathbf{0}_{3\times\dots} & \mathbf{I}_{3\times3} \end{bmatrix} \tag{4.23}$$

Dans cette jacobienne $-\mathbf{I}_{3\times3}$ est la dérivée de h_b par rapport à $\mathbf{b}_{s-1}$ et $\mathbf{I}_{3\times3}$ est la dérivée de cette même fonction par rapport au nouveau biais $\mathbf{b}_s$ ($\mathbf{I}$ est la matrice identité). Les matrices $\mathbf{0}_{3\times\dots}$ sont pour le reste du vecteur d'état qui n'est pas concerné par h_b (estimations d'amers, pose du véhicule et biais précédents).

Le gain de Kalman est ensuite calculé afin de pouvoir appliquer l'étape de mise à jour. Vu que l'innovation est nulle, uniquement la matrice de covariance de l'état sera affectée par la mise à jour. Cette dernière se fait de manière classique :

$$\mathbf{K} = \mathbf{P}\mathbf{H}_b^{\mathrm{T}} \left(\mathbf{H}_b \mathbf{P} \mathbf{H}_b^{\mathrm{T}} + \mathbf{R}_b\right)^{-1} \tag{4.24}$$

$$\mathbf{P} = \mathbf{P} - \mathbf{K}\mathbf{H}_b\mathbf{P} \tag{4.25}$$

où $\mathbf{P}$ est la matrice de covariance de l'état complet dans le filtre de Kalman étendu haut niveau.

Cette étape de mise à jour va connecter les deux biais consécutifs, mais pas seulement. En effet, en appliquant cette procédure pour chaque biais, ils deviennent alors tous connectés naturellement. Dans un exemple avec un vecteur d'état comportant 4 estimations du biais, la covariance aurait alors cette apparence :

$$\begin{bmatrix} \mathbf{P}_0 & \mathbf{P}_0 & \mathbf{P}_0 & \mathbf{P}_0 \\ \mathbf{P}_0 & \mathbf{P}_1 & \mathbf{P}_1 & \mathbf{P}_1 \\ \mathbf{P}_0 & \mathbf{P}_1 & \mathbf{P}_2 & \mathbf{P}_2 \\ \mathbf{P}_0 & \mathbf{P}_1 & \mathbf{P}_2 & \mathbf{P}_3 \end{bmatrix} \tag{4.26}$$

Cette forme serait celle obtenue après application d'une mise à jour du filtre de Kalman. Il serait possible de le faire à la main afin d'éviter cette étape du filtre. Or, ces liens sont altérés dès qu'une estimation du biais devient possible. De plus, l'intégration des amers et de la pose du véhicule complexifie encore davantage la construction à la main de cette matrice de covariance.

De manière similaire au biais, l'initialisation d'un amer (ou d'un véhicule) dans le haut niveau se fait par l'intermédiaire de l'EKF. Dans le cas d'un amer, celui-ci est tout d'abord inséré grâce à l'équation (4.3) définie dans la sous-section 4.2.1. Comme pour un nouveau biais, $\mathbf{P}$ est augmentée avec des variances infinies pour l'incertitude de l'amer. Afin d'affiner cette valeur, nous définissons une observation $\mathbf{z}_a$ avec sa covariance $\mathbf{R}_a$ basées sur l'amer provenant du bas niveau :

$$\mathbf{z}_a = \mathbf{a}_i \tag{4.27}$$

$$\mathbf{R}_a = \mathbf{P}_{a_i} \tag{4.28}$$

Soit h_a, la fonction d'observation non linéaire permettant de passer d'un amer prenant le biais en compte à sa version l'ignorant. Cette forme correspond à l'inverse de ce qui est décrit dans l'équation (4.3). Celle-ci se traduit ainsi :

$$h_a(\mathbf{b}_i, \mathbf{a}_{u_i}) = \mathbf{R}_z(b_{\Theta_i})^{\mathrm{T}} \left(\mathbf{a}_{u_i} - \begin{bmatrix} b_{x_i} \\ b_{y_i} \\ 0 \end{bmatrix} \right) \tag{4.29}$$

où $\mathbf{b}_i$ est l'estimation du biais associée à l'amer $\mathbf{a}_{u_i}$ au moment de sa convergence.

Il faut ensuite calculer la jacobienne $\mathbf{H}_a$ associée à h_a afin de pouvoir effectuer la mise à jour de Kalman. Dans ce calcul, uniquement l'amer en question $\mathbf{a}_{u_i}$ ainsi que son biais $\mathbf{b}_i$ sont concernés :

$$\frac{\partial h_a(\mathbf{b}_i, \mathbf{a}_{u_i})}{\partial \mathbf{a}_{j \neq i}} = 0 \tag{4.30}$$

$$\frac{\partial h_a(\mathbf{b}_i, \mathbf{a}_{u_i})}{\partial \mathbf{b}_{j \neq i}} = 0 \tag{4.31}$$

$$\frac{\partial h_a(\mathbf{b}_i, \mathbf{a}_{u_i})}{\partial \mathbf{v}_u} = 0 \tag{4.32}$$

La jacobienne $\mathbf{H}_a$ sera nulle sur les lignes ne correspondant pas à l'amer couramment traité ou à son biais. Si l'on considère que ce biais vient d'être inséré pour cet amer (ce qui n'est pas forcément le cas), alors $\mathbf{H}_a$ aura la forme suivante pour un état de n valeurs :

$$\mathbf{H}_a = \begin{bmatrix} \mathbf{0}_{3\times n-6} & \dfrac{\partial h_a}{\partial \mathbf{b}_i} & \dfrac{\partial h_a}{\partial \mathbf{a}_{u_i}} \end{bmatrix} \tag{4.33}$$

La dérivée concernant l'amer donne le résultat suivant :

$$\frac{\partial h_a}{\partial a_{u_i}} = \mathbf{R}_z(b_{\Theta_i})^{\mathrm{T}} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4.34}$$

De manière similaire pour le biais de position :

$$\frac{\partial h_a}{\partial b_{x:y_i}} = \mathbf{R}_z(b_{\Theta_i})^{\mathrm{T}} \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{4.35}$$

Le calcul des dérivées suivant le biais angulaire donne :

$$\frac{\partial h_a}{\partial b_{\Theta_i}} = \left(\frac{\partial \mathbf{R}_z(b_{\Theta_i})}{\partial b_{\Theta_i}} \right)^{\mathrm{T}} \left(\mathbf{a}_{u_i} - \begin{bmatrix} b_{x_i} \\ b_{y_i} \\ 0 \end{bmatrix} \right) \tag{4.36}$$

La mise à jour est ensuite faite comme précédemment (voir équations (4.24) et (4.25)). De même que pour la connexion des biais, l'innovation de Kalman sera nulle et n'affectera pas l'état.

À l'instar de ce qui a été fait pour les amers, il est possible de définir une fonction d'observation h_v permettant de lier la pose du véhicule à son biais. Celle-ci ne sera pas détaillée dans ce manuscrit car elle se déduit très facilement de ce qui a été développé pour les amers ci-dessus. Une fois ces étapes accomplies, l'ensemble du vecteur d'état côté haut niveau est entièrement lié par la dérive. L'estimation de celle-ci affectera l'ensemble de l'état. Le système est conscient de sa dérive. Ainsi, trouver des fermetures de boucle est facilité car le biais de localisation donne une forte indication sur le moment où elles peuvent intervenir. Il devient alors inutile d'avoir un processus qui cherche en permanence à reconnaître les zones déjà traversées puisque la connaissance de la dérive pourra guider cette recherche.

Que cela soit une fermeture de boucle, un amer géo-référencé ou encore un point provenant d'un autre véhicule, cette information peut facilement être fusionnée dans le SLAM haut niveau. Il s'agit d'une simple fusion point 3D - point 3D, l'un des amers étant déjà dans le vecteur d'état connecté à son estimation de biais. Considérons un amer $\mathbf{a}_{u_i}$ déjà dans le vecteur d'état. Un processus d'association de données nous renseigne qu'un amer nouvellement reçu $\mathbf{a}_j$ représente le même point 3D. Celui-ci a un biais $\mathbf{b}_j$ associé dans le vecteur d'état.

Nous pouvons alors définir celui-ci comme une observation $\mathbf{z}_f$:

$$\mathbf{z}_f = \mathbf{a}_j \tag{4.37}$$

$$\mathbf{R}_f = \mathbf{P}_{a_j} \tag{4.38}$$

où $\mathbf{P}_{a_j}$ est l'incertitude associée à $\mathbf{a}_j$.

Cela permet de définir la fonction d'observation h_f ainsi :

$$h_f(\mathbf{a}_{u_i}, \mathbf{b}_j) = \mathbf{R}_z(b_{\theta_j})^{\mathrm{T}} \left(\mathbf{a}_{u_i} - \begin{bmatrix} b_{x_j} \\ b_{y_j} \\ 0 \end{bmatrix} \right) \tag{4.39}$$

La jacobienne $\mathbf{H}_f$ associée à h_f est similaire à celle définie dans l'équation (4.33). Elle est nulle partout sauf à l'emplacement de l'amer $\mathbf{a}_{u_i}$ et du biais $\mathbf{b}_j$. Considérons que l'amer $\mathbf{a}_{u_i}$ soit situé à la position m pour un vecteur d'état de n lignes, alors $\mathbf{H}_f$ est de la forme :

$$\mathbf{H}_f = \begin{bmatrix} \mathbf{0}_{3\times m-1} & \dfrac{\partial h_f}{\partial \mathbf{a}_{u_i}} & \mathbf{0}_{3\times n-m-6} & \dfrac{\partial h_f}{\partial \mathbf{b}_j} \end{bmatrix} \tag{4.40}$$

Le calcul des dérivées partielles étant identique à celui détaillé dans les équations (4.34) à (4.36), il ne sera pas présenté ici.

Dans le cadre d'une fusion, l'innovation est non nulle. Celle-ci peut être calculée grâce à l'équation suivante :

$$\mathbf{\Delta} = \mathbf{z}_f - h_f(\mathbf{a}_{u_i}) \tag{4.41}$$

Le gain de Kalman est ensuite calculé comme pour l'équation (4.24) mais en replaçant $\mathbf{R}_b$ par $\mathbf{R}_f$ et $\mathbf{H}_b$ par $\mathbf{H}_f$. La mise à jour se fait ensuite via les équations traditionnelles de l'EKF.

Celle-ci corrigera l'ensemble des données composant le vecteur d'état en plus d'aider à estimer le biais courant. Avant d'étendre ce concept à plusieurs véhicules, des validations monorobot ont été effectuées. Elles seront décrites dans la section suivante.

4.3 Validation expérimentale

La validation du modèle de biais telle que présentée au cours de ce chapitre s'est faite de deux manières. Dans la première, il s'agissait de tester son comportement dans le cadre d'une fermeture de boucle avec des données réelles. Cette expérimentation est présentée en sous-section 4.3.1. Ensuite, le comportement de la procédure de mise à jour de la dérive a été éprouvé via l'intégration d'amers géo-référencés, montrant ainsi le fonctionnement de notre modèle en cas de fusion de données absolues. Les résultats de cette trajectoire sont exposés en sous-section 4.3.2.

Les deux expérimentations qui vont suivre partagent le même contexte. Un véhicule réel (VIPALab) équipé d'une caméra, d'un odomètre et d'un capteur d'angle de braquage est employé comme pour les expérimentations précédentes. De même, la plate-forme expérimentale urbaine PAVIN a été utilisée. Au niveau des capteurs, la caméra fonctionne à une fréquence d'acquisition de 15 Hz et le véhicule se déplace à environ 2 mètres par seconde. Un GPS RTK est également présent mais uniquement dans le but de fournir une vérité terrain afin de vérifier la qualité de la localisation calculée. Dans les expérimentations qui suivent, un ordinateur portable équipé d'un processeur Intel Core i5 opérant à 2,40 GHz est utilisé pour exécuter l'algorithme de SLAM décrit ici.

4.3.1 Fermeture de boucle

Dans cette première expérimentation, nous avons choisi de confronter l'algorithme au problème de la fermeture de boucle. Pour ce faire, une longue boucle d'approximativement 100 mètres a été faite par le VIPALab. Le véhicule a continué sur environ 10 mètres après avoir rejoint le point de départ de la trajectoire afin de rendre possible la détection de la boucle. Des amers ont été cartographiés tout au long de la trajectoire. Seulement ceux suffisamment précis ont été utilisés dans le SLAM haut niveau afin d'identifier la fermeture de boucle. Le seuil de convergence des amers est fixé ici comme un cumul des racines carrées des variances suivant les 3 dimensions de l'amer :

$$\sigma_{convergence} = \sqrt{\sigma^2_{a_{ix}}} + \sqrt{\sigma^2_{a_{iy}}} + \sqrt{\sigma^2_{a_{iz}}} < \tau \tag{4.42}$$

où τ est le seuil d'acceptabilité d'un amer, ici fixé à $0,5$m.

Pour cette trajectoire, environ 60 amers ont convergé et sont donc utilisés dans le SLAM gérant la dérive. Afin de bien comprendre l'importance de la prise en compte de la dérive, la figure 4.20 présente les résultats de localisation issus du SLAM bas niveau, c'est-à-dire sans l'intégration du biais. Sur cette figure, la fermeture de boucle n'est également pas accomplie.

Le point important à constater sur cette figure est que la trajectoire calculée par le SLAM monoculaire est sujette à une dérive en translation non négligeable. Les angles sont en revanche plutôt bien estimés, sauf vers la fin de la trajectoire où le véhicule commence tout doucement à dévier. L'incertitude estimée associée à la pose du véhicule est très faible, même après 100 mètres parcourus. Cela rend la détection de la fermeture de boucle impossible avec cette seule information. La localisation calculée est précise mais inconsistante. C'est cet état de fait qui a été décrit tout au long de ce chapitre et qui affecte les algorithmes de SLAM. On peut ainsi constater que, rapidement après le départ, l'intégrité de l'EKF est perdue car la vérité n'est plus englobée par l'incertitude calculée par le SLAM. La dérive de translation maximale mesurée ici est de plus de 4 mètres lors du dernier virage.

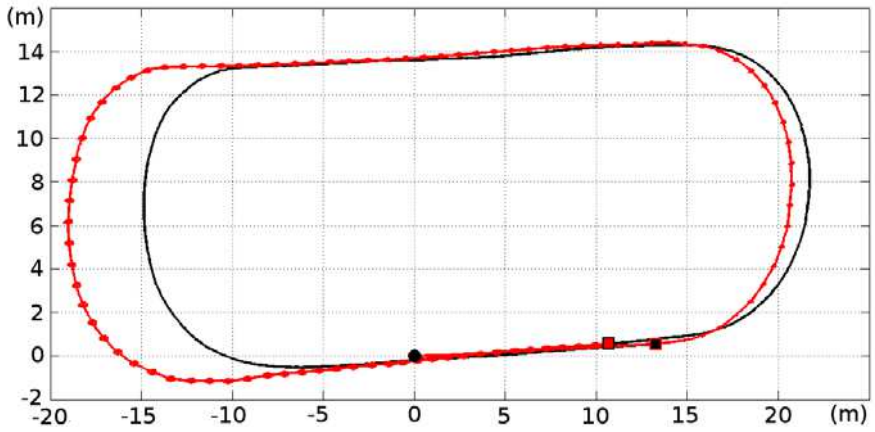


FIGURE 4.20 – Inconsistance et dérive du SLAM bas niveau. La courbe noire est la vérité terrain obtenue par le GPS RTK. La courbe rouge est la localisation calculée par le SLAM monoculaire. Les petites ellipses rouges sur cette trajectoire sont les incertitudes associées aux différentes poses du véhicule. Le point noir est le point de départ de la trajectoire (le véhicule part vers la droite). Le carré noir est la fin de trajectoire pour le GPS RTK et le carré rouge la fin de cette même trajectoire mais estimée par le SLAM.

En reprenant la même trajectoire mais en activant le haut niveau, il est possible de conserver l'intégrité de la localisation. Ces résultats sont exposés dans la figure 4.21. Comme pour la figure précédente, la fermeture de boucle n'est pas effective ici, expliquant ainsi que la dérive ne soit pas corrigée.

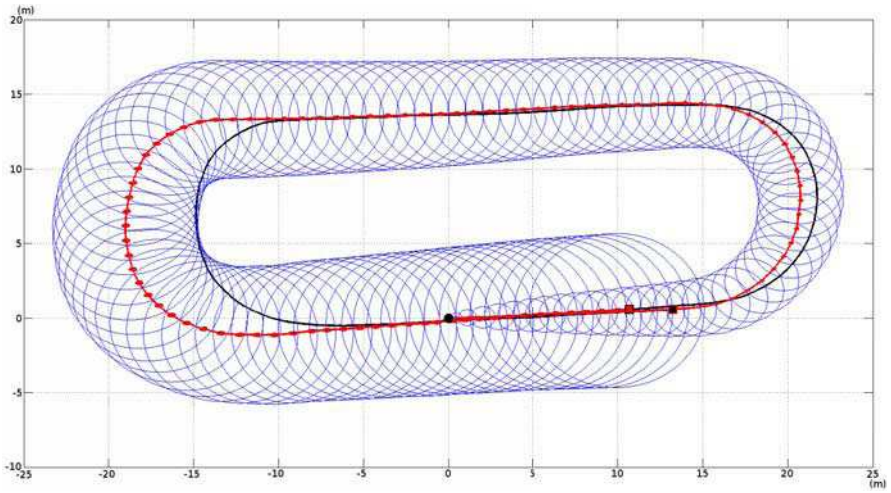


FIGURE 4.21 – Prise en compte de la dérive sans fermeture de boucle. Le code couleur est le même que pour la figure précédente. Les ellipses bleues correspondent à l'incertitude du véhicule donnée par le haut niveau en intégrant la dérive.

La prise en compte du biais montre que, dans ce cas, la consistance du SLAM est assurée : les ellipses grandissent tout au long de la trajectoire et inclues toujours la position réelle du véhicule. Bien entendu, le biais a été calibré pour le véhicule en fonction de la divergence moyenne observée sur plusieurs trajectoires. La recherche de fermetures de boucle est facilitée car un a priori fort est donné par la dérive. Il convient ensuite d'identifier des amers précédemment vus afin de pouvoir s'assurer de la fermeture de boucle et ainsi affiner la trajectoire calculée par le véhicule. Le résultat de localisation avec bouclage est disponible en figure 4.22.

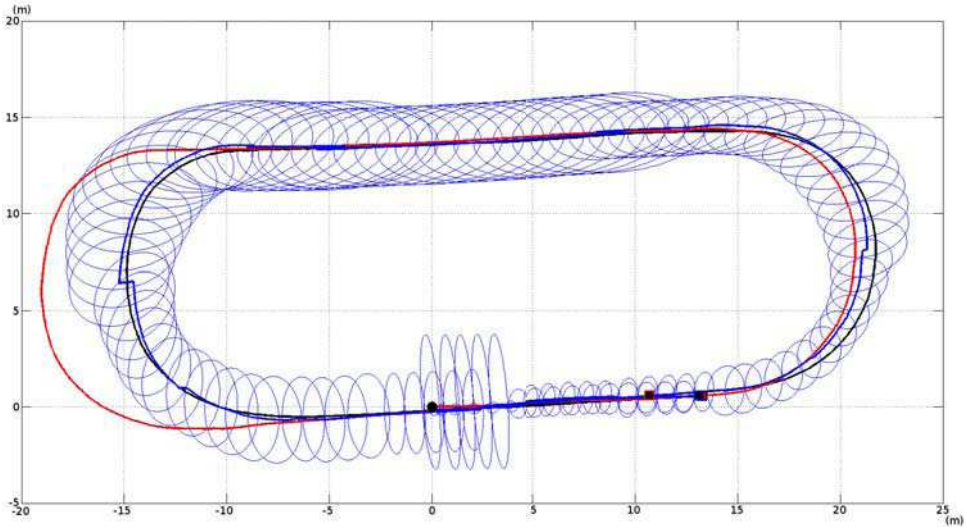


FIGURE 4.22 – Prise en compte de la dérive avec fermeture de boucle. Le code couleur est le même que pour la figure précédente. La trajectoire bleue est celle calculée avec la fermeture de boucle. Le carré bleu est le point d'arrêt calculé après fermeture de boucle.

On peut voir que la trajectoire avec bouclage vient beaucoup plus près de la vérité. L'angle est correctement corrigé à la fin de la trajectoire et les erreurs de translation ont quasiment disparu et ce sur toute la trajectoire grâce à la rétro-action naturelle causée par l'estimation du biais. Les sauts soudains, visibles dans la trajectoire, sont liés au fait que seulement quelques estimées de biais (16 au total) sont utilisées dans cette trajectoire. Ainsi, chaque changement de biais donne ce petit saut dans la localisation. Ce problème peut être résolu en initialisant plus de biais tout au long de la trajectoire, bien que cela engendrerait un surcoût au niveau du temps de calcul. Un autre aspect important est que l'incertitude concernant la pose du véhicule est grandement réduite. Cela est particulièrement visible autour du point de fermeture de la boucle. Il est important de noter que malgré la diminution de l'incertitude, la consistance est toujours assurée sur cette trajectoire.

Afin de vérifier la qualité de la localisation obtenue, celle-ci a été comparée à la position donnée par le GPS RTK. Ces résultats sont montrés en figure 4.23.

Ceux-ci montrent clairement le bénéfice que l'on peut retirer à intégrer la dérive

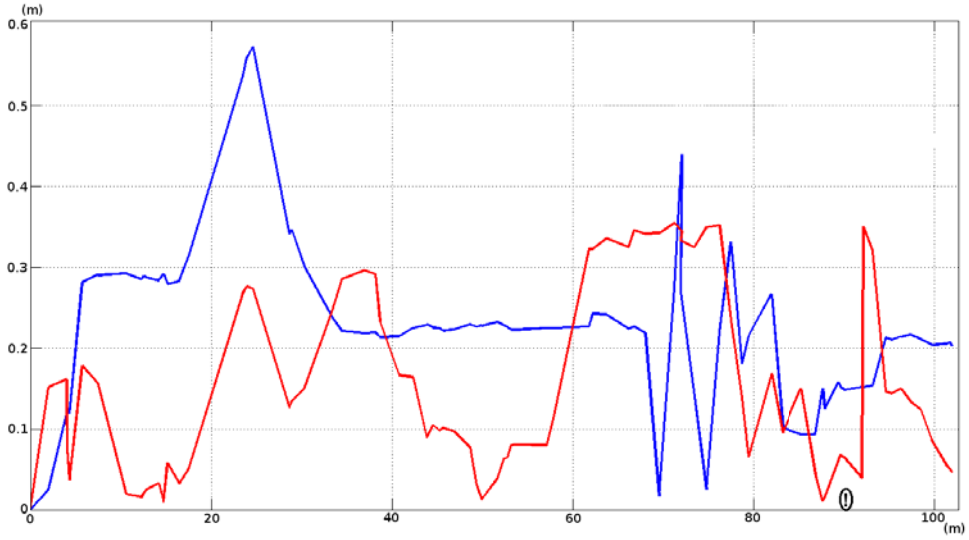


FIGURE 4.23 – Écart de position entre le GPS RTK et le SLAM dans le cadre d'une fermeture de boucle. En bleu, l'écart longitudinal en fonction de l'abscisse curviligne. En rouge, la même chose pour l'écart latéral. Le point d'exclamation sur l'abscisse est l'endroit où la boucle a été fermée.

dans le SLAM. L'erreur maximale est de 55 centimètres et est en moyenne autour de 20 centimètres. Les pics les plus importants ont eu lieu durant un changement d'estimée de biais. Il est important de garder en tête que la précision de la localisation est étroitement liée à la précision des amers. Celle-ci ayant été fixée à 50 centimètres, il est donc difficile d'arriver à avoir une meilleure estimation de la pose véhicule dans ce cas. Cela nécessiterait d'abaisser le seuil de convergence des amers. Pour ce faire, il faudrait également augmenter le nombre d'amers à suivre afin de pouvoir assurer leur convergence. Ici, uniquement 60 amers ont été utilisés. Enfin, en continuant la trajectoire plus longtemps, il aurait certainement été possible de trouver plus d'associations pour améliorer la qualité de la fermeture de boucle.

Ces premiers résultats montrent clairement l'intérêt qu'il y a à prendre en compte la dérive naturelle du SLAM. La fermeture de boucle a ainsi pu être facilement identifiée. De plus, le maintien de l'intégrité du filtre tout au long de la trajectoire renforce encore l'avantage que l'on peut avoir à intégrer le biais de localisation dans le calcul de la pose du véhicule.

4.3.2 Intégration d'amers géo-référencés

Une seconde expérimentation réelle a été menée afin de montrer l'impact que peut avoir l'intégration d'amers géo-référencés au cours d'une trajectoire. Avec l'architecture décrite précédemment, il est aisé de prendre en compte des informations absolues dans l'algorithme de SLAM sans risquer l'inconsistance. Des amers géo-référencés sont ici fusionnés dans le filtre de Kalman pour réduire la divergence.

Le GPS RTK a été utilisé dans ce contexte particulier pour définir des amers géo-référencés. La position du GPS RTK est employée afin de simuler un amer absolu tous les 10 amers. Pour ce faire, le point 3D est passé depuis le repère local vers le repère absolu grâce à la position du GPS RTK. Ces amers ne sont donc pas totalement géo-référencés puisqu'ils sont affectés par l'incertitude qu'ils ont en sortant du SLAM monoculaire. Ils restent donc soumis à une incertitude de 50 centimètres au maximum comme précédemment. L'erreur concernant la pose du véhicule pourra donc atteindre une précision similaire à ce qui a été montré pour la fermeture de boucle.

Il aurait été possible de fusionner directement les informations en sortie du GPS RTK avec la pose du véhicule. Le processus serait en tout point similaire à la fusion d'amers géo-référencés. La connexion de la pose du véhicule avec le reste du vecteur d'état aurait eu pour conséquence d'affecter l'intégralité des amers, tout en estimant la dérive sur toute la trajectoire. Ici, l'utilisation d'amers géo-référencés sert uniquement à illustrer l'une des nombreuses applications qui découlent de l'intégration d'informations absolues dans l'estimation de la dérive du SLAM.

Une trajectoire d'environ 100 mètres a été menée. Elle est composée de longues lignes droites ainsi que d'importants virages afin de confronter l'algorithme à des biais à la fois de position et d'angle. Au cours de cette trajectoire, 48 amers ont été cartographiés et 4 amers géo-référencés ont été fusionnés avec le filtre de Kalman étendu. La figure 4.24 montre les résultats obtenus par le SLAM sans intégrer les amers positionnés de manière absolue.

La trajectoire de SLAM dévie très nettement que cela soit d'un point de vue de la position ou du cap. De plus, l'incertitude calculée par le SLAM bas niveau est loin d'inclure la vérité. Très rapidement après le début de la trajectoire, le filtre devient inconsistant et l'incertitude estimée tend à rester faible même après plus de 100 mètres. Avec les mêmes paramètres de biais que ceux utilisés pour l'expérimentation précédente, l'intégrité du filtre est préservée. On peut voir que l'évolution de l'incertitude avec prise en compte du biais englobe toujours la vérité terrain fournie par le GPS RTK.

La figure 4.25 montre les résultats pour la même trajectoire mais dès lors que les 4 amers géo-référencés sont intégrés.

Les résultats de localisation obtenus sont grandement améliorés par l'utilisation de quelques amers géo-référencés. Ces informations absolues ont permis de réduire de façon conséquente l'incertitude associée à la pose du véhicule. On peut constater qu'autour des points de fusion, l'incertitude est encore plus petite que sur le reste de la trajectoire. La covariance associée à ces points étant très faible, il est logique que l'intégration d'un de ces points affecte fortement la covariance du véhicule. L'évolution de l'incertitude du biais fait ensuite de nouveau grossir l'ellipse entourant le véhicule, permettant ainsi d'intégrer la dérive pouvant intervenir entre la fusion de 2 amers géo-référencés. La consistance de la localisation est donc maintenue dans cet exemple même sans information absolue. Les petits sauts dans la trajectoire sont toujours causés par l'initialisation d'une nouvelle estimation du biais. Les résultats obtenus pourraient facilement être améliorés en utilisant plus d'amers et d'infor-

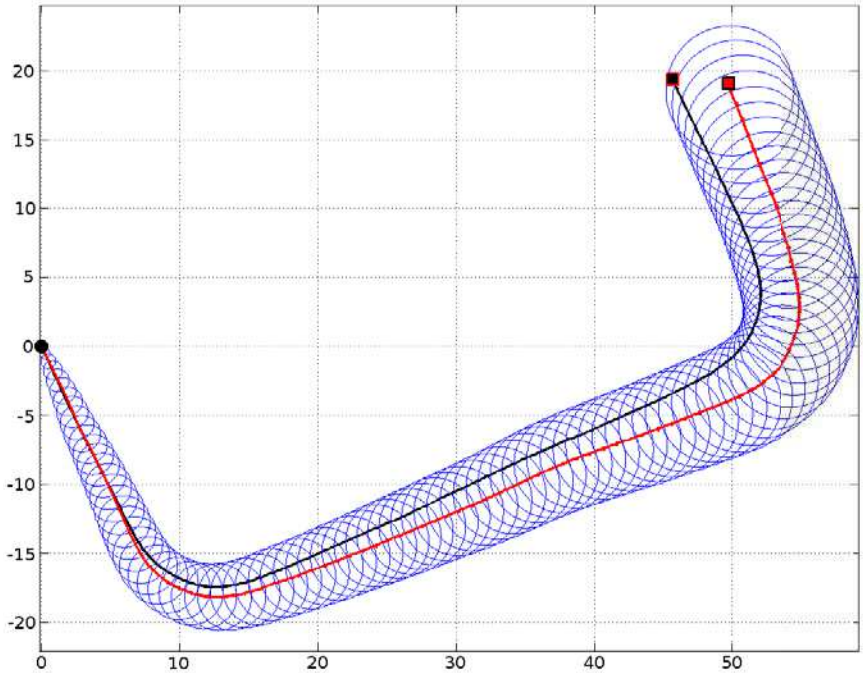


FIGURE 4.24 – Localisation obtenue sans intégrer les amers géo-référencés. La courbe noire est la vérité terrain obtenue par le GPS RTK. La courbe rouge est la localisation calculée par le SLAM monoculaire. Les petites ellipses rouges sur cette trajectoire (à peine visibles) sont les incertitudes associées aux différentes poses du véhicule. Les ellipses bleues sont les incertitudes quand la dérive est intégrée. Le point noir est le point de départ de la trajectoire. Le carré noir est la fin de trajectoire pour le GPS RTK et le carré rouge la fin mais estimée par le SLAM.

mations géo-référencées. Néanmoins, il est important de remarquer que, même avec seulement 4 amers absolus, l'algorithme est capable de fournir une localisation précise et intègre.

Comme pour l'expérimentation précédente, la qualité de la localisation calculée est un bon indicateur de la viabilité de notre solution. De plus, c'est également un moyen de montrer qu'avec seulement quelques informations absolues, il est possible d'atteindre une précision tout à fait satisfaisante. La figure 4.26 compare la position donnée par le GPS RTK à celle calculée par l'algorithme de SLAM.

L'erreur maximale (longitudinale ou latérale) est sous la barre des 40 centimètres ce qui témoigne bien de l'efficacité de la rétro-action par le biais. L'erreur moyenne est quant à elle autour de 20 centimètres comme pour la fermeture de boucle. Atteindre une meilleure précision est difficile dans ce contexte car les amers géo-référencés calculés sont toujours affectés par l'incertitude qu'ils ont en sortie du SLAM monoculaire. Le seuil de convergence étant encore fixé à 50 centimètres, il est normal que l'erreur de position soit bornée par cette valeur. Certains amers ont des précisions sous les 20 centimètres, ce qui explique que l'erreur moyenne soit autour

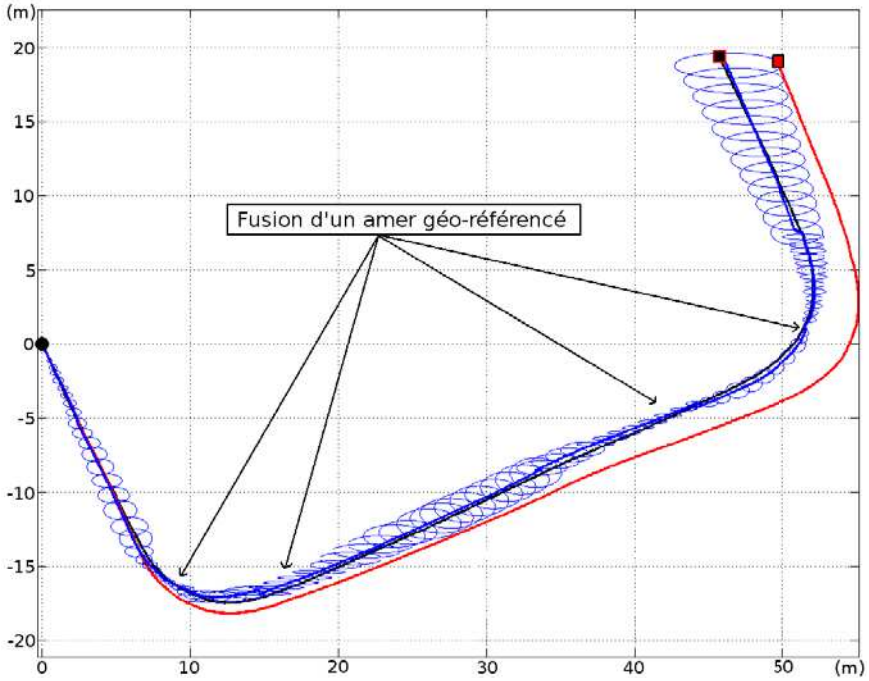


FIGURE 4.25 – Localisation obtenue avec intégration d'amers géo-référencés. Le code couleur est le même que pour la figure précédente. La courbe bleue est celle obtenue quand le biais est estimé par l'utilisation d'amers géo-référencés.

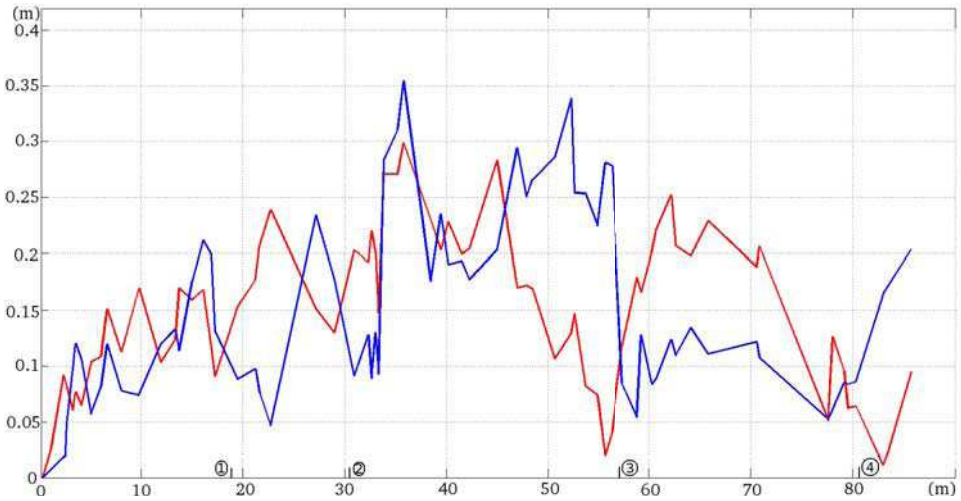


FIGURE 4.26 – Écart de position entre le GPS RTK et le SLAM lors de l'intégration d'amers géo-référencés. En bleu, l'écart longitudinal en fonction de l'abscisse curviligne. En rouge, la même chose pour l'écart latéral. Les numéros indiquent les fusions d'amers géo-référencés.

de cette valeur. Une fusion avantageuse entre deux estimations d'un amer (faible recoupement d'incertitude) contribue également à abaisser l'erreur moyenne de positionnement. L'utilisation de vraies informations géo-référencées ou l'abaissement du seuil de convergence pourrait conduire à une meilleure précision sur la trajectoire. Cette dernière contrainte est malheureusement difficile à satisfaire puisque la convergence des amers seraient plus difficile à obtenir. Néanmoins, avec uniquement 4 informations absolues, la qualité de positionnement obtenue est tout à fait correcte.

