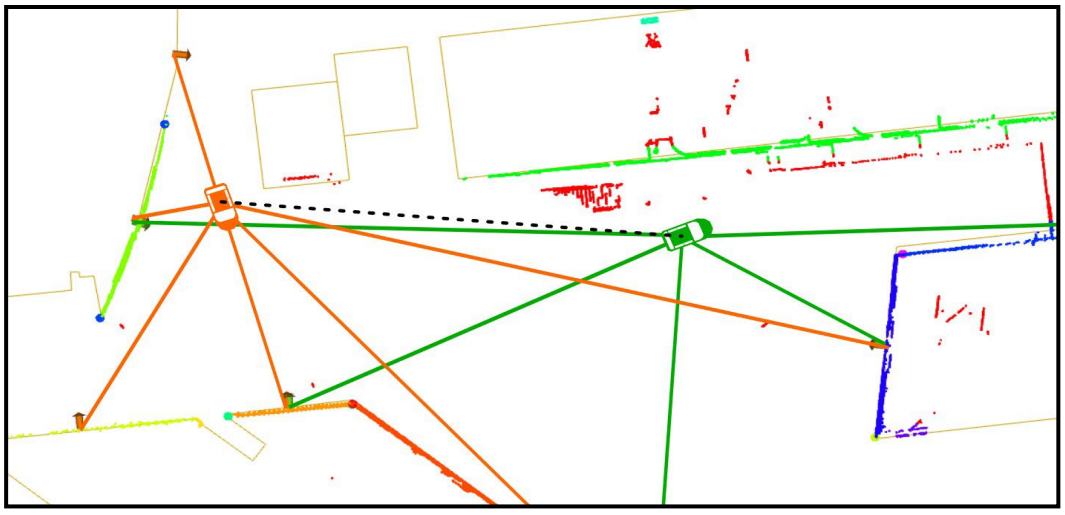
Par **Maxime ESCOURROU**

Méthode collaborative décentralisée pour la localisation et la mise à jour de cartes avec étude d'intégrité

Thèse présentée
pour l'obtention du grade
de Docteur de l'UTC



Soutenue le 20 décembre 2023
Spécialité : Automatique et Robotique : Unité de recherche Heudyasic (UMR-7253)
D2788

Méthode collaborative décentralisée pour la localisation et la mise à jour de cartes avec étude d'intégrité

Maxime Escourrou

Thèse présentée pour l'obtention du grade de Docteur de l'Université de Technologie de Compiègne

Spécialité : Automatique et Robotique

Thèse préparée au laboratoire Heudiasyc UMR 7352 UTC-CNRS

Thèse soutenue le mercredi 20 décembre 2023

Composition du jury :

Roland Chapuis	Professeur à l'Université Clermont Auvergne (rapporteur)
Simon Lacroix	Directeur de Recherche au LAAS-CNRS (rapporteur)
Véronique Cherfaoui	Professeure à l'UTC (présidente)
Maan El Badaoui El Najjar	Professeur à l'Université de Lille (examinateur)
Javier Ibanez-Guzman	Ingénieur de Recherche chez Renault (examinateur)
Joelle Al Hage	Maître de Conférences à l'UTC (Co-directrice)
Philippe Bonnifait	Professeur à l'UTC (Co-directeur)

Résumé

La capacité pour un robot de se localiser est un élément central en robotique mobile. Dans des environnements dans lesquels un système de positionnement satellitaire de type GNSS est indisponible ou avec des performances dégradées, les robots peuvent s'appuyer sur des caractéristiques cartographiques stables et géoréférencées détectées par des capteurs embarqués. Les caractéristiques considérées dans cette thèse sont obtenues à partir des façades des bâtiments et sont observées à l'aide de capteurs lidar 3D. Pour cela, une méthode d'extraction d'observation des nuages de points lidar est proposée. Cette méthode permet d'obtenir les poses correspondant aux milieux des façades, pour servir d'observation pour l'estimation d'état. Des cartes très précises et détaillées n'étant pas toujours accessibles, nous nous intéressons ici à l'utilisation de cartes participatives ouvertes pour la localisation des robots.

Dans ce contexte, nous nous attachons à proposer une méthode pour améliorer la localisation et la mise à jour des cartes en profitant de l'échange entre des robots communicants selon une architecture de collaboration et de fusion décentralisée qui permet aux robots de conserver un certain niveau d'autonomie. Nous traitons en particulier le cas de la collaboration indirecte, où les robots ne s'observent pas directement, mais collaborent via l'observation d'amers en commun. Ceci assure une augmentation de la portée de collaboration. La méthode se base sur l'utilisation du filtre de Schmidt-Kalman qui est bien adapté pour la gestion des corrélations entre les robots tout en limitant les communications. La fusion des états estimés par les robots lors des phases de collaboration est réalisée avec une moyenne de Kullback-Leibler pour maintenir une bonne consistance. Afin d'assurer l'intégrité de l'estimation d'état, l'approche met en œuvre une étape de détection et d'isolation des défauts. Grâce à la collaboration, les défauts de la carte peuvent être détectés, ce qui n'est pas possible lorsqu'ils sont observés séparément en navigation isolée.

La méthode est testée en simulation et sur des données réelles utilisant trois véhicules expérimentaux. Les résultats montrent l'intérêt de la collaboration pour améliorer la localisation des véhicules, l'estimation de la carte et la détection des défauts tout en maintenant un niveau élevé de consistance.

Mots-clés : Localisation collaborative, Mise à jour de carte, Filtre de Schmidt-Kalman, Tolérance aux défauts, Système multi-robots, Génération d'observation, Lidar, Observation d'amer

Abstract

The ability for a robot to localize itself is a central issue in mobile robotics. In environments where a GNSS-type satellite positioning system is unavailable or with degraded performance, robots can rely on stable geo-referenced map features detected by on-board sensors. The features considered in this PhD thesis are obtained from building facades and observed using 3D lidar sensors. For this purpose, an observation extraction method for lidar point clouds is proposed. This method provides the poses corresponding to the facade environments, to be used as observations for state estimation. As accurate and detailed maps are not always available, we are interested in the use of open-data and crowdsourced maps for robot localization.

In this context, we propose and study a method for improving localization and map update by taking advantage of the exchange between communicating robots according to a decentralized collaboration and fusion architecture that allows the robots to maintain a certain level of autonomy. In particular, we address the case of indirect collaboration, where robots do not observe each other directly, but collaborate via the observation of common landmarks. This increases the collaboration range. The method is based on the use of the Schmidt-Kalman filter, which is well suited to manage correlations between robots while limiting communication. The states estimated by the robots during the collaboration phases are merged using the Kullback-Leibler Average to maintain good consistency. To ensure the integrity of the state estimation, the approach implements a fault detection and isolation step. Through collaboration, map faults can be detected, which is not possible when they are observed separately in standalone navigation.

The method is evaluated in simulation and on real data using three experimental vehicles. The results show the added value of collaboration in improving vehicle localization, map estimation and fault detection, while maintaining a high level of consistency.

Keywords: Collaborative localization, Map update, Schmidt-Kalman filter, Fault tolerance, Multi-robot system, Observation generation, Lidar, Landmark observation

Remerciements

Les travaux présentés dans ce manuscrit ont été réalisés au laboratoire Heudiasyc, à l'Université de technologie de Compiègne. Je remercie le laboratoire ainsi que l'UTC pour m'avoir accueilli pendant toutes ces années.

Je remercie Monsieur Simon Lacroix, Directeur de Recherche au LAAS-CNRS et Monsieur Roland Chapuis, Professeur à l'Université Clermont-Auvergne pour avoir accepté de rapporter sur mon manuscrit, ainsi que pour leurs rapports très enrichissants et ouvrant de nombreuses perspectives.

Merci également aux membres du jury, Madame Véronique Cherfaoui, Professeure à l'Université de technologie de Compiègne, Monsieur Maan El Badaoui El Najjar, Professeur à l'Université de Lille, et Javier Ibanez-Guzman, Ingénieur de recherche chez Renault, pour avoir accepté d'examiner ma thèse, pour leurs questions pertinentes, ainsi que pour leurs présence et accompagnement pendant les Comités de Suivi Individuels.

Un grand merci à mes directeurs de thèse, Philippe Bonnifait, Professeur à l'Université de technologie de Compiègne, et Joelle Al Hage, Maitre de conférence à l'Université de technologie de Compiègne, pour leur encadrement tout au long de ces trois ans. Merci à Philippe pour ses remarques constructives, adoptant parfois un point de vue différent, rafraîchissant. Un immense merci à Joelle pour le temps qu'on a passé ensemble à débattre de la meilleure façon d'approcher les différents problèmes, nous étions tous les deux têtus, mais nous étions aussi capables d'admettre nos erreurs et le point de vue de l'autre. Ce genre d'échanges est ce qui me donne envie de faire de la recherche.

Je remercie aussi mes collègues ayant participé à la collecte des données réelles : Antoine, Stéphane, Thierry, Corentin et Rémy.

Merci au personnel administratif du laboratoire, ils ont été d'une grande aide lors de toutes les démarches : Sabine, Nathalie, Séverine et Bérengère.

Je remercie aussi mes collègues pour les moments passés au laboratoire, et en particulier Lyes et Maxime, ainsi qu'Antoine avec qui j'ai partagé un bureau. Merci beaucoup à mes collègues et amis, Stéphane, Rémy et Thibaud, pour leur présence et leur soutient tout au long de ma thèse.

Un grand merci à mes amis de longue date pour leur soutien, même de loin : Benjamin, Paul, Maëlle, Quentin, Théo et Florent.

Un immense merci à ma sœur et à mes parents qui ont fait de moi l'homme que je suis aujourd'hui, tant sur l'aspect humain que pour ma soif de savoir et ma curiosité.

Table des matières

Acronymes

BA	*Bundle Adjustment*
C-SLAM	SLAM collaboratif
CI	*Covariance intersection Filter*, filtre à intersection de covariance
CNN	*Convolutional Neural Network*, réseau de neurones convolutif
EIF	*Extended Information Filter*, filtre informationnel étendu
EKF	*Extended Kalman Filter*, filtre de Kalman étendu
FD	*Fault Detection*, détection de défauts
FDE	*Fault Detection and Exclusion*, détection et exclusion de défauts
FDI	*Fault Detection and Isolation*, détection et isolation de défauts
FDII	*Fault Detection, Isolation and Identification*, détection, isolation et identification de défauts
FDIR	*Fault Detection and Isolation and Recovery*, détection et isolation de défauts puis récupération
GNSS	*Global Navigation Satellite System*, Système de navigation mondial par satellites
ICP	*Iterative Closest Point*
IEKF	*Interlaced EKF*, EKF entrelacé
IF	*Information Filter*, filtre informationnel
KF	*Kalman Filter*, filtre de Kalman
KLA	*Kullback-Leibler Average*, moyenne de Kullback-Leibler
KLD	*Kullback-Leibler Divergence*, divergence de Kullback-Leibler

ND	*Normal Distribution*, distribution normale
NDT	*Normal Distributions Transform*, transformation via distributions normales
OOSM	*out-of-sequence measurements*, mesures hors-séquence
OSM	OpenStreetMap
PCL	*Point Cloud Library*
RAIM	*Receiver Autonomous Integrity Monitoring*
RANSAC	*Random Sampling Consensus*, consensus par échantillonnage aléatoire
ROS	*Robot Operating System*
SCI	*Split Covariance intersection Filter*, filtre à intersection de covariance partitionné
SEIF	*Sparse Extended Information Filter*, filtre informationnel étendu épars
SKF	*Schmidt-Kalman Filter*, filtre de Schmidt-Kalman
SLAM	*Simultaneous Localization and Mapping*, localisation et cartographie simultanée
SPoF	*Single Point of Failure*, point unique de défaillance
UKF	*Unscented Kalman Filter*, filtre de Kalman sans parfum
VO	*Visual Odometry*, odométrie visuelle

Introduction générale

Contexte et problématique

Une localisation précise est nécessaire pour de nombreuses applications en robotique mobile. Les erreurs d'estimation de pose doivent aussi souvent être maitrisées, et ce, pour des raisons de sécurité lorsque les robots peuvent causer des dégâts importants en cas de dysfonctionnement. C'est le problème d'une localisation précise et intègre.

En extérieur, un système de positionnement par satellites (*Global Navigation Satellite System*, GNSS) est généralement accessible gratuitement, via la réception des signaux de satellites GPS (*Global Positionning System*, système américain), Galileo (système européen), Glonass (système russe), ou Beidou (système chinois). Cependant, en présence de bâtiments, les signaux peuvent être altérés à cause des multi-trajets et des obstructions, comme illustré sur la figure 1.