Ces résultats montrent l'intérêt que l'on peut avoir à prendre en compte la dérive du SLAM. En effet, la consistance est garantie sur cette trajectoire, ce qui facilite l'intégration d'informations absolues sans risquer de divergence. L'amélioration de la pose du véhicule est très nette sur cet exemple. Il pourrait ainsi être envisageable d'avoir des solutions sur des plus grandes distances utilisant l'infrastructure pour améliorer la localisation du véhicule.

4.4 Remarques

Ce chapitre a présenté un point clef pour la résolution du SLAM multivéhicule : la prise en compte de la dérive. Chaque véhicule divergeant différemment, il est essentiel d'intégrer cette information afin d'aider le processus d'association de données.

Les résultats de ce chapitre ont montré que considérer ce biais de localisation pouvait également améliorer un SLAM classique monovéhicule. L'intégration d'éléments aidant à caractériser la dérive est facilitée par l'architecture mise en place. Cela permet également de maintenir la consistance de la carte et de la pose du véhicule.

Néanmoins, certains aspects pourraient être améliorés. Le premier concerne les sauts visibles lors de l'initialisation d'un nouveau biais. Bien que cela puisse être naturellement corrigé par l'utilisation de plus d'estimées de biais, cette solution n'est pas forcément adéquate car elle peut induire un surcoût en temps de calcul. En effet, la connexion d'un biais au reste de la carte peut devenir coûteuse dès lors que la taille de cette dernière devient conséquente (multiplications intervenant dans le calcul de la mise à jour du filtre de Kalman). Il serait intéressant de s'orienter vers un lissage de la localisation entre les changements de biais. Il faudrait cependant être vigilant à ce que cela n'affecte pas le processus de localisation en faussant l'intégrité du filtre à des points antérieurs de la trajectoire. De plus, le biais est amené à évoluer tout au long de la trajectoire dès que des informations permettent de l'estimer. Cela signifie que lisser les estimées de biais peut complètement changer la rétro-action qui leur sera appliquée avec une nouvelle donnée sur la divergence. Construire un tel algorithme peut alors vite s'avérer difficile.

La modélisation de l'évolution de la dérive introduite ici ne reflète pas réellement son comportement. En effet, il a été constaté de nombreuses fois que la dérive angulaire prédominait par rapport à la dérive de position. De plus, une erreur de cap a une influence considérable sur la position du véhicule. Ainsi, il pourrait être

intéressant de construire un modèle d'évolution liant les biais avec l'orientation du véhicule. De manière similaire, la dérive angulaire est majoritairement causée par les virages alors que dans le cas de la position, cela est davantage provoqué par de longues lignes droites. Notre modèle ne traduit pas cet aspect là. Le comportement du biais serait certainement amélioré avec une telle modélisation.

En l'état actuel, la représentation de la dérive est suffisante. Celle-ci doit être étendue à plusieurs véhicules afin d'en vérifier le bon fonctionnement. De plus, la construction d'un algorithme multirobot a de nombreuses autres contraintes. Celles-ci seront évoquées et traitées dans le prochain chapitre.

Chapitre 5

Extension du SLAM à une flotte de véhicules

Sommaire

L'intérêt d'étendre notre algorithme afin qu'il fonctionne avec une flotte de véhicules tient au fait que les applications qui en découlent sont nombreuses. Bien que ce domaine de recherche soit encore émergeant, certains travaux proposent déjà des solutions coopératives basées sur du SLAM. On peut par exemple citer [Pucihar & Coulton 2011] où les auteurs utilisent également une approche collaborative afin de faire de la réalité augmentée. Cette fois l'objectif est de partager la cartographie entre plusieurs téléphones pour que chacun puisse profiter des informations des autres. Toujours avec des téléphones portables, Barrena et al. proposent dans [Barrena *et al.* 2013] de faire de la reconstruction 3D dense en partageant les images issues de ces téléphones. Les images sont toutes stockées sur un serveur distant où est fait le calcul de la reconstruction. Dans [Kleiner & Sun 2007], Kleiner et al. décrivent une solution prévue pour fonctionner dans des incendies. Les capteurs classiques étant souvent inefficaces dans ce cas, des puces RFID sont utilisées afin de connaître la position des robots évoluant dans cet environnement dangereux. Le but est de déterminer la topologie des lieux avant une intervention humaine. Dans [Lee & Lee 2009] les auteurs proposent de localiser des véhicules sans que ceux-ci ne se voient directement. Pour ce faire, ce SLAM en intérieur se base sur l'observation du plafond par une caméra. Enfin, dans le domaine des ITS, l'approche de Li et al. [Li & Nashashibi 2011] propose un SLAM coopératif mixant vision et laser où les véhicules évoluent en convoi colonne. Afin de pouvoir anticiper les obstacles mobiles, le meneur envoie aux autres la partie de l'image que lui seul perçoit. Le reste de la flotte peut ainsi remplacer le véhicule de tête de l'image et voir les objets mobiles invisibles auparavant. Cela permet d'anticiper des piétons coupant la route par exemple en réagissant de façon adaptée.

Ces quelques applications sont loin d'être les seules existantes. Les algorithmes impliquant plusieurs véhicules touchent l'intégralité des branches de la robotique mobile. En effet, on peut trouver des applications pour la robotique terrestre en extérieur [Li & Nashashibi 2011], en intérieur [Lee & Lee 2009], pour des robots sous-marins [Diosdado & Ruiz 2007], des véhicules aériens [Ong *et al.* 2003], des équipes hétérogènes [Vidal-Calleja *et al.* 2011] ou encore des piétons [Kleiner & Sun 2007]. Même dans le cas où le but applicatif est similaire à celui visé par un seul véhicule, le fait d'utiliser plusieurs robots est un moyen soit d'accomplir des objectifs plus rapidement en explorant de façon séparée [Burgard *et al.* 2002][Burgard *et al.* 2005][Gifford *et al.* 2008], soit de le faire avec une précision supérieure si les véhicules se suivent [Rekleitis *et al.* 2000].

Tous ces exemples montrent bien que les algorithmes impliquant une flotte de véhicules sont des éléments cruciaux pour de nombreuses applications. Dans ce chapitre, nous commencerons tout d'abord par détailler en section 5.1 les approches de SLAM ou de localisation multivéhicule en fonction de quelques points clefs, certains parfois négligés dans la littérature. La section 5.2 décrira ensuite l'approche développée, tout en faisant attention à traiter les points importants soulevés pendant l'étude de l'état de l'art. La section 5.3 parlera de l'algorithme d'association de données construit afin de pouvoir reconnaître les amers cartographiés par les autres véhicules.

5.1 État de l'art du SLAM multivéhicule

5.1.1 Centralisé ou décentralisé

Avant même de commencer à étudier l'algorithme lui-même, il est important de considérer comment les véhicules vont coopérer. Cela signifie qu'il faut déterminer comment ils vont communiquer. Cela est fortement conditionné par l'environnement visé. Deux systèmes s'opposent : les approches centralisées et celles distribuées (ou décentralisées).

Dans le premier cas, toutes les communications sont faites avec une entité qui a pour charge d'agglomérer les données et de les fusionner (voir figure 5.1). Elle doit ensuite partager le résultat avec la flotte de véhicules. Cette entité peut être un des véhicules de l'équipe ou bien encore un ordinateur dédié. Cette organisation est souvent utilisée car elle facilite grandement l'extension d'un algorithme de SLAM à plusieurs véhicules puisqu'il suffit de mettre en place un système de communication et un traitement spécifique pour la fusion centrale. Beaucoup d'approches se sont orientées vers un système centralisé de par sa simplicité d'utilisation. Dans [Gil *et al.* 2010], il s'agit d'un SLAM visuel basé sur un filtre particulaire. Cet algorithme est le premier à avoir proposé un SLAM visuel multivéhicule (ici une paire stéréo). Dans celui-ci toutes les mesures sont envoyées à un agent central qui a la charge de construire la carte. Les positions initiales des robots doivent être approximativement connues afin de pouvoir se localiser dans cette carte commune.

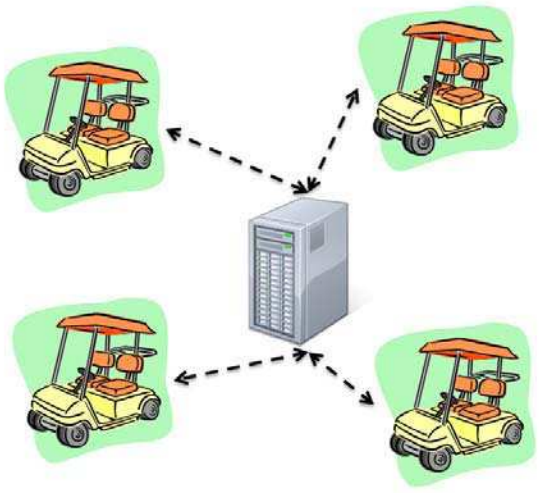


FIGURE 5.1 – Schéma simplifié d'un système centralisé

Les systèmes décentralisés supposent que chaque véhicule est capable de construire sa carte décentralisée tout en communiquant avec les autres robots de la flotte (voir figure 5.2). Un tel schéma est plus compliqué à gérer car il implique que les véhicules vont communiquer et échanger directement ensemble. Il faut donc contrô-

ler le flux d'informations qui circule. Les premiers algorithmes de SLAM décentralisé sont issus de travaux sur la fusion décentralisée. Le principe est le même, il s'agit de distribuer la fusion en répartissant le calcul sur différents processeurs. Certaines approches ont ainsi pu découler naturellement de ces algorithmes. Par exemple, les travaux de localisation multivéhicule de Nettleton et al. dans [Nettleton *et al.* 2006] sont basés sur des avancées significatives pour la fusion décentralisée de capteurs présentées dans [Nettleton *et al.* 2003] et [Durrant-Whyte 2000].

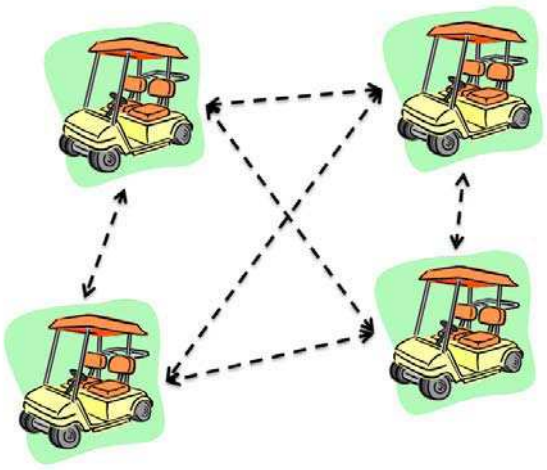


FIGURE 5.2 – Schéma simplifié d'un système décentralisé

Les systèmes distribués sont souvent préférés aux approches centralisées. Bien que plus complexes, ceux-ci ne sont pas affectés par les limitations qu'induit la centralisation des données. En effet, du point de vue de la robustesse, centraliser la construction de la carte est une mauvaise idée. Si la machine ayant la charge de cette tâche vient à tomber en panne, plus aucun véhicule ne pourra accéder à cette information ou encore profiter des nouvelles cartes envoyées par les autres robots. Dans le cas d'une application décentralisée, la panne d'un des véhicules n'empêche pas les autres de continuer à échanger et améliorer la carte commune. Hormis la robustesse, une entité centralisante pose aussi le problème de la quantité de données circulant sur le réseau. Dans le cas décentralisé, seules les informations envoyées par les véhicules passent par le réseau. Pour une approche centralisée, il faut ajouter à cela le retour de l'entité principale chargée d'agglomérer les données. En fonction du nombre de véhicules dans la flotte et de la taille des cartes, ce point peut avoir des répercussions importantes sur la bande passante nécessaire. Enfin, le fait que chaque véhicule doive être à portée de communication de l'unité centralisante est une limitation qui peut devenir gênante si l'on souhaite couvrir de grandes distances. Ce problème n'affecte pas les approches distribuées dans la même mesure puisque chaque véhicule est capable de relayer l'information à ses voisins.

Pour ces raisons, des algorithmes hybrides sont apparus. Ceux-ci sont essen-

tiellement décentralisés mais casse la complexité habituelle de ces approches en centralisant les calculs les plus critiques. C'est le cas de [Bailey *et al.* 2011] où les auteurs proposent une approche de localisation majoritairement décentralisée. Dans celle-ci les véhicules sont équipés de capteurs différents et la fusion de l'état est ainsi faite localement. L'estimation des distances entre les différentes membres de la flotte est, en revanche, faite par un serveur central. De manière similaire, dans [Vidal-Calleja *et al.* 2011], une approche de SLAM décentralisé basée sur des sous-cartes est proposée. Bien que la méthode soit présentée pour être entièrement décentralisée, les expérimentations montrent qu'un ordinateur central est utilisé pour calculer et enrichir la carte topologique globale gardant la trace de toutes les sous-cartes.

L'autre possibilité est de spécialiser la flotte en lui attribuant un meneur ayant un algorithme légèrement différent des autres. Ce cas de figure est extrêmement présent dans les scénarios impliquant des véhicules évoluant en convoi colonne (l'un derrière l'autre). Féraud et al. dans [Féraud *et al.* 2010] propose qu'un premier véhicule construise une carte qui sera ensuite utilisée par un autre (ou possiblement plusieurs). Le but est que des véhicules suiveurs puissent tirer parti des informations recueillies par le premier en se localisant dans la carte précédemment construite. L'objectif final est ici de faire de la conduite automatique. Dans [Avanzini *et al.* 2012], le contexte est similaire puisque les véhicules doivent évoluer dans un convoi colonne classique. Les véhicules suiveurs se servent de la localisation calculée par le véhicule de tête à un passage antérieur afin de pouvoir calculer les trajectoires permettant d'emprunter exactement le même chemin que le meneur.

Afin de produire la solution la plus générale possible, nous nous sommes tournés vers la construction d'un algorithme entièrement décentralisé. Cela signifie que les mêmes traitements sont faits par l'ensemble des véhicules de la flotte. Aucune entité externe n'est utilisée pour centraliser les informations. De même, aucun membre de la flotte n'a le statut de meneur : l'ordre d'un convoi peut donc être bouleversé et l'organisation de la flotte peut changer à tout moment. Ce choix implique que l'échange de données entre les véhicules doit être fait avec attention.

5.1.2 Consanguinité des données

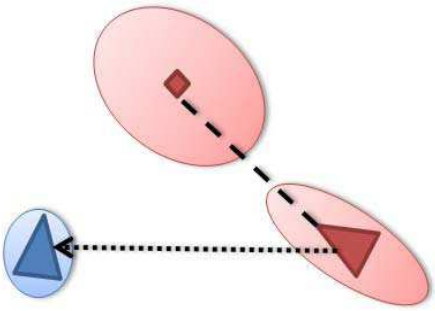
Parmi les difficultés communes à toutes les applications partageant des informations, la consanguinité des données (ou phénomène de rumeurs) est un problème qui doit être traité avec attention. La consanguinité consiste à prendre en compte une même information plusieurs fois, causant ainsi la sur-convergence de l'estimation. Avec les échanges fréquents d'informations par le réseau, le SLAM multivéhicule est particulièrement propice à l'apparition de ce phénomène. Il s'agit alors de contrôler comment les données venant de différents véhicules sont intégrées à l'intérieur du SLAM.

La consanguinité des données n'est pas un problème exclusif au SLAM décentralisé. Comme cela a été constaté dans [McLaughlin *et al.* 2003] et [Hua *et al.* 2010], les approches coopératives de localisation sont aussi affectées. Ces deux papiers pro-

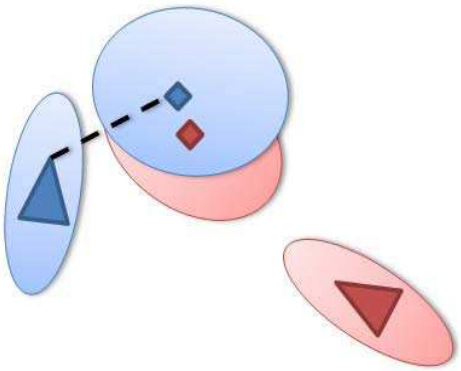
posent de résoudre ce problème en utilisant des états retardés. Ainsi, il devient possible de capturer les dépendances entre les estimées et donc de fusionner les informations en prenant en compte celles-ci. Le retard dans les fusions ne permet pas d'avoir la meilleure estimation de l'état du système à un moment donné. D'autres solutions existent. On peut par exemple citer [Bahr *et al.* 2009] où les auteurs proposent de stocker chaque état nouvellement reçu séparément de l'état courant du véhicule. En évitant de mélanger ainsi les données, le problème de la consanguinité est évité. Néanmoins, le stockage de tous les états peut s'avérer coûteux. Dans [Roumeliotis & Rekleitis 2003], Roumeliotis et al. proposent de résoudre le problème de la consanguinité pour la localisation en utilisant une mesure de pose relative entre les véhicules. L'intégralité de l'état du groupe ne peut être librement échangé. Les deux véhicules liés par la mesure de pose relative peuvent seulement communiquer la partie de l'état les concernant. Cette solution résout le problème de la consanguinité mais n'est pas optimale puisqu'elle n'intègre pas l'ensemble des informations provenant de tous les robots de la flotte. Enfin, concernant les approches de localisation, il est important de citer [Karam *et al.* 2006] où la notion de sous-états est apparue. Dans cette approche, les auteurs conservent les états des différents véhicules séparément, chacun formant un sous-état. La fusion de tous ces sous-états donne l'état global. Ce dernier n'est jamais échangé afin d'éviter la consanguinité. Dès lors qu'un nouveau sous-état est reçu, il vient directement remplacer l'ancien et l'état global est alors re-fusionné. Si chaque véhicule a les mêmes informations que les autres, la carte globale sera la même pour tous et celle-ci sera optimale.

Dans le cadre du SLAM, la consanguinité des données est plus complexe à aborder. En effet, en plus de la pose du véhicule, les différentes estimations des amers doivent également être surveillées. Les fréquentes mises à jour des amers pourraient facilement conduire à des situations de sur-convergence. Le phénomène de rumeurs pour le SLAM est illustré en figure 5.3.

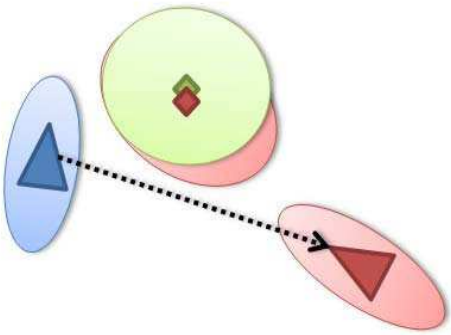
Dans cet exemple, un premier véhicule (rouge) cartographie un amer. Celui-ci est échangé via des communications à un second véhicule (bleu ici). Le véhicule bleu est ensuite capable d'observer cet amer à l'aide de ses capteurs (figure 5.3(b)) produisant ainsi sa propre estimation de l'amer qui est visible en bleu. Après cela, un processus d'association est enclenché et permet d'affirmer que ces deux estimations représentent le même amer. Ces dernières sont donc fusionnées afin de produire une estimation optimale, visible en vert sur la figure 5.3(c). Le véhicule bleu échange ensuite le nouvel amer fusionné au véhicule rouge. Ce dernier n'étant pas conscient que l'estimation reçue intègre déjà sa propre estimation, il déclenche également son algorithme d'association de données qui va fusionner l'amer reçu avec celui déjà dans sa propre carte. L'estimation produite intègre donc deux fois la donnée à la base échangée par le véhicule rouge, réduisant ainsi l'incertitude de ce point plus que cela ne devrait l'être (voir figure 5.3(d)). Cette correction va également affecter la pose du véhicule qui deviendra plus précise mais inconsistante. Dans cet exemple, les vraies positions du véhicule rouge et de l'amer vert (en grises ici) ne sont plus incluses dans les incertitudes calculées. Il y a consanguinité des données et sur-convergence des estimations.



(a) Les triangles représentent des véhicules et le losange un amer. Les ellipses sont les incertitudes associées. L'amer rouge a été cartographié par le véhicule rouge et est ensuite envoyé au véhicule bleu.



(b) L'amer bleu est cartographié par le véhicule bleu. L'amer rouge est celui qui a été reçu précédemment.



(c) L'amer vert représente la fusion entre les amers rouge et bleu. Celui-ci est échangé avec le véhicule rouge.

(d) Le triangle gris est la vraie position du véhicule rouge (même chose pour l'amer gris). Les ellipses en pointillées sont les incertitudes qui devraient affecter le véhicule rouge et l'amer vert. Les petites ellipses rouge et verte représentent l'incertitude après fusion de l'amer vert avec l'amer rouge (consanguinité).

FIGURE 5.3 – Consanguinité dans le SLAM

Cet exemple montre bien à quel point il est facile de mélanger des informations, surtout dans le cas d'amers. Dans une situation à seulement deux véhicules, il serait facile de remédier à ce problème en gardant une trace de ce qui est échangé. Avec n véhicules, cette solution n'est pas envisageable.

La communauté du SLAM a proposé quelques solutions pour résoudre la consanguinité. Dans [Nettleton *et al.* 2006], une architecture réseau dédiée est utilisée afin d'éviter de prendre en compte plusieurs fois la même information. Uniquement des communications point-à-point sont permises, forçant ainsi les informations à transiter par plusieurs véhicules avant d'atteindre l'autre bout du réseau. Aucune boucle n'est également autorisée dans ce réseau. Le but est que chaque véhicule contrôle les données envoyées par son voisin. Pour ce faire, la forme inverse du filtre de Kalman et un algorithme d'intersection de covariances sont employés. Ce dernier permet, à l'inverse de l'EKF, de produire des estimations pessimistes en considérant que les données sont potentiellement liées. Il permet de combiner plusieurs estimations d'une variable dans le filtre de Kalman lorsque les corrélations entre celles-ci sont inconnues. Par ailleurs, le filtre d'information est un choix intéressant pour le SLAM multivéhicule car la mise à jour est additive. Ainsi, l'ordre dans lequel les informations sont intégrées n'a plus d'importance, facilitant de fait la fusion de celles-ci. Le problème est que les résultats produits sont inconsistants dans le cas (finalement fréquent) où les données sont non corrélées. Bien que ces travaux donnent des résultats de simulation encourageants, l'architecture réseau dédiée est un frein important à son utilisation. L'approche de Williams et al. dans [Williams *et al.* 2002b] est intéressante car elle est dérivée d'un SLAM avec sous-cartes. Le principe est le même : chaque véhicule construit et échange ses sous-cartes avec le reste de la flotte. Une carte de plus haut niveau permet de référencer ces dernières dans un même repère. L'arrivée d'un nouveau véhicule dans la flotte déclenchera un processus chargé d'identifier où ses sous-cartes se positionnent dans la carte globale. Cette approche se base sur le fait que les sous-cartes sont uniquement échangées après clôture (et une seule fois) et que celles-ci ne sont jamais améliorées afin d'éviter la consanguinité des données. Les associations trouvées se font uniquement dans la carte globale. Ces derniers points rendent cet algorithme multivéhicule difficile à déployer pour des applications réelles car il ne prévoit pas qu'il puisse y avoir des pertes de données.

Les travaux de Vidal-Calleja et al. présentés dans [Vidal-Calleja *et al.* 2011] proposent également une extension d'un SLAM par sous-cartes. Il s'agit ici du SLAM hiérarchique initialement introduit dans [Estrada *et al.* 2005]. Le principe est similaire. Uniquement la carte globale topologique est échangée entre les membres de la flotte. Chaque événement modifiant cette carte est ainsi initié par les véhicules concernés. À chaque fois, les sous-cartes affectées sont closes, évitant ainsi la consanguinité. La conséquence est que la carte globale métrique n'est pas calculée pendant l'exécution de l'algorithme mais hors ligne. Dans [Cunningham *et al.* 2010], un SLAM multirobot basé sur des graphes est présenté. La carte de chaque véhicule est compressée et envoyée aux voisins. Un cache des données précédemment reçues permet de filtrer et ainsi d'éviter la consanguinité. La carte commune est ensuite calculée entre les voisins en fonction des amers communs. De manière similaire, dans [Aragues *et al.* 2010a][Aragues *et al.* 2010b], un consensus est cherché entre les véhicules voisins afin de trouver la meilleure carte qui évite de prendre en compte deux fois les mêmes informations. Le problème de ces approches reste la limitation de la communication à quelques véhicules voisins.

Enfin, les travaux de Li et al. [Li & Nashashibi 2013] proposent d'utiliser le Split Covariance Intersection Filter (SCIF), tout d'abord initié par Julier et al. dans [Julier & Uhlmann 2001a]. Ce filtre s'appuie sur l'intersection de covariance afin de produire des estimées consistantes, même en présence de corrélations inter-estimées inconnues. Les données de chaque véhicule peuvent ensuite être fusionnées sans craindre la consanguinité des données. Afin de rendre compatible ce filtre dans le cas de la localisation coopérative, les auteurs proposent que chaque véhicule ait son propre état composé de sa pose ainsi que de l'environnement voisin qui sera alors partagé avec le reste de la flotte. Le principe est assez proche de ce qui a été proposé par Karam et al. dans [Karam *et al.* 2006] mais dans le cadre d'un SLAM.

5.1.3 Problèmes de communication

En construisant un algorithme multivéhicule, d'aucuns doivent être prudents à propos des aspects concernant le réseau. En effet, idéalement, une application décentralisée devrait être indépendante du nombre de véhicules du point de vue de la bande passante. Néanmoins, cela revient à utiliser une architecture réseau dédiée qui limite les communications. Au lieu de cette contrainte, nous avons décidé de nous orienter vers une approche où la bande passante nécessaire à chaque véhicule est extrêmement faible, permettant ainsi d'augmenter la taille de la flotte sans que cela ne pose de problème. Ce point a déjà été évoqué précédemment lors du choix de la méthode de localisation (section 2.2). En effet, cette dernière a un impact conséquent sur le flot de données qui va être envoyé.

Les besoins en bande passante sont souvent négligés. C'est le cas par exemple dans l'approche de Williams et al. décrite précédemment [Williams *et al.* 2002b] où des sous-cartes complètes sont envoyées d'un seul coup. Dans [Pfingsthorn *et al.* 2007], les auteurs échangent des graphes dont les noeuds correspondent aux données brutes sorties d'un laser. En augmentant le nombre de véhicules, le nombre de nappes laser à échanger peut rapidement devenir conséquent comme le serait l'envoi des images en sortie de la caméra à tous les autres véhicules. Enfin, Chang et al. dans [Chang *et al.* 2007] propose de n'échanger les cartes que lorsque les véhicules sont capables de se détecter. C'est encore une fois problématique puisque le réseau sera alors énormément sollicité de manière ponctuelle pouvant ainsi conduire à des congestions et donc à des pertes de données. L'aspect volumineux des cartes est résolu dans les approches communiquant exclusivement des cartes topologiques de haut niveau qui elles sont légères. C'est le cas pour Vidal et al. [Vidal-Calleja *et al.* 2011] ou encore Cunningham et al. [Cunningham *et al.* 2010] dont les travaux ont été mentionnés précédemment.

En plus de la bande passante requise, la stratégie d'échange de données entre les véhicules doit prendre en compte de potentielles pertes de communication. Des retards dans la réception des données ou encore une désynchronisation des véhicules sont à intégrer afin de pouvoir pallier les faiblesses du matériel réseau. Quand elles ne sont pas gérées, les coupures dans les communications conduisent à des informations perdues de façon permanente comme c'est le cas dans [Pfingsthorn *et al.* 2007]

et [Williams *et al.* 2002b]. Les algorithmes bâtis autour de communications avec uniquement les voisins sont généralement capables de réclamer des données non reçues car les interactions sont limitées à un petit groupe [Cunningham *et al.* 2010]. Une autre possibilité est celle décrite dans [Martin & Emami 2010] où la notion de Lazy Belief Propagation est introduite. Celle-ci permet d'appliquer les observations dans un filtre particulaire sans se soucier de l'ordre, permettant ainsi d'intégrer des données manquantes tardivement sans que cela ne soit un problème.

C'est également un moyen de traiter les informations affectées d'une certaine latence. Toutes les approches basées sur le filtre d'information peuvent faire de même. En effet, la mise à jour étant additive, l'ordre d'arrivée des données n'a pas d'importance. C'est ainsi devenu un choix privilégié par de nombreux membres de la communauté multivéhicule [Diosdado & Ruiz 2007][Deaves *et al.* 2000][Thrun & Liu 2003] et ce malgré le surcoût induit par la conversion des données en forme inverse.

La difficulté de mettre en place des expérimentations pour plusieurs robots pousse souvent les auteurs à proposer principalement des résultats de simulation. Ainsi, l'aspect réseau est souvent oublié dans les différentes publications du domaine. La plupart du temps, les résultats sont issus de simulations Matlab [Fenwick *et al.* 2002][Hua *et al.* 2010][Cunningham *et al.* 2010][M.D.P. *et al.* 2010] prouvant seulement l'efficacité de méthodes de filtrage spécifiques mais pas la faisabilité d'un SLAM multivéhicule. Cela n'a pas empêché certains travaux d'être déployés à petite échelle [Lee & Lee 2009] (en intérieur) ou encore en environnement ouvert [Resende *et al.* 2012]. Dans ce dernier cas, des véhicules ont évolué pendant 3 mois dans la ville de La Rochelle afin de transporter des personnes sur des chemins bien définis. Néanmoins, la majorité de ces résultats se basent sur du SLAM centralisé. Dans l'approche précédente par exemple, un superviseur était nécessaire afin de s'assurer du bon comportement des véhicules, regroupant de fait une partie des données.

5.1.4 Association de données

L'absence d'expérimentations pour le SLAM multivéhicule est aussi liée à la difficulté de construire un algorithme d'association de données suffisamment robuste pour fonctionner avec des amers provenant de plusieurs sources. Celui-ci a pour but de lier les cartes des différents véhicules et d'affiner globalement la localisation de la flotte. Utiliser uniquement des descripteurs avec une corrélation est loin d'être assez restrictif et entraînerait de nombreuses erreurs d'associations. De plus, la bande passante étant limitée, les images mêmes ne peuvent être échangées. Le fait que les positions relatives des véhicules ne soient pas forcément connues complique encore plus la tâche du processus d'association de données.

Bien que cruciale, l'association de données dans un contexte multivéhicule n'est souvent pas considérée. En effet, il est fréquent que le problème soit évité en fournissant les distances séparant les différents robots de la flotte au début de la trajectoire. C'est le choix qui a été fait dans [Burgard *et al.* 2005] et [Nettleton *et al.* 2006]. Malheureusement, une telle décision limite les applications potentielles du SLAM

multivéhicule. Une approche différente mais avec des limitations semblables est décrite dans [Zhou & Roumeliotis 2006]. Dans celle-ci, il est question de définir un point de rendez-vous vers lequel tous les véhicules vont converger. Ainsi, ils auront la possibilité de s'observer directement afin de pouvoir mesurer leurs distances relatives.

Parmi les processus d'association de données prévus pour des applications à plusieurs véhicules, nous pouvons citer [Thrun & Liu 2003]. Dans cet article, les auteurs proposent d'analyser l'entourage de chaque amer afin de trouver des associations. Pour cela, les 3 amers les plus proches sont considérés. Des mesures de distances et d'angles sont ainsi établies. Grâce à celles-ci, il est possible de rechercher un amer provenant d'un autre véhicule partageant la même configuration. Cette étape est ici accomplie en explorant un arbre de possibilités et en gardant la meilleure correspondance trouvée. Le principal problème de cette méthode est qu'elle requiert que des amers cartographiés par des véhicules différents partagent le même voisinage (les 3 mêmes amers les plus proches). En fonction des capteurs utilisés, de l'environnement ou encore de l'orientation du véhicule au moment de la cartographie, ces suppositions sont difficiles à satisfaire. De manière similaire, dans [Cunningham & Dellaert 2012], les amers sont également regroupés par 3. Une triangulation de Delaunay est ensuite appliquée afin d'obtenir une carte unique. Les périmètres et aires de ces triangles sont ensuite fournis à un algorithme de RANSAC qui a pour but de trouver la meilleure correspondance entre 2 cartes provenant de 2 robots différents. Malheureusement, cette approche souffre du même problème que celle évoquée précédemment : elle s'appuie sur le fait que les mêmes amers seront cartographiés par deux véhicules passant approximativement au même endroit.

Les travaux de Li et al. [Li & Nashashibi 2012a][Li & Nashashibi 2012b] proposent une méthode originale pour résoudre le problème du positionnement initial des véhicules. Il s'agit de mettre en correspondance les cartes de différents robots. Pour ce faire, une représentation sous forme de grille d'occupation est utilisée. Un premier a priori quant à la localisation relative est fourni à l'aide d'un GPS bas coût, ce qui permet de contraindre l'erreur de positionnement à une dizaine de mètres. Ensuite, les grilles sont données à un algorithme génétique qui se charge de trouver le meilleur alignement possible entre les deux cartes. Il est ensuite possible de remonter à la position relative des véhicules. Néanmoins, cette solution est difficilement applicable dans notre cas puisque l'algorithme proposé s'appuie avant tout sur le fait qu'un capteur de distance est disponible. En effet, avec une unique caméra, construire une grille d'occupation qui soit suffisamment caractéristique est une tâche difficile.

Les algorithmes d'association de données capables de résoudre le problème du robot kidnappé (localiser un robot perdu à partir d'une carte et d'observations), même s'ils n'ont pas été développés pour des applications multivéhicules, sont intéressants. Trouver où se situe un robot dans une carte à partir d'un ensemble d'observations est très proche de notre problème, à savoir reconnaître des endroits explorés par d'autres robots. La méthode de JCBB [Neira & Tardós 2002], mentionnée en section 3.4 pour l'association de données monovéhicule, serait également envisageable

dans un contexte multirobot si les distances relatives étaient approximativement connues à l'avance. En effet, sans cette connaissance, cela signifie que les véhicules ne sont pas précisément localisés dans le repère commun et sont donc affublés d'incertitudes importantes. Ainsi, trouver des associations globalement compatibles du point de vue de la covariance n'est en rien discriminant.

Une alternative est le Geometric Constraints Branch and Bound (GCBB) présenté dans [Neira *et al.* 2003]. L'idée est de définir des contraintes géométriques entre des paires d'amers issus de la carte et des observations. Si deux observations partagent une caractéristique commune avec deux amers (distance les séparant par exemple) alors une association est possible. Bien évidemment, plus de 2 observations validant une contrainte seront nécessaires pour assurer une correspondance. Pour cela, un arbre des paires possibles est établi et exploré. À la fin, le sous-ensemble de correspondances le plus probable est retourné. L'avantage principal de cette méthode est que les contraintes géométriques sont robustes dès lors qu'elles sont validées par un ensemble d'amers, ce qui est le cas ici. L'algorithme de GCBB peut facilement être adapté à des associations entre véhicules. Pour cela, il suffit simplement de considérer les amers d'une seconde carte plutôt que des observations. Le deuxième aspect intéressant de cette approche est que seulement quelques amers doivent être communs entre deux cartes pour trouver une correspondance viable. L'algorithme récursif du GCBB est présenté en figure 5.4.