En plus des perturbations, il n'est pas toujours possible de capter les signaux GNSS, notamment dans les zones souterraines (tunnels, parkings, etc.) ou à l'intérieur de bâtiments par exemple. Dans ce cas, une exigence élevée de localisation précise peut demander l'installation de matériels spécifiques, avec par exemple des balises actives ou des systèmes de capture optiques fixes dans l'environnement. Ces systèmes sont en général coûteux et ont une couverture d'utilisation réduite. Une autre approche consiste à utiliser des systèmes de localisation qui utilisent des mesures sur l'environnement (qu'il soit naturel ou artificiel) mais sans avoir à ajouter d'équipements spécifiques pour réaliser la tâche de localisation. Cette approche, très classique en robotique, est une des clés de l'autonomie des robots.

Cette thèse s'inscrit dans ce contexte et étudie un système de localisation qui utilise des observations d'amers de l'environnement géoréférencés dans une carte, sans aucune source de localisation absolue, hormis pour l'initialisation. L'idéal est d'utiliser des caractéristiques cartographiques stables dans le temps et facilement détectables avec des capteurs embarqués.

L'application qui nous intéresse dans ces travaux est la localisation de véhicules intelligents dans des milieux défavorables aux systèmes GNSS tels que les milieux urbains. Dans de tels environnements, les façades des bâtiments sont des caractéristiques stables et facilement visibles. De plus, elles sont souvent cartographiées et géoréférencées dans un système de coordonnées géodésiques.

Les cartes « haute définition » (HD) (très précises et avec beaucoup de détails) ne contiennent pas à l'heure actuelle de description des bâtiments et ceci va certainement rester le cas pendant longtemps compte tenu du coût élevé de cartographie et de mise à jour. Nous nous intéressons donc ici à l'utilisation de cartes de type *crowdsourced* (ou

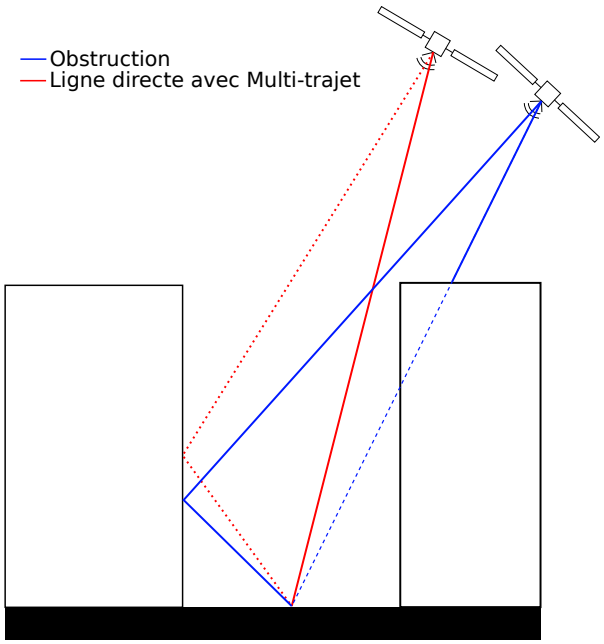


FIGURE 1. – Exemple de problèmes de propagation des signaux GNSS en milieu urbain. Lorsqu'il existe un masquage dans la ligne de vue directe d'un satellite, sa mesure peut être particulièrement faussée et engendrer un défaut de mesure.

open data) pour localiser des robots. Un exemple de ce genre très connu est la carte OpenStreetMap. Ce type de carte contient des descriptions des bâtiments assez variables en termes de précision et de détails d'un pays à l'autre. De plus, ces cartes sont sujettes à des erreurs et des défauts qui vont affecter la précision et l'intégrité de la localisation. En résumé, elles fournissent des informations à priori utiles et pertinentes, mais une correction et une mise à jour de leurs données est nécessaire pour obtenir un système de localisation performant.

Dans ce cadre, la collaboration entre robots (ou véhicules) semble être une approche intéressante pour profiter d'informations nombreuses et redondantes pour mettre à jour rapidement et fréquemment des cartes OpenStreetMap. De plus, en échangeant les caractéristiques observées et en partageant leurs cartes respectives, les robots peuvent améliorer simultanément leur localisation et leur cartographie. Il apparait alors une boucle vertueuse « localisation-cartographie » où l'amélioration de l'une améliore l'autre. Ce concept est connu depuis longtemps en robotique.

Développer un système collaboratif implique d'utiliser des communications, de définir une stratégie de collaboration et de choisir une architecture de fusion de données. Dans le contexte de cette thèse, nous nous intéressons en particulier à l'étude d'une collaboration entre robots à travers des mesures indirectes lorsqu'ils observent le ou les mêmes amers (figure 2). Cela permet de fournir une information directe pour la mise à jour de la carte. Cette stratégie a aussi l'avantage d'augmenter la portée de collaboration et d'être efficace en termes de communication comme nous le verrons plus tard.

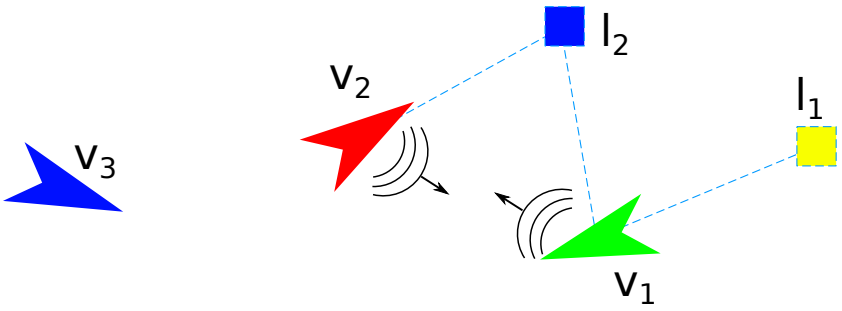


FIGURE 2. – Exemple de collaboration via l'observation d'amer en commun. Les robots v_1 et v_2 communiquent afin de collaborer après l'observation commune de l'amer l_2.

Les problèmes d'estimation d'état et de fusion de données présentent des caractéristiques particulières lorsqu'on aborde des problèmes collaboratifs. Ils doivent, par exemple, être efficaces compte tenu de la complexité qui peut apparaitre, pouvoir passer à l'échelle, maitriser les cycles d'échanges, gérer les pertes de données, les retards, etc. Cette question méthodologique fera l'objet d'une attention particulière dans ce manuscrit. Dans un environnement non contrôlé et encombré, les observations des capteurs ou le positionnement des amers dans la carte peuvent être entachées d'erreurs importantes, ce qui peut affecter la précision et l'intégrité de l'estimation d'état. Les méthodes d'estimation d'état et de fusion de données doivent donc être complétées d'étapes de détection et d'isolation de défauts.

Enfin, la collaboration dans un système multi-robots peut s'implémenter sous la forme de plusieurs architectures. L'architecture centralisée avec une unité centrale, crée une grande dépendance à l'égard de cette dernière, en particulier à une défaillance de sa part. Les architectures distribuées ou décentralisées résolvent ce problème mais ajoutent une plus grande complexité au niveau de la fusion de données et des échanges de données issus de la communication. Dans cette thèse, et compte tenu des applications robotiques envisagées, une architecture décentralisée pour la fusion de données semble donc être plus appropriée. Cette approche sera étudiée avec l'objectif de limiter les communications et le nombre d'échanges entre les robots.

Questions principales abordées dans la thèse

Sur la base de propositions méthodologiques, de simulations et d'expérimentations, cette thèse cherche donc à apporter des éléments de réponse aux questions suivantes :

— Comment localiser un ensemble de robots communicants dans un environnement défavorable au GNSS et avec une carte imparfaite ?
— Comment la collaboration entre les robots peut-elle aider à améliorer la carte et la localisation ? Quel est l'apport de la collaboration sur la précision et la consistance des estimations ?
— Comment faire en sorte que la localisation et la cartographie des robots soit tolérante

aux défauts, ainsi que résiliente aux pertes de communication pour que les robots conservent un certain niveau d'autonomie ?

Résumé des contributions

Cette thèse présente une méthode collaborative décentralisée pour la localisation de robots et la mise à jour de carte à travers des observations d'amers obtenues avec un capteur extéroceptif embarqué et avec la prise en compte de contraintes de communication.

Pour cela, la collaboration se réalise via des mesures indirectes entre robots à partir des observations d'amers en commun. Ces mesures permettent de mettre à jour la carte tout en améliorant la localisation des robots. Elles permettent une augmentation de la portée de collaboration et sans qu'une vue directe entre les robots soit nécessaire.

La méthode d'estimation se base sur l'utilisation du filtre de Schmidt-Kalman (SKF) qui est bien adapté aux problèmes d'estimation d'état suivant une architecture décentralisée. L'étape de fusion lors des phases de collaboration utilise la moyenne de Kullback-Leibler qui possède des propriétés intéressantes pour ce problème.

Une étape de détection et d'isolation des mesures capteurs erronées est proposée en se basant sur des résidus générés dans les espaces d'observation et d'état. Cette étape se fait encore suivant une architecture décentralisée. L'étape d'isolation utilise des matrices de signatures permettant jusqu'à l'isolation de deux défauts simultanés. L'apport de la collaboration pour la détection de défauts est mis en avant surtout pour les défauts cartes.

Une méthode de génération d'observations de poses relatives à partir de nuages de points lidar est aussi présentée. Cette méthode extrait des nuages de points 3D et 360° les poses représentant les centres des façades des bâtiments observés. Cette synthèse des façades sous forme de pose permet l'échange d'information de manière allégée, tout en conservant un système observable même avec une seule observation.

L'apport de la collaboration est démontré en simulation et expérimentalement, que ce soit pour la localisation des robots, pour l'estimation de la carte, pour la consistance de l'estimation ou pour la détection et l'isolation de défauts.

L'expérimentation sur données réelles, spécialement recueillies pour cette thèse, a mis en œuvre trois voitures expérimentales du laboratoire. Un scénario permettant la mise en avant de situations de collaboration, ainsi que de zones difficiles, a été réalisé sur le campus de l'université.

Plan de la thèse

Ce manuscrit est décomposé en deux parties. Une première partie méthodologique sur l'estimation d'état et la tolérance aux défauts pour une localisation collaborative multi-capteurs basée carte et une deuxième partie, plus applicative, qui porte sur l'adaptation de l'approche à des données réelles en utilisant des véhicules intelligents. Cette deuxième partie décrit aussi la méthodologie de génération des observations à partir de nuages de

points lidar.

Le chapitre 1 présente une revue de la littérature liée à la localisation et à la mise à jour de carte pour un cas multi-robots. Les différentes architectures de fusion de données sont discutées et sont comparées à travers plusieurs méthodes existantes. La littérature concernant la détection et l'isolation de défaut est aussi synthétisée, avec une limitation aux méthodes concernant les défauts de capteurs. Les méthodes de récupération d'un état sans défauts sont aussi explorées.

Ensuite, le cœur de la méthode est présenté au chapitre 2, via le filtre de Schmidt-Kalman. Le problème est initialement formalisé dans le cas d'une architecture centralisée, puis est adapté à une architecture décentralisée basée sur le SKF et la moyenne de Kullback-Leibler pour fusionner les différentes estimations des robots. Les limitations de communication sont prises en compte pour cette partie. La détection et l'isolation des défauts, permettant de récupérer un état sans défaut, sont aussi détaillées dans ce chapitre.

Cette méthode est évaluée en simulation au chapitre 3, grâce à un simulateur développé spécifiquement pour la thèse.

Le premier chapitre de la seconde partie (le chapitre 4) commence par un état de l'art sur les traitements des nuages de points lidar dans le but d'extraire des observations. Il s'agit des méthodes de segmentation sémantiques et géométriques, les méthodes d'extraction de *features* ou encore des méthodes permettant d'aligner des nuages de points entre eux ou avec une carte. La méthode proposée pour extraire les observations des façades du nuage de points lidar est détaillée également dans ce chapitre. Cette méthode utilise à la fois de la segmentation sémantique de nuage de points, de l'alignement avec la NDT, du groupement par façade, et du RANSAC dans le but de donner le meilleur résultat possible lors de la génération finale de l'observation.

La méthode globale est évaluée sur des données expérimentales au chapitre 5. Ces données ont été recueillies grâce aux véhicules du laboratoire sur un scénario montrant les différents aspects et avantages de la méthode.

Publications

Les travaux présentés dans cette thèse ont fait l'objet de plusieurs publications.

Tout d'abord, une étude préalable sur la localisation de véhicules en utilisant les bâtiments en milieu urbain a été publiée dans

> Maxime Escourrou, Joelle Al Hage et Philippe Bonnifait (août 2021). « NDT Localization with 2D Vector Maps and Filtered LiDAR Scans ». In : *European Conference on Mobile Robots (ECMR)*, p. 1-6

Une première version de la méthode théorique, appuyée par des résultats en simulation, a été publiée dans

> Maxime Escourrou, Joelle Al Hage et Philippe Bonnifait (juill. 2022). « Decentralized Collaborative Localization with Map Update Using Schmidt-Kalman Filter ». In : *25th International Conference on Information Fusion (FUSION)*, p. 1-8

Première partie

Principes méthodologiques pour la localisation collaborative avec mise à jour de cartes et tolérante aux défauts

Introduction

Cette première partie de la thèse porte sur l'approche théorique proposée pour la localisation collaborative avec mise à jour de cartes, basée sur le filtre de Schmidt Kalman. La collaboration se réalise d'une façon indirecte en observant des amers communs dans l'environnement, ceci suivant une architecture décentralisée qui évite des échanges fréquents et directs avec une unité centrale. Les amers sont préalablement cartographiés dans une carte imprécise. Les communications et les échanges d'informations entre les robots sont limités aux moments où ils observent un même amer.

Les mesures capteurs peuvent être entachées d'erreurs provenant du processus de génération des observations, de l'environnement ou du capteur lui-même. Ces défauts auront un impact direct sur la précision et sur l'intégrité de l'estimation d'état. Pour cela, une étape de détection et d'isolation de défauts est ajoutée tout en considérant la possibilité des défauts cartes qui se traduisent par des biais de positionnement au niveau de certains amers. Les défauts simultanés sont aussi considérés.

1. État de l'art sur la localisation et la mise à jour de carte collaborative en présence de défauts

Sommaire

Introduction

La localisation et la cartographie font partie du domaine de recherche de la fusion de données et de l'estimation d'état, et sont particulièrement utilisées en robotique.

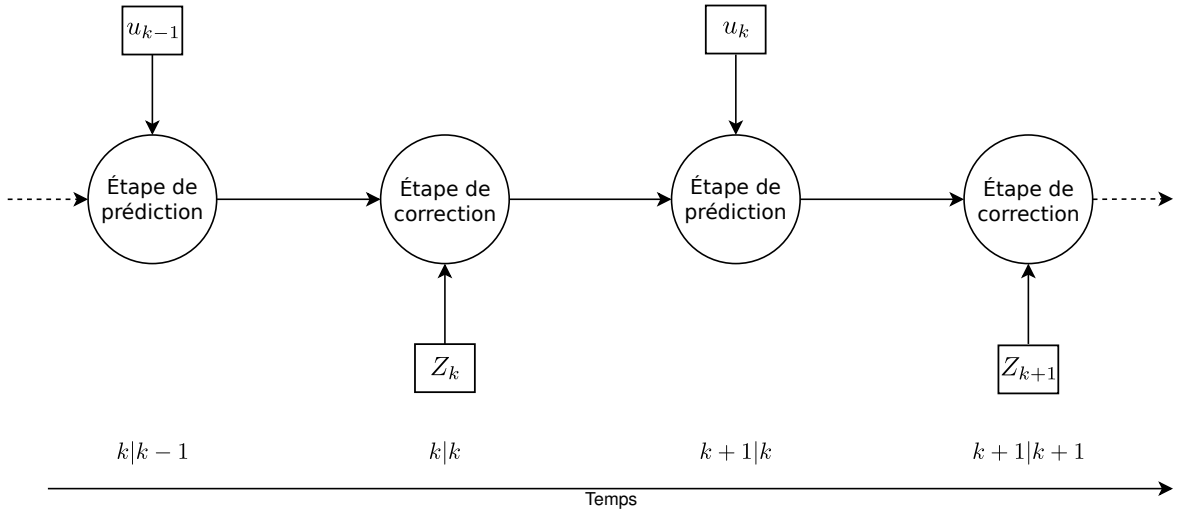


FIGURE 1.1. – Représentation d'un filtre bayésien sous forme de chaîne de Markov.

Dans ce chapitre, un état de l'art sur les méthodes de fusion de données basées sur des filtres est réalisé avant de discuter spécifiquement des méthodes de localisation et de cartographie simultanées. La recherche sur les méthodes de filtrage est encore très active. Nous présenterons ici les filtres les plus courants sans entrer dans le détail de toutes les variantes existantes. Un état de l'art sur les méthodes d'estimation d'état tolérantes aux défauts est aussi présenté, afin de robustifier le système en présence de mesures erronées. Pour finir, nous placerons notre approche dans la littérature existante.

L'aspect collaboratif de notre travail étant central, un accent particulier est mis sur les méthodes collaboratives, avec la localisation collaborative et le SLAM collaboratif.

1.1. Méthodes bayésiennes pour la fusion de données

L'état du système est représenté par son estimation, notée X, et sa matrice de covariance de l'erreur estimée, notée P. Les filtres pour la fusion de données sont issus du filtre Bayésien (MURPHY 2002) et sont constitués de deux étapes :

1. une première étape de prédiction de l'état à estimer en utilisant le modèle d'évolution associé,
2. et une deuxième étape utilisant une ou plusieurs observations de l'environnement pour corriger la prédiction, en utilisant le modèle d'observation.

Les filtres bayésiens peuvent être représentés sous forme de chaîne de Markov (MURPHY 2002), mettant en évidence leur récursivité, via les dépendances avec le pas de temps précédent ainsi qu'avec les mesures en entrée. Sur la figure 1.1, quelques étapes sont représentées. Les étapes de prédiction utilisent l'état du filtre X_k et de l'entrée u_k pour estimer l'état futur X_{k+1}. Les étapes de correction utilisent des observations de l'état du système par des capteurs, notées Z_k, pour corriger l'état prédit. Les étapes de prédiction et de correction ne sont pas nécessairement en alternance, elles suivent l'ordre des données.

Le modèle d'évolution permet d'estimer l'état du système à un temps futur (X_{k+1} et P_{k+1}) en utilisant l'état actuel (X_k et P_k) ainsi que, éventuellement, une entrée u_k. Cette entrée peut être, par exemple, la commande appliquée au système ou des mesures d'odométrie. Cela peut être représenté sous forme la forme :

$$X_{k+1} = f\left(X_k, u_k + \alpha_{u,k}\right) + \alpha_k, \tag{1.1}$$

avec u_k l'entrée avec un bruit $\alpha_{u,k}$ et où f est le modèle d'évolution. Dans plusieurs méthodes, ce modèle est supposé linéaire avec $f(X) = AX$. Le modèle est entaché d'un bruit d'évolution α_k, lié à l'imperfection du modèle (hypothèses utilisées, linéarisation, etc.).

L'étape de correction utilise un modèle d'observation sous la forme :

$$Z_k = h\left(X_k\right) + \beta_k, \tag{1.2}$$

avec h le modèle d'observation où l'hypothèse linéaire est considérée dans certains filtres et β_k est le bruit de mesure. L'objectif du modèle d'observation est de prédire une observation Z_k en fonction de la prédiction du modèle d'évolution.

Remarque. Nous utiliserons ici des filtres Bayésiens sous leur forme discrète. La notation de la prédiction $k+1|k$ se lit « à l'instant $k+1$ sachant l'instant k » et signifie que l'on prédit l'état du système à l'instant futur $k+1$ avec les informations connues jusqu'à l'instant k. Au contraire, la notation de la correction $k|k$ signifie que l'on corrige l'état du système à l'instant actuel k avec les informations connues jusqu'à l'instant k.

1.1.1. Définitions

La consistance du filtre est très importante pour assurer une estimation d'état précise et intègre, mais aussi pour être à même de prendre les meilleures décisions quant à la confiance à accorder à l'estimation. La consistance est définie comme suit :

Définition 1: Consistance. En statistiques, la propriété de consistance (convergence) d'un estimateur est le fait qu'il converge en probabilité vers la véritable valeur (Amemiya 1985). Dans le cas du filtrage, un filtre est dit consistant lorsque la covariance de l'erreur réelle est correctement estimée, par le filtre.

L'intégrité est un point essentiel pour la localisation de robots dans des conditions de sécurité. Elle se définit ainsi (N. Zhu et al. 2018) :

Définition 2. L'intégrité est la mesure de confiance qui peut être placée dans l'exactitude de l'information fournie par un système de navigation.

Dans nos travaux, nous nous focaliserons sur la tolérance aux défauts, permettant de conserver une bonne estimation et donc une bonne intégrité. C'est aussi ce qu'on appelle l'intégrité interne.

1.1.2. Filtre de Kalman et filtre de Kalman étendu

Le filtre le plus connu est sans doute le filtre de Kalman (KF) proposé dans (Kalman 1960). Le filtre de Kalman est une méthode d'estimation optimale pour des systèmes linéaires avec des bruits gaussiens non biaisés. L'hypothèse centrale du KF est que l'estimation d'état suit une distribution normale, avec une moyenne de X et une covariance de P.

La version non-linéaire est appelée le filtre de Kalman étendu (*Extended Kalman Filter*, EKF) et repose sur la linéarisation des modèles autour de l'estimation, ce qui fait perdre au filtre son optimalité. Cela rajoute une approximation, en plus de rendre la qualité de l'estimation directement dépendante de l'erreur de l'estimation précédente. La non-linéarité peut être engendrée par le modèle d'évolution ou par le modèle d'observation. Une explication approfondie de ces deux filtres peut être trouvée dans (Welch et Bishop 1997). Dans la suite, la version étendue est présentée. Pour la version linéaire, il suffit d'utiliser des modèles linéaires.

L'étape de prédiction utilise le modèle d'évolution f :

$$X_{k+1|k} = f(X_{k|k}, u_k), \tag{1.3}$$
$$P_{k+1|k} = F_k P_{k|k} F_k^T + G_k Q_{u,k} G_k^T + Q, \tag{1.4}$$

où Q_u est la matrice de covariance du bruit sur l'entrée $\alpha_{u,k}$ et Q la matrice de covariance du bruit de modèle α_k.

Les matrices jacobiennes F_k et G_k sont utilisées pour la linéarisation, et sont les suivantes :

$$F_k = \left.\frac{\partial f}{\partial X}\right|_{X_{k|k}} = \begin{pmatrix} \frac{\partial f_1}{\partial X_1} & \cdots & \frac{\partial f_1}{\partial X_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial X_1} & \cdots & \frac{\partial f_n}{\partial X_n} \end{pmatrix}_{X_{k|k}}, \tag{1.5}$$

$$G_k = \left.\frac{\partial f}{\partial u}\right|_{u_k}. \tag{1.6}$$

L'étape de correction utilise des observations Z_k pour corriger la prédiction faite précédemment. Pour cela, le modèle d'observation h est utilisé :

$$X_{k|k} = X_{k|k-1} + K_k \left(Z_k - h(X_{k|k-1})\right), \tag{1.7}$$
$$P_{k|k} = (I - K_k H_k) P_{k|k-1}. \tag{1.8}$$

H est la jacobienne associée au modèle d'observation non-linéaire,

$$H = \left.\frac{\partial h}{\partial X}\right|_{X_{k|k-1}}, \tag{1.9}$$

et K est le gain de Kalman. Il est calculé comme suit :

$$K_k = P_{k|k-1} H_k^T \left(H_k P_{k|k-1} H_k^T + R \right)^{-1}, \tag{1.10}$$

où R est la matrice de covariance du bruit d'observation.

1.1.3. Filtre informationnel

Le filtre de Kalman a été décliné dans d'autres formes, la plus connue étant la forme informationnelle (IF), et sa version étendue (*Extended Information Filter*, EIF) (Hugh Durrant-Whyte 2002). Ce filtre est mathématiquement équivalent au KF. L'état du système n'est plus représenté sous la forme de moments de la distribution normale, mais sous forme canonique (ou forme informationnelle). La matrice informationnelle est définie comme $Y = P^{-1}$, et le vecteur informationnel sous la forme $y = P^{-1}X$. Cette forme a le grand avantage de permettre la correction sous forme de somme de contributions informationnelles, rendant ainsi le processus commutatif (et donc asynchrone), mais également de permettre de grandes incertitudes lorsque $Y \to \mathbb{0}$. Cependant, l'étape de prédiction est complexifiée par rapport au filtre de Kalman. Elle est donc régulièrement effectuée dans la forme du KF, ce qui implique l'inversion de tout Y avant et après la prédiction. Un IF est donc intéressant lorsqu'il y a beaucoup d'observations et peu de prédictions.