L'étude de la littérature montre que des palliatifs aux différents points soulevés existent. Néanmoins, le SLAM multivéhicule semble manquer d'une solution unifiée permettant de résoudre tous les problèmes à la fois. Peu d'approches ont donc pu être testées dans des conditions réelles. La section suivante va présenter une solution au problème du SLAM décentralisé traitant les aspects clefs évoqués ici.

5.2 Conception d'un algorithme de SLAM décentralisé

L'état de l'art a montré que la conception d'un algorithme décentralisé devait s'articuler autour d'un certain nombre de points cruciaux. D'autres aspects, pas forcément cités dans la section précédente, sont également à prendre en compte. On peut notamment penser à comment gérer le fait que les positions initiales des véhicules soient inconnues ou encore, dans notre cas, comment intégrer la dérive de chaque véhicule dans une architecture décentralisée. Bien sûr, la solution globale devra également éviter la consanguinité des données, tout en traitant les problématiques liées à la communication évoquées précédemment.

Afin de faciliter la compréhension, nous commencerons par présenter l'architecture générale de l'application dans la sous-section 5.2.1. Il s'agit de l'extension de l'organisation exposée dans le chapitre précédent et qui permettait de prendre en compte la dérive du SLAM. Sa version décentralisée devra conserver la généricité qu'a cette dernière afin que n'importe quel SLAM puisse être utilisé à la place de l'approche monoculaire employée ici. C'est d'ailleurs la première fois à notre connaissance qu'un algorithme décentralisé dans un contexte purement mo-

```
% appelée avec : GCBB([ ], 0)
procedure GCBB(H, i)
    % H : hypothèse courante
    % i : nouvel amer à tester
    % m : nombre de nouveaux amers à tester
    % n : nombre d'amers dans la carte
    if i ≥ m then
        % meilleure association trouvée ?
        if NbCorrespondances(H) > NbCorrespondances(best) then
            best = H
        end if
    else
        for j = 0 à n − 1 do
            if contrainteUnaire(i, j) & contrainteBinaire(i, j, H) then
                % amers i et j associés
                GCBB([H j], i + 1)
            end if
        end for
        % peut faire mieux ?
        if NbCorrespondances(H)+m − (i + 1) > NbCorrespondances(best) then
            % correspondance (i,j) refusée
            GCBB([H −1], i + 1)
        end if
    end if
end procedure
```

FIGURE 5.4 – Algorithme GCBB

noculaire est présenté. La sous-section 5.2.2 montera comment il est possible de considérer les positions initiales des véhicules afin de construire une carte commune. Enfin, la sous-section 5.2.3 illustrera l'efficacité de notre méthode avec quelques résultats préliminaires ne prenant pas en compte la dérive du SLAM et sans algorithme d'association de données spécifique. Les expérimentations générales feront l'objet du prochain chapitre. Ces travaux ont fait l'objet de deux publications [Bresson *et al.* 2012a][Bresson *et al.* 2013a].

5.2.1 Architecture générale

5.2.1.1 Notion de sous-cartes pour le SLAM décentralisé

L'objectif de cette architecture est triple : sa première fonction est d'éviter la consanguinité des données. Une gestion particulière des données provenant des autres véhicules est nécessaire afin d'éviter de prendre en compte une même information plusieurs fois. Nous considérons en effet que le filtre fondé sur l'intersection de covariance est sous-optimal car pessimiste. Cette architecture doit également per-

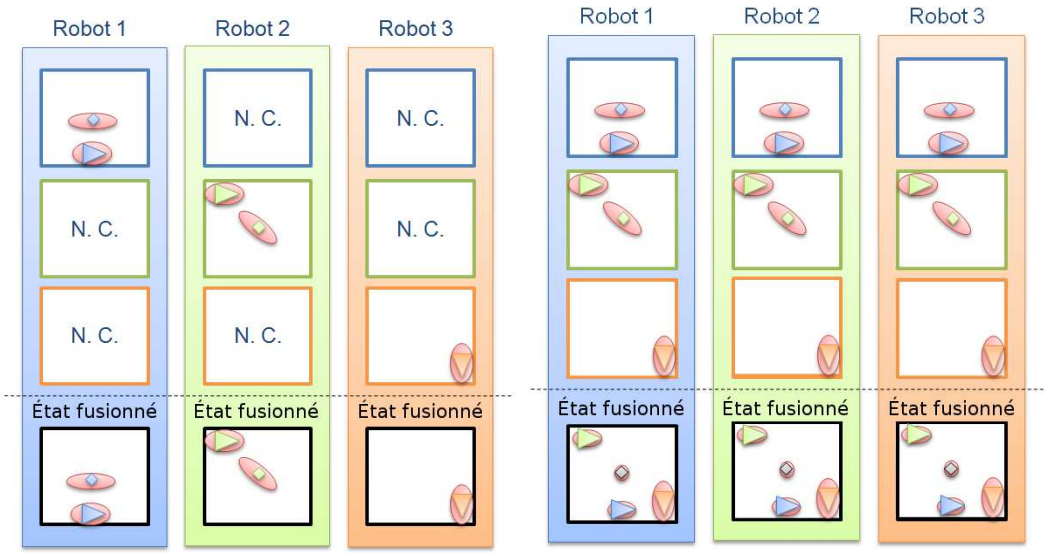
mettre de contrer la majorité des problèmes liés à la communication. En effet, des mécanismes structurels peuvent être mis en place afin de ne pas être affecté par des retards dans les échanges tout en évitant de surcharger le réseau. De même, une perte de données causée par des ruptures dans la communication ne doit pas être critique et les informations perdues doivent pouvoir être récupérées ultérieurement. Enfin, le dernier objectif derrière la construction d'une telle architecture est de conserver les avantages de la structure présentée dans le chapitre 4. Cette dernière permet la gestion de la dérive naturelle du SLAM et offre la possibilité d'utiliser n'importe quel algorithme de SLAM bas niveau. Ces aspects sont importants et nous considérons qu'ils doivent toujours être présents dans l'architecture multivéhicule.

Concernant la consanguinité des données, la stratégie fréquemment employée consiste à ne jamais échanger des informations déjà fusionnées. C'est ce qui est fait dans [Karam *et al.* 2006] où les états des véhicules sont séparés en sous-états. L'idée est que ces derniers peuvent être librement échangés sans avoir à craindre une quelconque sur-convergence. Quand un sous-état est reçu, s'il est plus récent que celui disponible, il le remplace (les messages réseau sont datés). Les sous-états sont fusionnés régulièrement afin d'obtenir un état global qui tire parti de toute l'information disponible. Cet état fusionné n'est jamais envoyé aux autres véhicules. Néanmoins, chaque véhicule partage les mêmes sous-états avec l'ensemble de la flotte, le rendant ainsi capable de calculer le même état global que les autres.

Bien que conçue pour de la localisation multivéhicule, cette méthode peut être adaptée afin d'être pertinente dans un contexte de SLAM. Le changement le plus important est que les cartes des véhicules doivent être envoyées, en plus de leur pose. La différenciation de l'état global et des sous-états doit également être faite avec les cartes. Ainsi, la carte de chaque véhicule devient une sous-carte (terme à distinguer de la technique présentée en 2.2.2.2) dans le traitement décentralisé, la carte globale étant la fusion de toutes les sous-cartes. Un exemple à 3 véhicules du fonctionnement de cette stratégie est visible en figure 5.5.

Dans cet exemple, sans communication (figure 5.5(a)), la carte globale obtenue par chaque véhicule est logiquement sa propre carte. Après l'avoir partagée avec les autres, la fusion des sous-cartes va permettre de les exprimer dans un même repère. De cette façon, trouver des amers communs entre les sous-cartes permettra de corriger la dérive et également d'estimer les distances relatives entre les véhicules. Dans l'exemple de la figure 5.5, deux véhicules (le bleu et le vert) possèdent une estimation du même amer. La fusion des sous-cartes va améliorer cette estimation sans pour autant la rendre consanguine. La carte globale obtenue ne sera, quant à elle, jamais échangée. On peut constater qu'une fois toutes les cartes communiquées (5.5(b)), même le véhicule orange (robot 3), qui n'avait pas d'amer dans sa carte, est capable d'avoir l'estimation améliorée de l'amer partagé par les autres. Les cartes globales calculées par chaque véhicule de manière décentralisée sont optimales dès lors que toutes les informations sont partagées.

Cependant, les cartes grossissent en taille et s'améliorent au cours du temps, rendant le mécanisme de remplacement décrit dans l'approche de Karam et al. [Karam *et al.* 2006] inapplicable ici. En effet, envoyer un amer à tous les véhicules



(a) Avant communication (b) Après communication

FIGURE 5.5 – Séparation en sous-cartes afin d'éviter la consanguinité. La sous-carte du robot 1 est celle encadrée en bleu. Celle du robot 2 est en vert et celle du robot 3 en orange.

d'une flotte à chaque fois qu'il est mis à jour n'est pas une technique viable. En plus d'un surcoût calculatoire pour la fusion et l'intégration de celui-ci dans la carte globale, cela induirait une surcharge du réseau. Le fait de remplacer un amer dans sa sous-carte nécessite de reconstruire la carte décentralisée avec les nouvelles informations. Or cette étape peut avoir un coût non négligeable. À cause de cette contrainte, nous avons décidé de conserver le mécanisme de l'architecture de biais qui consistait à n'envoyer dans le haut niveau (ici, la sous-carte du véhicule) que les amers précis. Comme précédemment, les mises à jour faites dans le bas niveau ne seront pas reportées dans les sous-cartes. Cela représente néanmoins une perte d'information faible car les amers transférés dans les sous-cartes sont déjà précis.

Cette méthode permet également d'éviter de casser la carte globale. En effet, uniquement les nouveaux amers (venant du bas niveau ou arrivant par le réseau) seront ajoutés, complétant ainsi la carte décentralisée. Cela renforce la séparation établie au cours de la section 4.2. L'algorithme de SLAM bas niveau peut utiliser n'importe quelle méthode de filtrage ou capteur du moment qu'il fournit la pose du véhicule, les amers de la carte et les incertitudes associées à ces estimations, ce qui permet de conserver le caractère générique de l'approche. L'exemple ayant précédemment servi à expliquer l'organisation de l'architecture (figure 4.17) est ici repris et complété dans la figure 5.6 pour un cas à 2 véhicules.

La figure 5.6 montre clairement que la structure précédemment développée est conservée. Le bloc gris en haut représente toujours le SLAM classique bas niveau. Comme dans l'exemple précédent, l'état est composé de la pose du véhicule et de

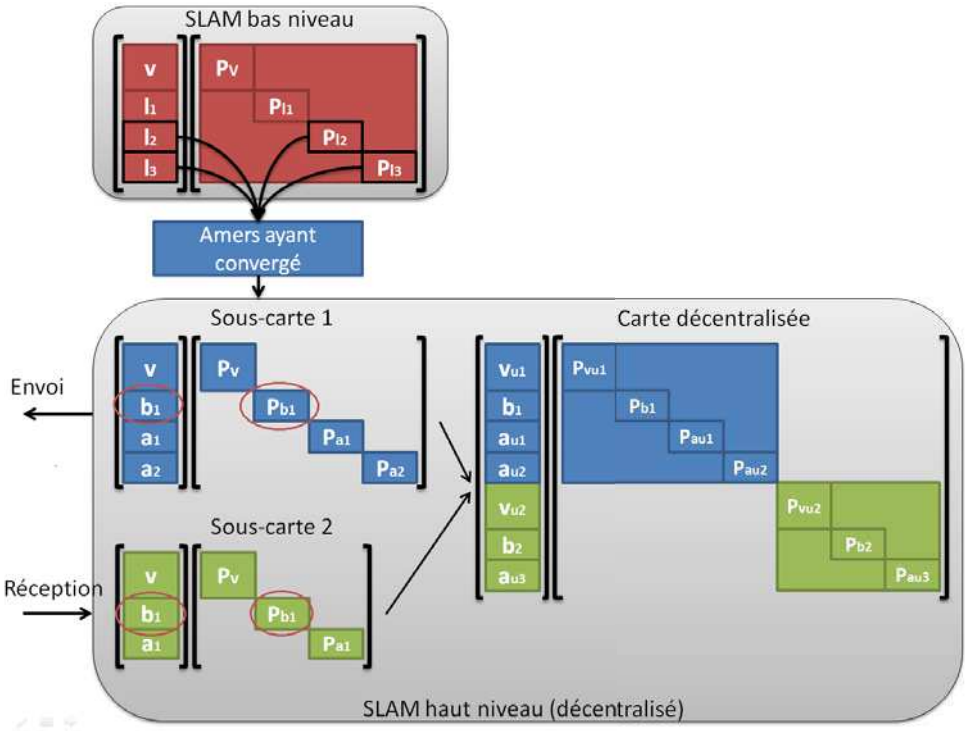


FIGURE 5.6 – Architecture de SLAM décentralisée pour un exemple à 2 véhicules

3 amers. Deux d'entre eux ont d'ailleurs convergé (ceux encadrés en noir) et sont envoyés au haut niveau avec la pose du véhicule. Cela permet de créer la sous-carte 1 (sous-carte bleue). Une nouvelle estimée de biais est alors initialisée afin de représenter la dérive affectant ces amers. Celle-ci est également insérée dans la sous-carte (entourée en rouge sur la sous-carte bleue). Cette dernière peut être diffusée aux autres véhicules de l'équipe sans craindre la sur-convergence. Dans cet exemple, un second véhicule a communiqué sa sous-carte (en vert ici). La pose de ce véhicule a été reçue avec un amer et l'estimation du biais les affectant (entouré en rouge dans la sous-carte verte). Comme pour le premier véhicule, la réception de ces données entraîne la création d'une sous-carte spécifique au véhicule vert. Si ce robot avait déjà échangé des informations, la sous-carte aurait déjà été disponible et aurait seulement été augmentée des nouveaux amers et d'une nouvelle estimation du biais si cela s'avérait nécessaire.

Dans tous les cas, en fusionnant les deux sous-cartes disponibles avec un filtre de Kalman étendu, la carte globale décentralisée (partie droite du bloc gris du bas sur la figure 5.6) est obtenue. Cette fusion correspond en fait à la prise en compte du biais telle que décrite précédemment. Le but est ici de connecter l'état d'un véhicule au biais l'affectant. Néanmoins, au lieu de le faire directement comme cela était le cas précédemment, la couche de sous-cartes a été ajoutée afin d'éviter la consanguinité. La sous-carte d'un véhicule ne correspond donc qu'à un espace de

stockage puisqu'aucun traitement n'est fait sur celle-ci. La réception d'un nouvel amer par le réseau n'implique que son stockage comme étape supplémentaire par rapport à ce qui a été présenté auparavant. L'intégration de son biais dans la carte globale est strictement identique au processus décrit dans le chapitre 4.

On peut noter dans la covariance de la carte décentralisée que les informations concernant chaque véhicule sont reliées entre elles (bloc bleu et bloc vert dans la matrice de covariance). Par conséquent, l'intégration de données absolues ou encore la détection d'une fermeture de boucle est toujours possible. Cette séparation forte entre les 2 véhicules dans la covariance est normale. Tant qu'aucun amer commun n'a été trouvé, il n'est pas possible de lier ces deux véhicules. C'est le but de l'algorithme d'association de données qui sera détaillé en section 5.3. De manière simplifiée, chaque nouvel amer, avant d'être ajouté à la carte décentralisée, est testé afin de vérifier s'il peut s'associer avec un autre amer déjà présent dans la carte ou non. Si c'est le cas, celui-ci sera alors fusionné avec l'amer existant via le filtre de Kalman. Cette fusion est identique à une fermeture de boucle. En effet, dans les deux cas (fusion d'amers et fermeture de boucle), la mise à jour du filtre de Kalman se chargera de répercuter la mise à jour sur les biais concernés et donc sur le reste de l'état. Dans le premier cas, la valeur du biais d'un véhicule pourra être estimée. Dans le second cas, la position des véhicules dans le repère commun de la carte globale sera affinée et des liens apparaîtront alors dans la covariance, traduisant ainsi la relation entre les deux véhicules. La connexion entre ces deux robots fera que n'importe quelle information venant affecter l'un, aura également un impact sur l'autre.

Tout comme pour l'intégration de la dérive, il n'est pas nécessaire ici d'envoyer les covariances croisées. En effet, les liens entre les amers étant faibles en sortie du SLAM monoculaire, ils en deviennent négligeables. De plus, la divergence commune est bien plus forte. Ainsi, la connexion des amers au biais permet de récupérer naturellement les covariances croisées.

5.2.1.2 Prise en compte de l'aspect réseau

Les applications réelles impliquent d'être robuste contre les échecs du réseau. Concernant les désynchronisations, celles-ci sont principalement causées par le fait que chaque capteur délivre des informations à une vitesse particulière. Dans un contexte décentralisé, il est aisé de comprendre que le nombre de véhicules pourrait potentiellement mener à une situation ingérable si toutes les données issues des capteurs transitaient par le réseau. Dans l'architecture proposée ici, cela n'est pas le cas puisque le bas niveau filtre déjà les données et ne transmet que les informations de plus haut niveau (amers au lieu de mesures brutes). La gestion de la désynchronisation est alors facilitée puisqu'il suffit désormais que les véhicules partagent une base de temps commune. Pour ce faire, nous avons utilisé le Network Time Protocol (NTP). Les données échangées sont datées et il devient alors possible, avec un modèle à vitesse constante par exemple, d'avoir une estimation de la pose de chaque véhicule dès que cela est nécessaire (par exemple au moment d'une fusion

d'amers). Dans tous les cas, les amers étant relayés une seule et unique fois depuis le bas niveau, une seule version de chaque amer va transiter entre les véhicules. Ainsi, peu importe la désynchronisation, les amers ne seront pas concernés par celle-ci.

Les latences qui peuvent affecter les informations envoyées ne sont plus un problème avec la stratégie d'échange mise en place. En effet, comme cela a été expliqué précédemment, les amers passent du SLAM bas niveau au côté décentralisé une seule fois. Cela signifie qu'une seule version de chaque amer existe et sera donc envoyée aux autres véhicules. En conséquence, le moment où il est reçu importe peu car il ne sera plus mis à jour. Si un amer est réceptionné deux fois (relayé par un troisième véhicule par exemple), il ne sera simplement pas considéré la seconde fois. Pour ce faire, nous avons décidé d'envoyer chaque amer avec un identifiant unique. Celui-ci est composé du véhicule dont il est originaire ainsi que de son ordre d'arrivée dans le haut niveau. Le système est ainsi capable de distinguer les amers. Si jamais un amer est reçu avec du retard, son intégration ne posera pas de problème puisque sa position est définitive et n'est donc pas liée à une date particulière. Concernant la pose véhicule, celle-ci est toujours datée. La pose la plus récente est utilisée et remplace la précédente. Ainsi, si une pose véhicule est reçue tardivement, soit la pose courante sera plus récente, auquel cas la pose retardée ne sera pas utilisée, soit ce n'est pas le cas et alors la pose affectée d'une latence est employée. Un modèle d'évolution basé sur la vitesse linéaire et angulaire du véhicule est utilisé pour prédire la pose courante du robot à partir de la donnée retardée.

L'indice unique par amer a également l'avantage de permettre la gestion des pertes de données suite à des ruptures dans la communication. L'indice est incrémenté à chaque fois qu'un nouvel amer est reçu dans le haut niveau. Ainsi, il sera facile pour les véhicules distants de constater une perte de données. À la réception d'un amer par le réseau, le véhicule aura juste à vérifier si l'indice de l'amer reçu est bien consécutif à celui reçu juste avant. Si cela n'est pas le cas, une requête peut être envoyée au véhicule concerné qui répondra avec les amers manquants. Ainsi, même après un arrêt des communications de plusieurs secondes, tous les véhicules seront à même de récupérer l'intégralité des données perdues. De plus, chaque véhicule peut relayer, sans risque de consanguinité, les données provenant des autres véhicules afin d'accélérer la récupération des données.

L'extension de l'architecture mise en place pour contrer la dérive du SLAM est assez naturelle. La structure de base, en plus de gérer la dérive inhérente à chaque véhicule, avait en effet été pensée afin de pouvoir facilement intégrer les données de plusieurs robots correctement. De plus, une attention particulière a été portée aux communications réseau. Ainsi, nous pouvons éviter les problèmes qui affectent la plupart des approches décentralisées. Il faut maintenant pouvoir prendre en compte le fait que les positions initiales des véhicules de la flotte peuvent être inconnues.

5.2.2 Positions initiales inconnues

Le modèle dynamique du biais, présenté en section 4.2, convient tout à fait aux besoins de l'application décentralisée puisque chaque véhicule dérive indépendamment des autres. Cependant, il est en plus nécessaire d'exprimer l'état global de la flotte (véhicules, amers et biais) dans un repère commun. En effet, dans le bas niveau (et la sous-carte correspondante), tout est exprimé par rapport à un repère local initialisé sur le point de départ de la trajectoire.

Le moyen le plus simple de contourner ce problème serait que chaque véhicule exprime dans son propre repère de référence l'équipe complète. Néanmoins, pour ce faire, il est requis de connaître les distances séparant les véhicules. Considérer cette connaissance comme un pré-requis limite drastiquement les applications potentielles. Au lieu de forcer le système à démarrer avec des positions connues, nous avons décidé de tirer avantage de cette connaissance seulement si elle est disponible.

L'intégration se fait sous la forme d'un biais statique affectant le véhicule. Celui-ci correspond en fait à la valeur initiale du biais. Si la distance entre un véhicule et un repère commun à toute la flotte est approximativement connue, cette information peut être utilisée pour initialiser la première estimation du biais insérée dans la partie décentralisée du SLAM. Les approximations peuvent être prises en compte dans la matrice de covariance associée. Prenons un exemple où deux véhicules évoluant dans une zone carrée de 10 mètres par 10 mètres. Si nous définissons le repère de référence comme étant le centre de cette aire, chaque covariance de biais pourra être initialisée avec une incertitude circulaire ayant un diamètre d'un peu plus de 14 mètres ($10\sqrt{2}$) afin d'être certain que cela inclura la vraie position du véhicule par rapport à ce repère commun (voir véhicule rouge de la figure 5.7). Avec un a priori sur la distance au repère commun, il est possible de réduire cette incertitude. C'est le cas du véhicule bleu qui a un biais statique de 8 mètres de rayon. En l'absence de connaissances précises, les deux véhicules de la figure 5.7 ont tout deux leur estimée initiale positionnée sur l'origine du repère commun.

L'avantage principal d'intégrer une part statique au biais est de représenter n'importe quel a priori disponible avant le commencement d'une trajectoire, que cela soit une valeur ou une incertitude. Un autre aspect essentiel est la préservation de la consistance. En effet, même sans connaissance a priori sur la distance entre les véhicules, les matrices de covariance peuvent toujours être initialisées avec des valeurs infinies. Cela offre également un avantage lors de la conception d'un algorithme d'association de données comme cela sera expliqué dans la section 5.3.

Deux types d'initialisation de biais sont possibles. Ils sont exposés sur la figure 5.8 dans un exemple à deux véhicules. Dans la figure 5.8(a), le repère de référence pour la flotte est celui du véhicule bleu. Avec la distance initiale entre les véhicules bleu et vert inconnue, exprimer le robot vert dans le repère commun lui fera penser qu'il démarre sa trajectoire sur le robot bleu. Cependant, avec une matrice de covariance appropriée, la vraie position sera toujours incluse. Le véhicule bleu commencera en revanche avec une estimée de biais ainsi qu'une covariance nulles. Dans le second exemple (figure 5.8(b)), l'objectif est le même : maintenir l'intégrité. Ici, le repère

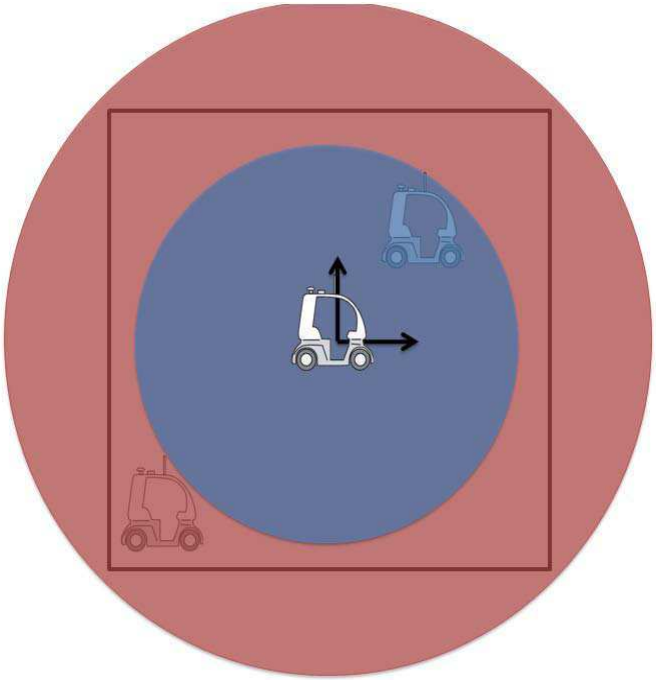
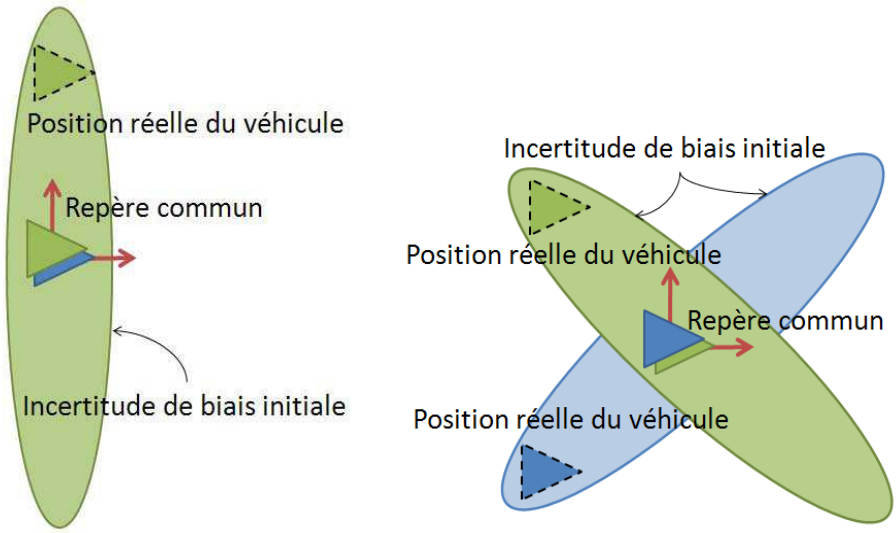


FIGURE 5.7 – Exemple d'inclusion d'un a priori dans le biais. Les véhicules en transparence représentent les vraies positions des deux véhicules estimées sur le repère commun. L'incertitude bleue couvre le véhicule en haut à droite et la rouge celui en bas à gauche.

de référence de la flotte n'est pas celui d'un des véhicules. Cela signifie que chaque véhicule devra commencer avec une valeur initiale de biais non nulle ou, comme dans cet exemple, avec une incertitude initiale couvrant sa pose de départ.

Quel que soit le type d'initialisation, l'intégration des informations disponibles a priori est aisée. Dans les résultats qui seront donnés dans ce manuscrit, nous avons choisi de définir le repère commun toujours comme le point de départ de l'un des véhicules (figure 5.8(a)) car cela permet de commencer avec un véhicule déjà précisément localisé dans le repère commun. Il serait aussi possible d'utiliser la représentation de la figure 5.8(b) avec comme repère commun le référentiel WGS84 du GPS. Cela permettrait de faciliter l'intégration de ce capteur dans la localisation car l'état décentralisé serait déjà fourni dans ce même repère.

Les associations entre amers de différents véhicules vont permettre d'estimer plus précisément ce biais initial et donc les distances relatives entre les membres de la flotte. Afin de valider l'aspect statique du biais, nous avons conduit deux simulations qui seront détaillées dans la sous-section suivante.



(a) Repère commun basé sur le repère local du véhicule bleu

(b) Repère commun indépendant des repères locaux

FIGURE 5.8 – Initialisations possibles pour le biais. Les triangles en pointillés sont les vraies positions des véhicules. Les triangles pleins représentent les estimations des positions. Les ellipses sont les incertitudes associées. Le repère commun est en rouge.

5.2.3 Résultats préliminaires

Les résultats qui vont être présentés ont tous été obtenus grâce à un simulateur (le même que précédemment) pour des raisons de commodité. Encore une fois, l'environnement de la place de Jaude a été utilisé afin de mener ces scénarios. Il est important de rappeler que le simulateur sert uniquement à générer des données capteur. Dans ces résultats, deux véhicules ont été employés. Les traitements SLAM de chaque robot sont faits sur des ordinateurs séparés avec de réelles communications réseau. Pour ces deux trajectoires, chaque véhicule était équipé d'une caméra fonctionnant à 10 Hz ainsi que d'un odomètre et d'un capteur d'angle de braquage. Les véhicules se déplaçaient encore une fois à environ 2 mètres par seconde. Côté SLAM monoculaire, 20 amers étaient suivis par image et seulement ceux ayant convergés ont été utilisés dans l'algorithme décentralisé. Dans ces simulations, le biais dynamique n'était pas utilisé. Le but était de tester la phase d'initialisation avec des incertitudes importantes ainsi que le bon fonctionnement de notre architecture. L'association de données en place ici est la même que celle utilisée dans le SLAM monoculaire. Le but est de tester la robustesse que peut avoir un tel algorithme dans un contexte décentralisé.

La première trajectoire, présentée en 5.2.3.1, voit deux véhicules évoluer en colonne. Pour la seconde trajectoire, illustrée quant à elle en 5.2.3.2, deux véhicules avancent côte à côte, dans ce que l'on appelle un convoi ligne.

5.2.3.1 Convoi colonne

Concernant le convoi colonne, l'idée est de voir si les véhicules sont capables d'estimer la distance les séparant sans utiliser d'observations directes mais uniquement des amers communs. Sur cette trajectoire, le véhicule de queue parcourt 70 mètres et le véhicule de tête 85 mètres. Les robots sont à l'origine séparés par environ 5 mètres. Ceux-ci vont évoluer en ligne droite pendant toute la trajectoire. Le repère global est fixé sur le véhicule de tête. Cela signifie que le véhicule de queue débutera sa trajectoire en pensant qu'il est situé à la même position que le robot de tête. Pour ce véhicule de derrière, l'initialisation du biais est faite de manière à intégrer sa position réelle avec une incertitude circulaire de 7 mètres de rayon. En revanche, le véhicule de devant étant parfaitement localisé au départ, il n'est affublé d'aucun biais.

Au cours de cette trajectoire, le véhicule de queue a cartographié 61 amers alors que celui de tête en a conservé 71 dans sa carte. Ce sont ces amers qui sont échangés au fil de la trajectoire. Les trajectoires accomplies par les deux véhicules, ainsi que les amers, sont visibles sur la figure 5.9.

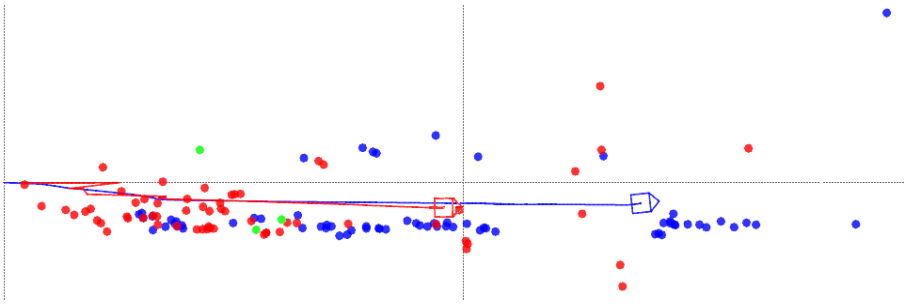


FIGURE 5.9 – Trajectoire en convoi colonne. La ligne rouge est la trajectoire du véhicule de queue et la ligne bleue celle accomplie par celui de tête. Les amers suivent le même code couleur. Les amers en vert sont ceux associés et donc communs aux deux véhicules.

Nous pouvons constater que le véhicule de queue a bien été capable d'estimer la distance le séparant du meneur. En effet, après quelques mètres, on peut voir que la position du véhicule est corrigée et ramenée derrière le véhicule de tête. En revanche, uniquement 3 amers communs ont été trouvés. Cela est dû au fait que les seuils servant à certifier une correspondance ont été durcis afin d'éviter de faux appariements. L'algorithme d'association n'est donc pas adapté à une utilisation multivéhicule.

Malgré ces difficultés, les quelques associations trouvées ont permis d'estimer correctement le biais statique. La qualité des résultats de localisation est indiquée dans la table 5.1 et ce après chaque association faite. Pour ce faire, la valeur estimée du biais statique est comparée, après chaque association, à la vraie distance séparant les véhicules au début de la trajectoire (donnée fournie par le simulateur).

	x	y	z	Ψ	Φ	Θ
Association 1	3,1282	1,0247	0,2300	0,0369	0,0110	0,0815
Association 2	1,9790	0,3605	0,2357	0,0168	0,02033	0,0113
Association 3	0,3123	0,2599	0,2081	0,0152	0,0207	0,0074

TABLE 5.1 – Erreur quadratique moyenne du biais pour le convoi colonne

Nous pouvons constater que chaque association a contribué à affiner l'estimation du biais. Après les 3 associations, l'erreur de position est d'environ 25 centimètres pour les 3 axes. L'erreur d'orientation quant à elle est en dessous de 0,01 radian. Il est important de rappeler que le seuil utilisé pour la convergence des amers est toujours fixé à 50 centimètres et influe donc grandement sur la qualité des mesures obtenues. Néanmoins, les résultats sont satisfaisants.

Les temps de calcul induits par la surcouche décentralisée sont très faibles. Cela est dû au fait que la réception, l'envoi et la fusion d'informations se font en parallèle. Les temps obtenus sont affichés sur la figure 5.10. Les temps sont donnés pour chaque appel fait à la fonction concernée (envoi, réception, fusion), ce qui signifie que l'algorithme décentralisé est inactif la majeure partie du temps. Le pire cas observé fait état de 100 μs de temps de traitement (moins de 1% du temps total de calcul). Il est cependant important de garder en tête que l'association de données et la gestion du biais dynamique ne sont ici pas incluses.

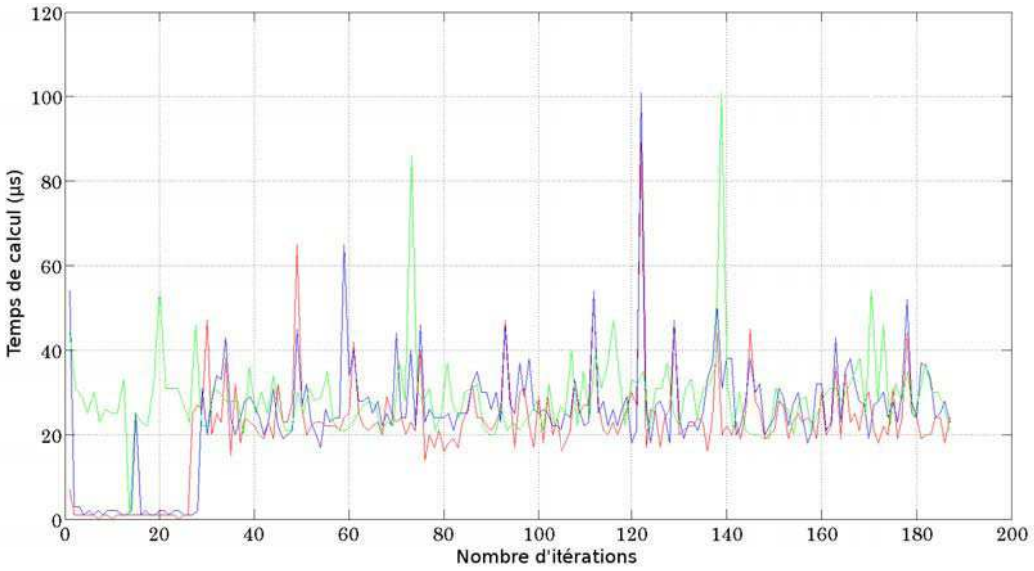


FIGURE 5.10 – Temps de calcul requis par le processus décentralisé. La courbe rouge est le temps nécessaire à l'envoi des amers. La bleue est celle indiquant le temps consacré à la réception de ceux-ci. Enfin, la courbe verte est le coût calculatoire de l'étape de fusion décentralisée.

Enfin, la bande passante requise pour cette simulation était relativement faible (voir figure 5.11). Chaque véhicule envoyait moins de 2 ko de données par seconde. Encore une fois, il est important de garder en tête que le biais n'est ici pas communiqué et que le nombre d'amers cartographiés est peu conséquent. Néanmoins, ces résultats démontrent la faisabilité de notre approche.

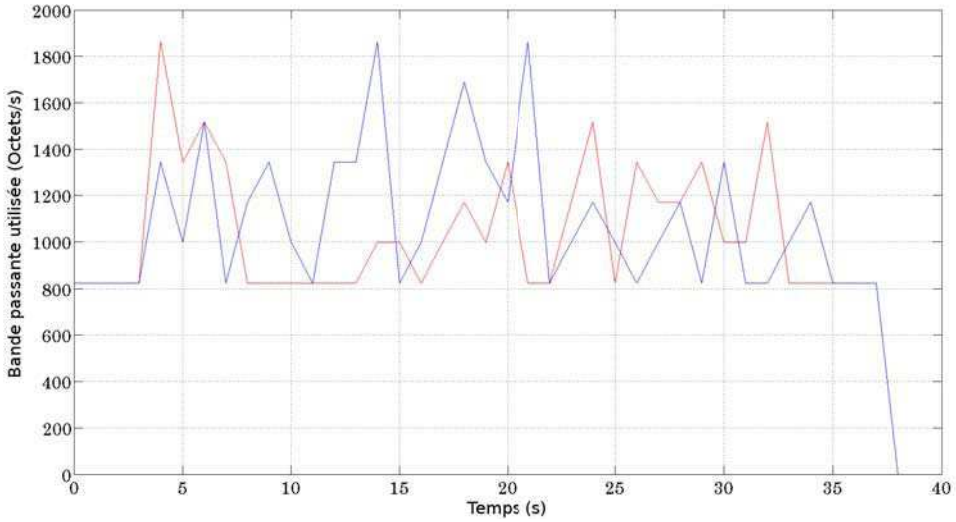


FIGURE 5.11 – Bande passante utilisée lors de la trajectoire. La courbe rouge est la bande passante utilisée par le véhicule de queue. La courbe bleue représente celle consommée par le véhicule de tête.

5.2.3.2 Convoi ligne

Le dernier scénario testé dans cette section présente deux véhicules se déplaçant côte à côte. Cette simulation est intéressante car il est impossible pour les véhicules de s'observer directement et l'estimation de l'état de la flotte ne peut être faite qu'en trouvant des points de l'environnement en commun. Par rapport à l'approche colonne précédente, les amers communs vont ici être situés dans le champ recouvrant des caméras des deux véhicules plutôt que partout dans l'image. La difficulté est donc de trouver des points communs en bordure de l'image.

Concernant les trajectoires, les deux véhicules ont approximativement parcourus 90 mètres chacun en ligne droite. Encore une fois, le véhicule rouge (de gauche) est celui qui démarre avec un biais statique puisqu'il se localise en fonction du repère initial du véhicule bleu (de droite) dont la position est inconnue. Le robot de droite est quant à lui bien localisé et n'est donc pas affecté par un biais statique. Les deux robots sont séparés par environ 2,5 mètres au début de la simulation. L'incertitude du biais statique a été fixé en fonction de cela afin de former un cercle d'environ 6 mètres de diamètre pour le véhicule rouge.

Ce dernier a pu cartographier 70 amers sur ces 90 mètres alors que le bleu lui

n'en a que 34 dans sa carte. Cela est dû au fait que le véhicule bleu n'a quasiment aucun objet proche contrairement au rouge qui a des façades de bâtiments sur sa gauche (voir figure 3.20(d)). Ainsi, il est plus difficile de faire converger des amers pour le véhicule bleu. Les trajectoires suivies sont disponibles en figure 5.12.

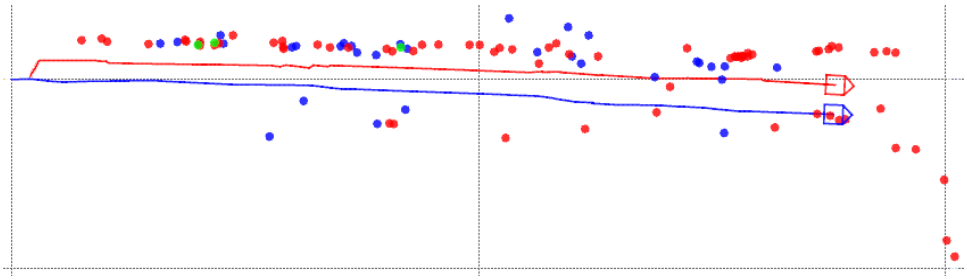


FIGURE 5.12 – Trajectoire en convoi ligne. La ligne rouge est la trajectoire du véhicule de gauche et la ligne bleue celle accomplie par celui de droite. Les amers suivent le même code couleur. Les amers en vert sont ceux associés et donc communs aux deux véhicules.

Encore une fois, uniquement 3 amers communs ont pu être identifiés pour les mêmes raisons que le scénario précédent. Néanmoins, ceux-ci ont permis d'estimer la distance séparant les véhicules correctement. On peut en effet constater qu'après quelques mètres, la position du véhicule rouge est corrigée et traduit bien la configuration initiale des véhicules.

L'estimation du biais obtenue est néanmoins de bonne qualité comme en témoigne la table 5.2. L'erreur concernant la distance initiale séparant les véhicules est très faible. Elle est sous la barre des 5 centimètres pour les axes liés à la position. L'erreur d'orientation est elle inférieure à 0,002 radian.

	x	y	z	Ψ	Φ	Θ
Association 1	0,1679	0,0571	0,0025	0,0002	0,0006	0,0026
Association 2	0,0519	0,0970	0,0007	0,0017	0,0002	0,0052
Association 3	0,0435	0,1215	0,0255	0,0077	0,0011	0,0019

TABLE 5.2 – Erreur quadratique moyenne du biais pour le convoi ligne

La qualité des amers cartographiés a permis d'atteindre une bonne précision pour le biais statique. Les résultats en termes de temps de calcul et de bande passante utilisée sont eux aussi similaires à ceux de la trajectoire précédente et ne seront donc pas exposés ici.

Ces scénarios ont montré que la notion de biais statique permet d'assurer la consistance du filtre à son initialisation. La distance entre les véhicules peut ainsi être retrouvée avec une précision satisfaisante. Les coûts en temps de calcul et en bande passante montrent également que notre approche décentralisée est tout à fait

viable. Elle peut facilement être étendue à plus de véhicules sans aucun problème. Ces premiers résultats illustrent bien à quel point l'utilisation du biais sous sa forme dynamique est nécessaire dans un algorithme multivéhicule, que cela soit pour assurer l'intégrité du filtre ou faciliter l'association de données. Ces résultats indiquent d'ailleurs qu'il est nécessaire de construire un processus d'association de données plus robuste que celui mis en place dans le SLAM monoculaire. En effet, avec plusieurs véhicules et des incertitudes importantes, cet algorithme représente un point crucial de l'approche décentralisée.

5.3 Association de données pour le multivéhicule

Les résultats précédents illustrent bien la nécessité de réfléchir à un tel processus par rapport au contexte dans lequel il va être utilisé. Dans notre cas, cela sera avec des amers de différents véhicules et affectés d'incertitudes importantes. Il faudra donc être prudent afin de s'assurer que les correspondances trouvées soient justes, le filtre de Kalman étendu pouvant complètement diverger après de mauvaises associations.

Dans cette courte section, nous commencerons tout d'abord par présenter l'algorithme construit (sous-section 5.3.1). Ensuite, celui-ci sera éprouvé sur une première approche simple afin d'en illustrer le fonctionnement et d'en tester les capacités. Cela sera l'objet de la sous-section 5.3.2.

5.3.1 Choix de la méthode

L'analyse de l'état de l'art fournie en sous-section 5.1.4 donne un aperçu des algorithmes existants qui pourraient convenir à une approche décentralisée et générique. Ce dernier point est important car c'est cet aspect qui nous a conduit à utiliser un EKF afin de limiter le nombre d'amers à envoyer. Cela signifie néanmoins que l'association de données est plus difficile car faite avec moins de points. Ainsi, les algorithmes reposant énormément sur des configurations spatiales similaires sont à bannir. Bien que l'organisation géométrique des amers soit importante pour reconnaître des endroits déjà visités, il faut bien garder en tête qu'uniquement quelques amers seront communs entre deux cartes du même espace. En effet, le point du vue ou même les conditions d'observation peuvent être différents, changeant ainsi les résultats du sélecteur de points et donc les amers qui arriveront dans le haut niveau.

La deuxième conséquence, liée à la construction d'une solution générale, est que les associations doivent se faire presque exclusivement entre des amers 3D. La décorrélation complète entre le SLAM bas niveau et son pendant haut niveau fait qu'il est difficile de faire transiter des informations propres aux capteurs sans casser la structure générale de l'approche. Ainsi, il n'est pas possible de venir projeter dans l'image des points 3D provenant d'un autre véhicule. Cependant, il reste possible d'utiliser les descripteurs issus du SLAM monoculaire (zone de pixels autour des points initialisés par Harris) afin d'améliorer les performances de notre algorithme

d'association de données. Le but est ici de construire une méthode d'association qui soit en adéquation avec la structure générale proposée jusque là.

L'algorithme de JCBB [Neira & Tardós 2002], présenté lors de l'état de l'art, couplé avec l'intégration du biais statique, pourrait être pertinent. Cependant, si l'a priori initial concernant l'organisation de la flotte est faible, la compatibilité jointe n'aidera pas. Les incertitudes seraient trop grandes et s'intersecteraient quasiment toutes, rendant impossible le fait de trouver une configuration d'amers correcte.

Néanmoins, avec la consistance assurée par le biais, l'utilisation des incertitudes des amers est un bon moyen pour diminuer le nombre d'associations potentielles. Après une première mise à jour, une fois que la covariance initiale du biais aura été grandement réduite, l'utilisation des incertitudes des amers va devenir bien plus discriminante et ainsi énormément aider à trouver des associations. Bien que l'utilisation du JCBB soit compliquée ici, l'incertitude des amers pourra être considérée dans l'algorithme final.

Parmi les autres algorithmes présentés, le GCBB a été cité comme intéressant lors de l'étude de l'état de l'art. Un arbre des différentes possibilités d'association (entre les nouveaux amers et la carte décentralisée dans notre cas) est construit et exploré. Des contraintes sont définies afin de trouver le sous-ensemble d'observations qui correspond le mieux à la carte globale. Bien qu'il soit possible de définir de nouvelles contraintes, le GCBB est principalement basé sur une contrainte géométrique binaire. Afin d'être ajoutée à l'hypothèse d'association courante, l'observation et sa correspondance potentielle dans la carte doivent maintenir une organisation spatiale cohérente avec les observations déjà sélectionnées (hypothèse en cours) et les amers avec lesquels elles ont été associées. Une hypothèse est considérée correcte dès lors que le nombre d'amers la composant est supérieur à seuil (généralement fixé à 5 ou 6 amers communs). Ce processus est expliqué au travers d'un exemple dans la figure 5.13.

Quand cet algorithme d'association de données a été présenté [Neira *et al.* 2003], les auteurs ont remarqué que celui-ci pourrait être coûteux en temps de calcul à cause du parcours de l'arbre. En effet, avec un nombre important d'amers, la quantité de solutions à tester augmente rapidement, tout comme le temps nécessaire à cette opération. Ici, nous avons ajouté un premier filtrage afin d'élaguer les branches de l'arbre peu plausibles avant que l'arbre ne soit exploré, réduisant ainsi le temps nécessaire à son parcours grâce à l'intégration de la dérive. Il devient alors possible de limiter le nombre d'amers qui peuvent être associés à ceux qui sont individuellement compatibles de par leur incertitude. La compatibilité individuelle ou Individual Compatibility (IC) entre deux amers $\mathbf{a}_{u_i}$ et $\mathbf{a}_{u_j}$ se définit comme suit :

$$D_{ij}^2 = \nu_{ij}^T \mathbf{C}_{ij}^{-1} \nu_{ij} < \chi_{d,\alpha}^2 \tag{5.1}$$

avec :

$$\begin{aligned} \nu_{ij} &= \mathbf{a}_{u_i} \quad \mathbf{a}_{u_j} \\ \mathbf{C}_{ij} &= \mathbf{P}_{\mathbf{a}_{u_i}} + \mathbf{P}_{\mathbf{a}_{u_j}} \end{aligned} \tag{5.2}$$

Le seuil d'acceptation (en-dessous duquel une association est possible) est basé sur une distribution du Khi-deux $\chi^2_{d,\alpha}$. Ici, $d = 3$ (degrés de liberté des amers) et la confiance souhaitée (α) a été fixée à $0,95$. Uniquement les amers individuellement compatibles resteront dans l'arbre afin de rendre son exploration plus rapide.

Il est possible d'ajouter d'autres contraintes dans le but de renforcer la fiabilité des solutions obtenues. Dans notre contexte monoculaire particulier, nous avons spécialisé l'algorithme afin d'intégrer une information visuelle, permettant ainsi d'être plus discrimant envers les amers associés. Les descripteurs extraits autour

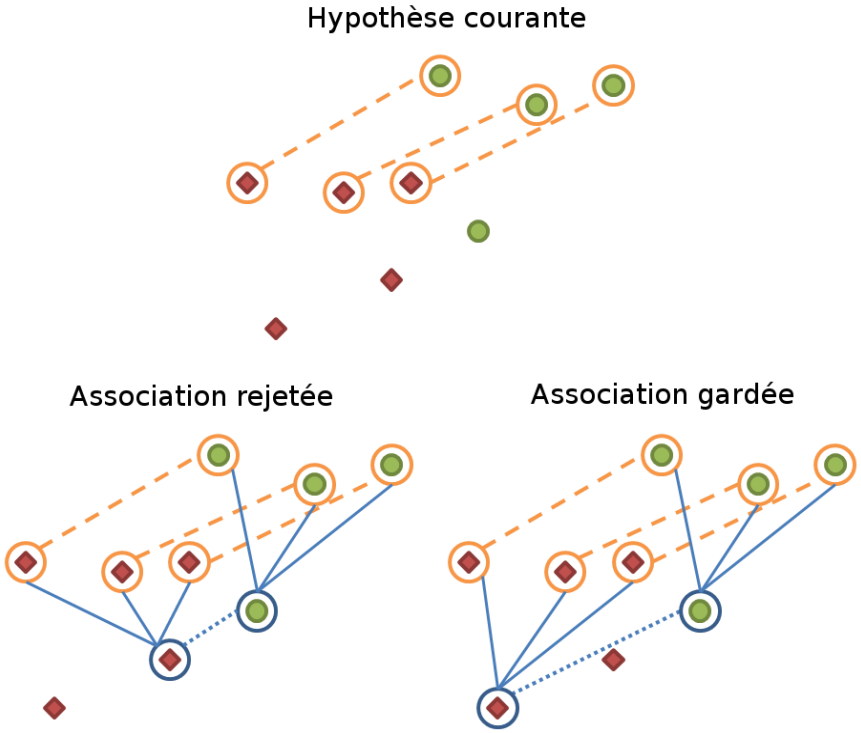


FIGURE 5.13 – Exemple d'utilisation de l'algorithme GCBB. L'hypothèse d'association courante est indiquée en orange. Les amers pour l'instant associés sont reliés par des lignes pointillées oranges. Les losanges rouges sont les amers de la carte. Les cercles verts représentent des observations (ici des amers reçus d'autres véhicules). Les points entourés en bleu sont ceux dont l'association est en cours de test. Afin d'être confirmée, les distances aux points dans l'hypothèse (lignes pleines bleues) doivent être similaires entre les observations et la carte. La nouvelle association potentielle est indiquée par la ligne pointillée bleue.

des points caractéristiques de l'image au moment de l'initialisation sont comparés par ZNCC. Cela permet d'éliminer de mauvaises associations de façon peu coûteuse. Cette contrainte unaire, associée à celle binaire concernant l'organisation spatiale des amers, va permettre de guider la recherche dans l'arbre.

L'algorithme récursif construit est intéressant car il tire parti de toutes les informations disponibles. Le premier filtrage suivant les incertitudes, en plus d'accélérer le processus d'association de données, utilise le fait que l'état des véhicules est consistant. Les estimations du biais permettent d'utiliser les covariances des amers comme une première étape de sélection. À l'inverse des algorithmes classiques de fermeture de boucle [Williams *et al.* 2009], tous les amers de la carte ne seront pas utilisés pour rechercher des associations. Uniquement ceux dont les incertitudes témoignent d'une possibilité de correspondance avec des observations seront conservés. L'exploration de l'arbre se fera ensuite par la validation des contraintes unaire et binaire. Pour la première, l'information visuelle disponible est utilisée. Une caméra donne des informations riches et facilement distinguables les unes des autres. Enfin, la contrainte binaire permet d'utiliser l'organisation géométrique des amers. Il s'agit certainement du point le plus discriminant de l'algorithme. Assurer une cohérence spatiale entre deux ensembles d'amers est loin d'être anodin et avec un seuil autour de 5 à 6 amers, les associations obtenues peuvent être considérées comme justes. L'algorithme d'association de données exploite à la fois l'incertitude, l'aspect visuel et l'organisation spatiale des amers. Il maintient également la structure générale de l'approche décentralisée.

5.3.2 Premiers résultats

Le processus développé peut être utilisé pour détecter des fermetures de boucle (observations associées avec la carte du même véhicule) et des endroits déjà cartographiés par un membre de la flotte (carte du véhicule en train d'explorer associée avec celle établie par le véhicule qui a déjà traversé la zone). L'association de données est déclenchée intelligemment pour éviter des traitement inutiles. Uniquement les nouveaux points sont testés avec la carte globale. Si aucune association n'est trouvée, l'amer est directement ajouté à la carte et lié au reste du vecteur d'état via le biais l'affectant. Sinon, celui-ci est conservé dans le processus d'association de données et si assez de correspondances sont déjà disponibles, la fusion de ces amers est faite. Comme cela a été énoncé auparavant, un véhicule traversant une zone déjà explorée ne cartographiera pas nécessairement les mêmes amers. Au lieu d'analyser les amers communs à un seul moment, nous avons choisi de rechercher les associations sur des portions de la carte. Cela signifie que les amers qui ont été appariés sont gardés dans le processus d'association pendant un certain temps. Si durant cette période, assez d'amers sont associés pour certifier qu'un endroit a été reconnu, alors une fusion sera réalisée. Si ce n'est pas le cas, ces amers seront ajoutés à la carte globale.

Une expérimentation a été menée afin de tester l'algorithme construit dans des conditions réelles. Un VIPALab a accompli une trajectoire de 150 mètres avant de

retourner à son point de départ et d'enchaîner avec un second tour de la même boucle. La partie de la trajectoire qui nous intéresse ici est le moment où le véhicule commence à traverser la zone qu'il a déjà cartographiée. Après 150 mètres, la dérive affectant la pose du véhicule peut être importante, il est ainsi pertinent de tester la robustesse de l'algorithme dans ces conditions. Bien évidemment, le but ici est de fermer la boucle. Néanmoins, les traitements sont identiques à une association entre deux véhicules différents. La figure 5.14 montre la portion de la trajectoire où le véhicule traverse une zone préalablement visitée.

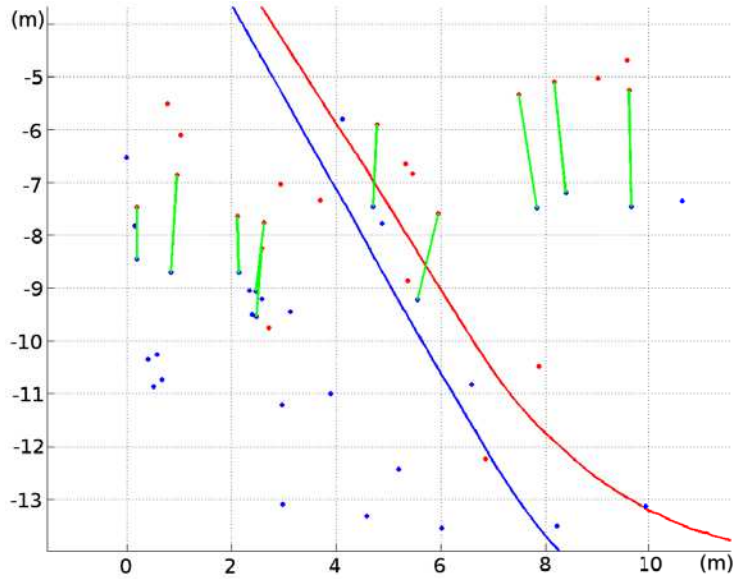


FIGURE 5.14 – Associations trouvées entre deux passages. La courbe bleue est la trajectoire calculée lors du premier passage. La rouge est celle obtenue au deuxième tour. Les amers correspondant à chaque passage suivent le même code couleur. Les amers trouvés comme communs entre les cartes sont reliés par des lignes vertes.

On peut constater qu'entre les deux passages, le véhicule a dérivé. En effet, bien que le chemin soit le même la deuxième fois, la position du véhicule est éloignée de quelques mètres de celle calculée lors du premier tour. Sur la portion commune, le véhicule a cartographié un peu plus de 30 amers sur le premier passage et autour de 20 lors du second. L'algorithme d'association a permis de trouver 10 amers comme étant communs aux deux tours. On peut constater que ces amers partagent une configuration géométrique similaire.

Ces résultats sont encourageants puisqu'ils montrent qu'avec seulement quelques amers, le processus d'association trouve tout de même des correspondances entre les deux passages. D'autres expérimentations, impliquant plusieurs véhicules, doivent cependant être conduites afin de le valider définitivement et seront montrées dans le prochain chapitre.

Ce chapitre a présenté un algorithme de SLAM décentralisé. Celui-ci adresse

les problématiques courantes de ces approches tout en étant suffisamment général pour pouvoir utiliser n'importe quel processus de SLAM bas niveau existant. La consanguinité des données est évitée par une stratégie d'échange couplée à une architecture dédiée. Cette dernière permet par ailleurs de gérer les échecs du réseau (désynchronisations, latences, coupures de communication). L'application conçue intègre également la dérive de chaque véhicule sous la forme d'un biais de localisation. Sa partie statique, liée à son initialisation, permet également de mettre dans un même repère les différents véhicules de la flotte et ainsi de résoudre le problème des positions initiales inconnues. Enfin, un algorithme d'association de données pour le multivéhicule a été exposé. Celui-ci tire parti des spécificités de l'application construite. Chaque point du SLAM décentralisé a été validé individuellement par des expérimentations. Le prochain chapitre va maintenant proposer des résultats mettant en jeu l'intégralité de l'application développée, et ce dans des conditions et configurations variées.

Chapitre 6

Expérimentations et résultats pour le multivéhicule

Sommaire

Au cours de ce chapitre, les différentes expérimentations et simulations menées vont être exposées. Celles-ci vont à chaque fois utiliser l'application complète de SLAM décentralisé. Afin de pleinement valider notre algorithme, celui-ci sera confronté à diverses situations. Le contexte multivéhicule rend néanmoins la mise en place d'expérimentations avec plusieurs véhicules difficile. En effet, cela nécessite d'avoir tous les véhicules de la flotte opérationnels à un moment donné. De même, la communication sans fil doit être fonctionnelle et les conditions météorologiques propices à la mise en route des robots. Le support technique doit donc être important. Ainsi, dans un premier temps, et afin d'éviter ces contraintes, notre solution a été testée grâce au simulateur présenté plus tôt dans ce manuscrit. Ces simulations feront l'objet de la section 6.1. Après cette première validation, l'algorithme sera alors utilisé dans des expérimentations avec des données réelles. Des configurations différentes et des résultats à 2 et 3 véhicules seront présentés dans la section 6.2. Dans toutes ces expérimentations, le modèle de biais utilisé n'intègre pas la translation à l'origine puisque les différentes fusions ont lieu a proximité de l'origine et n'ont donc pas d'impact significatif sur les résultats. Enfin, la section 6.3 discutera les résultats obtenus en dressant un bilan rapide de notre approche.

6.1 Résultats avec simulateur

Deux scénarios ont été réalisés avec l'aide du simulateur. Comme pour ses utilisations précédentes, le simulateur sert uniquement à générer des données capteur et ne fait aucun traitement supplémentaire sur celles-ci. Chaque véhicule simulé est géré par une instance de l'algorithme décentralisé de SLAM et doit donc gérer le partage des données au travers d'un vrai réseau. Ici, plutôt que l'environnement de la place de Jaude, nous nous sommes orientés vers celui de PAVIN simulé. Ce dernier est identique à la plate-forme utilisée pour les expérimentations réelles et est plus représentatif de l'environnement urbain que ne peut l'être la place de Jaude qui est très ouverte. Quelques vues de PAVIN simulé sont disponibles en figure 6.1 et quelques images caméra prises en cours de trajectoire sont visibles en figure 6.2.

Dans les deux scénarios simulés, chaque véhicule est équipé d'une caméra (située à 1,20 mètres dans l'habitacle) fournissant des images en niveaux de gris à 10 Hz et d'un odomètre. Un GPS RTK simulé est également présent afin de servir de vérité terrain. Les véhicules se déplacent à approximativement 2 mètres par seconde. Ces véhicules sont les équivalents simulés de l'Aroco. Ce dernier est un engin tout-terrain habituellement utilisé dans le domaine agricole. Ici, son comportement simulé est très proche du VIPALab présenté précédemment. Tous les résultats présentés ici, et dans la prochaine section, sont ceux obtenus après application de la rétro-action naturelle sur l'ensemble du vecteur d'état décentralisé lorsque des associations sont trouvées.

(a) (b)

FIGURE 6.1 – Environnement PAVIN sous le simulateur

(a) (b)

FIGURE 6.2 – Quelques exemples d'images issues de la caméra virtuelle sur PAVIN

Dans la sous-section 6.1.1, un convoi colonne classique à deux véhicules sera présenté. Le but est ici de valider l'application et d'en comprendre clairement les tenants et aboutissants. Le deuxième scénario qui sera décrit en sous-section 6.1.2 est lui plus complexe. Il s'agit d'une longue trajectoire au cours de laquelle le véhicule de tête sera interverti avec celui de queue.

6.1.1 Convoi colonne

Afin de montrer le fonctionnement de notre application, la première simulation effectuée est relativement simple. Deux véhicules, séparés par environ 20 mètres, évoluent sur PAVIN en se suivant. La trajectoire accomplie par chaque véhicule est d'une longueur d'environ 60 mètres. Celle-ci est constituée de 2 grandes lignes droites et d'un virage à 90°. Bien que le contexte de cette simulation soit simple, les véhicules ne savent pas où ils se situent l'un par rapport à l'autre. Estimer qu'ils sont séparés par 20 mètres est loin d'être trivial et nécessite un algorithme robuste. Comme pour toutes les expérimentations qui suivront, les véhicules ne sont pourvus

d'aucun algorithme permettant de détecter les autres véhicules directement. Ainsi, l'estimation de l'écart entre eux est uniquement basée sur la mise en correspondance d'amers. Un aperçu de la trajectoire parcourue par chaque véhicule est disponible en figure 6.3.

FIGURE 6.3 – Aperçu des trajectoires du scénario convoi colonne simulé

Le repère de référence pour l'état décentralisé est fixé sur le point de départ du véhicule de tête (vert). Avec cette configuration, cela signifie que le véhicule de derrière n'est initialement pas correctement localisé dans le repère commun et doit trouver des amers communs avec le robot de tête de manière à pouvoir estimer la distance les séparant. Afin d'être le plus clair possible, les résultats présentés ici (et pour tout le chapitre) seront uniquement ceux d'un seul des véhicules. En effet, les informations partagées par la flotte sont les mêmes, rendant les cartes décentralisées calculées par chacun quasiment identiques. Ici, les résultats sont ceux provenant du véhicule de tête.

La figure 6.4 montre les trajectoires de chaque véhicule obtenues par le SLAM bas niveau, c'est-à-dire sans intégration de la dérive ou encore d'association de données haut niveau. La trajectoire affichée pour le véhicule de derrière correspond à ses poses successives communiquées au véhicule de tête via le réseau.

Cette figure montre entre autres l'écart entre la trajectoire perçue par le véhicule de derrière et la vérité. En effet, bien que les robots se suivent, la localisation calculée ne le reflète pas. Avec le repère commun fixé (point de départ de la tête), et sans trouver d'amers communs, le véhicule de queue croit commencer sa trajectoire au point de départ du véhicule vert. Ainsi, celui-ci n'est pas correctement localisé dans le repère de référence.

On peut également constater sur la figure 6.4 que la consistance n'est pas maintenue durant ce scénario. Les incertitudes (affichées sur la figure 6.4 mais à peine visibles) sont très faibles. Même la trajectoire verte (premier véhicule), qui est proche de la vérité, n'est pas intègre. La dérive, bien que mineure sur une si courte distance,

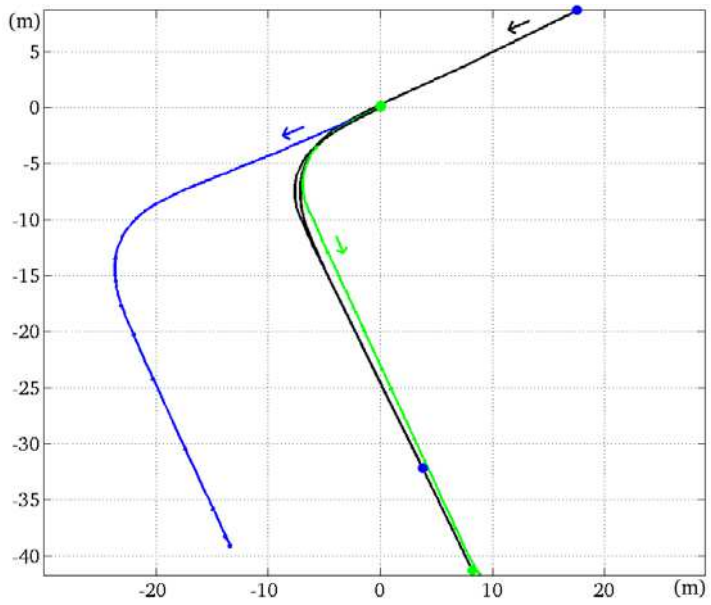


FIGURE 6.4 – Trajectoires obtenues par le SLAM bas niveau pour le scénario convoi colonne simulé. En vert, la trajectoire calculée par le véhicule de tête. En bleu, celle du suiveur (échangée via des communications réseau). En noir, les trajectoires réellement accomplies par les véhicules (vérité terrain). Points : départ et arrivée réels de chaque véhicule.

empêche les solutions calculées par un SLAM classique d'être consistantes.

Afin de clairement montrer la taille des ellipses d'incertitude, des zooms sur les deux trajectoires sont proposés en figure 6.5 et 6.6. Les incertitudes obtenues avec l'intégration du biais (mais toujours sans fusion) sont également visibles sur ces mêmes figures.

On peut voir sur ces deux figures que les incertitudes en sortie du SLAM monoculaire sont particulièrement petites, confirmant de fait l'inconsistance de la localisation bas niveau. L'intégration du biais dynamique sur la figure 6.5 permet de prendre en compte même les petites dérives en fournissant des incertitudes adaptées. Le cas de la figure 6.6 est un peu différent car le véhicule commence la trajectoire en étant localisé de manière très imprécise. L'initialisation du biais permet de s'assurer que, dans le repère commun à la flotte, la vraie position du véhicule est bien incluse dans son incertitude. En l'absence d'information précise, l'incertitude statique du biais a été initialisée de manière à ce que l'ellipse générée soit un cercle de 22 mètres de rayon. Cette initialisation permet d'assurer que la vraie pose du véhicule est dans un rayon de 22 mètres par rapport au $(0, 0)$. Nous avons fixé une valeur nulle pour le biais statique afin de pouvoir tester la fiabilité de notre algorithme d'association de données et la robustesse du modèle de dérive. Il est aussi possible de constater sur la figure 6.6 que l'intégrité du filtre est également maintenue tout le long de le trajectoire du véhicule de queue.

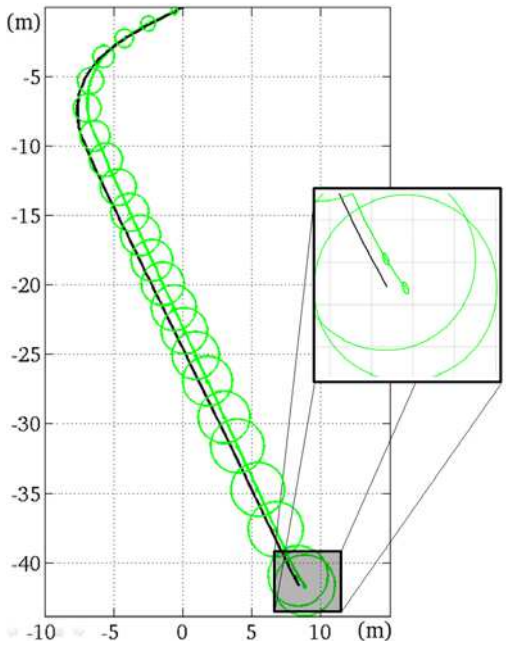


FIGURE 6.5 – Intégration du biais pour le véhicule de tête dans le scénario convoi colonne simulé

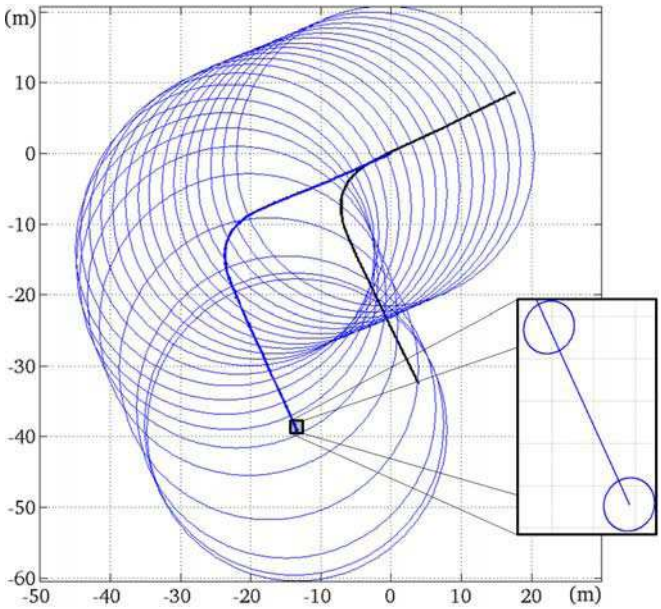


FIGURE 6.6 – Intégration du biais pour le véhicule de queue dans le scénario convoi colonne simulé
Sur les deux figures, les petites incertitudes sont celles en sortie du SLAM monoculaire. Les plus grosses sont celles intégrant la dérive. Le code couleur est le même que précédemment.