Un des problèmes de l'EIF est que pour connaître une sous partie de l'état, if faut effectuer une marginalisation coûteuse, où toute la matrice informationnelle doit être connue. Pour marginaliser la variance de la variable a dans un état contenant les variables a et b, il faut utiliser l'équation suivante :

$$P_{aa} = \left(Y_{aa} - Y_{ab} Y_{bb}^{-1} Y_{ba} \right)^{-1} \tag{1.11}$$

Il est ainsi impossible d'agir sur une composante en particulier sans passer par cette étape d'inversion, sauf à utiliser des formes particulières permettant l'accès aux éléments dans un ordre précis (Mu et al. 2011).

1.1.4. Filtre de Schmidt-Kalman

Un autre filtre dérivé du filtre de Kalman est le filtre de Schmidt-Kalman (SKF ; Schmidt 1966), aussi appelé « *consider Kalman filter* » (Zanetti et D'Souza 2013). Il permet de limiter les besoins calculatoires lorsque l'état est peu observable et qu'il n'est pas nécessaire de l'estimer dans son entièreté en permanence. Avec ce filtre, il est ainsi possible de n'estimer qu'une partie de l'état si l'accès au reste de l'état est limité. Il fera l'objet du chapitre 2.

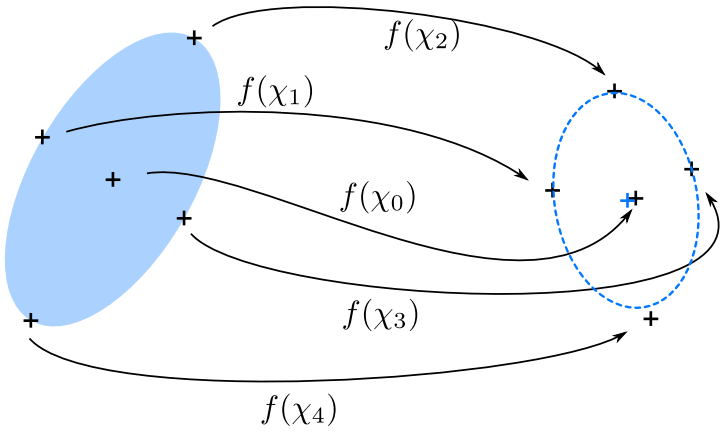


FIGURE 1.2. – Principe de la transformation non parfumée.

1.1.5. Filtre de Kalman sans parfum

L'EKF et l'EIF reposent sur la linéarisation des modèles non-linéaires pour réaliser leur estimation. Cette linéarisation génère des erreurs lorsque l'estimation est trop erronée (en particulier sur l'orientation), car l'approximation linéaire au point d'estimation ne tient pas lorsque l'erreur est trop grande.

Le filtre de Kalman sans parfum (*Unscented Kalman Filter*, UKF) répond à ce problème. Ce filtre utilise un sous-échantillonnage, les « points-sigma », et applique les modèles non-linéaires dessus (S. J. JULIER et J. K. UHLMANN 1997a). Ainsi, les points-sigma χ_i sont choisis de telle façon que leur moyenne et leur covariance correspondent à la distribution représentée. Il y en a $2L+1$ avec L la taille de l'état (S. J. JULIER et J. K. UHLMANN 1997a) et ils ont des poids W_i différents :

$$\begin{aligned} \chi_0 &= \bar{X} & W_0 &= \tfrac{\lambda}{L+\lambda} \\ \chi_i &= \bar{X} + \left(\sqrt{(L+\lambda)P}\right)_i & W_i &= \tfrac{1}{2(L+\lambda)} \quad i = 1, ..., L \\ \chi_{i+L} &= \bar{X} - \left(\sqrt{(L+\lambda)P}\right)_i & W_{i+L} &= \tfrac{1}{2(L+\lambda)} \quad i = 1, ..., L \end{aligned} \tag{1.12}$$

où λ est un paramètre d'échelle, $\sqrt{P}$ représente la décomposition de Cholesky : $P = \sqrt{P}\sqrt{P}^T$, et $(A)_i$ représente la i-ème colonne de A.

La transformation par un modèle d'évolution se fait en appliquant le modèle sur chaque point-sigma. Ensuite, la moyenne et la covariance sont retrouvées empiriquement sur les

points-sigma transformés, en pondérant les sommes avec W_i définis plus tôt (figure 1.2).

$$\chi_{i,k+1|k} = f(\chi_{i,k|k}, u_k), \tag{1.13}$$

$$X_{k+1|k} = \sum_{i=0}^{2L} W_i \chi_{i,k+1|k}, \tag{1.14}$$

$$P_{k+1|k} = \sum_{i=0}^{2L} W_i (\chi_{i,k+1|k} - X_{k+1|k})(\chi_{i,k+1|k} - X_{k+1|k})^T. \tag{1.15}$$

Pour l'étape de correction, le modèle d'observation est appliqué à chaque point-sigma de la même manière afin d'obtenir une prédiction de l'observation par point-sigma, puis une prédiction unique via la moyenne et la covariance :

$$\mathcal{Z}_{i,k} = h(\chi_{i,k|k-1}), \tag{1.16}$$

$$\hat{Z}_k = \sum_{i=0}^{2L} W_i \mathcal{Z}_{i,k}. \tag{1.17}$$

L'innovation est ensuite calculée puis multipliée à un gain K, afin de corriger l'estimation :

$$X_{k|k} = X_{k|k-1} + K_k(Z_k - \hat{Z}_k), \tag{1.18}$$

$$P_{k|k} = P_{k|k-1} - K_k P_{zz} K_k^T, \tag{1.19}$$

$$K_k = P_{XZ} P_{ZZ}^{-1}, \tag{1.20}$$

$$P_{XZ} = \sum_{i=0}^{2L} W_i (\chi_{i,k|k-1} - X_{k|k-1})(\mathcal{Z}_{i,k} - \hat{Z}_k)^T, \tag{1.21}$$

$$P_{ZZ} = R_k + \sum_{i=0}^{2L} W_i (\mathcal{Z}_{i,k} - \hat{Z}_k)(\mathcal{Z}_{i,k} - \hat{Z}_k)^T. \tag{1.22}$$

Après l'étape de correction, les χ_i sont ré-échantillonnés avec (1.12).

La force de l'UKF est qu'il n'y a pas besoin de linéariser pour les problèmes non-linéaires, et il propage mieux les covariances dans un système non-linéaire que l'EKF.

1.1.6. Filtre à intersection de covariance

Un autre type de filtre pour la fusion de données est le filtre à intersection de covariance (CI) présenté dans (S. J. Julier et J. K. Uhlmann 1997b). Il permet de fusionner de manière robuste lorsque les corrélations entre les données à fusionner sont inconnues, tout étant assez simple. Cependant, il est pessimiste. Le principe est de combiner de manière convexe (la somme des poids doit faire 1) deux données a et b afin de minimiser

la trace ou le déterminant de la covariance du résultat de la fusion, noté c.

$$P_c^{-1} = wP_a^{-1} + (1-w)P_b^{-1} \tag{1.23}$$
$$c = P_c\left[wP_a^{-1}a + (1-w)P_b^{-1}b\right] \tag{1.24}$$

où $w \in [0,1]$ est le paramètre libre qui doit être optimisé pour minimiser la trace ou le déterminant de la covariance P_c. Ce filtre est très robuste à la consanguinité de données (double compte d'information), car il considère la corrélation complètement inconnue.

Afin de combiner cette force avec l'optimalité du KF, une version mixte avec ce denier existe, il s'agit du filtre à intersection de covariance partitionné (*Split Covariance Intersection*, SCI) (H. Li, Nashashibi et Yang 2013 ; Lima et al. 2021 ; S. J. Uhlmann J. K. 2009) qui repose sur une décomposition de la covariance P entre une partie dépendante P_d et une partie indépendante P_i, telle que $P = P_d + P_i$. Les équations de fusion sont un peu plus complexes et nécessitent plus de réglages.

$$P_a = P_{a,d}/w + P_{a,i}, \tag{1.25}$$
$$P_b = P_{b,d}/(1-w) + P_{b,i}, \tag{1.26}$$
$$P_c^{-1} = P_a^{-1} + P_b^{-1}, \tag{1.27}$$
$$c = P_c\left[P_a^{-1}a + P_b^{-1}b\right], \tag{1.28}$$
$$P_{c,i} = P_c\left[P_a^{-1}P_{a,i}P_a^{-1} + P_b^{-1}P_{b,i}P_b^{-1}\right]P_c, \tag{1.29}$$
$$P_{c,d} = P_c - P_{c,i}, \tag{1.30}$$

où w est optimisé pour minimiser la trace ou le déterminant de P_c.

Le SCI est moins pessimiste que le CI, et reste résilient à la consanguinité de données. Cependant, le choix de la décomposition n'est pas trivial, car il peut être difficile de séparer les variables dépendantes et celles qui ne le sont pas. La forme du KF n'est pas visible directement, mais il se devine sous sa forme simplifiée $P_{c,i}^{-1} = P_{a,i}^{-1} + P_{b,i}^{-1}$, s'il n'y avait pas les parties dépendantes.

1.1.7. Filtre particulaire

Le filtre particulaire utilise des méthodes de Monte-Carlo. Il permet d'estimer un état dans un problème non linéaire et non gaussien. Via un grand nombre de points dans l'espace d'état, appelés ici « particules », le filtre particulaire simule l'évolution de ces points lors des étapes de prédiction et de correction pour avoir une représentation de la distribution de l'état sous forme de nuage de points.

Le modèle de prédiction est appliqué à chaque particule avec un tirage aléatoire représentant le bruit. Lorsqu'il y a une observation, le poids de chaque particule est ajusté en fonction de sa vraisemblance. Les particules sont ré-échantillonnées, soit autour d'une particule existante avec la plus forte vraisemblance (maximum a posteriori), soit autour de la moyenne des particules pondérées par leurs vraisemblances (espérance a posteriori). Les particules sont redistribuées autour de cette particule retenue afin de recentrer la recherche.

1.2. Localisation et cartographie simultanées

Après avoir vu, de manière non exhaustive, les filtres bayésiens, cette section présente la localisation et la cartographie simultanées, qui utilise parfois ces filtres. La localisation et cartographie simultanées (*Simultaneous Localization and Mapping*, SLAM), ont commencé à être l'objet de recherches à la fin des années 1980 et sont devenues populaires dans les années 1990. Plusieurs définitions peuvent être trouvées dans la littérature, plus ou moins strictes. Nous pouvons relever tout d'abord une définition dans (G. Bresson et al. 2017) qui se veut volontairement large pour englober un maximum de méthodes liées à la localisation et à la cartographie :

> Les approches de SLAM sont composées au minimum d'un module d'odométrie et d'un module de cartographie.

Suivant cette définition, il n'y a pas de contrainte sur la simultanéité du traitement : la carte peut être donnée initialement, elle peut ne subir que des changements mineurs, voir aucun.

L'étude de la littérature (Cadena et al. 2016) propose une définition portant à la fois sur la simultanéité, et la construction de la carte :

> Le SLAM comprend l'estimation de l'état du robot équipé de capteurs simultanément à la construction d'un modèle de l'environnement (carte).

Cette définition met de côté les approches qui ne construisent pas une carte, mais utilisent et corrigent une carte déjà existante.

La définition adoptée dans ce manuscrit est donnée par (Lajoie et al. 2022), elle limite un peu moins les approches concernées :

Définition 3. Le SLAM est l'estimation conjointe de l'état du robot et de son environnement, tout en traitant les données de manière séquentielle.

Dans cette définition, l'accent est mis sur le caractère simultané de l'estimation de l'état du robot et de son environnement.

Dans (G. Bresson et al. 2017), les auteurs identifient six critères qui doivent être remplis pour qu'une approche de SLAM soit viable pour les véhicules autonomes.

1. *Précision* : l'erreur doit idéalement être inférieure à un seuil de 20 cm, elle peut être assouplie dans des situations moins dangereuses.
2. *Mise à l'échelle* : l'algorithme doit pouvoir fonctionner en temps constant et en mémoire constante, afin de permettre un déploiement sur un plus grand nombre de véhicules et une zone plus grande. La carte construite doit être suffisamment légère pour pouvoir s'étendre sur de grandes surfaces.
3. *Disponibilité* : capacité à produire immédiatement une localisation correcte, sans recourir à premier passage.
4. *Rétablissement* : capacité à se relever d'une défaillance ainsi que d'être capable de savoir où le véhicule se trouve initialement (problème du robot kidnappé).