La figure 6.7 présente les résultats de localisation obtenus quand le processus d'association de données est actif. Pour plus de lisibilité, uniquement quelques ellipses sont affichées. Dès lors que l'algorithme est capable de certifier une fusion de données (assez d'amers communs entre les 2 cartes), on peut voir que la localisation est correctement corrigée. L'écart entre les véhicules est alors rétabli et les incertitudes sont en conséquence grandement réduites. Un point intéressant à noter est que peu d'amers ont été cartographiés. Le véhicule bleu en totalise 99 et de son côté, le véhicule vert en compte 116. Ces faibles nombres compliquent la tâche de l'association de données car les sous-ensembles d'amers communs sont peu nombreux. Enfin, et c'est un aspect majeur, les localisations fournies par les véhicules avant et après l'association de données sont toujours intègres.

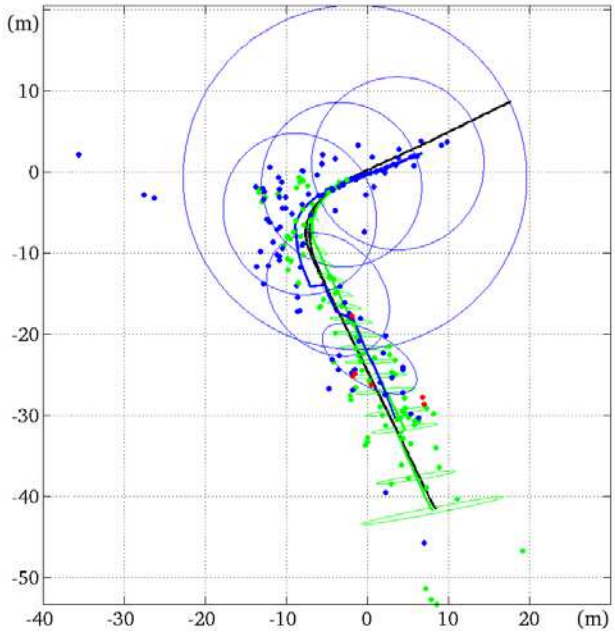


FIGURE 6.7 – Localisation avec association de données pour le convoi colonne simulé. Le code couleur est le même que précédemment. Les points bleus sont les amers cartographiés par le véhicule de derrière. De manière similaire, les points verts proviennent du véhicule de tête. Les points rouges sont les amers trouvés communs entre les deux véhicules par le processus d'association de données.

La qualité de la localisation est mesurée en comparant l'écart entre les véhicules. Pour ce faire, la distance réelle est établie à l'aide des GPS RTK des véhicules. Celle-ci est, à chaque instant, comparée à celle estimée par notre algorithme de SLAM décentralisé. Il est important de rappeler que, pour tous les résultats de ce chapitre, l'estimation de cette distance est uniquement faite à l'aide des amers communs et sans l'aide d'une mesure directe d'interdistance. Les résultats pour ce scénario sont disponibles en figure 6.8 où l'évolution de la distance entre les deux véhicules en fonction du temps est montrée. On peut voir que la distance estimée par le SLAM

décentralisé s'approche de la vérité après que des associations entre les robots aient été trouvées (moins de 20 centimètres d'erreur). La distance mesurée en amont du point de fusion est assez éloignée de l'écart réel (environ 5 mètres d'écart). Cela est principalement dû au fait que les associations ont été faites vers la fin de la trajectoire, rendant ainsi l'impact de la rétro-action moins important au début de la trajectoire. Augmenter le nombre d'amers à suivre permettrait d'aider à trouver des associations plus tôt afin de pouvoir estimer plus précisément la distance entre les véhicules. Bien évidemment, l'écart estimé par le SLAM bas niveau n'est pas correct car celui-ci n'est pas conscient de sa distance à l'autre véhicule.

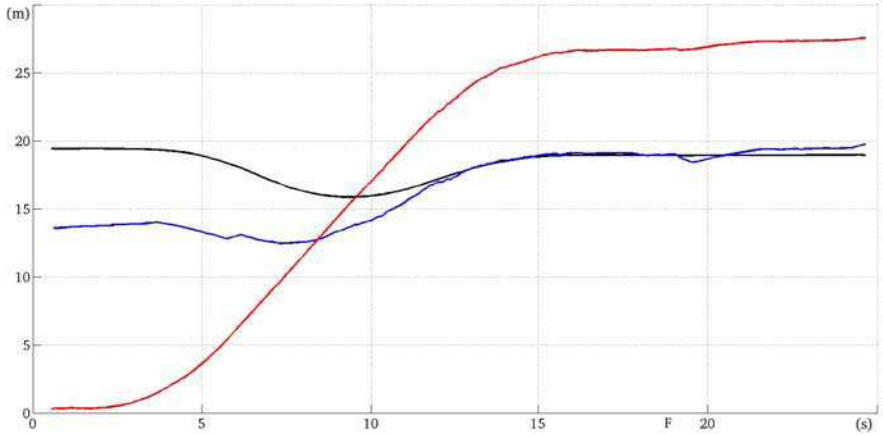


FIGURE 6.8 – Distance entre les véhicules en fonction du temps pour le convoi colonne simulé. En rouge est la distance estimée sans association de données. La courbe noire est le véritable écart séparant les véhicules. Enfin, en bleu est la distance estimée par notre algorithme décentralisé. Le F sur l'abscisse indique quand une association de données a été trouvée (au moins 5 amers communs) et donc une fusion réalisée.

Un autre aspect intéressant est la bande passante utilisée par notre SLAM décentralisé. La figure 6.9 expose la quantité de données envoyée par les deux véhicules durant l'ensemble de la trajectoire. On peut voir que peu de bande passante a été consommée. En effet, approximativement 10 ko sont émis par seconde et par véhicule, ce qui est très loin des verrous technologiques actuels. Les standards étant à plus de 5 Mo par seconde, il est possible d'augmenter le nombre de véhicule sans risquer une congestion du réseau (jusqu'à 500 robots dans notre cas ici). La nature éparse des cartes que l'on construit aide à maintenir ces chiffres bas puisque la bande passante requise est fortement liée au nombre d'amers échangés.

Ce premier scénario montre que le nombre d'amers dans la carte joue un rôle clef pour l'estimation de la distance entre les véhicules. Néanmoins, l'écart conséquent séparant les véhicules à tout de même pu être estimé assez précisément. L'algorithme doit maintenant être éprouvé sur de plus longues distances. Cela sera l'objet de la prochaine simulation

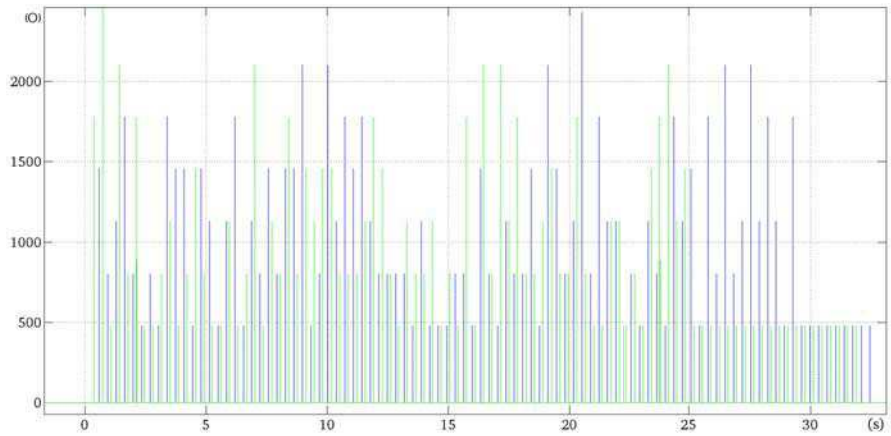


FIGURE 6.9 – Quantité de données envoyées en fonction du temps pour le convoi colonne simulé. La courbe verte représente les données envoyées par le véhicule de tête et en bleu celles émises par le véhicule arrière. La quantité de données est exprimée en octets.

6.1.2 Changement de l'ordre du convoi

En plus de pouvoir gérer des convois très espacés, notre SLAM est entièrement décentralisé et peut donc fonctionner sans véhicule meneur. Cet aspect sera illustré sur la prochaine trajectoire. L'idée est d'inverser l'ordre d'un convoi colonne au cours de la trajectoire. Le véhicule de tête devient celui de queue et inversement. C'est aussi un moyen de montrer une trajectoire plus complexe et plus longue. Le contexte est exactement le même que pour le scénario précédent. Les véhicules sont équipés des mêmes capteurs fonctionnant aux mêmes vitesses. Les deux trajectoires mesurent approximativement 110 mètres chacune. Une vue de dessus des trajectoires effectuées est disponible en figure 6.10. Les deux véhicules sont séparés initialement par 11 mètres et démarrent en configuration classique colonne. Le véhicule vert est la tête du convoi et le bleu le suiveur. Après le premier virage, le véhicule vert fait le tour du rond-point pendant que le véhicule bleu passe devant. Le robot vert s'arrête quelques secondes dans le rond-point afin de laisser le nouveau véhicule de tête prendre quelques mètres d'avance. Les trajectoires continuent ensuite dans cette formation jusqu'à l'arrêt final.

On peut constater que les portions communes ne sont pas très longues et demandent donc que l'association de données fonctionne efficacement. En gardant cela à l'esprit, nous avons augmenté le nombre de points caractéristiques suivis dans les images afin d'avoir plus d'amers dans l'algorithme décentralisé. Concernant l'initialisation du biais, le point de départ du véhicule de devant est encore une fois utilisé comme repère commun. Le robot à l'arrière est donc initialement localisé de manière imprécise. Son biais statique a été fixé de façon à générer un cercle de 21 mètres de rayon afin de ne pas avoir un a priori trop précis sur la distance entre les véhicules.

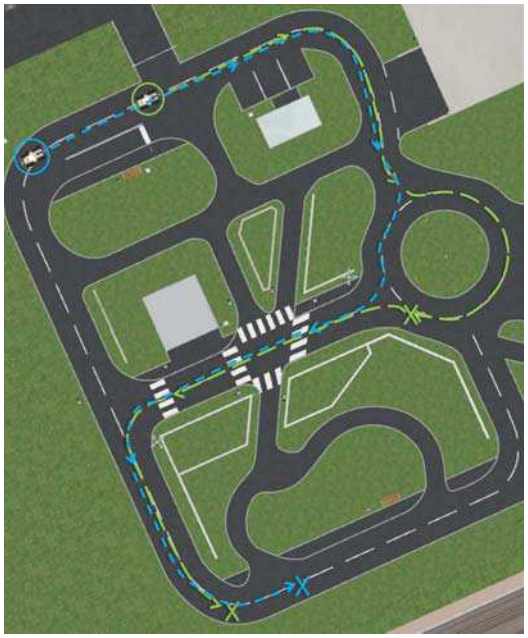


FIGURE 6.10 – Aperçu (approximatif) des trajectoires du scénario d'inversion de convoi. Les croix correspondent à des arrêts du véhicule et les cercles aux points de départ.

La figure 6.11 montre les trajectoires calculées par le bas niveau avec les incertitudes associées et les vérités terrain.

Les résultats sont assez proches de ceux de la simulation précédente. La trajectoire bleue estimée est décalée de 11 mètres dans le repère commun. Les incertitudes restent, pour les deux véhicules, très faibles tout au long du scénario. L'intégrité des localisations n'est pas préservée, et ce même pour le véhicule vert dont la pose est assez bien estimée. En plus de l'écart initial, les deux véhicules sont affectés par un léger biais angulaire, écartant les trajectoires de leur vérité terrain. Les figures 6.12 et 6.13 montrent comment l'intégration du biais affecte les incertitudes. Un zoom sur les incertitudes du bas niveau est également visible pour chacune des trajectoires.

L'intégration du biais permet de maintenir la consistance de la localisation tout au long des deux trajectoires. Dans le cas de la figure 6.12, l'évolution de l'incertitude de biais est bien visible car le véhicule part parfaitement localisé puis dérive tout doucement, diminuant ainsi la précision de sa position. La vue rapprochée vers la fin de la trajectoire montre que le SLAM bas niveau est loin d'intégrer le véritable chemin suivi par le véhicule alors que le biais le permet. L'évolution de l'incertitude du biais est moins évidente sur la figure 6.13. En effet, le biais statique est important et masque l'évolution de sa partie dynamique. Le zoom en fin de trajectoire montre encore une fois l'impossibilité du SLAM classique à maintenir l'intégrité.

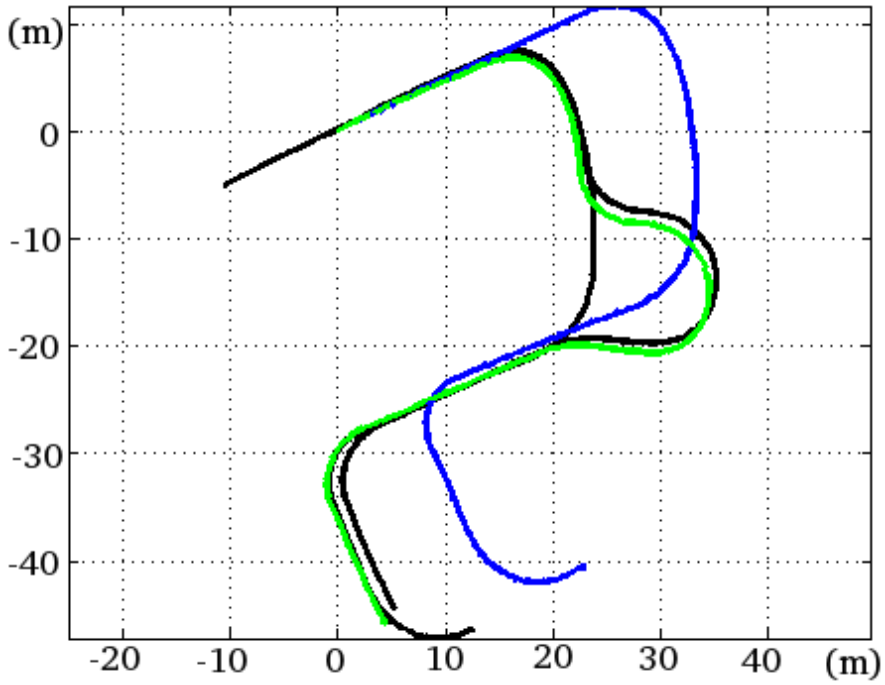


FIGURE 6.11 – Trajectoires obtenues par le SLAM bas niveau pour le scénario d'inversion de convoi. En vert, la trajectoire calculée par le véhicule initialement devant. En bleu, la même chose pour l'autre véhicule (échangée via des communications réseau). En noir, les trajectoires réellement accomplies par les véhicules (vérité terrain).

Les trajectoires calculées lorsque des associations sont cherchées sont exposées en figure 6.14. Pour des raisons de clarté, les amers ne sont pas visibles, uniquement les trajectoires et les incertitudes le sont. À l'inverse, les trajectoires avec les amers cartographiés, mais sans les incertitudes, sont montrées dans la figure 6.15.

La localisation obtenue est proche de la vérité terrain, et ce pour les deux véhicules. Les incertitudes du biais ont été grandement réduites, surtout pour le véhicule bleu qui démarrait la trajectoire avec une covariance initiale importante. Les poses des véhicules sont consistantes, même avec ces incertitudes réduites.

Le fait de trouver des portions communes de cartes est facilité par un nombre d'amers cartographiés plus important. La première association est essentielle car c'est celle qui va donner la première estimation de la distance réelle entre les véhicules et donc réduire drastiquement l'incertitude de biais initiale. Celle-ci intervient ici rapidement après le début de la trajectoire. Avec une idée précise de cette distance, identifier les amers communs est plus facile car les incertitudes sont plus petites et donc plus discriminantes. Cela a permis à l'algorithme de trouver des associations dans la deuxième grande ligne droite après le rond-point. Il est tout de même important de mentionner que peu d'amers sont communs entre les deux vé-

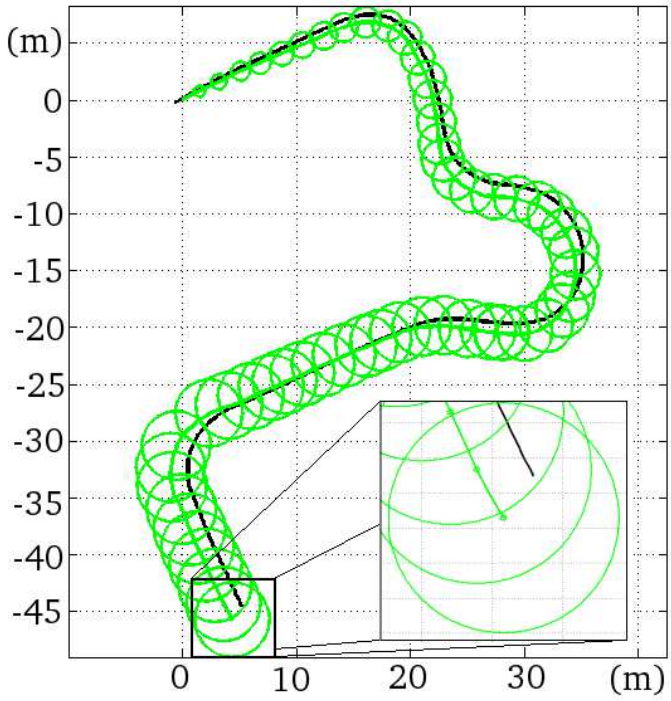


FIGURE 6.12 – Intégration du biais dans le scénario d'inversion de convoi (1/2)

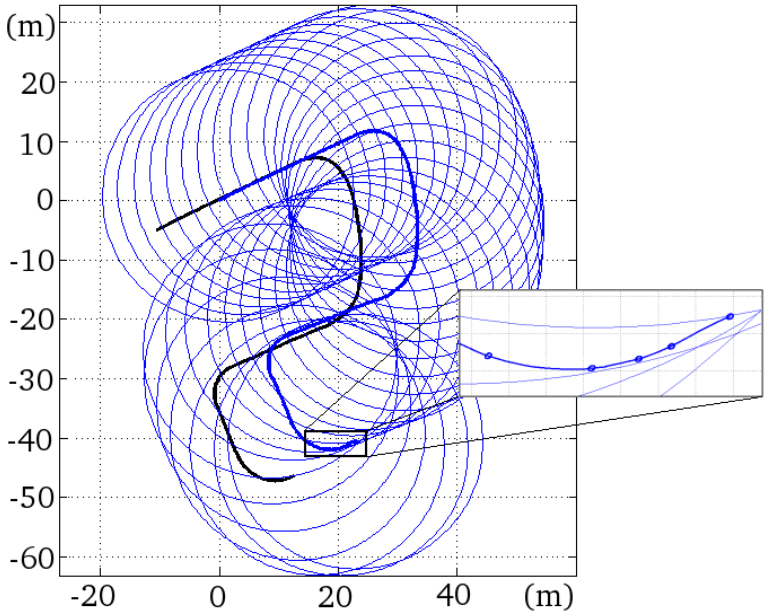


FIGURE 6.13 – Intégration du biais dans le scénario d'inversion de convoi (2/2)
Sur les deux figures, les petites incertitudes sont celles en sortie du SLAM monoculaire. Les plus grosses sont celles intégrant la dérive.

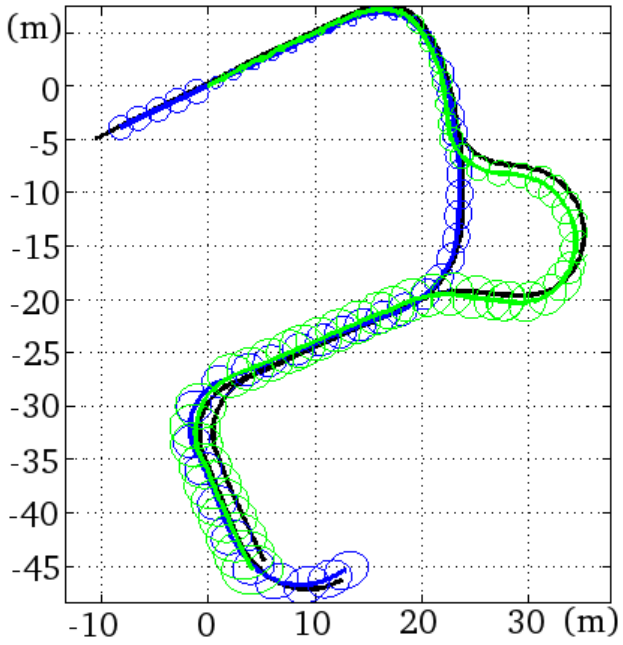


FIGURE 6.14 – Localisation avec association de données pour l'inversion de convoi

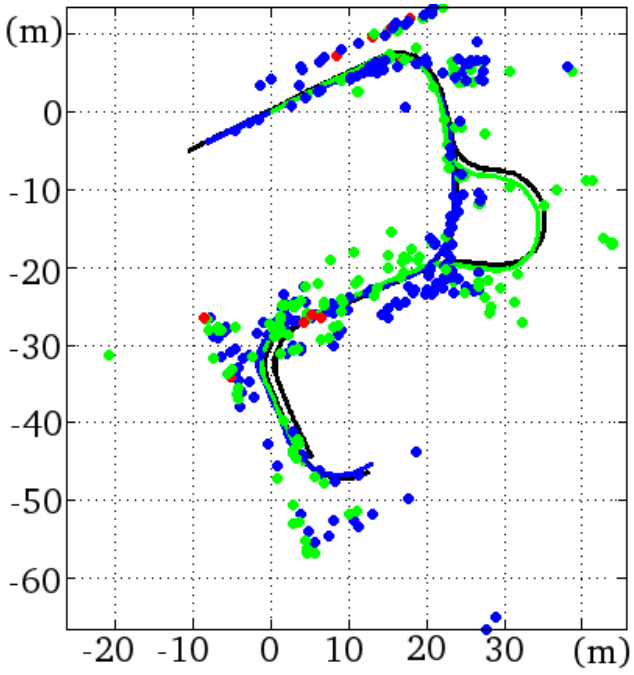


FIGURE 6.15 – Amers cartographiés pour le scénario d'inversion de convoi. Le code couleur est le même que précédemment.

hicules. En effet, avec une orientation du véhicule légèrement différente, la sélection de points caractéristiques peut totalement changer. C'est par exemple le cas après le rond-point où le véhicule vert tend à cartographier des points sur le bord droit de la route alors que le véhicule bleu n'a pour ainsi dire que des points sur le bord gauche.

La justesse de la distance estimée est présentée en figure 6.16. Elle est comparée à la vérité terrain ainsi qu'à l'écart calculé par le SLAM sans l'intégration de la dérive. La qualité de la localisation de chaque véhicule n'est pas donnée car peu pertinente. En effet, sans fermeture de boucle ou information absolue, il n'est pas possible de corriger complètement la dérive du SLAM. La distance entre les véhicules évoluant beaucoup durant ce scénario, c'est un bon indicateur de la qualité du modèle de biais.

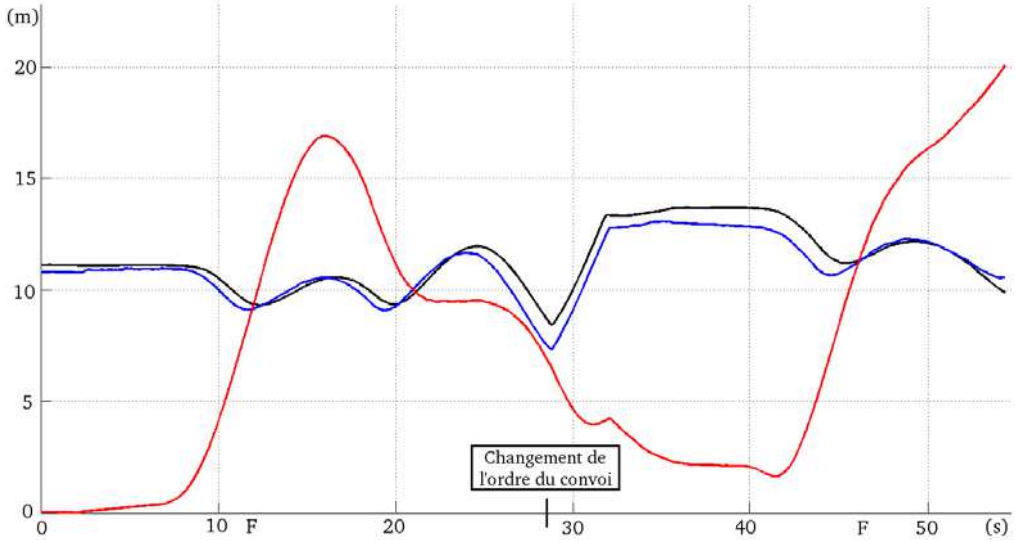


FIGURE 6.16 – Distance entre les véhicules en fonction du temps pour le scénario d'inversion de convoi. En rouge est la distance estimée sans association de données. La courbe noire est le véritable écart séparant les véhicules. Enfin, en bleu est la distance estimée par notre algorithme décentralisé. Les F sur l'abscisse indiquent quand une association de données a été trouvée (au moins 5 amers communs) et donc une fusion réalisée.

On peut constater que l'estimation de la distance suit de très près le véritable écart (moins de 20 centimètres d'erreur sur la première moitié de la trajectoire et 80 centimètres sur la deuxième partie). Aucun pic majeur n'est à noter. La rétro-action appliquée aux distances passées donne de meilleurs résultats que pour le scénario précédent. Cela est principalement dû au fait que plus d'amers ont été associés et ce dès le début de la trajectoire. Le plus gros écart dans l'estimation (autour de 1,5 mètres) intervient au moment où aucune association n'est trouvée.

Au niveau des communications réseau, les taux de transfert de données sont

similaires à ceux observés précédemment. Ils sont renseignés en figure 6.17.

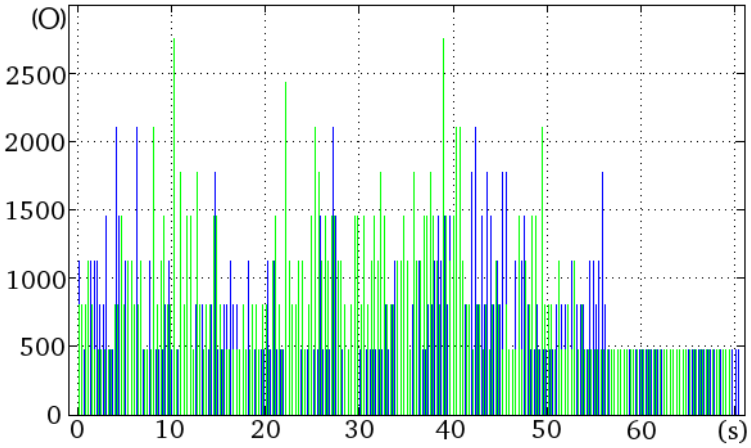


FIGURE 6.17 – Quantité de données envoyées en fonction du temps pour l'inversion de convoi. La courbe verte représente les données envoyées par le véhicule de tête et en bleu celles émises par le véhicule arrière. La quantité de données est exprimée en octets.

Moins de 10 ko par seconde sont consommés bien que plus d'amers aient été cartographiés. Comme ils sont envoyés régulièrement, ils ne s'accumulent pas évitant ainsi des gros pics ponctuels.

Les résultats de cette section confirment que notre approche décentralisée est viable. En termes de temps de calcul, un ordinateur portable avec un processeur Intel Core i5 à 2,4 GHz a suffit à le faire fonctionner en temps réel. Les débits réseau sont aussi très faibles et s'accommodent donc bien au nombre de véhicules de la flotte. L'utilisation de capteurs bas coûts n'a pas empêché l'algorithme de donner de bons résultats. Ceux-ci doivent maintenant être validés avec des expérimentations réelles.

6.2 Résultats avec données réelles

Le contexte des expérimentations qui vont suivre est très similaire à celui des simulations. Les véhicules VIPALab sont toujours équipés d'une unique caméra, d'un odomètre et d'un capteur d'angle de braquage. La plate-forme PAVIN reste l'environnement de ces expérimentations. Dans celles-ci, la caméra fonctionne à 15 Hz. Chaque véhicule embarque aussi un GPS RTK pour fournir une vérité terrain. Comme pour les simulations, l'algorithme de SLAM est exécuté sur un ordinateur portable possédant un processeur Intel Core i5. Chaque véhicule utilise un ordinateur différent de manière complètement décentralisée. Dans toutes les trajectoires qui vont être présentées, les VIPALab sont conduits manuellement.

Les 3 expérimentations menées sont très différentes. La première, qui sera présentée en sous-section 6.2.1, correspond au convoi colonne effectué sous simulateur

en sous-section 6.1.1. Le but est ici de valider les résultats obtenus par le simulateur. Ensuite, la sous-section 6.2.2 illustrera une approche sans meneur avec un convoi ligne. Enfin, la sous-section 6.2.3 exposera une expérimentation réelle avec 3 véhicules.

6.2.1 Convoi colonne à deux véhicules

La première expérimentation effectuée est donc très proche de la première simulation de ce chapitre. Le but est de s'assurer que les résultats obtenus grâce au simulateur sont similaires à ceux calculés à partir de données réelles. Il s'agit d'un convoi colonne où les deux véhicules sont initialement séparés par 13 mètres. Chaque trajectoire est longue de 80 mètres et a été accomplie à une vitesse de 2 mètres par seconde. Un aperçu des trajectoires suivies au cours de cette expérimentation est donné en figure 6.18.

FIGURE 6.18 – Aperçu des trajectoires du scénario convoi colonne
Les cercles correspondent aux points de départ des véhicules et les croix aux arrêts.

Comme lors de la simulation, nous avons utilisé le point de départ du véhicule de tête comme repère de référence. Cela signifie que le véhicule bleu suiveur ne sera pas correctement localisé au début de la trajectoire. La figure 6.19 montre le résultat de localisation de ces deux véhicules sans la prise en compte du biais ou d'associations de données.

Les incertitudes calculées par l'algorithme de SLAM sont aussi visibles sur cette figure. Elles sont d'ailleurs plus importantes que pour les simulations. Cela vient du fait que les capteurs réels sont moins précis que ceux inclus dans le simulateur. Les bruits d'évolution (odomètre et angle volant) et d'observation (caméra) ont donc été gonflés en conséquence. Néanmoins, le SLAM ne produit toujours pas des estimées intègres. L'aspect des trajectoires est assez proche des données fournies par le GPS RTK. On peut noter une dérive de position pour le véhicule de tête. Bien évidemment, de par le repère commun choisi, le robot de queue voit lui sa trajectoire décalée de 13 mètres. L'intégration du biais (principalement statique

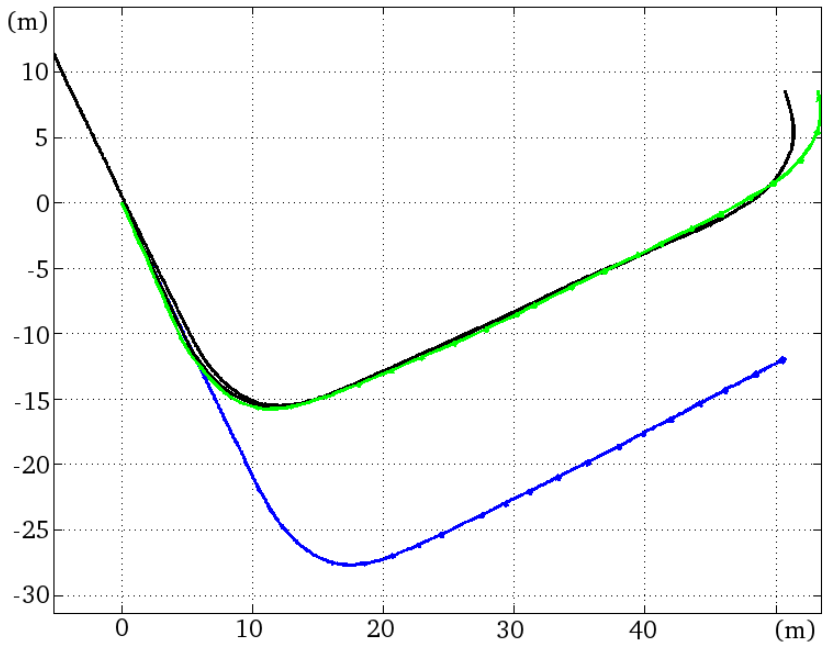


FIGURE 6.19 – Trajectoires obtenues par le SLAM bas niveau pour le scénario convoi colonne. En vert, la trajectoire calculée par le véhicule de tête. En bleu, celle du suiveur (échangée via des communications réseau). En noir, les trajectoires réellement accomplies par les véhicules (vérité terrain).

pour ce véhicule) va permettre de corriger cela. Son évolution a été calibrée une seule et unique fois en fonction du comportement du VIPALab (divergence quadratique observée en fonction de la distance parcourue). Les mêmes paramètres d'évolution seront donc utilisés pour toutes les expérimentations qui vont suivre.

Les figures 6.20 et 6.21 présentent les résultats de localisation avec l'intégration du biais respectivement pour le véhicule de tête et de queue. Pour le véhicule de derrière, la partie statique du biais a été initialisée avec une incertitude de 20 mètres de rayon. Cela est largement supérieur à la distance réelle et permet donc d'assurer la consistance de la localisation dès le début de la trajectoire.

On peut voir sur ces figures que l'intégrité de la localisation est garantie tout au long des trajectoires. Néanmoins, hormis le biais statique conséquent, la dérive reste légère car la distance parcourue est faible. Avec le processus d'association de données actif, la distance réelle entre les véhicules peut alors être estimée conduisant ainsi aux trajectoires de la figure 6.22.

Les résultats sont similaires et même meilleurs que ceux obtenus grâce au simulateur. La trajectoire du véhicule arrière est correctement corrigée et les incertitudes réduites en conséquence. Cela n'empêche d'ailleurs pas la pose des véhicules d'être intègre tout au long des trajectoires. En l'absence d'associations vers la fin du scénario, l'incertitude du biais recommence à grossir afin d'intégrer une dérive potentielle.

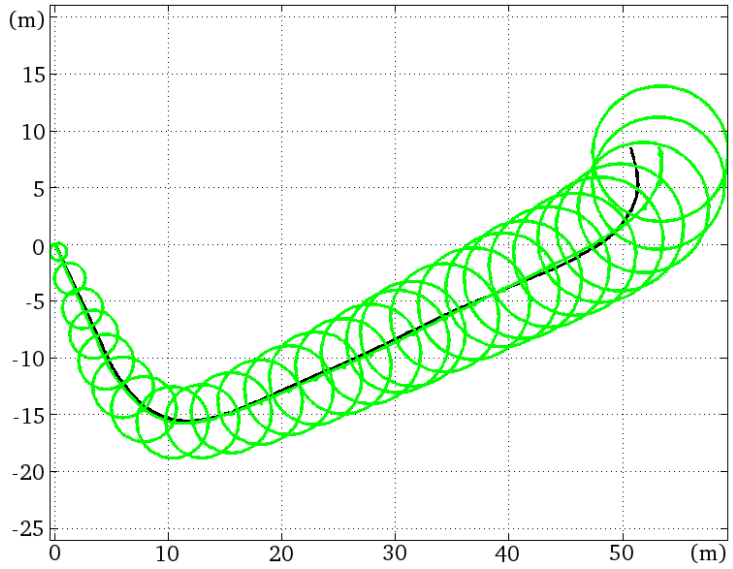


FIGURE 6.20 – Intégration du biais pour le véhicule de tête dans le scénario convoi colonne

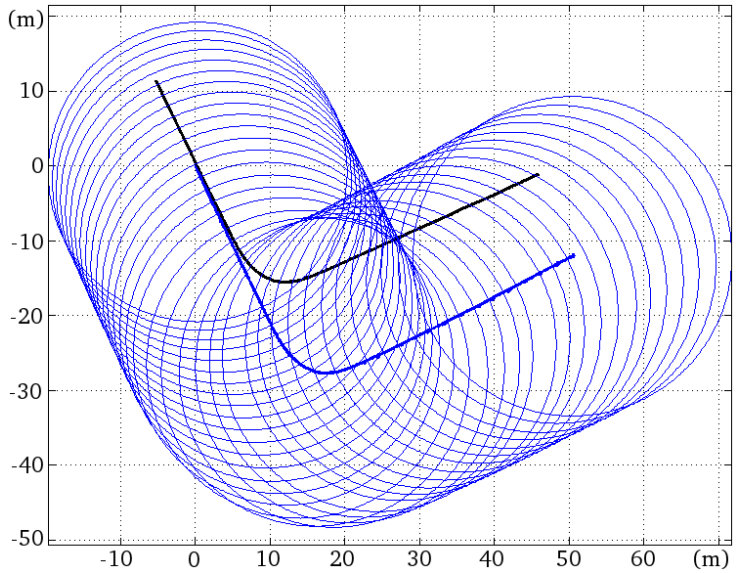


FIGURE 6.21 – Intégration du biais pour le véhicule de queue dans le scénario convoi colonne

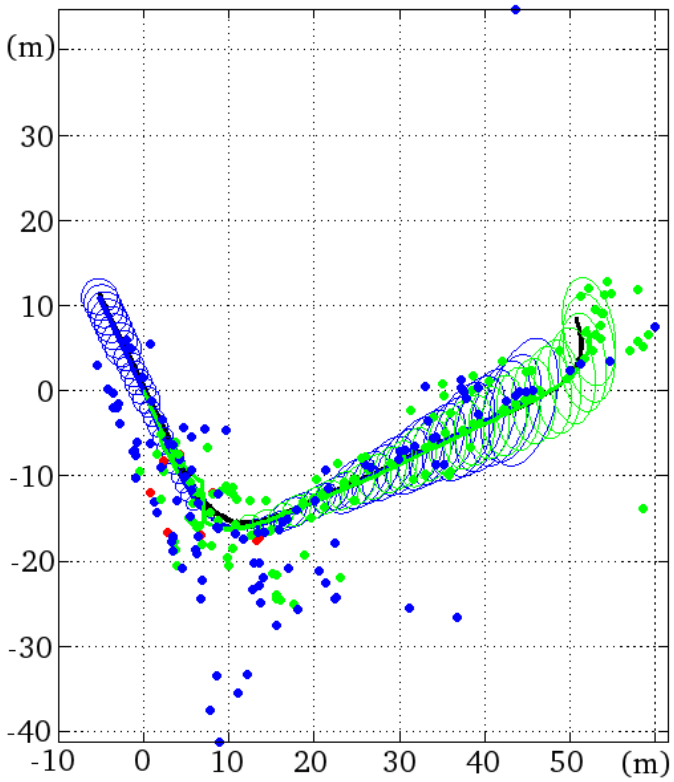


FIGURE 6.22 – Localisation avec association de données pour le convoi colonne réel. Le code couleur est le même que précédemment.

L'écart entre les véhicules, mesuré par des GPS RTK, peut être comparé à celui estimé par notre algorithme. Cela permet d'avoir une idée de la qualité de localisation obtenue en fusionnant des amers (voir figure 6.23).

On peut voir que les résultats donnés par notre algorithme décentralisé sont très proches de la vérité. L'erreur est en moyenne de 30 centimètres. L'erreur maximale observée est de 70 centimètres et est située au démarrage de la trajectoire, loin des zones de fusion. En connectant les différentes estimées du biais ensemble via le filtre de Kalman, l'algorithme est toujours capable de fournir une estimation correcte de la distance séparant les véhicules et ce même lorsque qu'aucune association n'est trouvée.

Les résultats de cette expérimentation sont meilleurs que ceux obtenus avec des trajectoires similaires mais simulées. Une des raisons est le fait que plus d'amers ont été cartographiés ici. Cela facilite la découverte de portions communes entre les deux véhicules et permet donc d'avoir une meilleure estimation de la distance les séparant. Ces résultats confirment également la validité des simulations conduites dans la sous-section précédente. Même avec des réglages légèrement différents, les localisations fournies en simulation par notre algorithme sont très proches de celles obtenues avec des données réelles. Comme pour les simulations, l'application de

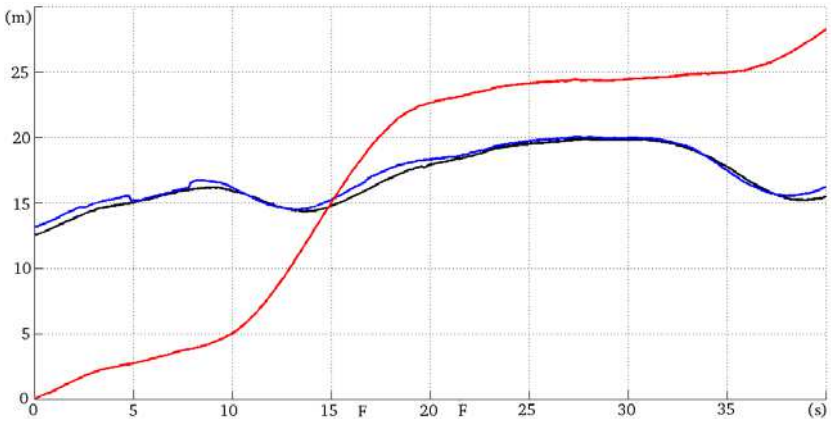


FIGURE 6.23 – Distance entre les véhicules en fonction du temps pour le convoi colonne. En rouge est la distance estimée sans association de données. La courbe noire est le véritable écart séparant les véhicules. Enfin, en bleu est la distance estimée par notre algorithme décentralisé. Les F sur l'abscisse indiquent quand une association de données a été trouvée (au moins 5 amers communs) et donc une fusion réalisée.

SLAM fonctionnait en temps réel pour les deux véhicules et a utilisé une bande passante comparable (voir figure 6.24).

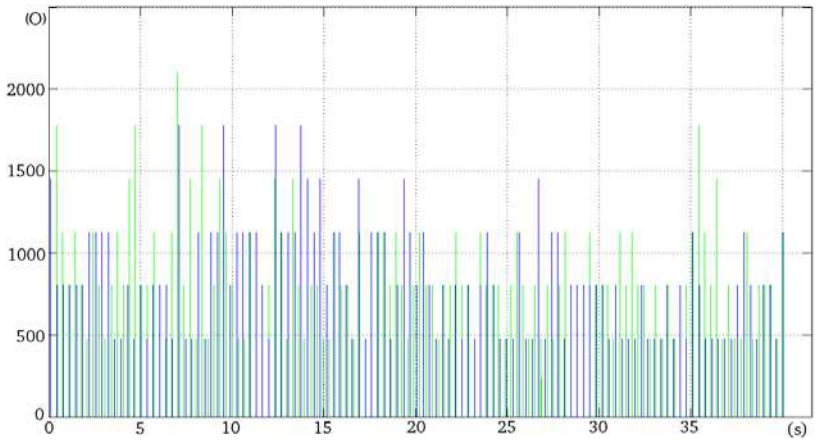


FIGURE 6.24 – Quantité de données envoyées en fonction du temps pour le convoi colonne. La courbe verte représente les données envoyées par le véhicule de tête et en bleu celles émises par le véhicule arrière. La quantité de données est exprimée en octets.

6.2.2 Convoi ligne à deux véhicules

Un des avantages que l'on a en construisant une application entièrement décentralisée est que l'on n'est pas cantonné à une formation de type colonne comme cela est souvent le cas avec d'autres approches. L'absence de meneur permet de bouleverser l'ordre d'un convoi comme cela a été précédemment montré, ou encore de faire évoluer deux véhicules côte à côte.

Cette configuration est intéressante car elle offre de nombreuses possibilités d'applications (agriculture, transport...). Afin de tester ce schéma de coopération avec l'algorithme de SLAM décentralisé complet, deux véhicules se déplaçant côte à côte sur notre plate-forme expérimentale PAVIN ont été utilisés. À cause de la largeur de la route, uniquement une longue ligne droite d'environ 40 mètres a été accomplie. Les deux véhicules étaient séparés par 2 à 3 mètres en fonction de la vitesse des robots qui pouvait varier légèrement au cours de ce scénario. Un aperçu des trajectoires effectuées est disponible en figure 6.25. Les véhicules étaient là encore une fois conduits manuellement.

FIGURE 6.25 – Aperçu des trajectoires du scénario convoi ligne. Les cercles correspondent aux points de départ des véhicules et les croix aux arrêts.

Comme pour l'expérimentation précédente, le point de départ du véhicule vert est utilisé comme repère commun entre les deux robots. Ici, le véhicule vert évolue sur la droite de la route et le bleu sur la partie gauche. Les trajectoires calculées par le bas niveau sont exposées en figure 6.26.

Les incertitudes affichées pour ces trajectoires sont à peine visibles sur la figure et n'incluent donc pas la vérité, ce qui démontre, encore une fois, l'inconsistance des algorithmes de SLAM. On peut en plus constater la dérive et ce même pour le véhicule vert. L'orientation étant mal estimée, la localisation calculée s'éloigne donc progressivement de la vérité. La trajectoire bleue quant à elle est décalée d'environ 2 mètres sur la droite. Afin de couvrir l'erreur de translation initiale du véhicule bleu, le biais a été initialisé avec une incertitude de 3 mètres pour l'erreur latérale et d'1 mètre pour l'erreur longitudinale. L'évolution du biais est ici identique à celle de l'expérimentation précédente. Les figures 6.27 et 6.28 montrent l'intégration du biais pour chacun des véhicules.

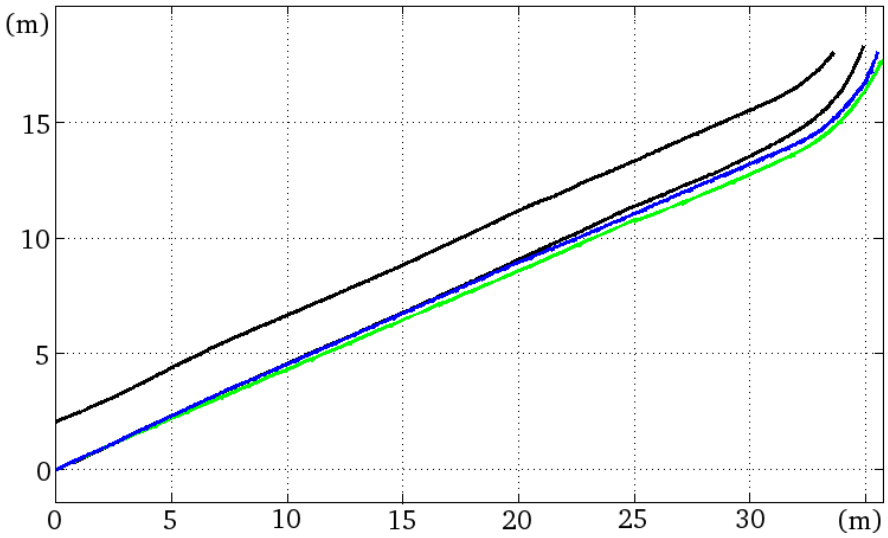


FIGURE 6.26 – Trajectoires obtenues par le bas niveau pour le scénario convoi ligne. En vert, la trajectoire calculée par le véhicule de droite. En bleu, celle du robot de gauche (échangée via des communications réseau). En noir, les trajectoires réellement accomplies par les véhicules (vérité terrain).

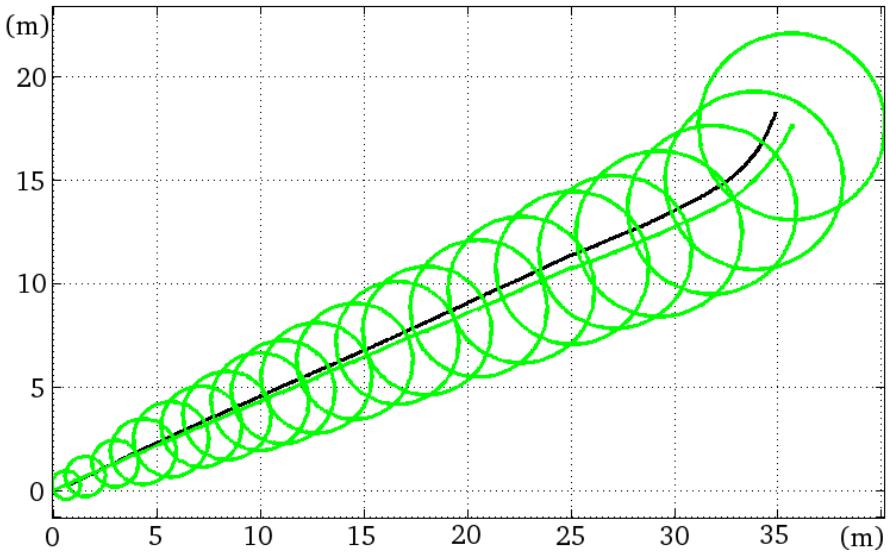


FIGURE 6.27 – Intégration du biais pour le véhicule de droite dans le convoi ligne

L'intégrité du filtre est ainsi maintenue tout au long des deux trajectoires. La valeur initiale utilisée pour l'incertitude du biais permet de s'assurer que le véhicule bleu intègre bien sa vraie position au départ du scénario. Les résultats de localisation lorsque le processus d'association de données est actif sont présentés en figure 6.29.

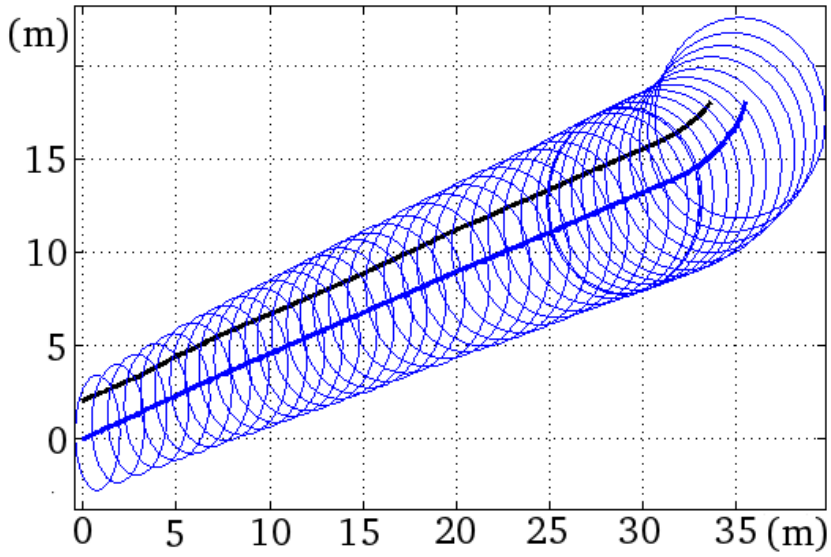


FIGURE 6.28 – Intégration du biais pour le véhicule de gauche dans le convoi ligne

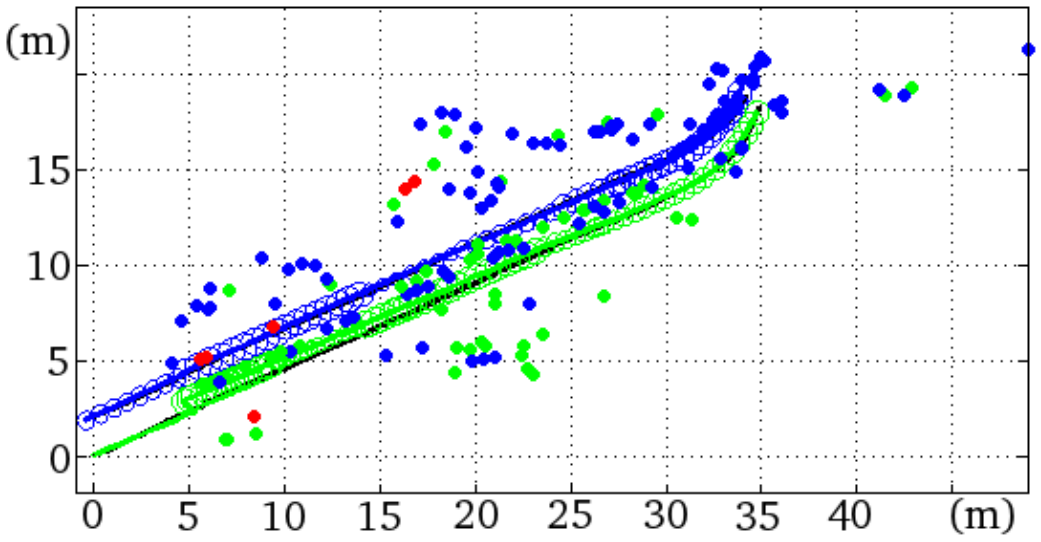


FIGURE 6.29 – Localisation avec association de données pour le convoi ligne. Le code couleur est le même que précédemment.

La distance entre les deux véhicules a pu être estimée grâce à plusieurs associations d'amers. Les incertitudes ont logiquement été grandement réduites. On peut constater que le véhicule vert a un petit saut dans sa localisation après quelques mètres. Celui-ci est causé par une association entre des amers affectés par une incertitude suffisamment importante pour permettre un tel saut. Néanmoins, la localisation est toujours consistante, même lorsque l'incertitude du véhicule augmente pour prendre en compte cette imprécision. L'évolution de la distance entre les véhicules est fournie en figure 6.30.

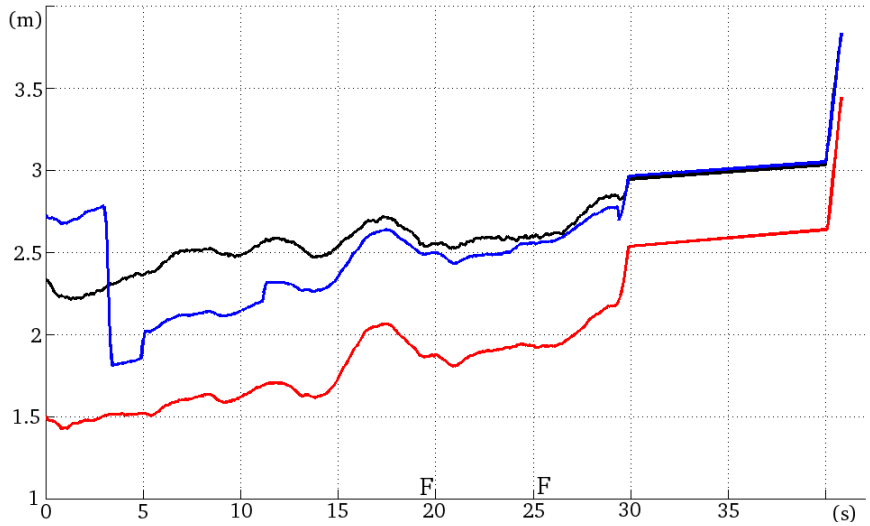


FIGURE 6.30 – Distance entre les véhicules en fonction du temps pour le convoi ligne. En rouge est la distance estimée sans association de données. La courbe noire est le véritable écart séparant les véhicules. Enfin, en bleu est la distance estimée par notre algorithme décentralisé. Les F sur l'abscisse indiquent quand une association de données a été trouvée (au moins 5 amers communs) et donc une fusion réalisée.

Pendant la première partie des trajectoires, la distance calculée par l'algorithme décentralisé oscille avec une erreur maximum de 50 centimètres. Les différentes associations trouvées, ainsi que l'initialisation de plusieurs estimées de biais, causent ces sauts autour de la vraie valeur (situation similaire à celle décrite en section 4.3). Quand des associations sont faites dans la deuxième partie du scénario, l'écart entre les véhicules est bien estimé et stabilisé. L'erreur descend sous les 10 centimètres. Bien évidemment, en l'absence d'association, la distance estimée suit celle du SLAM bas niveau puisque les deux véhicules évoluent côte à côte. On peut noter que la distance initiale entre les deux robots n'est pas nulle. Cela est dû au fait que le véhicule de droite commence à bouger avant celui de gauche. Quand ce dernier commence enfin à se déplacer, l'autre a déjà parcouru 1,5 mètres d'où cet écart en début de scénario pour le SLAM bas niveau.

Il est intéressant de noter que le profil de distance établi par le SLAM monoculaire ne suit pas complètement celui mesuré par les GPS RTK. En effet, chaque

véhicule dérive différemment faisant ainsi varier l'écart les séparant en fonction de la divergence. Cet effet est contraint dans le SLAM haut niveau quand des associations sont trouvées.

Comme cela était attendu, le taux d'échange reste très faible pour cette expérimentation (voir figure 6.31). La bande passante utilisée est inférieure à 5 ko/s ce qui reste loin des limitations technologiques actuelles.

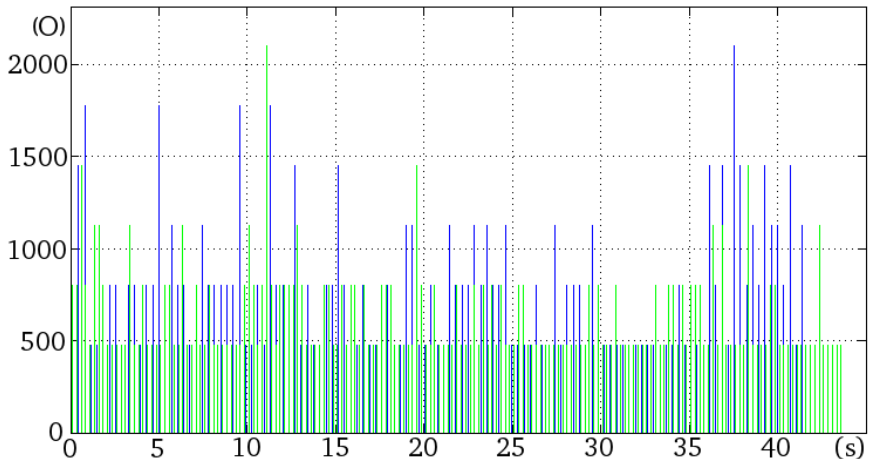


FIGURE 6.31 – Quantité de données envoyées en fonction du temps pour le convoi ligne. La courbe verte représente les données envoyées par le véhicule de droite et en bleu celles émises par le véhicule de gauche. La quantité de données est exprimée en octets.

6.2.3 Convoi à trois véhicules

La dernière expérimentation de ce chapitre présente un exemple de scénario avec 3 véhicules. Une telle configuration est plus complexe à gérer car les flux de données ne viennent plus uniquement d'un véhicule, mais de deux autres. Cela veut également dire que les associations devront être cherchées par rapport à une carte décentralisée plus importante mais moins précise au démarrage. Chaque véhicule se déplaçait approximativement à 2 mètres par seconde sur une trajectoire d'environ 125 mètres. Un aperçu des trois trajectoires accomplies est visible en figure 6.32.

Les 3 véhicules évoluent en convoi colonne quand bien même le dernier robot (ici en bleu) commence avec une orientation différente. La tête et la queue du convoi sont séparés par environ 25 mètres et le véhicule du milieu est 15 mètres devant le dernier. Comme les 3 véhicules ne démarrent pas exactement au même moment, ces écarts initiaux changent très rapidement. Dans ce contexte à 3 véhicules, 2 d'entre eux commencent leur trajectoire avec une position initiale inconnue. Nous avons utilisé le point de départ du véhicule de tête (en magenta ici) comme le repère commun de la flotte. Cela signifie que les deux autres robots sont localisés de manière imprécise dans ce repère. De plus, le véhicule à l'arrière du convoi commence avec

FIGURE 6.32 – Aperçu des trajectoires du scénario à 3 véhicules. Les cercles correspondent aux points de départ des véhicules et les croix aux arrêts.

une erreur angulaire importante (presque 180 °). Les trajectoires de SLAM calculées sans aucune association ou prise en compte du biais sont visibles sur la figure 6.33.

Avec des distances parcourues plus longues, la divergence devient bien plus présente que dans les scénarios précédents. Cela est particulièrement le cas pour la dérive angulaire qui rend les virages plus lâches que ce qu'ils ne devraient. L'erreur affectant le véhicule bleu est encore plus flagrante puisque celui-ci commence la trajectoire à l'envers par rapport au repère commun, expliquant ainsi le fait que celle-ci soit totalement divergente. Les deux autres véhicules terminent aussi leur trajectoire loin de la vérité et ce à cause d'une dérive angulaire majeure. Comme pour les trajectoires précédentes, les ellipses marquant les incertitudes estimées des véhicules sont trop petites pour être visibles. Néanmoins, il est clair que celles-ci n'englobent pas la vérité terrain.

Afin de rester le plus clair possible, nous ne montrerons pas les 3 trajectoires avec la prise en compte du biais. Nous présenterons uniquement celle dont la localisation est le plus affectée par le biais : le véhicule de queue. L'incertitude quant à sa pose avec l'intégration du biais est disponible en figure 6.34.

Il est possible de noter que les valeurs utilisées pour initialiser l'incertitude de biais couvrent bien plus que la position réelle du véhicule bleu. En effet, l'idée était de mettre le processus d'association de données dans une situation difficile avec de nombreuses correspondances potentielles. L'évolution de l'incertitude quant à la position des véhicules n'inclut pas la vérité terrain tout au long de la trajectoire. Cela s'explique très simplement par le fait que la majorité de la dérive est liée à une

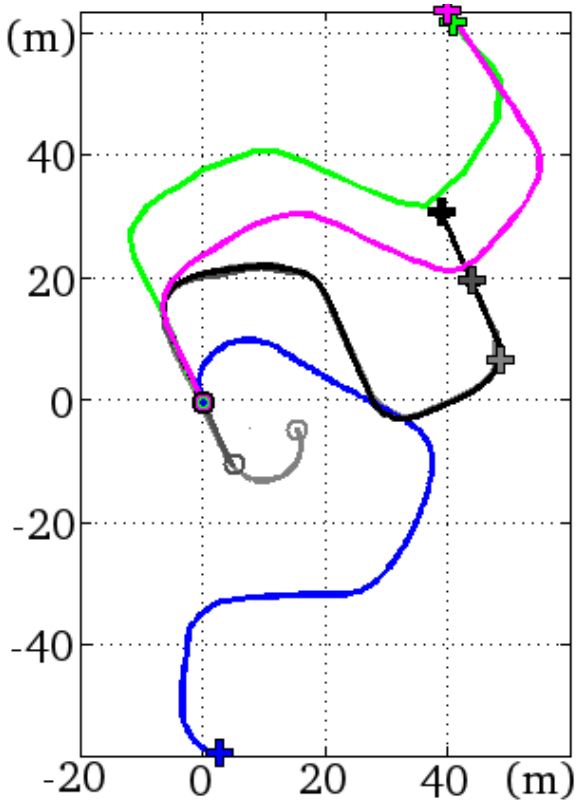


FIGURE 6.33 – Trajectoires obtenues par le bas niveau pour le convoi à trois véhicules. En magenta, la trajectoire calculée par le véhicule de tête. En vert, celle du véhicule au milieu et en bleu, celle du robot de queue. En noir, la vérité terrain du véhicule magenta. En gris foncé, celle du véhicule vert et en gris clair, celle du robot bleu. Les cercles indiquent les points de départ et les croix les arrêts.

divergence angulaire. Celle-ci est prise en compte par le véhicule mais ne peut pas être représentée graphiquement. L'incertitude initiale du biais angulaire a été fixée de telle sorte que le véhicule puisse être orienté dans n'importe quelle direction. En prenant en compte ce biais angulaire, la consistance est bien garantie puisque cela signifie que la trajectoire estimée par le SLAM bas niveau peut être complètement tournée afin de s'approcher de la vérité terrain. Des résultats similaires ont été observés pour les deux autres véhicules.

En considérant les associations de données, l'algorithme de SLAM décentralisé calcule les trajectoires visibles dans la figure 6.35. Pour des raisons de visibilité, les incertitudes ne sont pas affichées sur cette figure.

On peut constater que notre approche décentralisée est capable d'estimer correctement les écarts séparant les véhicules. La dérive angulaire n'est bien sûr pas corrigée car elle est commune aux trois véhicules. Sans une information absolue ou une fermeture de boucle, il est impossible de corriger l'orientation fournie par le

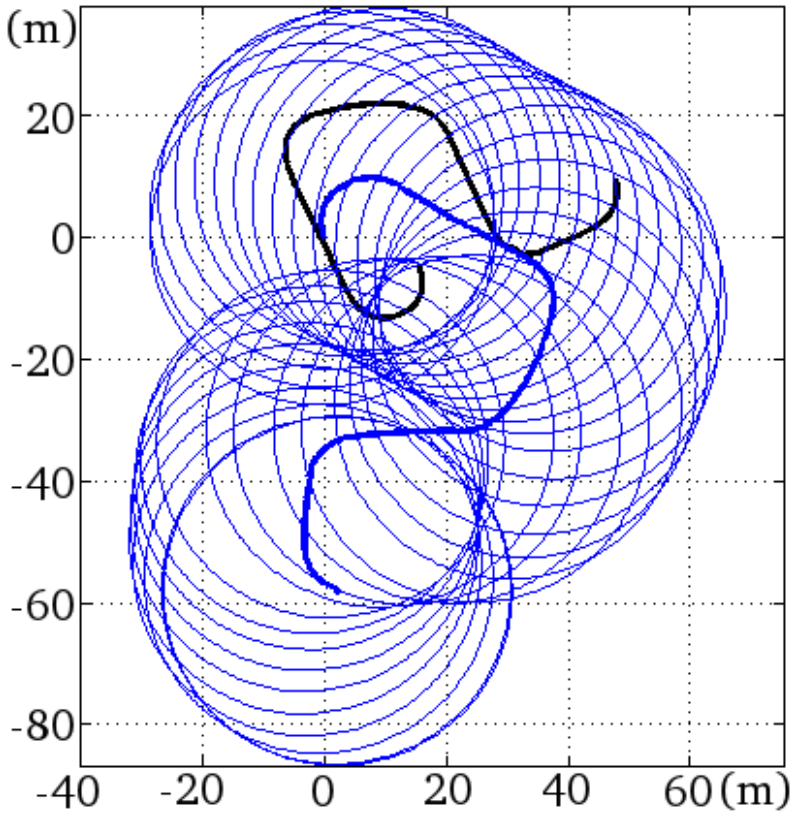


FIGURE 6.34 – Intégration du biais pour le véhicule de queue dans le scénario convoi à trois véhicules

SLAM bas niveau. À défaut, la dérive angulaire est tout de même contrainte dans la flotte par les associations.

Malgré le fait que la trajectoire du véhicule bleu soit globalement corrigée, on peut noter qu'une erreur de position l'affecte toujours au départ de la trajectoire. Pour trouver des associations, il est nécessaire que le véhicule de queue passe par des zones cartographiés par les deux autres véhicules. Quand cela est possible, l'incertitude du biais a déjà bien augmentée. Ainsi, la rétro-action appliquée sur le début de la trajectoire ne permet pas d'estimer entièrement la dérive de position.

On peut également observer un petit écart entre la trajectoire du véhicule bleu et celles des autres sur le reste de la trajectoire. Cela est dû au fait que celui-ci démarre dans un virage alors que les deux autres débutent en ligne droite. La conséquence est que la dérive est différente au début de la trajectoire pour le véhicule bleu. Lorsque des associations sont trouvées, les incertitudes des biais sont déjà trop importantes pour retrouver entièrement la distance séparant les véhicules. Néanmoins, celle-ci est globalement bien estimée même avec ce petit écart.

Pour les mêmes raisons que celles évoquées précédemment, uniquement les incertitudes et la trajectoire du véhicule bleu sont montrées sur la figure 6.36.

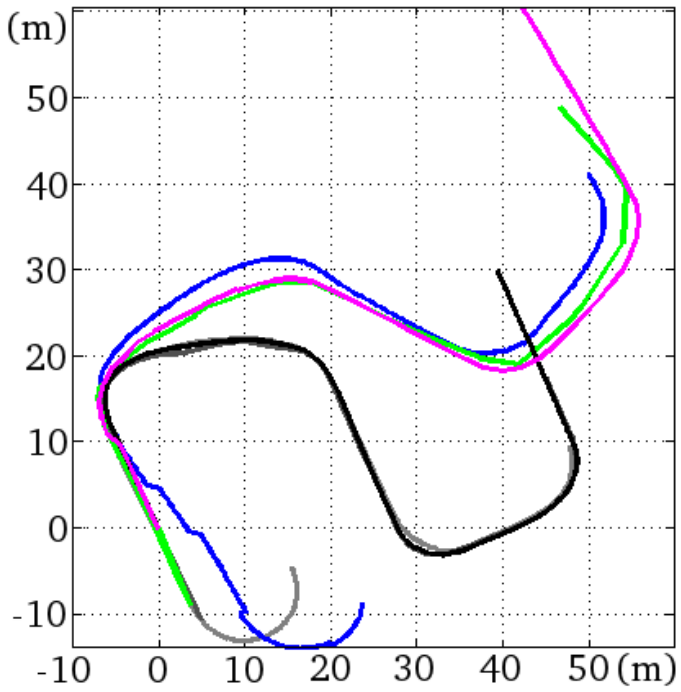


FIGURE 6.35 – Localisation avec associations pour le convoi à trois véhicules. Le code couleur est le même que précédemment.

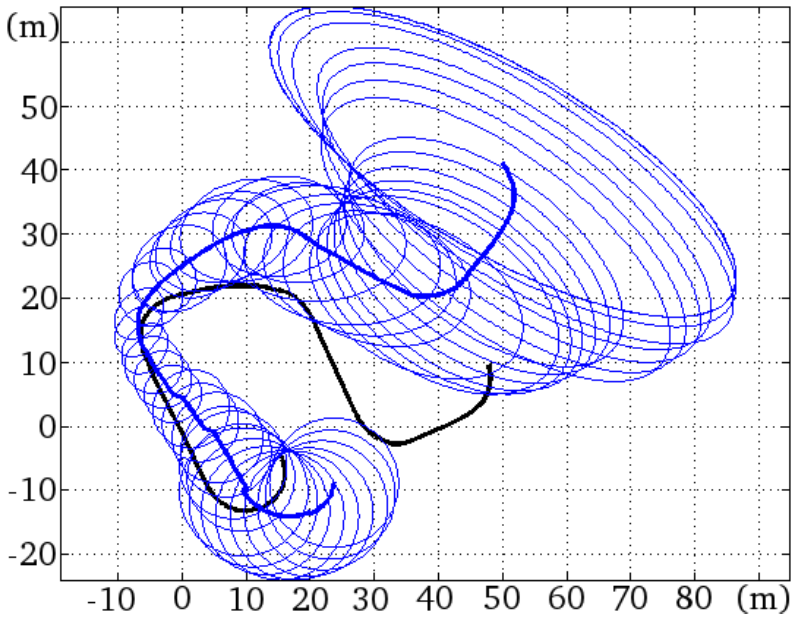


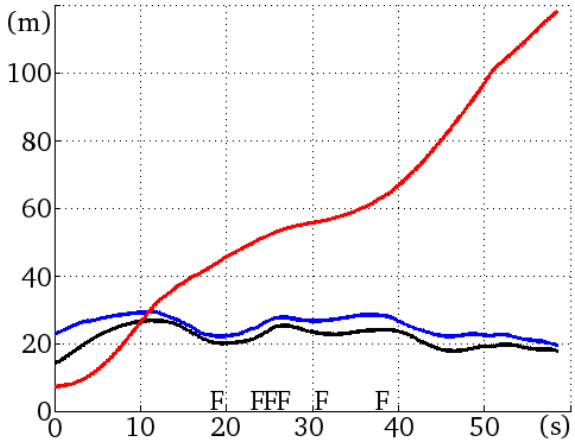
FIGURE 6.36 – Localisation avec association de données pour le véhicule de queue dans le convoi à trois véhicules

Les incertitudes affectant la pose du véhicule sont encore une fois grandement réduites une fois que des associations sont trouvées. Cela est surtout le cas durant la première moitié de la trajectoire. À la fin du scénario, l'incertitude du véhicule commence à grandir de nouveau car moins d'amers sont disponibles, empêchant ainsi de trouver des correspondances avec le reste de la flotte. Tout comme pour la figure 6.34, l'incertitude angulaire ne peut pas être représentée ici. Il s'agit de celle affectant le plus le véhicule. Celle-ci est néanmoins intégrée dans le biais afin d'assurer la consistance de la pose véhicule. À la fin de la trajectoire, l'incertitude du biais concernant le cap du véhicule est d'environ 60 °. Cela signifie que la position finale du robot peut être corrigée afin de venir s'aligner sur la vérité car l'incertitude d'orientation le permet. Ainsi, la consistance de la pose est respectée même lorsque le biais de localisation ne suffit plus à inclure la vérité terrain. Les profils de distance pour les différents cas possibles sont illustrés en figure 6.37. La qualité de chaque écart peut ainsi être quantifiée.