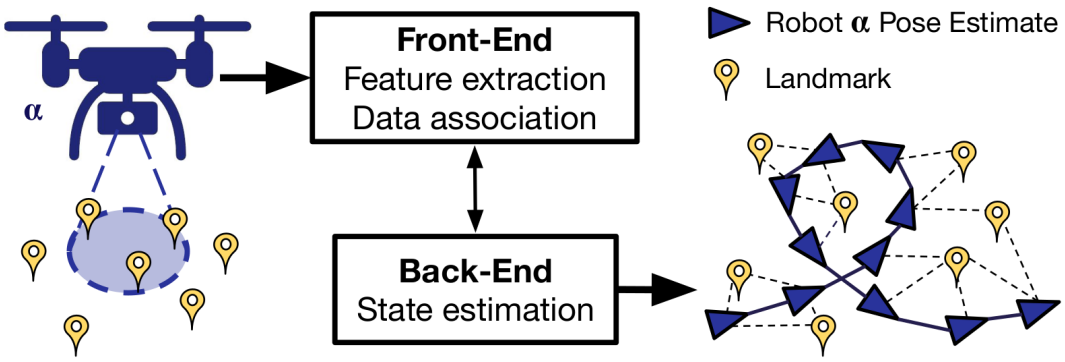


FIGURE 1.3. – Architecture du SLAM. Illustration de (LAJOIE et al. 2022).

5. *Possibilité de mise à jour* : capacité à déterminer les changements à effectuer sur la carte, notamment en gardant seulement les changements durables.
6. *Dynamicité* : capacité à gérer des environnements dynamiques (obstacles, météo, changements de saison...).

Dans (LAJOIE et al. 2022), les auteurs présentent le SLAM avec une architecture en deux parties (figure 1.3), chacune étant l'objet d'un domaine d'étude différent.

Le *front-end* est la partie en charge du traitement des données capteurs, afin d'en extraire les observations des amers. Il est aussi en charge de l'association de données. Cela relève de la recherche en traitement du signal, vision par ordinateur, alignement de nuage de points, photogrammétrie, etc. L'état de l'art relatif à cette partie sera fait au chapitre 4 lorsqu'on traitera le cas des données réelles.

Le *back-end* est la partie qui produit l'estimation d'état finale, à partir des observations fournies par le *front-end*. Cela relève de la recherche en optimisation, en filtrage (probabilités), en théorie des graphes, etc. L'état de l'art sur ces méthodes est développé dans le présent chapitre.

En pratique, la frontière peut être très floue entre les deux parties. Le *front-end* peut s'occuper de l'odométrie visuelle pendant que le *back-end* s'occupe de l'estimation jointe entre la carte et le robot.

L'objectif du SLAM est donc d'être capable d'estimer l'état du robot tout en construisant la carte. Il y a cependant deux principaux types d'approches sur la manière d'estimer l'état du robot (G. BRESSON et al. 2017).

Les premières approches, généralement basées sur du filtrage, dites d'*online SLAM*, visent à estimer l'état courant du robot, sans se soucier de corriger les estimations précédentes. Il y est fait usage de filtres Bayésiens, récursifs, propices à une exécution en temps réel (MURPHY 2002).

Les deuxièmes approches, généralement basées sur de l'optimisation, dites de *full SLAM*, visent à estimer la trajectoire complète du robot depuis son départ en considérant toutes les mesures et commandes présentes et passées. Ces approches consistent à optimiser la trajectoire complète pour satisfaire les contraintes induites par les mesures.

Les avantages et inconvénients de ces deux types de méthodes seront discutés en section 1.2.4.

1.2.1. Méthodes basées sur du filtrage

Les méthodes de SLAM basées sur du filtrage, dans le cadre de l'*online SLAM*, utilisent, lors de l'étape de correction, l'observation d'éléments existant dans la carte pour calculer une erreur et corriger l'estimation. Si les éléments observés n'existent pas dans la carte, ils y sont rajoutés.

L'utilisation de filtre pour faire du SLAM a été particulièrement exploitée entre les années 1990 et 2000 (T. Bailey et H. Durrant-Whyte 2006). Peu après, les méthodes par optimisation commencent à devenir l'un des choix le plus privilégié pour le SLAM.

Le SLAM utilisant l'EKF n'est pas récent, et des problèmes de consistance ont été relevés, notamment dans (Tim Bailey et al. 2006 ; S. Julier et J. Uhlmann 2001). Le lecteur pourra s'y référer pour la modélisation du système et le fonctionnement de l'EKF-SLAM. Le problème de la perte de consistance est structurel, il advient quoi qu'il en soit. Cela se manifeste, tout d'abord, par le gain d'information trop important, en particulier sur l'orientation, ce qui conduit à une sous-évaluation de la variance de l'orientation. Cela est causé par une variation de l'estimation (notamment à cause du bruit), et donc une Jacobienne calculée autour du mauvais point, qui entraine une baisse de l'incertitude. La perte de consistance se manifeste aussi par des discontinuités dans l'estimation de la trajectoire, dues aussi à la linéarisation autour d'une mauvaise estimation. Les deux symptômes disparaissent lorsque les Jacobiennes sont calculées autour du vrai état. Cependant, (Tim Bailey et al. 2006) concluent tout de même sur le fait que les linéarisations standards de l'EKF produisent des inconsistances qui s'accumulent lentement et restent maîtrisables tant que l'incertitude d'orientation est faible et que le filtre est robuste aux bruits d'évolution et d'observation ainsi qu'aux non-linéarités et qu'il est assisté par un bruit de stabilisation.

Une manière de limiter l'inconsistance est de diviser la carte en sous-cartes plus petites. Dans (Paz et al. 2007), un EKF-SLAM divisant la carte en sous-cartes selon un arbre binaire est présenté, appelé *Divide and Conquer EKF-SLAM*. Chaque étage de l'arbre divise la carte en deux sous-cartes de même taille jusqu'à avoir des sous-cartes suffisamment petites. Cela permet de travailler dans de petites cartes, considérées indépendantes, et ainsi limiter l'incertitude dans chaque sous-carte. Via des simulations, les auteurs ont montré que leur méthode est plus consistante qu'un EKF-SLAM standard. Cette méthode permet aussi de réduire la complexité, passant de $O(n^3)$ à $O(n^2)$ pour le full SLAM, comparé à un EKF-SLAM standard.

Dans (WooYeon Jeong et Kyoung Mu Lee 2005), les auteurs présentent un SLAM utilisant les points de Harris comme observations d'un EKF, provenant d'une caméra observant le plafond. Grâce à l'EKF, ils sont capables d'estimer une carte 3D du plafond. Le robot est ensuite localisé en associant les points de l'image à la carte de *features*. La méthode présentée propose aussi une relocalisation grâce à la transformation de Hough (Hough 1962). La relocalisation permet de déterminer la localisation du robot

de manière indépendante, ce qui est utile lors de l'initialisation ou lors d'une grande dérive du robot.

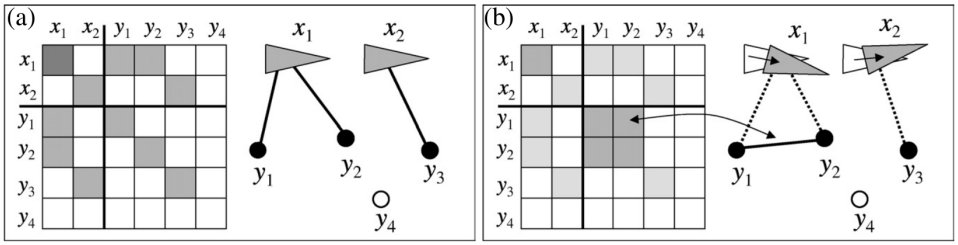
L'EIF a souvent été utilisé pour du SLAM, notamment dans (Lázaro et al. 2013), présenté plus tard dans la partie sur le SLAM collaboratif. Il est aussi utilisé dans (Sebastian Thrun, Koller et al. 2004) où l'EIF sert de point de départ pour clairsemer (*sparsify*) la matrice informationnelle, permettant de meilleures performances d'un point de vue calculatoire. Il s'agit du SEIF (*Sparse Extended Information Filter*), dont l'étape de *sparsification*, illustrée en figure 1.4b, permet de limiter les dépendances entre les amers de la carte en supprimant éléments diagonaux trop faibles. Les éléments de la carte et la pose du robot sont représentés par un champ aléatoire de Markov gaussien (*Gaussian Markov Random Field*, GMRF), qui est une représentation de l'évolution des états du filtre à travers le temps. Dans (Sebastian Thrun, Koller et al. 2004), ce GMRF est maintenu épars, c'est-à-dire qu'il y a une étape de retrait des liens entre les éléments afin de garder la matrice informationnelle creuse (contenant une majorité d'éléments nuls). Cela se traduit par la suppression des inter-covariances trop petites (remplacées par 0).

Ce processus se fait en trois étapes. Tout d'abord, la mise à jour via les observations : des liens sont ajoutés dans le GMRF entre le robot et les amers observés (figure 1.4a, à gauche). Ensuite, lors de l'étape de prédiction, les poids des liens robot-amers sont transférés vers les liens entre les amers eux-mêmes (figure 1.4a, à droite), c'est l'étape d'affaiblissement. Enfin, lorsque de nouveaux amers sont observés, les vieux liens entre le robot et les amers non observés sont retirés pour garder la matrice informationnelle creuse (figure 1.4b) et de nouveaux sont ajoutés. C'est l'étape de *sparsification*.

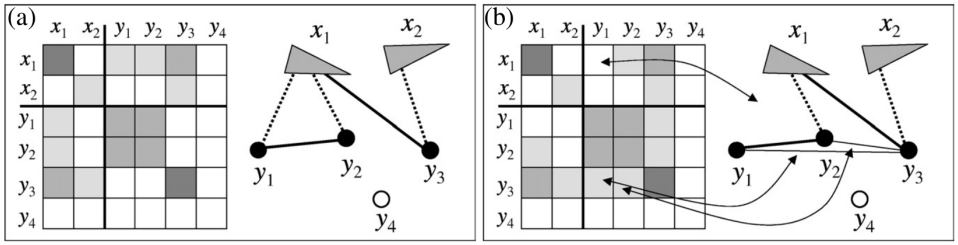
Des approches basées sur l'UKF sont utilisées en SLAM, mais, comme l'EKF, elles ne sont pas optimales d'un point de vue calculatoire (la complexité est de l'ordre du cube de la taille de la carte). (G. P. Huang, Mourikis et Stergios I. Roumeliotis 2009) propose des modifications à l'UKF pour l'adapter aux problématiques de SLAM. Il s'agit de modifier les étapes de prédiction et de correction en tirant parti du fait que seul l'état du robot change pendant la prédiction et que seulement certains amers sont concernés pendant la correction. Les auteurs n'effectuent la transformation sans parfum que sur ces composantes de l'état et non sur l'état complet. Ils proposent aussi une amélioration sur la consistance du SLAM par UKF.

Dans sa thèse, (Delobel 2018) utilise le SCI ainsi que des réseaux bayésiens afin d'estimer la localisation du véhicule ainsi que la carte d'amers tout en évitant la sur-convergence du filtre de Kalman en présence de consanguinité de données. Chaque élément est considéré indépendamment des autres, donc la localisation impliquant un seul véhicule et un seul amer ne fait pas appel au reste de la carte, ce qui allège les calculs et évite la sur-convergence.

Le filtre particulaire a été utilisé par (Gao, G. Battistelli et L. Chisci 2020), allié à un *Random Finite Set* (RFS) représentant la carte. Un RFS est un ensemble avec un nombre aléatoire (mais fini) de variables aléatoires. Une densité de probabilité d'hypothèse



(a) Étape de prédiction.



(b) Étape de *sparsification*.

FIGURE 1.4. – Principales étapes du SEIF (Sebastian THRUN, KOLLER et al. 2004) dans le cas multi-robot. Illustrations issues de (Sebastian THRUN et LIU 2005) car de meilleure qualité.

(*Probability Hypothesis Density*, PHD) est ensuite calculée sur le RFS permettant de représenter la probabilité d'observer un amer en chaque endroit. Cela permet de calculer les vraisemblances des observations pour chaque particule, et donc d'ajuster leurs poids.

Les méthodes de SLAM utilisant du filtrage présentées dans cette section offrent différentes approches et différentes solutions aux problèmes propres à l'EKF. Son principal problème est son inconsistance. Afin de limiter ce problème, il est possible de réduire la taille du problème afin de garder une consistance locale, ou d'utiliser la forme informationnelle couplée à une étape de *sparsification*. L'UKF permet de traiter les problèmes de non-linéarités, présents dès lors qu'une pose est estimée et par conséquent de limiter l'inconsistance. Le filtre à particule permet de manipuler des distributions complexes, mais il est souvent dans l'impossibilité d'avoir une estimation fiable de la covariance, nécessaire pour gérer l'intégrité de l'estimation d'état. Enfin, le SCI apporte une fusion consistante, au prix d'une perte d'optimalité.

Le filtrage permet d'estimer l'erreur, via l'estimation de la covariance, et donc possède un indice de confiance dans l'estimation. Cela ouvre la porte aux méthodes de détection de défauts, abordées plus tard dans ce chapitre, pour assurer la précision et l'intégrité de l'estimation d'état.