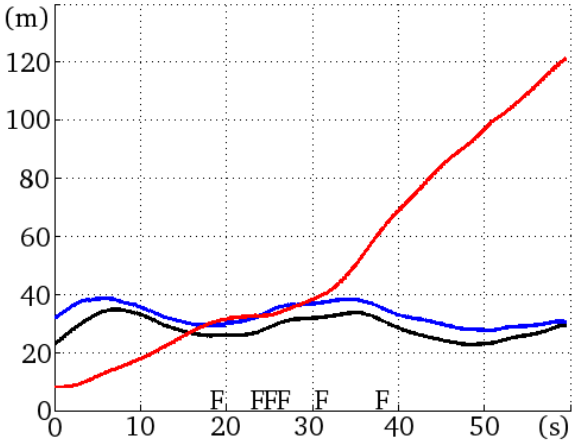
Ces différents résultats illustrent que notre algorithme décentralisé est capable de calculer un profil de distance qui est très similaire à la réalité. Les imperfections dans ces profils peuvent être expliquées par deux raisons. La première est liée à l'évolution de la dérive. Chaque véhicule est affecté par une dérive qui, bien que similaire, n'est pas la même pour toute la flotte. Ainsi, il n'est pas possible de retrouver la vraie distance qui sépare les véhicules à cause des fortes incertitudes affublant chaque pose. La seconde raison pour ces erreurs est l'utilisation d'uniquement un algorithme d'association de données haut niveau. La conséquence est que la précision ne peut descendre en dessous du seuil de convergence fixé. Cependant, les distances estimées sont assez proches des écarts réels. En moyenne, l'erreur est de 3 mètres pour l'écart entre les véhicules bleu et vert (figure 6.37(a)), 5 mètres entre les robots bleu et magenta (figure 6.37(b)) et 70 centimètres entre le vert et le magenta (figure 6.37(c)). Les écarts plus importants par rapport au véhicule bleu sont logiques car il est initialement localisé de façon très imprécise (erreur angulaire de presque 180 °). De plus, il est celui le plus éloigné du véhicule de tête, ce qui rend l'estimation précise de son écart à celui-ci plus délicate à obtenir.

De façon similaire aux expérimentations précédentes, la figure 6.38 montre les nécessités en bande passante avec les trois véhicules opérant ensemble.

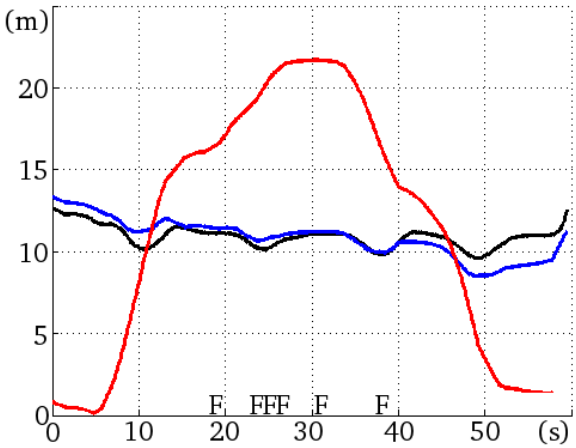
Encore une fois, les résultats sont très satisfaisants avec un taux d'échange faible. On peut observer que le véhicule magenta envoie plus d'informations d'un coup que les autres. Cela est dû au fait que chaque véhicule communique avec des fréquences qui ne sont pas les mêmes. En effet, afin de montrer que les désynchronisations n'affectent pas notre système, nous avons utilisé des temporisations différentes pour les envois de données. Celle du véhicule magenta était plus importante que les autres, expliquant ainsi qu'il communique plus de données chaque fois. Chaque véhicule a utilisé moins de 10 ko par seconde.



(a) Distance entre les véhicules bleu et vert



(b) Distance entre les véhicules bleu et magenta



(c) Distance entre les véhicules vert et magenta

FIGURE 6.37 – Profil de distance en fonction du temps pour le convoi à trois véhicules. En rouge est la distance estimée sans association de données. La courbe noire est le véritable écart séparant les véhicules. Enfin, en bleu est la distance estimée par notre algorithme décentralisé. Les F sur l'abscisse indiquent quand une association de données a été trouvée (au moins 5 amers communs) et donc une fusion réalisée.

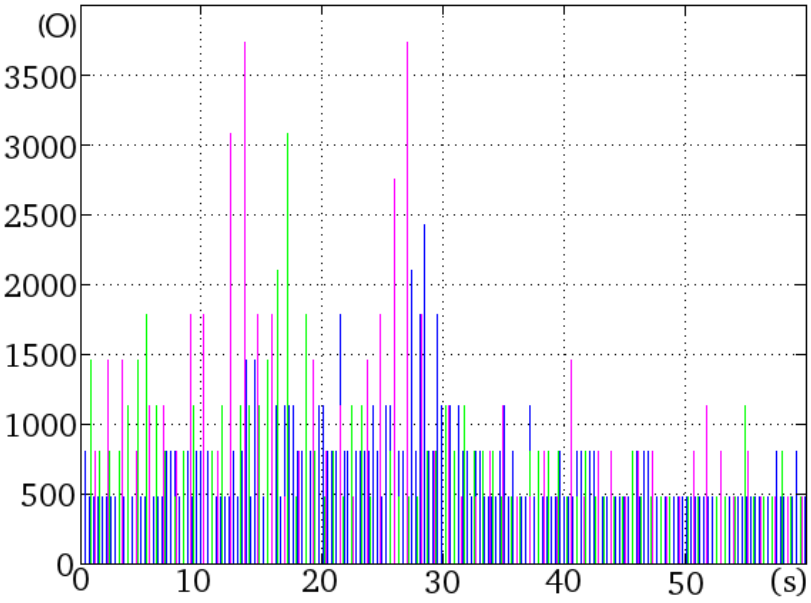


FIGURE 6.38 – Quantité de données envoyées au cours du temps pour le convoi à trois véhicules. Les courbes bleue, verte et magenta sont respectivement associées aux véhicules commençant à l'arrière, au milieu et au début du convoi. La quantité de données est exprimée en octets.

La qualité des localisations obtenues pourrait être améliorée en cartographiant plus d'amers. Bien que les besoins en bande passante ne soient pas un problème, le temps de calcul requis pourrait en être un. Au-delà de 40 amers suivis par image, et avec le coût de la surcouche décentralisée, il est difficile d'augmenter davantage le nombre de points dans le SLAM. Néanmoins, quelques optimisations, évoquées dans les perspectives de ce manuscrit (section 7.2), permettraient de lever cette restriction.

6.3 Analyse des résultats

La réalisation de ces trajectoires permet de tirer une analyse de notre application de SLAM décentralisé. Le premier aspect important est que notre approche est viable pour faire du SLAM décentralisé. Les résultats obtenus montrent clairement que l'estimation de la pose de chaque véhicule dans un repère commun est faisable, même avec un a priori extrêmement faible. L'algorithme d'association de données développé aide grandement en fournissant des correspondances fiables entre plusieurs amers. Celles-ci permettent de réaliser des mises à jour du biais afin de corriger l'ensemble du vecteur d'état décentralisé.

La consistance concernant la localisation des véhicules et des amers est assurée. Dans un premier temps, l'initialisation de la partie statique du biais permet de gérer le problème des positions initiales inconnues. Ainsi, l'intégrité du filtre est

maintenue en début de trajectoire. Ensuite, le biais dynamique permet de conserver la consistance en faisant évoluer l'incertitude afin de couvrir une dérive potentielle. Les résultats montrent encore une fois que ces deux aspects sont efficaces. À l'aide de correspondances, le biais peut être estimé afin de corriger les positions dans la carte globale. Celui-ci étant connecté à l'intégralité de l'état, la mise à jour affecte la carte entière. De plus, les diverses expérimentations ont montré que, même après plusieurs mises à jour, les localisations calculées sont toujours intègres. Le modèle de biais développé est donc bien adapté à l'évolution de la dérive.

Bien que les résultats ne le mettent pas avant, en présence d'une erreur angulaire importante, il est possible que l'EKF produise une mise à jour erronée lors de la première estimation du biais statique. En effet, des erreurs de linéarisation peuvent intervenir car l'intégration par le modèle proposé est non linéaire dès lors qu'une dérive angulaire existe. Bien que ce comportement ne se soit pas manifesté au cours des expérimentations (et tout particulièrement dans la dernière où un biais statique angulaire majeur est présent), il est possible que la mise à jour initiale du biais statique soit fausse. Pour éviter cela, une solution consisterait à effectuer la première estimation du biais statique hors du filtre de Kalman. C'est en effet celle-ci qui permet de réduire grandement l'incertitude du biais angulaire initial. Après cette première mise à jour, la valeur du biais est suffisamment proche de la vérité pour être de nouveau estimée dans le filtre de Kalman. Ainsi, les erreurs de linéarisation sont en grande partie évitées. L'estimation hors du filtre de Kalman pourra se faire en calculant la transformation entre les amers provenant de deux véhicules différents pour ensuite remonter à la pose relative.

L'ajout d'un algorithme de détection des véhicules, à l'aide d'un télémètre laser par exemple, permettrait de faciliter l'estimation du biais de localisation. En effet, l'intégration d'une telle information est aisée dans la structure conçue et affinerait l'estimation d'interdistance faite jusqu'ici uniquement avec des amers communs. Dans le cas où la configuration des robots l'autoriserait, la mesure directe de l'écart entre les véhicules serait un atout considérable.

Au niveau des contraintes liées au réseau, la bande passante utilisée au cours de ces différents scénarios est très faible. Cela implique que notre approche peut facilement s'accommoder d'un nombre de véhicules plus important sans pour autant risquer la congestion du réseau. Les aspects problématiques habituellement associés au SLAM décentralisé sont ici parfaitement pris en compte, que cela soit la perte de communication, les latences ou encore les désynchronisations. Ces aspects ont d'ailleurs pu être éprouvés au cours d'expérimentations réelles.

Les contextes de chaque scénario étant assez différents, nous avons pu éprouver notre algorithme dans diverses conditions. Celui-ci s'est avéré capable de gérer tout aussi bien des distances intervéhicules importantes que des convois en ligne. Le classique convoi en colonne a également fait l'objet de bons résultats. Enfin, un grand convoi à 3 véhicules a permis de montrer la faisabilité de ce genre d'approches en se basant uniquement sur des associations de points 3D. Bien que le nombre d'amers cartographiés soit faible, suffisamment d'associations fiables sont trouvées.

Il s'agit, à notre connaissance, du tout premier SLAM monoculaire décentra-

lisé présenté. De plus, il est très rare que des expérimentations mettant en jeu des données réelles soient utilisées pour valider un algorithme de SLAM distribué. Les difficultés de mise en oeuvre limitent généralement ce point. Ici, nous avons pu montrer que de telles approches sont possibles dès lors que l'on considère bien l'intégralité du problème et pas seulement certains aspects.

Enfin, le faible nombre d'amers évoqués au cours de ces expérimentations permet d'avoir une empreinte mémoire raisonnable. En effet, seulement quelques centaines de points 3D sont requis par l'algorithme décentralisé pour des trajectoires de l'ordre de la centaine de mètres. Au vu des autres approches citées au cours de l'état de l'art, ces résultats sont tout à fait satisfaisants du point de vue de la quantité de mémoire consommée par l'algorithme. Concernant le coût calculatoire, l'algorithme a opéré en temps réel pour les différentes trajectoires effectuées. Le bas niveau occupait approximativement un tiers du temps de calcul contre deux tiers pour la surcouche décentralisée. Néanmoins, quelques optimisations permettraient de pouvoir augmenter le nombre d'amers à suivre ou bien encore d'améliorer certains traitements avec des étapes de vérification supplémentaires. Celles-ci seront évoquées dans le prochain chapitre.

Des environnements non urbains pourraient également être testés afin de confronter notre algorithme à des conditions différentes. Également, plus de scénarios pourraient être effectués. Cependant, les résultats obtenus ici sont plus qu'encourageants et la transposition à d'autres environnements ne devrait pas poser de problèmes majeurs.

Ce chapitre a présenté divers résultats pour le SLAM décentralisé. Ceux-ci ont été abondamment commentés et analysés. Il en ressort que notre approche est viable pour des flottes de plusieurs véhicules. La qualité des localisations est bonne en l'absence de connaissance initiale sur la position des véhicules. Le prochain chapitre donnera quelques clefs pour comprendre comment notre SLAM décentralisé peut évoluer à l'avenir.

Chapitre 7

Conclusion et perpespectives

Sommaire

7.1 Conclusion

L'objectif de cette thèse était de réussir à mettre en place un SLAM entièrement décentralisé et basé sur des capteurs peu onéreux. Pour ce faire, chaque aspect impliqué dans la conception de cet algorithme a été présenté, justifié et amplement testé. Les différents chapitres de ce manuscrit se sont attardés sur les innovations apportées afin de rendre possible l'utilisation du SLAM au sein d'une flotte de véhicules coopérants.

Les visées applicatives d'un tel algorithme sont nombreuses : transport de personnes, agriculture, réalité augmentée... Au-delà de ces applications concrètes, notre objectif était de fournir une solution de localisation qui ne soit pas liée à des capteurs spécifiques ou à une configuration particulière de la flotte. Ainsi, le nombre de véhicules du groupe peut être important (plus d'une dizaine) et ne devait pas être restreint par les envois de données sur le réseau. De plus, les environnements, bien que statiques, peuvent être variés et devaient être pris en compte afin d'assurer la robustesse de l'algorithme. Au niveau de la justesse de localisation, il est évident qu'avec une approche générale il n'est pas possible d'atteindre une précision de l'ordre de la dizaine de centimètres sur de grandes distances. Néanmoins, l'application devait pouvoir être spécialisée pour atteindre une telle précision. Pour cela, nous nous sommes attachés à traiter chaque point particulier de l'algorithme afin de le rendre le plus efficace possible au vu des critères rappelés ici.

Tout d'abord, l'aspect SLAM monovéhicule a été étudié avec attention. Le but était de fournir un algorithme de base qui soit léger aussi bien du point de vue de la quantité de mémoire nécessaire que du temps de calcul. De même, et afin de rendre possible une extension multivéhicule, les capteurs utilisés devaient être peu chers. En combinant ces contraintes, nous nous sommes tournés vers l'utilisation d'un filtre de Kalman étendu basé sur une seule caméra. Bien évidemment, ce schéma apporte son lot de problématiques qui doivent être traitées. L'EKF est très sensible aux erreurs de linéarisation et le fait de n'avoir qu'une seule caméra tend à favoriser ces dernières. En effet, la profondeur d'un amer étant inconnue après son initialisation, la linéarisation de la fonction d'observation autour de cette estimée peut donner des résultats aberrants. Ceux-ci ont d'ailleurs été largement commentés au cours de ce manuscrit. Pour contrer cet effet indésirable, nous avons proposé, sur la base d'anciens travaux, des correctifs permettant de quasiment supprimer l'impact des mauvaises linéarisations.

Dans un premier temps, une approche géométrique a été introduite afin que le suivi des points ne soit plus dépendant de la fonction d'observation. Celle-ci permet d'obtenir une fenêtre de recherche dans l'image qui correspond parfaitement à l'incertitude 3D du point.

Par ailleurs, concernant la mise à jour du filtre de Kalman, il a été constaté que celle-ci pouvait mal se dérouler et appliquer des corrections fausses à l'état. Un palliatif a été introduit afin de limiter l'impact de ces mises à jour. Cela se traduit sous la forme d'un correctif du gain de Kalman qui permet de contrôler la mise à jour quand celle-ci s'appuie sur une mauvaise linéarisation.

Les différents points soulevés ici ont d'abord été validés individuellement puis ensemble via des expérimentations variées. La qualité des cartes peu denses construites a ainsi pu être validée. De même, la robustesse de ce SLAM a été éprouvée en le confrontant à diverses conditions (caméra plus lente, environnement ouvert ou encore conditions défavorables comme un fort ensoleillement). Dans chacun de ces scénarios, les résultats obtenus étaient très satisfaisants.

Malgré les améliorations apportées au SLAM monoculaire, les résultats obtenus indiquaient clairement une dérive progressive de la localisation. Ce problème classique a été constaté à de nombreuses reprises dans la littérature sans que des solutions satisfaisantes ne soient proposées. La divergence observée dans la localisation rend cette dernière inconsistante, c'est-à-dire que l'estimation de la pose du véhicule n'intègre pas la vérité. Cela pose le problème de la prise en compte des fermetures de boucle ou encore d'informations absolues. En effet, la fusion de données non intègres peut produire des résultats aberrants. Cela est encore plus délicat avec plusieurs véhicules puisque chacun va diverger différemment des autres. La mise en commun des données dans une carte partagée requiert que cette information de divergence soit intégrée afin de pouvoir positionner les membres de la flotte les uns par rapport aux autres.

Pour ce faire, nous avons proposé de modéliser la dérive affectant le SLAM via un modèle dynamique représentant celle-ci comme un biais de localisation. Bien entendu, la dérive entre deux instants ne pouvant être connue (sauf cas particuliers), il était nécessaire de pouvoir représenter ce manque de connaissance sous la forme d'une incertitude afin de garantir la consistance de la localisation. De même, une architecture spécifique a été conçue afin d'éviter que l'incertitude provenant du biais ne soit mélangée avec celle du véhicule. Ainsi, nous pouvons garantir l'intégrité des données en sortie de cette architecture dédiée. Le design de cette dernière s'est fait en gardant l'aspect général que devait avoir notre approche. En décorrélant totalement le SLAM classique, dit bas niveau, de l'estimation du biais, il devient possible de facilement changer la technique de filtrage ou encore les capteurs utilisés. La connexion du biais au reste du vecteur d'état (amers et pose véhicule) a ensuite été introduite. Celle-ci se fait par l'intermédiaire d'un filtre de Kalman étendu qui permet d'établir le lien de dérive affectant l'ensemble des données. Le coût calculatoire de l'intégration du biais est quasiment nulle dès lors que l'on considère le bas niveau comme une odométrie visuelle fournissant des amers précis.

La prise en compte de la dérive a été validée au travers de plusieurs expérimentations. Celles-ci mettaient en scène un seul véhicule face à des situations permettant d'estimer ce biais de localisation. Pour le premier scénario, une trajectoire bouclant sur elle-même a été présentée. La deuxième trajectoire proposait d'intégrer des amers géo-référencés afin d'aider la localisation. Les résultats obtenus dans les deux cas montrent que la consistance de la localisation est conservée. De plus, notre système permet de corriger l'ensemble du vecteur d'état, dès lors qu'une estimation du biais est faite, sans aucun traitement particulier.

Enfin, les concepts présentés dans le cadre monorobot ont été étendus à plusieurs véhicules. Pour cela, il a été nécessaire de prendre en compte les contraintes induites

par un tel contexte. On peut en lister un certain nombre. En nous orientant vers une approche entièrement décentralisée, le point le plus important était d'éviter la consanguinité des données. Ce phénomène, qui correspond au fait de prendre en compte plusieurs fois la même information, nécessite soit des méthodes de filtrage adaptées, soit une architecture dédiée, pour être évité. Nous avons opté pour la seconde solution en étendant celle introduite pour la gestion du biais. En séparant les données provenant de plusieurs véhicules en sous-cartes, nous évitons ainsi qu'elles ne soient utilisées plusieurs fois. Chaque véhicule a ensuite la charge de reconstruire la carte globale prenant en compte tous les sous-états. Le coût calculatoire engendré par cette étape peut être important. Aussi, nous avons décidé d'uniquement enrichir cette dernière en ne transmettant que des amers précis et définitifs. Les raisons de ce choix sont multiples : moins de communication entre les véhicules, le coût calculatoire est plus faible, le biais affectant ces amers est bien supérieur aux mises à jour ratées.

Cette architecture permet d'ajouter naturellement la composante multivéhicule au SLAM traditionnel. Néanmoins, cette structure reste générale et le SLAM bas niveau pourra être changé à loisir. La combinaison de cette architecture avec la prise en compte de la dérive permet aussi de résoudre les problèmes liés aux communications réseau. En effet, le fait de ne communiquer qu'une seule version de chaque amer permet de facilement régler les pertes de données et de ne pas être affecté par les retards dans les échanges. Enfin, et c'est certainement le plus important, la bande passante nécessaire par véhicule est très faible. En effet, l'intégration de la dérive permet de n'envoyer que les variances des points, sans les covariances croisées, celles-ci étant négligeables par rapport au biais de localisation. De plus, le fait de ne communiquer qu'une version d'un amer allège encore considérablement la charge d'informations à transmettre. Le choix de l'EKF joue également un rôle crucial dans cette faible bande passante puisque ce filtre ne requiert que quelques amers pour produire des résultats précis.

L'extension de la modélisation de la dérive à plusieurs véhicules a permis d'introduire la notion de biais statique. Il s'agit de l'erreur initiale affectant les véhicules d'une flotte vis-à-vis de leur repère commun. Ainsi, le SLAM décentralisé développé peut intégrer n'importe quel a priori de localisation que cela soit une distance au repère commun ou encore une incertitude quant à la position de celui-ci. Un algorithme d'association de données, adapté à plusieurs véhicules, a également été introduit. Celui-ci tire parti à la foi de la configuration géométrique des amers, de leur incertitude et de leur aspect visuel. En combinant ces informations, les résultats obtenus sont très satisfaisants même avec peu d'amers communs dans la carte.

L'application complète a été testée sur de nombreux scénarios en simulation et avec des données réelles. Nous avons ainsi prouvé la viabilité de notre approche pour diverses situations : un convoi colonne avec une distance intervéhicules importante, un convoi avec inversion de l'ordre en milieu de trajectoire, un convoi en ligne ou encore un convoi à 3 véhicules. À chaque fois, les résultats, obtenus en temps réel, ont été très satisfaisants.

Néanmoins, certains points peuvent être améliorés afin d'atteindre de meilleures

performances. Cela implique autant des aspects concernant l'algorithme de SLAM bas niveau que l'architecture décentralisée. Certains ont déjà été évoqués auparavant mais une synthèse en est proposée dans la prochaine section.

7.2 Perspectives

La construction d'un algorithme de SLAM décentralisé a entraîné certains compromis afin de pouvoir traiter tous les problèmes affectant habituellement ce genre d'approche. Bien que les résultats de localisation obtenus soient satisfaisants, des aspects de l'application pourraient être améliorés.

L'approche construite est générale. Cela signifie que n'importe quel SLAM classique peut être utilisé. Néanmoins, nous avons proposé dans ce manuscrit une version adaptée aux contraintes induites par le schéma décentralisé. Les critères majeurs étaient le coût (qu'il soit financier ou calculatoire) et la quantité de données amenée à être échangée. Pour ces raisons, nous nous sommes tournés vers un SLAM monoculaire basé sur une EKF itératif. Ce dernier point induit une forte sensibilité aux mauvaises associations de données. Cet aspect a déjà été évoqué auparavant et une solution pourrait être de mettre en place des descripteurs plus discriminants. Il reste également envisageable d'utiliser une solution permettant de retourner en arrière dans le cas où une fausse association est faite. En rendant l'association de données du SLAM bas niveau plus robuste, il sera alors possible d'avoir des amers en plus grande quantité et plus précis dans le haut niveau.

Néanmoins, le surcoût que peuvent impliquer ces techniques requiert de passer par une phase d'optimisation du bas niveau. Dans tous les cas, accélérer le SLAM monoculaire peut être intéressant afin de libérer du temps pour les traitements décentralisés. Bien que le SLAM monoculaire soit capable de fonctionner à plus de 30 Hz, le fait de lui adjoindre la partie décentralisée nécessite de baisser la fréquence autour de 15 Hz. Un gain de temps substantiel pourrait être dégagé en migrant les traitements purement vision (détecteur de Harris, ZNCC) vers le GPU. En effet, ces algorithmes se prêtent bien à du calcul parallèle et sont les plus coûteux du bas niveau. Il ne s'agit pas ici d'un verrou scientifique mais davantage d'un problème technique.

Le SLAM conçu est prévu pour fonctionner dans des environnements statiques. Aucune méthode particulière n'a été envisagée pour étendre notre méthode à la détection d'objets mobiles. Ce n'est pas forcément nécessaire mais en fonction des applications visées, cela peut le devenir. Par exemple, dans le cas des ITS, le but est de transporter des gens d'un point A à un point B. Il est fort probable que sur ces trajets des piétons, voire des voitures, soient présents. Ne pas prendre en compte ces informations peut conduire le SLAM à de fausses estimations. En effet, il est fort probable que des amers soient initialisés sur ces objets mobiles. Le suivi est alors altéré et le déplacement de l'objet va être confondu avec celui du véhicule, créant ainsi des erreurs de localisation. La prise en compte d'un environnement dynamique est une étape cruciale pour le déploiement en milieu urbain. Une des

possibilités serait l'extension des techniques de SLAMMOT (SLAM and Moving Object Tracking pour SLAM et suivi d'objets mobiles) à un contexte décentralisé.

Concernant la dérive du SLAM, nous avons déjà évoqué la possibilité de construire un modèle plus adapté. La dérive angulaire est dépendante de l'angle volant. Plus le virage sera serré, plus il y a de chances que le cap du véhicule diverge. Une solution possible pour améliorer notre modèle serait de faire intervenir l'angle volant directement dans l'estimation de la dérive.

Cela n'est cependant pas l'aspect du biais le plus problématique. Chaque nouvelle information, que cela soit une pose véhicule ou un amer, doit être connectée au biais courant dans la carte décentralisée. Le but est de lier toutes les données affectées par la divergence. Si l'étape de connexion est négligeable au niveau du temps de calcul en début de trajectoire, cela n'est plus du tout le cas vers la fin. La carte décentralisée étant alimentée par les différents membres de la flotte, elle ne fait que grossir. L'intégration du biais par l'EKF nécessite une étape de mise à jour. Bien qu'uniquement l'estimation courante du biais et la donnée concernée soient considérées dans la fonction d'observation, les équations du filtre de Kalman impliquent des multiplications par la matrice de covariance complète et la jacobienne. Ces dernières ne faisant qu'augmenter en taille, les multiplications vont prendre de plus en en plus de temps. Une solution est de ne pas garder les points dépassés depuis une certaine distance. Dans le cas d'un convoi où les membres ne sont pas trop espacés, cette solution est envisageable. Néanmoins, il sera alors impossible d'effectuer des fermetures de boucle une fois les amers oubliés. Une autre solution serait de connecter le biais différemment aux amers afin d'éviter la mise à jour du filtre de Kalman. Pour cela, il serait nécessaire de connaître la valeur du biais en n'importe quelle abscisse curviligne, et ce même après des fusions, ce qui est loin d'être trivial. Cependant, un tel gain de temps sur les traitements décentralisés permettrait d'augmenter le nombre d'amers à suivre et ainsi faciliterait les associations de données.

L'estimation des distances relatives entre les véhicules d'une flotte, et ce avec des positions initiales inconnues, est un point majeur de notre application. Les résultats obtenus montrent qu'il est possible, même sans a priori fort, de retrouver la configuration d'une équipe de robots. Cependant, le guidage automatique d'une flotte n'est actuellement pas possible. En effet, les précisions atteintes sont insuffisantes pour de la conduite autonome. Cela était prévisible puisque l'application a été construite afin d'être la plus générale possible. Il a donc été décidé de ne pas spécialiser l'association de données. Cela implique qu'il n'y pas de retour des points 3D vers l'image dans notre cas. C'est néanmoins cette étape qui permettrait d'obtenir une précision supérieure pour pouvoir faire du rejeu automatique. La mise en place de la projection des points décentralisés dans l'image pose d'autres problèmes que la perte de généricité. En effet, les amers seraient alors mis à jour. En déportant cette action uniquement dans la carte décentralisée, cela pourrait être envisageable. Les améliorations apportées à ce niveau par un véhicule ne pourraient alors pas être communiquées aux autres. Ce genre de spécialisation pour des applications particulières est tout à fait possible. C'est d'ailleurs dans ce but qu'une approche générale a été favorisée tout au long de cette thèse.

Dans tous les cas, l'utilisation du modèle de biais permet un couplage facile avec un GPS ou tout autre capteur de positionnement absolu. De même, la prise en compte de données en provenance de l'infrastructure est aisée. On peut par exemple penser à des capteurs RFID sur la route dont on connaît la position et qui permettraient de préciser les poses des véhicules. Ces deux extensions sont intéressantes car l'architecture est pensée pour celles-ci et offre donc une facilité d'intégration. De plus, l'estimation de la dérive via ces moyens permettraient un important gain en précision. De manière similaire, la détection des membres de la flotte (via un télémètre laser par exemple) apporterait une information supplémentaire sur la configuration du groupe. Étant donné que la position de chaque robot est consistante, la prise en compte de cette information serait également simple. Les exemples donnés dans ce paragraphe ouvrent encore un peu plus les différentes applications dans lesquelles notre algorithme de SLAM décentralisé pourrait être utilisé.

Bibliographie

[Aires *et al.* 2008] Kelson R. T. Aires, Andre M. Santana et Adelardo A. D. Medeiros. *Optical Flow Using Color Information : Preliminary Results.* In Proceedings of the ACM Symposium on Applied Computing, pages 1607–1611, 2008. (Cité en page 21.)

[Aragues *et al.* 2010a] Rosario Aragues, Jorge Cortes et Carlos Sagues. *Dynamic Consensus for Merging Visual Maps under Limited Communications.* In IEEE International Conference on Robotics and Automation, pages 3032–3037, 2010. (Cité en page 132.)

[Aragues *et al.* 2010b] Rosario Aragues, Eduardo Montijano et Carlos Sagues. *Consistent Data Association in Multi-Robot Systems with Limited Communications.* In Robotics : Science and Systems, pages 51–58, 2010. (Cité en page 132.)

[Avanzini *et al.* 2012] Pierre Avanzini, Benoit Thuilot et Philippe Martinet. *Manual Convoying of Automated Urban Vehicle Relying on Monocular Vision.* In IEEE Intelligent Vehicles Symposium, pages 19–24, 2012. (Cité en page 129.)

[Bahr *et al.* 2009] Alexander Bahr, Matthew R. Walter et John J. Leonard. *Consistent Cooperative Localization.* In IEEE International Conference on Robotics and Automation, pages 3415–3422, 2009. (Cité en page 130.)

[Bailey & Durrant-Whyte 2006] Tim Bailey et Hugh Durrant-Whyte. *Simultaneous Localization and Mapping (SLAM) : Part II.* IEEE Robotics and Automation Magazine, vol. 13, no. 3, pages 108–117, 2006. (Cité en page 21.)

[Bailey *et al.* 2006a] Tim Bailey, Juan Nieto, Jose Guivant, Michael Stevens et Eduardo Nebot. *Consistency of the EKF-SLAM Algorithm.* In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 3562–3568, 2006. (Cité en page 87.)

[Bailey *et al.* 2006b] Tim Bailey, Juan Nieto et Eduardo Nebot. *Consistency of the FastSLAM Algorithm.* In IEEE International Conference on Robotics and Automation, pages 424–429, 2006. (Cité en page 30.)

[Bailey *et al.* 2011] Tim Bailey, M. Bryson, Hua Mu, J. Vial, L. McCalman et H. Durrant-Whyte. *Decentralised Cooperative Localisation for Heterogeneous Teams of Mobile Robots.* In IEEE International Conference on Robotics and Automation, pages 2859–2865, 2011. (Cité en page 129.)

[Bailey 2002] Tim Bailey. *Mobile Robot Localisation and Mapping in Extensive Outdoor Environments.* PhD thesis, Australian Centre for Field Robotics - The University of Sydney, 2002. (Cité en page 26.)

[Bailey 2003] Tim Bailey. *Constrained Initialisation for Bearing-Only SLAM.* In IEEE International Conference on Robotics and Automation, volume 2, pages 1966–1971, 2003. (Cité en page 52.)

[Baker & Matthews 2004] Simon Baker et Iain Matthews. *Lucas-Kanade 20 Years on : A Unifying Framework.* International Journal of Computer Vision, vol. 56, no. 3, pages 221–255, 2004. (Cité en page 84.)

[Bar-Shalom *et al.* 2001] Yaakov Bar-Shalom, X. Rong Li et Thiagalingam Kirubarajan. Estimation with Applications to Tracking and Navigation. Wiley-Interscience, 2001. (Cité en pages 25 et 86.)

[Barrena *et al.* 2013] Nagore Barrena, Jairo R. Sánchez et Alejandro García-Alonso. *A Distributed and Collaborative vSLAM Framework for Real-Time Localisation in Huge Environments for Mobile Devices.* In Eurographics, 2013. (Cité en page 126.)

[Bay *et al.* 2008] Herbert Bay, Andreas Ess, Tinne Tuytelaars et Luc Van Gool. *Speeded-Up Robust Features (SURF).* Computer Vision and Image Understanding, vol. 110, no. 3, pages 346–359, 2008. (Cité en page 41.)

[Bekris *et al.* 2006] Kostas E. Bekris, Max Glick et Lydia E. Kavraki. *Evaluation of Algorithms for Bearing-Only SLAM.* In IEEE International Conference on Robotics and Automation, pages 1937–1943, 2006. (Cité en page 54.)

[Blanco *et al.* 2009] J. L. Blanco, J. González et J.-A. Fernández-Madrigal. *Subjective Local Maps for Hybrid Metric-Topological SLAM.* Robotics and Autonomous Systems, vol. 57, no. 1, pages 64–74, 2009. (Cité en page 28.)

[Bosse *et al.* 2003] Michael Bosse, Paul Newman, John Leonard, Martin Soika, Wendelin Feiten et Seth Teller. *An Atlas Framework for Scalable Mapping.* In IEEE International Conference on Robotics and Automation, volume 2, pages 1899–1906, 2003. (Cité en page 27.)

[Bresson *et al.* 2011] Guillaume Bresson, Thomas Féraud, Romuald Aufrère, Paul Checchin et Roland Chapuis. *A New Strategy for Feature Initialization in Visual SLAM.* In IEEE/RSJ International Conference on Intelligent Robots and Systems Workshop on Perception and Navigation for Autonomous Vehicles in Human Environment, pages 115–120, 2011. (Cité en page 56.)

[Bresson *et al.* 2012a] Guillaume Bresson, Romuald Aufrère et Roland Chapuis. *Real-time Decentralized Monocular SLAM.* In International Conference on Control, Automation, Robotics and Vision, 2012. (Cité en page 137.)

[Bresson *et al.* 2012b] Guillaume Bresson, Thomas Féraud, Romuald Aufrère, Paul Checchin et Roland Chapuis. *Parsimonious Real Time Monocular SLAM.* In IEEE International Conference on Intelligent Vehicles, pages 511–516, 2012. (Cité en pages 56 et 65.)