1.2.2. Méthodes basées sur de l'optimisation

Les méthodes de SLAM par optimisation sont divisées en deux catégories (G. BRESSON et al. 2017). Le *bundle adjustment* (BA) qui consiste à optimiser conjointement une

structure 3D, représentant les points caractéristiques (*features*) ou les amers de l'environnement, avec la pose de la caméra via une fonction objectif. Il s'agit à la base d'une méthode de *structure from motion*, c'est-à-dire de reconstruction d'un environnement 3D couplée à l'estimation des paramètres intrinsèques de la caméra. La deuxième catégorie est le *Graph SLAM* qui repose sur l'optimisation d'un graphe bayésien représentant les liens entre les différents nœuds du graphe (poses de robots et amers) via une fonction de coût, sans chercher à déterminer les caractéristiques de la caméra. Le BA est apparu en premier et a été rejoint par le *Graph SLAM* à la fin des années 2000. Les deux méthodes sont proches, car les deux reviennent à une optimisation d'une fonction de coût avec le maximum de vraisemblance. La différence entre ces deux approches semble être dans les éléments optimisés. Dans le cas du BA, la trajectoire des robots ainsi que la carte composée d'amers sont optimisés en considérant toutes les variables, y compris la calibration du capteur. Le *Graph SLAM* optimise un graphe bayésien, en minimisant les distances entre les nœuds.

Le *bundle adjustment* consiste à utiliser toutes les observations afin d'obtenir une estimation de l'environnement et de la pose des robots via l'optimisation d'une fonction (Campos et al. 2021 ; Triggs et al. 2000). L'optimisation peut se faire via la minimisation de l'erreur de reprojection pour les méthodes utilisant des *features*, ou bien par la minimisation de l'erreur photométrique entre des ensembles de pixels, pour les méthodes utilisant directement les images des caméras.

Le BA permet d'associer et d'optimiser les données sur plusieurs échelles de temps :

— À court terme, c'est-à-dire les dernières secondes, permettant notamment de faire de l'odométrie visuelle.
— À moyen terme, cela concerne les éléments proches de la caméra, mais qui aurait dérivé. Le BA permet de limiter cette dérive sans fermeture de boucle.
— À long terme, ce qui concerne les fermetures de boucle, donc de la reconnaissance de lieu, ce qui permet de réduire la dérive de manière plus importante.

Il s'agit d'un processus d'optimisation qui peut utiliser les moindres carrés non-linéaires, où l'objectif sera de minimiser une fonction de coût sous la forme d'une somme d'erreurs quadratiques. Il a été très utilisé pour des applications de photogrammétrie et d'imagerie. Le BA est une méthode éprouvée, qui donne de très bons résultats et peut être très efficiente lorsque optimisée correctement (Triggs et al. 2000). Un inconvénient est qu'il faut calculer la covariance de l'erreur en plus du processus d'optimisation pour avoir les mêmes informations qu'à l'issue d'un filtre.

Le *Graph SLAM* repose surtout sur l'optimisation d'un graphe de poses à long terme, au niveau de la fermeture de boucle. Un exemple est présenté dans (Mendes, Koch et Lacroix 2016) avec le *Pose-Graph SLAM*, une des versions les plus connues de *Graph SLAM*, qui construit un graphe bipartite où des poses sont liées entre elles par des transformations (figure 1.5). À chaque pose du graphe est associée une *keyframe*, c'est-à-dire un repère et un nuage de points servant de référence locale. Lorsqu'il y a un nouveau nuage de points lidar, ce dernier est aligné sur la dernière *keyframe* grâce à l'*Iterative Closest Point* (ICP ; Segal, Haehnel et Thrun 2009). Ainsi, la pose du

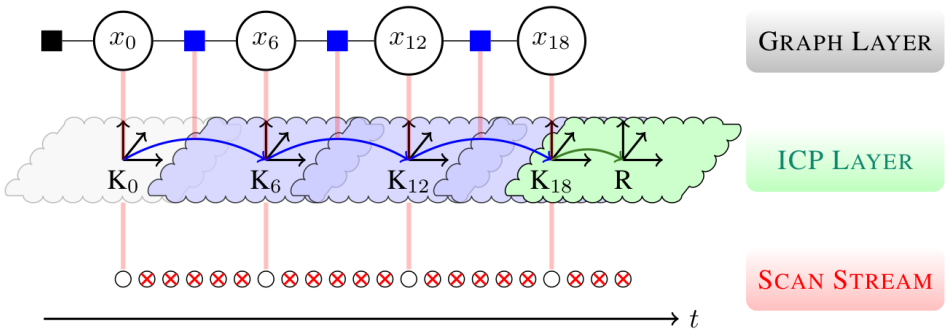


FIGURE 1.5. – Principe du *Pose-Graph SLAM*. Certains scans lidar (cercles sans croix dans la partie basse de l'illustration) sont utilisés pour créer des *keyframes* (partie intermédiaire), associées à un nœud de pose dans le *pose-graph* (partie supérieure). Le scan vert est le scan actuel, il est associé à la dernière *keyframe* (K_{18}). Illustration de (MENDES, KOCH et LACROIX 2016).

véhicule est obtenue dans le repère de la *keyframe* et une localisation peut être calculée en réalisant la transformation dans le repère global. Lorsque la proportion de chevauchement entre le nouveau nuage de points lidar et le nuage de points de la *keyframe* est trop petite (inférieure à 75 %), une nouvelle *keyframe* est créée, avec la nouvelle pose qui constitue le repère et le nouveau nuage de points qui sert de référence pour les prochains. Elle est aussi reliée à la *keyframe* précédente par la transformation obtenue via l'ICP. Tout cela repose sur la qualité de l'alignement, il n'y a pas d'optimisation de graphe pour l'instant.

Lors de la création d'une nouvelle *keyframe*, un autre système complémentaire entre en jeu : la fermeture de boucle. Les transformations de proche en proche entre les nuages de points et les *keyframes* constituent un système d'odométrie visuelle (*Visual Odometry*, VO), sujet à une dérive en position avec le temps. Pour corriger cela, des correspondances entre les nouvelles *keyframes* et les anciennes vont être recherchées, ce qui correspond à des boucles (le robot est revenu sur une zone connue) qu'il s'agira de fermer. Le nuage de points de la nouvelle *keyframe* est donc associé à des *keyframes* candidates sélectionnées pour leur proximité avec la nouvelle *keyframe*, tout en excluant les m dernières pour éviter de créer de toutes petites boucles. Si une correspondance est trouvée, une nouvelle transformation est rajoutée dans le *pose-graph* entre les deux *keyframes* associées, ce qui rajoute une contrainte. Enfin, le graphe est optimisé pour réduire la dérive du robot sur la trajectoire parcourue via la résolution d'un problème de moindres carrés non linéaire.

(DELLENBACH et al. 2022) va un peu plus loin dans la définition de la *keyframe*. Elle est construite avec tous les nuages de points, via l'agrégation dans des voxels. Le nombre de points et la densité sont limités dans les voxels pour limiter la place mémoire nécessaire (la densité maximale est d'un point tous les 10 cm). La fermeture de boucle est réalisée via une simplification de la grille en grille d'élévation, ce qui permet de rechercher plus de *keyframes* candidates.

En l'absence de fermeture de boucle, le SLAM est sujet à une dérive importante. Cela

peut nuire à la consistance de l'estimation. Dans (Guillaume Bresson, Aufrère et Chapuis 2015), les auteurs développent, entre autres, une méthode pour estimer la dérive en fonction de l'abscisse curviligne du robot, pour la prendre en compte dans la covariance et rester consistant.

Les approches de SLAM par optimisation peuvent utiliser les données brutes des capteurs, généralement denses, mais également utiliser le résultat d'un traitement des données capteurs afin d'en retirer des observations.

1.2.3. Traitement des données

Les méthodes de filtrage s'appuient sur des observations obtenues via un traitement ou des données brutes de capteurs simples. Certaines méthodes de SLAM par optimisation utilisent des données brutes (via l'alignement avec l'ICP ou la NDT). Que ce soit en filtrage ou en optimisation, des méthodes utilisent des *features*.

Dans ce manuscrit, nous parlerons de *features* lorsque ces observations sont issues de caractéristiques géométriques (points de fort contraste par exemple) dénuées de sens sémantique, tandis que les amers (*landmarks* en anglais) ont un sens (panneau, arbre, poteau, façade, etc.) et sont composés de plusieurs *features*. L'extraction de ces *features* sera traité dans le chapitre 4.

1.2.4. Comparaison entre les méthodes basées filtrage et basées optimisation

Dans (Strasdat, Montiel et Davison 2010), les auteurs comparent les deux classes de méthodes dans le cadre du SLAM utilisant des caméras monoculaires. Dans leur analyse, basée essentiellement sur le temps de calcul, les approches par optimisation s'avèrent plus efficace en termes de complexité algorithmique pour une même précision. Il est fait mention que les approches par optimisation ont une complexité proportionnelle au temps tandis que les approches par filtrage ont une complexité augmentant avec le carré de la taille de la carte.

Cependant, cet article est très orienté sur un seul type de capteur et ne prend pas tous les paramètres en compte, notamment en réalisant l'évaluation sur le SLAM monoculaire uniquement. Ce genre de SLAM utilise un grand nombre de *features* et donc possède une carte particulièrement importante. Dans les approches utilisant un nombre plus faible d'amers, l'argument de la taille de la carte est affaibli (Lázaro et al. 2013). De plus, il existe des méthodes de filtrage vues précédemment qui limitent la taille des matrices à inverser (SCI, SEIF, SKF).

Un avantage important des méthodes de filtrage est l'estimation de la covariance de l'erreur directement, ce qui rend ces méthodes plus adaptées pour la détection de défauts et pour la définition de niveaux d'incertitude utilisés pour l'étude des systèmes nécessitant un haut niveau d'intégrité et de sécurité. Les méthodes par optimisation prennent rarement en compte l'incertitude, et quand c'est le cas, ce sont des opérations supplémentaires qui sont par nature très sensibles à la convergence dans un minimum

local (par exemple, dans le cas de l'utilisation de l'inverse de la Hessienne) (Akai et al. 2017 ; Vemprala et Saripalli 2021). Ces méthodes possèdent aussi un côté heuristique dans la manière de déterminer ces incertitudes, notamment lorsqu'elles utilisent un échantillon pour calculer une incertitude empirique.

Une étude plus récente montre que les SLAM, avec un EKF invariant ou basés sur de l'optimisation, se comportent de manière similaire en termes de précision et de consistance (Y. Zhang, Teng Zhang et S. Huang 2018). L'EKF invariant pallie les problèmes de l'inconsistance de l'EKF grâce au groupe de Lie. (C. Chen et al. 2018) montre que ce sont plutôt les méthodes par optimisation qui sont plus précises et que les méthodes par filtrage sont plus efficaces d'un point de vue calculatoire. Le point faible de ces deux articles est le nombre très limité de méthodes de SLAM par filtrage et par optimisation testées.

Le choix entre les méthodes par optimisation ou par filtrage repose donc sur plusieurs critères et sur l'application envisagée. Elles ont chacune des avantages et des inconvénients. L'utilisation de l'EKF directement n'est pas une bonne solution, mais des modifications existent pour résoudre certains cas problématiques (consistance, temps de calcul, etc.). Dans le cadre d'un système devant estimer son erreur (via la covariance), les méthodes par filtrage sont particulièrement adaptées, et permettent la mise en place simple d'une étape de détection de défauts.

Dans le cadre de cette thèse, nous utilisons des méthodes basées filtrage, car ces travaux s'inscrivent dans une recherche liée à l'intégrité. Il est donc important de pouvoir estimer l'erreur du système, à la fois pour permettre une détection et un traitement des défauts, mais également pour fournir des informations de consistance pour prendre la décision d'utiliser ou non l'estimation.

1.3. Collaboration multi-robots

1.3.1. Introduction sur les systèmes multi-robots

La collaboration entre robots permet aux méthodes de SLAM de s'améliorer sur plusieurs plans. Tout d'abord, le temps nécessaire pour explorer un environnement est réduit dans le cas d'une stratégie d'exploration coordonnée. Ensuite, la précision de la localisation est améliorée via la correction des cartes entre les robots, multipliant les observations de l'environnement par plusieurs acteurs pouvant ainsi s'entre-corriger. Enfin, la collaboration apporte une meilleure confiance dans l'existence ou non d'éléments de la carte, via la multiplication des points de vue.

La différence entre la « collaboration » et la « coopération » n'est pas standardisée, et la littérature est assez ambigüe sur les termes utilisés (Vicentini 2020). Les auteurs, qui travaillent dans le champ de la sécurité en robotique, ont relevé que la « collaboration » représente l'interaction robot/robot (ou humain/robot dans le cas des robots d'usine) lorsque les deux partagent une tâche et un but, tout en ayant des interactions physiques. La « coopération » correspond à la réalisation de tâches en parallèle, mais sans contact.

La frontière reste floue dans la littérature liée à la sécurité en robotique, essentiellement appliquée aux robots industriels en contact avec l'humain. À notre connaissance, il n'y a pas de consensus sur l'utilisation de ces termes pour la robotique mobile, ces deux termes étant parfois totalement intervertis dans certains articles.

Si l'on se base sur les définitions des dictionnaires, le dictionnaire Cambridge fait la différence. Un comportement « coopératif » correspond à une situation dans laquelle une personne est prête à aider ou faire ce qu'on lui demande [1], tandis qu'un comportement « collaboratif » doit impliquer au moins deux personnes qui travaillent ensemble pour un but spécifique [2].

Au vu de ces éléments, nous utilisons les définitions suivantes.

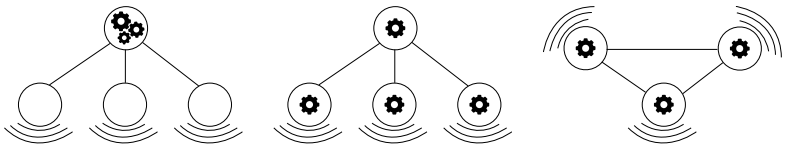
Définition 4. La coopération consiste à répartir des tâches entre les robots, qu'ils accomplissent individuellement, tout en utilisant des données fournies par d'autres robots. Ces tâches peuvent être différentes d'un robot à l'autre.

Définition 5. La collaboration entre des robots consiste à exécuter une tâche dans un but commun via le dialogue.

Les systèmes multi-robots peuvent être décrits par les caractéristiques suivantes, définies dans (Dudek et al. 1993) et rappelées dans (Lajoie et al. 2022) :

- *Taille du groupe* : du point de vue du contrôle, l'augmentation de la taille du groupe affecte l'efficacité de la tâche : le groupe devient plus dur à contrôler, mais peut subdiviser la tâche entre plus d'agents. Du point de vue de la fusion de données, cela fait plus de données, donc une meilleure estimation de l'état du système.
- *Portée de communication* : il s'agit de la capacité pour les robots à communiquer en permanence ou seulement à une certaine portée, ce qui va influencer la topologie du réseau et donc les algorithmes utilisables.
- *Topologie du réseau* : la capacité des robots à communiquer (portée, obstacles, perturbations, etc.) va définir si la topologie du réseau va être densément connectée ou faiblement connectée.
- *Bande passante* : c'est la quantité de données pouvant être échangée au même moment. Cela affecte grandement les approches utilisables en cas de bande passante limitée.
- *Reconfiguration du groupe* : c'est la capacité des robots à se réagencer les uns par rapport aux autres. Cela ouvre des possibilités intéressantes, notamment pour modifier la topologie du réseau, mais également pour redistribuer des tâches.
- *Capacité de calcul par robot* : cela peut affecter les architectures possibles, notamment si les robots ne sont pas capables d'effectuer toutes les tâches nécessaires.
- *Nature du groupe* : le groupe peut être homogène ou hétérogène, c'est-à-dire, respectivement, que tous les robots sont identiques, ou qu'il y a des différences de capteurs, de capacités de calcul ou de communication, etc.

1. `https://dictionary.cambridge.org/dictionary/english/cooperative`
2. `https://dictionary.cambridge.org/dictionary/english/collaborative`



(a) Centralisée. (b) Distribuée. (c) Décentralisée.

FIGURE 1.6. – Architectures de fusion.

Dans notre travail, nous avons choisi de considérer la collaboration, dans le but d'améliorer une carte de l'environnement avec tous les robots. Un premier but de chaque robot est de se localiser lui-même sans se soucier de la localisation des autres robots. La tâche de localisation peut être qualifiée ici de coopérative. Avec la coopération, les robots s'aident mutuellement et l'ensemble des robots est mieux localisé. L'objectif reste ainsi la localisation de chaque robot individuellement, l'amélioration de l'ensemble des robots étant un effet secondaire positif. Le système complet est collaboratif, car un second objectif est d'affiner la carte avec l'aide de tous les robots, via la fusion de leurs estimations de manière régulière lorsqu'ils échangent des données. Une amélioration de la carte améliore ensuite la localisation des robots.

1.3.2. Architectures de fusion

La fusion de données multi-capteurs peut être réalisée suivant plusieurs architectures. Il existe plusieurs définitions, notamment sur les architectures distribuées, utilisées dans la recherche en réseau ou en informatique. Dans (LAJOIE et al. 2022), la notion d'architecture réseau centralisée/décentralisée est séparée de la notion de partage de la charge de calcul. Pour eux, un système avec des robots partageant la charge de calcul avec une unité centrale est un système à la fois centralisé (car il y a une unité centrale) et distribué (car les robots partagent la charge de calcul). Le problème est qu'un système entièrement décentralisé (sans unité centrale) réparti nécessairement la charge de calcul entre les robots, ce qui le rend distribué suivant cette définition. Nous adopterons plutôt ici la terminologie utilisée dans (Hugh DURRANT-WHYTE 2002), dont les définitions sont présentées ci-après.

Définition 6: ARCHITECTURE CENTRALISÉE. Un système de fusion de données centralisé (figure 1.6a) est constitué d'un réseau de nœuds qui ne font qu'envoyer les données brutes à un centre de fusion unique. Ils ne disposent pas de capacité de calcul et peuvent donc être peu gourmand en ressources. Cependant, tout le système dépend du bon fonctionnement du centre de fusion et du système de communication.

Définition 7: ARCHITECTURE DISTRIBUÉE. Un système de fusion de données distribué (figure 1.6b) est constitué d'un réseau de nœuds avec des capacités de calcul propre, mais un centre de fusion unique produit la fusion finale. Les capacités de calcul des nœuds servent donc à décharger le nœud central en extrayant l'information utile des données brutes, ou en réalisant une partie de la fusion : le calcul est distribué sur les nœuds. La charge réseau est donc réduite par rapport à un système centralisé. Le réseau peut aussi

être hiérarchique, avec des nœuds de traitement intermédiaires entre les nœuds et le centre de fusion unique.

Définition 8: Architecture décentralisée. Un système de fusion de données décentralisé (figure 1.6c) est constitué d'un réseau de nœuds, chacun avec sa propre capacité de calcul, et ne nécessitant pas de fusion centralisée ou de communication centralisée. Chaque nœud fusionne localement des données en se basant sur les informations locales et sur celles communiquées par les autres nœuds. Par conséquent, cette architecture ne comporte pas de centre de fusion et aucun nœud n'est absolument nécessaire à la réussite des opérations du groupe. De même, les communications sont uniquement en pair-à-pair (nœud à nœud) et les nœuds ne disposent pas de connaissance sur la topologie du réseau et ne communiquent souvent qu'avec leurs voisins.

Ces différentes architectures privilégient soit la résilience du système en limitant les points de défaillance uniques (*Single Point of Failure*, SPoF) pour les architectures décentralisées, soit une certaine simplicité et l'accès à des ressources calculatoires quasi illimitées permettant de meilleures performances dans le cas des architectures centralisées. Les architectures décentralisées permettent un fonctionnement du système plus longtemps en cas de défaillance d'un de ses composants (fonctionnement potentiellement dégradé) et évitent l'échange permanent avec une unité centrale. Elles répartissent la charge de calcul sur les différents nœuds, qui devient plus facilement quantifiable, comparé à une charge de calcul centralisée qui peut avoir de grandes variations en fonction de la situation.

Une architecture décentralisée pose quatre challenges d'après (Dubois et al. 2020) :

1. La communication peut présenter plusieurs problèmes comme la perte de contact, la synchronisation des horloges, la mise à l'échelle du système à un grand nombre de robots, etc.
2. L'échange d'information peut monopoliser les ressources calculatoires des robots, ce qui empêcherait la bonne exécution de leurs tâches principales.
3. En l'absence de nœud central pour fixer un repère global, il faut estimer les transformations entre chaque robot. Il n'y a pas de « chef » permettant de prendre une décision simplement, ce qui est à la fois un avantage et un inconvénient.
4. La « consanguinité de donnée » peut être présente, et se traduit par une boucle d'information, et donc l'utilisation d'une même information plusieurs fois.

Dans le cas des méthodes basées filtrage, la consanguinité est un problème important car elle peut engendrer une sur-convergence du filtre. Certains filtres y sont immunisés, comme le CI ou le SCI, mais ce n'est pas le cas de ceux établis à partir du filtre de Kalman (EKF, EIF).

Pour répondre à ce problème, le *channel filter* (que l'on pourrait traduire par « filtre par canal ») est présenté dans (Bourgault et Hugh F Durrant-Whyte 2004 ; Hugh Durrant-Whyte 2002). Il s'agit de comptabiliser l'information échangée entre deux robots sur un canal de communication, afin de quantifier l'information commune entre les deux robots et la soustraire lorsqu'on recevra de nouvelles informations de la part de

cet autre robot. Cela fonctionne parfaitement lorsque le réseau de communication est soit complet (l'information est partagée en permanence, toute l'information est connue), soit en arbre (l'information passe forcément par un unique chemin entre deux nœuds). Le problème se pose lorsque la topologie du réseau est changeante ou lorsqu'il y a des cycles dans le graphe de communication. En effet, l'information peut utiliser différents chemins, rendant le *channel filter* incapable de connaître l'information commune avec l'autre nœud. Dans ce cas, une couche intermédiaire doit être ajoutée pour définir un arbre couvrant, pour revenir à une topologie d'arbre.

1.3.3. Localisation coopérative et collaborative

Avant de traiter le SLAM collaboratif, dans cette section, nous allons nous focaliser sur la localisation coopérative et collaborative. Il s'agit pour un ensemble de robots de se localiser avec l'aide des autres, en utilisant des observations inter-robot.

Dans (Kurazume, Nagata et Hirose 1994), le groupe de robots est divisé en deux sous-groupes. Pendant qu'un groupe avance, l'autre reste statique, puis les rôles s'échangent, de façon collaborative. Cela permet d'utiliser le groupe stationnaire comme point de repère fixe.

(Howard, Matark et Sukhatme 2002) utilise aussi les robots comme amers pour qu'ils se repèrent entre eux, mais sans groupe stationnaire. La méthode, utilisant une architecture centralisée, emploie l'estimateur du maximum de vraisemblance (*Maximum Likelihood Estimator*), et les trajectoires et observations des robots sont représentées dans un graphe. La localisation collaborative est obtenue via une optimisation du graphe pour maximiser la vraisemblance des estimations.

Dans (S. Roumeliotis et Bekey 2002), une méthode distribuée pour la localisation coopérative d'un groupe de robots, via un EKF, est proposée. Elle repose sur l'estimation locale de la localisation de chaque robot, et par l'échange des observations lors des phases de correction, où les robots observent des inter-distances entre eux. Les inter-covariances sont décomposées pour être estimées pendant la prédiction, mais doivent être reconstituées lors de la phase de correction, induisant une communication avec tous les robots.

(Al Hage, E. El Najjar et Pomorski 2017) propose une architecture de fusion distribuée, tolérante aux défauts, utilisant l'EIF. Chaque robot estime localement sa localisation grâce à ses capteurs. Lorsque des observations inter-robots sont effectuées, un filtre central fusionne les contributions informationnelles provenant de chaque robot de manière coopérative, avec une étape de détection et d'exclusion de défauts.

(Mu et al. 2011) localisent une flotte de robots de manière coopérative. Certains sont équipés de GNSS, ce qui leur permet de corriger leur dérive d'odométrie. Tous les robots sont capables de mesurer les distances avec les autres ou avec des amers et peuvent communiquer avec l'ensemble de la flotte. Ils obtiennent des résultats équivalents à un EKF, mais avec du délai lié aux latences de communication.

Pour palier le problème de l'EIF qui rend impossible l'accès à la covariance d'un robot sans connaître l'état complet, les auteurs utilisent la décomposition de Cholesky. Elle permet de déterminer l'état de chaque robot en établissant une chaine d'inversion, où chaque robot va inverser une partie de la matrice, puis la transmettre au suivant dans la chaîne. Il faut donc que l'ordre des robots dans l'état du système soit le même pour tous.