[Bresson *et al.* 2013a] Guillaume Bresson, Romuald Aufrère et Roland Chapuis. *Consistent Multi-robot Decentralized SLAM with Unknown Initial Positions.* In 16th International Conference on Information FUSION, 2013. (Cité en page 137.)

[Bresson *et al.* 2013b] Guillaume Bresson, Romuald Aufrère et Roland Chapuis. *Loop Closing in a Drift-Aware Monocular SLAM.* In IFAC Intelligent Autonomous Vehicles Symposium, 2013. (Cité en page 86.)

[Bresson *et al.* 2013c] Guillaume Bresson, Romuald Aufrère et Roland Chapuis. *Making Visual SLAM Consistent with Geo-Referenced Landmarks.* In IEEE International Conference on Intelligent Vehicles, 2013. (Cité en page 86.)

[Burgard *et al.* 2002] Wolfram Burgard, Mark Moors, Dieter Fox, Reid Simmons et Sebastian Thrun. *Collaborative Multi-Robot Exploration.* In IEEE International Conference on Robotics and Automation, volume 1, pages 476–481, 2002. (Cité en page 126.)

[Burgard *et al.* 2005] Wolfram Burgard, Mark Moors, Cyrill Stachniss et Frank Schneider. *Coordinated Multi-Robot Exploration.* IEEE Transactions on Robotics, vol. 21, no. 3, pages 376–386, 2005. (Cité en pages 126 et 134.)

[Castellanos *et al.* 2004] José A. Castellanos, José Neira et Juan D. Tardós. *Limits to the Consistency of EKF-Based SLAM.* In 5th IFAC Symposium on Intelligent Autonomous Vehicles, 2004. (Cité en page 88.)

[Castellanos *et al.* 2007] José A. Castellanos, Ruben Martinez-Cantin, Juan D. Tardós et José Neira. *Robocentric Map Joining : Improving the Consistency of EKF-SLAM.* Robotics and Autonomous Systems, vol. 55, no. 1, pages 21–29, 2007. (Cité en page 88.)

[Chang *et al.* 2007] H. Jacky Chang, C. S. George Lee, Y. Charlie Hu et Yung-Hsiang Lu. *Multi-Robot SLAM with Topological/Metric Maps.* In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 1467–1472, 2007. (Cité en page 133.)

[Charmette *et al.* 2010] Baptiste Charmette, Éric Royer et Frédéric Chausse. *Efficient Planar Features Matching for Robot Localization Using GPU.* In IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, pages 16–23, 2010. (Cité en page 84.)

[Checklov *et al.* 2006] D. Checklov, M. Pupilli, W. Mayol-Cuevas et A. Calway. *Real-Time and Robust Monocular SLAM using Predictive Multi-Resolution Descriptors.* Advances in Visual Computing, vol. 4292, pages 276–285, 2006. (Cité en page 29.)

[Chli & Davison 2009] Margarita Chli et Andrew J. Davison. *Automatically and Efficiently Inferring the Hierarchical Structure of Visual Maps.* In IEEE International Conference on Robotics and Automation, pages 387–394, 2009. (Cité en page 28.)

[Civera *et al.* 2007] Javier Civera, Andrew J. Davison et J. M. M. Montiel. *Inverse Depth to Depth Conversion for Monocular SLAM.* In IEEE International Conference on Robotics and Automation, pages 2778–2783, 2007. (Cité en page 54.)

[Civera *et al.* 2008] Javier Civera, Andrew J. Davison et J. M. Martinez Montiel. *Inverse Depth Parametrization for Monocular SLAM.* IEEE Transactions on Robotics, vol. 24, no. 5, pages 932–945, 2008. (Cité en page 53.)

[Civera *et al.* 2009] Javier Civera, Oscar G. Grasa, Andrew J. Davison et J. M. M. Montiel. *1-Point RANSAC for EKF-Based Structure from Motion.* In

IEEE/RSJ International Conference on Intelligent Robot and Systems, pages 3498–3504, 2009. (Cité en page 89.)

[Civera *et al.* 2010] Javier Civera, Oscar G. Grasa, Andrew J. Davison et J. M. M. Montiel. *1-Point RANSAC for EKF Filtering. Application to Real-Time Structure from Motion and Visual Odometry.* Journal of Fields Robotics, vol. 27, no. 5, pages 609–631, 2010. (Cité en pages 70, 74, 78, 79, 80, 81, 82, 83 et 84.)

[Clemente *et al.* 2007] Laura A. Clemente, Andrew J. Davison, Ian D. Reid, José Neira et Juan D. Tardós. *Mapping Large Loops with a Single Hand-Held Camera.* In Robotics : Science and Systems, 2007. (Cité en pages 25 et 42.)

[Cole & Newman 2006] David M. Cole et Paul M. Newman. *Using Laser Range Data for 3D SLAM in Outdoor Environments.* In IEEE International Conference on Robotics and Automation, pages 1556–1563, 2006. (Cité en page 90.)

[Conte & Doherty 2009] Gianpaolo Conte et Patrick Doherty. *Vision-Based Unmanned Aerial Vehicle Navigation Using Geo-Referenced Information.* EURASIP Journal On Advances In Signal Processing, pages 10–32, 2009. (Cité en page 89.)

[Cunningham & Dellaert 2012] Alex Cunningham et Frank Dellaert. *Large Scale Experimental Design for Decentralized SLAM.* In Unmanned Systems Technology XIV, 2012. (Cité en page 135.)

[Cunningham *et al.* 2010] Alexander Cunningham, Manohar Paluri et Frank Dellaert. *DDF-SAM : Fully Distributed SLAM using Constrained Factor Graphs.* In IEEE/RSJ International Conference on Intelligent Robots and Systems, 2010. (Cité en pages 132, 133 et 134.)

[Davison *et al.* 2007] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton et Olivier Stasse. *MonoSLAM : Real-time Single Camera SLAM.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 29, no. 6, pages 1052–1067, 2007. (Cité en pages 42 et 55.)

[Davison 2003] Andrew J. Davison. *Real-Time Simultaneous Localisation and Mapping with a Single Camera.* In IEEE International Conference on Computer Vision, pages 1403–1410, 2003. (Cité en pages 25, 42, 52, 53 et 55.)

[Deaves *et al.* 2000] R. H. Deaves, D. Nicholson, D. W. Gough, L. A. Binns, P. Vangasse et P. Greenway. *Multiple Robot System for Decentralized SLAM Investigations.* In Sensor Fusion and Decentralized Control in Robotic Systems III, volume 4196, pages 360–369, 2000. (Cité en page 134.)

[Dellaert *et al.* 1999] Frank Dellaert, Dieter Fox, Wolfram Burgard et Sebastian Thrun. *Monte Carlo Localization for Mobile Robots.* In IEEE Internationational Conference on Robotics and Automation, volume 2, pages 1322–1328, 1999. (Cité en page 30.)

[Diosdado & Ruiz 2007] Jose Vazquez Diosdado et Ioseba Tena Ruiz. *Decentralised Simultaneous Localisation and Mapping for AUVs.* In 2nd SEAS DTC Technical Conference, page A14, 2007. (Cité en pages 126 et 134.)

[Dissanayake *et al.* 2000] M. W. M. G. Dissanayake, P. Newman, H. F. Durrant-Whyte, S. Clark et M. Csorba. *An Experimental and Theoretical Investigation into Simultaneous Localization and Map Building.* In The Sixth International Symposium on Experimental Robotics VI, pages 265–274, 2000. (Cité en page 24.)

[Dissanayake *et al.* 2001] M. W. M. G. Dissanayake, P. Newman, H. F. Durrant-Whyte, S. Clark et M. Csorba. *A Solution to the Simultaneous Localization and Map Building (SLAM) Problem.* IEEE Transactions on Robotics and Automation, vol. 17, no. 3, pages 229–241, 2001. (Cité en pages 24 et 86.)

[Durrant-Whyte & Bailey 2006] Hugh Durrant-Whyte et Tim Bailey. *Simultaneous Localization and Mapping : Part I.* IEEE Robotics and Automation Magazine, vol. 13, no. 2, pages 99–110, 2006. (Cité en page 21.)

[Durrant-Whyte 2000] Hugh Durrant-Whyte. *A Beginner's Guide to Decentralised Data Fusion.* Rapport technique, Australian Centre for Field Robotics, The University of Sydney, Australia, 2000. (Cité en page 128.)

[Eade & Drummond 2006] E. Eade et T. Drummond. *Scalable Monocular SLAM.* In IEEE Computer Society Conference on Computer Vision and Pattern Recognition, volume 1, pages 469–476, 2006. (Cité en page 30.)

[Estrada *et al.* 2005] Carlos Estrada, José Neira et Juan D. Tardós. *Hierarchical SLAM : real-time accurate mapping of large environments.* IEEE Transactions on Robotics, vol. 21, no. 4, pages 588–596, 2005. (Cité en pages 27, 88 et 132.)

[Eustice *et al.* 2005] Ryan Eustice, Matthew Walter et John J. Leonard. *Sparse Extended Information Filters : Insights Into Sparsification.* In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 3281–3288, 2005. (Cité en page 29.)

[Fenwick *et al.* 2002] John W. Fenwick, Paul M. Newman et John J. Leonard. *Cooperative Concurrent Mapping and Localization.* In IEEE International Conference on Robotics and Automation, volume 2, pages 1810–1817, 2002. (Cité en page 134.)

[Féraud *et al.* 2010] Thomas Féraud, Paul Checchin, Romuald Aufrère et Roland Chapuis. *Communicating Vehicles in Convoy and Monocular Vision-based Localization.* In 7th Symposium on Intelligent Autonomous Vehicles, volume 7, 2010. (Cité en page 129.)

[Féraud *et al.* 2011a] Thomas Féraud, Roland Chapuis, Romuald Aufrère et Paul Checchin. *Improving Results of Non-Linear Observation Function Using a Kalman Filter Correction.* In International Conference on Information Fusion, 2011. (Cité en page 65.)

[Féraud *et al.* 2011b] Thomas Féraud, Roland Chapuis, Romuald Aufrère et Paul Checchin. *Kalman Filter Correction with Rational Non-Linear Functions : Application to Visual-SLAM.* In European Conference on Mobile Robots, pages 232–238, 2011. (Cité en page 65.)

[Féraud 2011] Thomas Féraud. *Rejeu de chemin et localisation monoculaire : Application du Visual SLAM sur carte peu dense en environnement extérieur contraint.* PhD thesis, Université Blaise Pascal - Clermont-Ferrand II, 2011. (Cité en pages ix, 26, 27 et 56.)

[Fischler & Bolles 1981] Martin A. Fischler et Robert C. Bolles. *Random Sample Consensus : A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography.* Communications of the ACM, vol. 24, no. 6, pages 381–395, 1981. (Cité en page 23.)

[Frese 2006] Udo Frese. *A Discussion of Simultaneous Localization and Mapping.* Autonomous Robots, vol. 20, no. 1, pages 25–42, 2006. (Cité en page 87.)

[Gifford *et al.* 2008] Christopher M. Gifford, Russell Webb, James Bley, Daniel Leung, Mark Calnon, Joe Makarewicz, Bryan Banz et Arvin Agah. *Low-Cost Multi-Robot Exploration and Mapping.* In IEEE International Conference on Technologies for Practical Robot Applications, pages 74–79, 2008. (Cité en page 126.)

[Gil *et al.* 2010] Arturo Gil, Óscar Reinoso, Mónica Ballesta et Miguel Juliá. *Multi-robot visual SLAM using a Rao-Blackwellized particle filter.* Robotics and Autonomous Systems, vol. 58, no. 1, pages 68–80, 2010. (Cité en page 127.)

[Guivant & Nebot 2001] Jose Guivant et Eduardo Nebot. *Optimization of the Simultaneous Localization and Map Building Algorithm for Real-Time Implementation.* IEEE Transactions on Robotics and Automation, vol. 17, no. 3, pages 242–257, 2001. (Cité en page 25.)

[Guivant *et al.* 2000] Jose Guivant, Eduardo Nebot et Stephan Baiker. *Autonomous Navigation and Map Building Using Laser Range Sensors in Outdoor Applications.* Journal of Robotic Systems, vol. 17, no. 10, pages 565–583, 2000. (Cité en page 25.)

[Gutmann *et al.* 2010] Jens-Steffen Gutmann, Ethan Eade nad Philip Fong et Mario E. Munich. *A Constant-Time Algorithm for Vector Field SLAM Using an Exactly Sparse Extended Information Filter.* In Robotics : Science and Systems, 2010. (Cité en page 29.)

[Harris & Stephens 1988] Chris Harris et Mike Stephens. *A Combined Corner and Edge Detector.* In 4th Alvey Vision Conference, pages 147–151, 1988. (Cité en page 41.)

[Hartley & Zisserman 2000] Richard Hartley et Andrew Zisserman. Multiple View Geometry in Computer Vision. Cambridge Univ Press, 2000. (Cité en page 22.)

[Hua *et al.* 2010] Mu Hua, Tim Bailey, Paul Thompson et Hugh Durrant-Whyte. *Decentralised Solutions to the Cooperative Multi-Platform Navigation Problem.* IEEE Transactions on Aerospace and Electronic Systems, 2010. (Cité en pages 129 et 134.)

[Huang & Dissanayake 2007] Shoudong Huang et Gamini Dissanayake. *Convergence and Consistency Analysis for Extended Kalman Filter Based SLAM.* IEEE Transactions on Robotics, vol. 23, no. 5, pages 1036–1049, 2007. (Cité en page 87.)

[Julier & Uhlmann 1997] Simon J. Julier et Jeffrey K. Uhlmann. *New Extension of the Kalman Filter to Nonlinear Systems.* In AeroSense'97, pages 182–193, 1997. (Cité en page 28.)

[Julier & Uhlmann 2001a] S. Julier et J. Uhlmann. General Decentralized Data Fusion with Covariance Intersection. CRC Press, 2001. Chapter 12. (Cité en page 133.)

[Julier & Uhlmann 2001b] Simon J. Julier et Jeffrey K. Uhlmann. *A Counter Example to the Theory of Simultaneous Localization and Map Building.* In IEEE International Conference on Robotics and Automation, pages 4238–4243, 2001. (Cité en pages 25 et 87.)

[Julier & Uhlmann 2004] Simon J. Julier et Jeffrey K. Uhlmann. *Unscented Filtering and Nonlinear Estimation.* Proceedings of the IEEE, vol. 92, no. 3, pages 401–422, 2004. (Cité en page 29.)

[Kalman & Bucy 1961] Rudolph E. Kalman et Richard S. Bucy. *New Results in Linear Filtering and Prediction Theory.* Journal of Basic Engineering, vol. 83, no. 3, pages 95–108, 1961. (Cité en page 25.)

[Kalman 1960] Rudolph E. Kalman. *A New Approach to Linear Filtering and Prediction Problems.* Journal of Basic Engineering, vol. 82, no. 1, pages 35–45, 1960. (Cité en page 23.)

[Karam *et al.* 2006] Nadir Karam, Frédéric Chausse, Romuald Aufrère et Roland Chapuis. *Localization of a Group of Communicating Vehicles by State Exchange.* In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 519–524, 2006. (Cité en pages 130, 133 et 138.)

[Kleiner & Sun 2007] Alexander Kleiner et Dali Sun. *Decentralized SLAM for Pedestrians without direct Communication.* In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 1461–1466, 2007. (Cité en page 126.)

[Kwok & Dissanayake 2004] N. M. Kwok et G. Dissanayake. *An Efficient Multiple Hypothesis Filter for Bearing-Only SLAM.* In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 736–741, 2004. (Cité en page 53.)

[Lee & Lee 2009] Hee Seok Lee et Kyoung Mu Lee. *Multi-Robot SLAM Using Ceiling Vision.* In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 912–917, 2009. (Cité en pages 126 et 134.)

[Leonard & Durrant-Whyte 1991] John J. Leonard et Hugh Durrant-Whyte. *Simultaneous Map Building and Localization for an Autonomous Mobile Robot.* In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 1442–1447, 1991. (Cité en page 20.)

[Leonard & Feder 2000] John J. Leonard et Hans Jacob S. Feder. *A Computationally Efficient Method for Large-Scale Concurrent Mapping and Localization.* In International Symposium on Robotics Research, volume 9, pages 169–178, 2000. (Cité en page 25.)

[Leonard & Newman 2003] John J. Leonard et Paul Newman. *Consistent, Convergent and Constant-Time SLAM.* In International Joint Conferences on Artificial Intelligence, pages 1143–1150, 2003. (Cité en page 26.)

[Lewis 1995] J. P. Lewis. *Fast Normalized Cross-Correlation.* In Vision Interface, pages 120–123, 1995. (Cité en page 42.)

[Li & Nashashibi 2011] Hao Li et Fawzi Nashashibi. *Multi-vehicle Cooperative Perception and Augmented Reality for Driver Assistance : A Possibility to 'See' Through Front Vehicle.* In IEEE International Conference on Intelligent Transportation Systems, 2011. (Cité en page 126.)

[Li & Nashashibi 2012a] Hao Li et Fawzi Nashashibi. *A new method for occupancy grid maps merging : Application to multi-vehicle cooperative local mapping and moving object detection in outdoor environment.* In International Conference on Control, Automation, Robotics and Vision, pages 632–637, 2012. (Cité en page 135.)

[Li & Nashashibi 2012b] Hao Li et Fawzi Nashashibi. *Multi-vehicle cooperative localization using indirect vehicle-to-vehicle relative pose estimation.* In IEEE International Conference on Vehicular Electronics and Safety, pages 267–272, 2012. (Cité en page 135.)

[Li & Nashashibi 2013] Hao Li et Fawzi Nashashibi. *Cooperative multi-vehicle localization using split covariance intersection filter.* IEEE Intelligent Transportation Systems Magazine, vol. 5, no. 2, pages 33–44, 2013. (Cité en page 133.)

[Lindeberg 1998] Tony Lindeberg. *Feature Detection with Automatic Scale Selection.* International Journal of Computer Vision, vol. 30, no. 2, pages 79–116, 1998. (Cité en page 41.)

[Liu & Thrun 2003] Yufeng Liu et Sebastian Thrun. *Results for Outdoor-SLAM Using Sparse Extended Information Filters.* In IEEE International Conference on Robotics and Automation, volume 1, pages 1227–1233, 2003. (Cité en page 29.)

[Lovregrove *et al.* 2011] Steven Lovregrove, Andrew J. Davison et Javier Ibanez-Guzmán. *Accurate Visual Odometry from a Rear Parking Camera.* In IEEE Intelligent Vehicles Symposium, pages 788–793, 2011. (Cité en page 21.)

[Lowe 1999] David G. Lowe. *Object Recognition from Local Scale-Invariant Features.* In International Conference on Computer Vision, pages 1150–1157, 1999. (Cité en page 41.)

[Lowe 2004] David G. Lowe. *Distinctive Image Features from Scale-Invariant Keypoints.* International Journal of Computer Vision, vol. 60, no. 2, pages 91–110, 2004. (Cité en page 41.)

[Mahon *et al.* 2008] Ian Mahon, Stefan B. Williams, Oscar Pizarro et Matthew Johnson-Roberson. *Efficient View-Based SLAM Using Visual Loop Closures.* IEEE Transactions on Robotics, vol. 24, no. 5, pages 1002–1014, 2008. (Cité en page 29.)

[Malartre 2011] Florent Malartre. *Perception intelligente pour la nvaigation rapide de robots mobiles en environnement naturel.* PhD thesis, Université Blaise Pascal - Clermont-Ferrand II, 2011. (Cité en page 25.)

[Martin & Emami 2010] Adrian Martin et M. Reza Emami. *Just-in-time Cooperative Simultaneous Localization and Mapping.* In International Conference on Control, Automation, Robotics and Vision, pages 479–484, 2010. (Cité en page 134.)

[Martinelli *et al.* 2005] Agostino Martinelli, Nicola Tomatis et Roland Siegwart. *Some Results on SLAM and the Closing the Loop Problem.* In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 2917–2922, 2005. (Cité en page 90.)

[Maybeck 1982] Peter S. Maybeck. Stochastic Models, Estimation and Control. Elsevier, 1982. (Cité en page 29.)

[McLaughlin *et al.* 2003] Samuel P. McLaughlin, Robin J. Evans et Bikram Krishnamurthy. *Data Incest Removal in Survivable Estimation Fusion Architecture.* In International Conference on Information Fusion, volume 1, pages 229–236, 2003. (Cité en page 129.)

[M.D.P. *et al.* 2010] Moratuwage M.D.P., Wijerupage Sardha Wijesoma, Bharath Kalyan, Nicholas M. Patrikalakis et Peyman Moghadam. *Collaborative Multi-Vehicle Localization and Mapping in High Clutter Environments.* In International Conference on Control, Automation, Robotics and Vision, pages 1422–1427, 2010. (Cité en page 134.)

[Mikolajczyk & Schmid 2004] Krystian Mikolajczyk et Cordelia Schmid. *Scale & Affine Invariant Interest Point Detectors.* International Journal of Computer Vision, vol. 60, no. 1, pages 63–86, 2004. (Cité en page 41.)

[Mikolajczyk & Schmid 2005] Krystian Mikolajczyk et Cordelia Schmid. *A Performance Evaluation of Local Descriptors.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, no. 10, pages 1615–1630, 2005. (Cité en page 41.)

[Mikolajczyk *et al.* 2005] Krystian Mikolajczyk, Tinne Tuytelaars, Cordelia Schmid, Andrew Zisserman, Jiri Matas, Frederik Schaffalitzky, Timor Kardir et Luc Van Gool. *A Comparison of Affine Region Detectors.* International Journal of Computer Vision, vol. 65, no. 1-2, pages 43–72, 2005. (Cité en page 41.)

[Mohan & Madhava 2010] Mahesh Mohan et Krishna K. Madhava. *Mapping Large Scale Environments by Combining Particle Filter and Information Filter.* In International Conference on Control, Automation, Robotics and Vision, pages 1000–1005, 2010. (Cité en page 30.)

[Molton *et al.* 2004] Nicholas D. Molton, Andrew J. Davison et Ian D. Reid. *Locally Planar Patch Features for Real-Time Structure from Motion.* In BMVC, pages 1–10, 2004. (Cité en page 84.)

[Montemerlo *et al.* 2002] Michael Montemerlo, Sebastian Thrun, Daphne Koller et Ben Wegbreit. *FastSLAM : A Factored Solution to the Simultaneous Localization And Mapping Problem.* In AAAI/IAAI, pages 593–598, 2002. (Cité en page 30.)

[Montiel *et al.* 2006] J.M.M. Montiel, Javier Civera et Andrew J. Davison. *Unified Inverse Depth Parametrization for Monocular SLAM.* In Robotics : Science and Systems, 2006. (Cité en pages ix, 53, 54 et 74.)

[Mouragnon *et al.* 2006] Etienne Mouragnon, Maxime Lhuillier, Michel Dhome, Fabien Dekeyser et Patrick Sayd. *Real Time Localization and 3D Reconstruction.* In IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 363–370, 2006. (Cité en pages 23 et 55.)

[Murphy 1999] Kevin P. Murphy. *Bayesian Map Learning in Dynamic Environments.* In Neural Information Processing Systems, pages 1015–1021, 1999. (Cité en page 30.)

[Neira & Tardós 2002] José Neira et Juan D. Tardós. *Data Association in Stochastic Mapping Using the Joint Compatibility Test.* IEEE Transactions on Robotics and Automation, vol. 17, no. 6, pages 890–897, 2002. (Cité en pages 84, 135 et 151.)

[Neira *et al.* 2003] José Neira, Juan D. Tardós et José A. Castellanos. *Linear time vehicle relocation in SLAM.* In IEEE International Conference on Robotics and Automation, volume 1, pages 427–433, 2003. (Cité en pages 136 et 151.)

[Nettleton *et al.* 2003] Eric Nettleton, Hugh Durrant-Whyte et Salah Sukkarieh. *A Robust Architecture for Decentralised Data Fusion.* In International Conference on Advanced Robotics, 2003. (Cité en page 128.)

[Nettleton *et al.* 2006] Eric Nettleton, Sebastian Thrun, Hugh Durrant-Whyte et Salah Sukkarieh. *Decentralised SLAM with Low-Bandwidth Communication for Teams of Vehicles.* In Field and Service Robotics, pages 179–188, 2006. (Cité en pages 128, 132 et 134.)

[Newman & Leonard 2003] Paul Newman et John J. Leonard. *Pure Range-Only Sub-Sea SLAM.* In IEEE International Conference on Robotics and Automation, volume 2, pages 1921–1926, 2003. (Cité en page 25.)

[Newman *et al.* 2002] P. Newman, J. Leonard, Juan D. Tardós et José Neira. *Explore and Return : Experimental Validation of Real-Time Concurrent Mapping and Localization.* In IEEE International Conference on Robotics and Automation, volume 2, pages 1802–1809, 2002. (Cité en page 25.)

[Newman *et al.* 2006] Paul Newman, David Cole et Kin Ho. *Outdoor SLAM Using Visual Appearance and Laser Ranging.* In IEEE International Conference on Robotics and Automation, pages 1180–1187, 2006. (Cité en page 25.)

[Nistér *et al.* 2006] David Nistér, Oleg Naroditsky et James Bergen. *Visual Odometry for Ground Vehicle Applications.* Journal of Fields Robotics, vol. 23, no. 1, pages 3–20, 2006. (Cité en page 23.)

[Ong *et al.* 2003] Lee Ling (Sharon) Ong, Matthew Ridley, Jong-Hyuk Kim, Eric Nettleton et Salah Sukkarieh. *Six DoF Decentralised SLAM.* In Australasian Conference on Robotics and Automation, pages 10–16, 2003. (Cité en page 126.)

[Parsley & Julier 2008] Martin P. Parsley et Simon J. Julier. *Avoiding Negative Depth in Inverse Depth Bearing-Only SLAM.* In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 2066–2071, 2008. (Cité en pages 55 et 65.)

[Paz *et al.* 2008] Lina M. Paz, Juan D. Tardós et José Neira. *Divide and Conquer : EKF SLAM in O(n).* IEEE Transactions on Robotics, vol. 24, no. 5, pages 1107–1120, 2008. (Cité en pages 28 et 41.)

[Petrovskaya & Thrun 2009] Anna Petrovskaya et Sebastian Thrun. *Model Based Vehicle Detection and Tracking for Autonomous Urban Driving.* Autonomous Robots, vol. 26, no. 2, pages 123–139, 2009. (Cité en page 55.)

[Pfingsthorn *et al.* 2007] Max Pfingsthorn, Bayu Slamet et Arnoud Visser. *A Scalable Hybrid Multi-Robot SLAM Method for Highly Detailed Maps.* In RoboCup 2007 : Robot Soccer World Cup XI, pages 457–464, 2007. (Cité en page 133.)

[Piniés & Tardós 2008] Pedro Piniés et Juan D. Tardós. *Large Scale SLAM Building Conditionally Independent Local Maps : Application to Monocular Vision.* IEEE Transactions on Robotics, vol. 24, no. 5, pages 1094–1106, 2008. (Cité en page 28.)

[Press *et al.* 1992] WH Press, SA Keukolsky, WT Vettering et BP Flannery. *Levenberg-Marquardt Method.* Numerical Recipes in C : The Art of Scientific Computation, pages 542–547, 1992. (Cité en page 22.)

[Pucihar & Coulton 2011] Klen Copic Pucihar et Paul Coulton. *Towards Collaboratively Mapped Multi-View Mobile Augmented Reality.* In Workshop on Mobile Augmented Reality : Design Issues and Opportunities, Mobile Human Computer-Interaction, 2011. (Cité en page 126.)

[Rekleitis *et al.* 2000] Ioannis M. Rekleitis, Gregory Dudek et Evangelos E. Milios. *Multi-Robot Collaboration for Robust Exploration.* In IEEE International Conference on Robotics and Automation, volume 4, pages 3164–3169, 2000. (Cité en page 126.)

[Resende *et al.* 2012] Paulo Resende, Fawzi Nashashibi, François Charlot, Carlos Holguin, Laurent Bouraoui et Michel Parent. *A Cooperative Personal Automated Transport System - A CityMobil Demonstration in Rocquencourt.*

In International Conference on Control, Automation, Robotics and Vision, 2012. (Cité en page 134.)

[Roumeliotis & Rekleitis 2003] S. I. Roumeliotis et I. M. Rekleitis. *Analysis of Multirobot Localization Uncertainty Propagation.* In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 1763–1770, 2003. (Cité en page 130.)

[Royer *et al.* 2005] Éric Royer, Maxime Lhuillier, Michel Dhome et Thierry Chateau. *Localization in Urban Environments : Monocular Vision Compared to a Differential GPS Sensor.* In IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 114–121, 2005. (Cité en pages ix, 22, 23 et 26.)

[Schmid *et al.* 2000] Cordelia Schmid, Roger Mohr et Christian Bauckhage. *Evaluation of Interest Point Detectors.* International Journal of Computer Vision, vol. 37, no. 2, pages 151–172, 2000. (Cité en page 41.)

[Seo *et al.* 2005] Dae-Sung Seo, Daeheui Won, Gwang-Woong Yang, Moo-Sung Choi, Sang-Ju Kwon et Joon Woo Park. *A Probabilistic Approach for Mobile Robot Localization under RFID Tag Infrastructures.* In International Conference on Control, Automation and Systems, pages 1797–1801, 2005. (Cité en page 89.)

[Shi & Tomasi 1994] Jianbo Shi et Carlo Tomasi. *Good Features to Track.* In Computer Vision and Pattern Recognition, pages 593–600, 1994. (Cité en page 41.)

[Smith & Cheeseman 1986] Randall Smith et Peter Cheeseman. *On the Representation and Estimation of Spatial Uncertainty.* The International Journal of Robotics Research, vol. 5, no. 4, pages 56–68, 1986. (Cité en page 21.)

[Smith *et al.* 1990] Randall Smith, Matthew Self et Peter Cheeseman. *Estimating Uncertain Spatial Relationships in Robotics.* In Autonomous Robot Vehicles, pages 167–193. Springer, 1990. (Cité en page 20.)

[Solà *et al.* 2005] Joan Solà, André Monin, Michel Devy et Thomas Lemaire. *Undelayed Initialization in Bearing Only SLAM.* In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 2499–2504, 2005. (Cité en page 53.)

[Stewenius *et al.* 2006] Henrik Stewenius, Christopher Engels et David Nistér. *Recent Developments on Direct Relative Orientation.* ISPRS Journal of Photogrammetry and Remote Sensing, vol. 60, no. 4, pages 284–294, 2006. (Cité en page 84.)

[Strasdat *et al.* 2010a] Hauke Strasdat, J. M. M. Montiel et Andrew J. Davison. *Real-time Monocular SLAM : Why Filter ?* In IEEE International Conference on Robotics and Automation, pages 2657–2664, 2010. (Cité en pages 26 et 30.)

[Strasdat *et al.* 2010b] Hauke Strasdat, J. M. M. Montiel et Andrew J. Davison. *Scale Drift-Aware Large Scale Monocular SLAM.* In Robotics : Science and Systems, 2010. (Cité en page 20.)

[Strasdat *et al.* 2012] Hauke Strasdat, José M. M. Montiel et Andrew J. Davison. *Visual SLAM : Why Filter ?* Image and Vision Computing, vol. 30, no. 2, pages 65–77, 2012. (Cité en page 30.)

[Tessier *et al.* 2006] C. Tessier, C. Cariou, C. Debain, F. Chausse, R. Chapuis et C. Rousset. *A Real-Time, Multi-Sensor Architecture for Fusion of Delayed Observations : Application to Vehicle Localization.* In IEEE Conference on Intelligent Transportation Systems, pages 1316–1321, 2006. (Cité en page 35.)

[Thrun & Liu 2003] Sebastian Thrun et Yufeng Liu. *Multi-Robot SLAM With Sparse Extended Information Filers.* In 11th International Symposium of Robotics Research, pages 254–266, 2003. (Cité en pages 134 et 135.)

[Thrun *et al.* 2000] Sebastian Thrun, Wolfram Burgard et Dieter Fox. *A Real-Time Algorithm for Mobile Robot Mapping With Applications to Multi-Robot and 3D Mapping.* In IEEE International Conference on Robotics and Automation, volume 1, pages 321–328, 2000. (Cité en page 30.)

[Thrun *et al.* 2004] Sebastian Thrun, Yufeng Liu, Daphne Koller, Andrew Y. Ng, Zoubin Ghahramani et Hugh Durrant-Whyte. *Simultaneous Localization and Mapping with Sparse Extended Information Filters.* The International Journal of Robotics Research, vol. 23, no. 7-8, pages 693–716, 2004. (Cité en page 29.)

[Thrun *et al.* 2005] Sebastian Thrun, Wolfram Burgard et Dieter Fox. Probabilistic Robotics, volume 1. MIT Press Cambridge, 2005. (Cité en page 93.)

[Triggs *et al.* 2000] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley et Andrew W. Fitzgibbon. *Bundle Adjustment - A Modern Synthesis.* In Vision Algorithms : Theory and Practice, pages 298–372. Springer, 2000. (Cité en page 21.)

[Vidal-Calleja *et al.* 2011] Teresa A. Vidal-Calleja, Cyrille Berger, Joan Solà et Simon Lacroix. *Large Scale Multiple Robot Visual Mapping with Heterogeneous Landmarks in Semi-structured Terrain.* Robotics and Autonomous Systems, vol. 59, no. 9, pages 654–674, 2011. (Cité en pages 126, 129, 132 et 133.)

[Vivet 2011] Damien Vivet. *Perception de l'environnement par radar hyperfr'equence. Application à la localisation et la cartographie simultanées, à la détection et au suivi d'objets mobiles en milieu extérieur.* PhD thesis, Université Blaise Pascal - Clermont-Ferrand II, 2011. (Cité en page 19.)

[Walter *et al.* 2007] Matthew R. Walter, Ryan M. Eustice et John J. Leonard. *Exactly Sparse Extended Information Filters for Feature-Based SLAM.* The International Journal of Robotics Research, vol. 26, no. 4, pages 335–359, 2007. (Cité en page 29.)

[Wan & Merwe 2000] Eric A. Wan et Rudolph Van Der Merwe. *The Unscented Kalman Filter for Nonlinear Estimation.* In Adaptive Systems for Signal

Processing, Communications and Control Symposium, pages 153–158, 2000. (Cité en page 29.)

[Williams & Reid 2010] Brian P. Williams et Ian D. Reid. *On Combining Visual SLAM and Visual Odometry.* In IEEE International Conference on Robotics and Automation, pages 3494–3500, 2010. (Cité en page 89.)

[Williams *et al.* 2002a] Stefan B. Williams, Gamini Dissanayake et Hugh Durrant-Whyte. *An Efficient Approach to the Simultaneous Localisation and Mapping Problem.* In IEEE International Conference on Robotics and Automation, pages 406–411, 2002. (Cité en page 26.)

[Williams *et al.* 2002b] Stefan B. Williams, Gamini Dissanayake et Hugh Durrant-Whyte. *Towards Multi-Vehicle Simultaneous Localisation and Mapping.* In IEEE International Conference on Robotics and Automation, pages 2743–2748, 2002. (Cité en pages 132, 133 et 134.)

[Williams *et al.* 2009] Brian Williams, Mark Cummins, José Neira, Paul Newman, Ian Reid et Juan Tardós. *A comparison of loop closing techniques in monocular SLAM.* Robotics and Autonomous Systems, vol. 57, no. 12, pages 1188–1197, 2009. (Cité en page 153.)

[Zhou & Roumeliotis 2006] Xun S. Zhou et Stergios I. Roumeliotis. *Multi-robot SLAM with Unknown Initial Correspondance : The Robot Rendezvous Case.* In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 1785–1792, 2006. (Cité en page 135.)