Dans (Panzieri, Pascucci et Setola 2006), les auteurs utilisent un EKF entrelacé (*Interlaced EKF*, IEKF) afin d'estimer la position de plusieurs robots de manière coopérative pouvant observer des amers bien localisés et mesurer des distances inter-robots. L'IEKF consiste à utiliser plusieurs EKF en parallèle, travaillant sur des parties différentes de l'état. Chaque robot estime sa propre localisation avec un EKF local, tout en utilisant les estimations que les autres robots (et donc les autres EKF) fournissent. Il en va de même avec les observations des distances inter-robots, où la pose estimée de l'autre robot est prise en compte ainsi que sa covariance. Il n'y a cependant aucune mémoire sur les corrélations créées lors de l'intégration de ces observations, ce qui détériore la qualité de l'estimation.

Dans (Luft et al. 2016), le SKF est utilisé afin de localiser plusieurs robots par rapport à des amers fixes connus, mais aussi en utilisant des mesures de distances inter-robots, coopérativement. Contrairement à l'IEKF utilisé par (Panzieri, Pascucci et Setola 2006) qui néglige les inter-corrélations, les corrélations entre les robots sont prises en compte. Les auteurs ont imposé la contrainte de limiter le nombre de communications aux seuls moments où les robots peuvent mesurer une inter-distance entre eux. Ainsi, les robots ne peuvent pas partager en permanence leur état. Les états des autres robots ne sont pas connus, mais leurs inter-covariances continuent d'être estimées, via leur décomposition après chaque coopération.

(Carrillo-Arce et al. 2013) utilise un CI pour fusionner des observations inter-robot de manière coopérative. Grâce au CI, il n'est pas nécessaire d'estimer les inter-covariances entre les robots, mais cela entraine une fusion pessimiste. La complexité de la fusion est linéaire avec le nombre de robots, $O(n)$, ce qui la rend facilement utilisable sur du matériel limité en calcul.

Dans (Z. Zhu et al. 2022), les robots se localisent de manière coopérative avec la force du signal des communications reçues (*Received Signal Strength Indicator*, RSSI). Le RSSI est converti via un modèle pour déterminer la distance de l'émetteur, et ainsi constituer une observation d'inter-distance. Les auteurs utilisent de l'optimisation pour définir ce modèle, permettant de déterminer pour un RSSI mesuré, la distance correspondante. Cette optimisation est précédée d'un RANSAC (*Random Sampling Consensus*) afin d'enlever des valeurs aberrantes et robustifier le modèle. Les distances sont ensuite utilisées dans un estimateur de maximum de vraisemblance qui permet de localiser les robots. En outre, il n'est pas fait mention de la présence d'obstacles (murs, objets, etc.) qui pourraient dégrader la force du signal sans augmenter la distance.

1.3.4. Localisation et cartographie collaboratives

Après avoir présenté des méthodes de localisation collaborative, cette section se focalise sur les méthodes de SLAM collaboratif (C-SLAM), qui permettent de constituer et de maintenir à jour une représentation de l'environnement. La différence entre les approches avec ou sans carte à priori sera soulignée.

Les approches de SLAM avec une carte à priori sont peu nombreuses. Il y a d'un côté les méthodes qui font uniquement de la localisation dans une carte, sans la modifier (donc ce n'est pas du SLAM), et de l'autre, les méthodes de SLAM qui construisent leur carte sans à priori. Les approches entre ces deux types de méthodes sont rares.

Dans (Ouyang et al. 2021), les auteurs ont développé une méthode de SLAM visuel et collaboratif suivant une architecture distribuée. Chaque robot opère un système de SLAM local utilisant du *bundle adjustment* (BA) permettant de mettre à jour une carte locale, constituée de *keyframes* et d'amers. La collaboration s'effectue uniquement via l'envoi de cette carte à un serveur central qui va la fusionner avec la sienne via une optimisation par graphe, puis les modifications sont renvoyées aux robots. Une optimisation a été faite pour n'envoyer aux robots que les amers vus par leur caméra afin de limiter la bande passante utilisée.

Une autre méthode distribuée est présentée dans (Tianjun Zhang et al. 2022), avec un SLAM visuel permettant la construction d'une carte dense sans utilisation de capteurs de profondeur (RGB-D ou lidar). Chaque robot exécute en local un algorithme d'odométrie visuelle. Toute la partie d'optimisation de graphe pour les fermetures de boucles est déportée sur un serveur central. Ce serveur reçoit les données brutes venant des robots, puis construit un graphe de *keyframes* propre à chacun, avant d'effectuer la fermeture de boucle à la fois intra-robot, mais aussi inter-robots. La collaboration s'effectue donc à la fois sur la carte, mais aussi indirectement via les observations puisque toutes les données sont prises en compte.

(Vemprala et Saripalli 2021) propose une architecture décentralisée qui utilise du BA intégré à un filtre de Kalman pour localiser une flotte de micro-robots aériens tout en construisant sa carte et en l'affinant. La collaboration s'effectue via la fusion des cartes de tous les robots, mais également par l'estimation de la position relative des robots entre eux par la mise en correspondance des *features* communes. Il s'agit donc d'observations indirectes de la pose d'autres robots. Cependant, cette méthode ne prend pas en compte l'incertitude dans la carte et la covariance de l'estimation est calculée via l'inversion de la matrice Hessienne utilisée pour l'optimisation. Ce procédé d'estimation de la covariance est sujet à une sous-estimation de l'erreur en cas de convergence dans un minimum local. Il est donc heuristique et constitue une estimation peu fiable de l'erreur. Le KF est utilisé pour fusionner les observations fournies par le BA et pour rejeter des défauts. Afin de ne pas avoir à gérer les corrélations entre chaque robot, un CI est utilisé afin d'intégrer les observations inter-robots.

Dans (Dong et al. 2015), un système de SLAM collaboratif décentralisé est présenté. Plusieurs robots construisent d'abord leur carte locale, et lorsqu'ils se rencontrent,

seules les cartes sont échangées puis associées pour trouver la transformation entre les repères locaux de chaque robot. Les robots sont localisés les uns par rapport aux autres uniquement via la fusion de leurs cartes. Les robots ne connaissent pas leur position initiale et ils déterminent leur trajectoire les uns par rapport aux autres lors des étapes de fermeture de boucle.

Le SEIF a été utilisé pour du SLAM dans (Sebastian Thrun, Koller et al. 2004) puis adapté en collaboratif dans (Sebastian Thrun et Liu 2005). Chaque robot opère indépendamment, construit sa propre carte, sans connaître sa position initiale. Les robots communiquent en permanence entre eux pour essayer de trouver des correspondances entre les cartes. La fusion, basée sur l'EIF, est réalisée si elle satisfait un critère portant sur la vraisemblance et le nombre d'amers à fusionner. Une fois que les amers ont été transformés dans le repère du robot qui opère la fusion, l'opération est simple et repose sur l'additivité de l'EIF. La collaboration ne s'effectue donc que par la mise en correspondance des cartes, ce qui influe sur la localisation des robots sans échanger les observations.

(Cieslewski, Choudhary et Scaramuzza 2017) propose un SLAM visuel décentralisé qui utilise un réseau de neurones, NetVLAD, afin de produire une représentation compacte des images. Ces images compactes sont ensuite utilisées dans un algorithme de reconnaissance de lieu (*place recognition*) qui permet de trouver des correspondances avec les cartes des autres robots. En parallèle, un algorithme d'odométrie visuelle construit la carte et se localise par rapport à elle. La collaboration utilise les cartes des autres robots pour se localiser dedans, mais ne fusionne pas les cartes. Cela se revient à envoyer ses observations à l'autre robot pour se localiser dans sa carte.

Pour terminer, (Lázaro et al. 2013) présente un algorithme décentralisé de SLAM utilisant une carte à priori constituée de *features* positionnées avec incertitude. Les robots fonctionnent en formation. Un EIF est utilisé pour représenter les robots et la carte, qui ne sera pas augmentée. Afin d'éviter les problèmes de consanguinité de données, les communications des robots sont synchronisées. De plus, la carte est découpée en zones, et lorsqu'un robot passe d'une zone à l'autre, il partage ses dernières observations avec les robots de la zone précédente, puis se synchronise avec les robots de la zone suivante. Les échanges entre les robots se composent des contributions informationnelles des observations plutôt que des mesures brutes. La collaboration s'effectue uniquement via l'échange des contributions informationnelles, les cartes ne sont pas fusionnées autrement.

1.3.5. Limites actuelles du SLAM collaboratif

Une des principales limites du SLAM collaboratif est l'adaptation aux environnements dynamiques (Lajoie et al. 2022), certains éléments de la carte peuvent changer (voiture garée partant de sa place, arbre taillé, etc.). Ensuite, ce SLAM collaboratif est généralement très dépendant des communications, certaines méthodes reposant uniquement dessus. Enfin, le problème de la complexité du calcul est à considérer lors de l'utilisation de robots qui possèdent des ressources calculatoires limitées.

Un axe de recherche encore largement ouvert concerne le SLAM actif, c'est-à-dire qu'il va choisir des zones à explorer ou à affiner. De même, la représentation de la carte n'est pas encore fixée dans le domaine et dépend de l'application. L'utilisation d'objets sémantiques est envisagée, permettant d'alléger la carte et d'améliorer la compréhension de l'environnement.

Dans notre travail, nous considérons une méthode collaborative pour la localisation et la mise à jour de carte suivant une architecture décentralisée. La collaboration, dans notre cas, se réalise d'une façon indirecte à partir des observations des amers communs dans l'environnement. Ces mesures sont utilisées pour améliorer la localisation de chaque robot mais aussi pour mettre à jour la carte d'une façon collaborative. En effet, la carte à priori peut avoir des imprécisions qui doivent être correctement gérées. Notre architecture sera bien adaptée pour gérer les problèmes de communication sans la nécessité d'une dépendance avec une unité centrale.

1.4. Localisation tolérante aux défauts

L'idée de diagnostic de défauts est apparue depuis très longtemps dans les systèmes d'informations, notamment lors du développement des ordinateurs embarqués dans des satellites ou des fusées (Avižienis 1967 ; Von Neumann 1956). Elle reposait surtout sur de la redondance matérielle. Le but de ce diagnostic est d'éviter la propagation des erreurs dans le système, pouvant avoir des conséquences désastreuses dans le cas des fusées. Le domaine de la fusion de données utilise des mécanismes de diagnostic, notamment depuis le *Receiver Autonomous Integrity Monitoring* (RAIM), utilisé pour l'estimation de position par GNSS (Brown 1992). De nos jours, différentes méthodes de diagnostic existent (Ding 2013) et vont être discutées dans cette section. Un accent particulier sera mis sur la tolérance aux défauts capteurs dans le but d'améliorer la précision et l'intégrité de localisation.

1.4.1. Introduction au diagnostic

Le diagnostic par redondance matérielle compare la sortie de plusieurs composants identiques réalisant la même tâche. Un défaut est détecté si la sortie d'un des composants est différente de tous les autres. Le principal avantage de cette méthode est qu'elle est très fiable et que le défaut est directement isolable, il suffit de ne pas prendre en compte la sortie du composant défectueux. Cependant, cela implique la multiplication du matériel, et donc un coût plus élevé. Cette méthode est réservée aux composants clés d'un système critique.

Ensuite, il existe le diagnostic par traitement du signal. Il s'agit de mesurer des caractéristiques du signal (amplitude, valeurs limites, moments statistiques, analyse fréquentielle, etc.) et de les comparer aux signaux précédents.

Enfin, le diagnostic par redondance analytique, repose sur la correspondance des mesures à un modèle du système, réduisant ainsi le coût matériel et calculatoire lié à

cette étape. Cette méthode sera utilisée dans notre cas en se basant sur les modèles fournis par le filtre.

Le diagnostic utilise plusieurs termes qu'il convient de définir (Al Hage 2016 ; Ding 2013).

Définition 9. Un résidu est une différence entre les variables mesurées et leur estimation. La métrique utilisée pour mesurer cette différence peut varier en fonction du système et de l'application.

Définition 10. La détection de défauts dans un système consiste à observer la présence d'*au moins un* défaut qui conduit à un comportement indésirable.

Définition 11. L'isolation de défauts consiste à localiser la source du défaut. Il peut aussi s'agir de plusieurs défauts simultanés, dans ce cas, il faudra les localiser individuellement.

Il existe plusieurs méthodes pour isoler des défauts (Ragot 2021). Une première méthode, l'isolation séquentielle, repose sur le calcul des résidus en retirant séquentiellement des variables. Il faut que les défauts soient indépendants les uns des autres. Une deuxième méthode consiste à examiner de manière exhaustive toutes les combinaisons de variables pour déterminer les variables à utiliser, celles sans défauts. Cette méthode a un coût calculatoire élevé. Enfin, une troisième méthode repose sur la matrice de signatures. Il s'agit de déterminer les résidus sensibles à certains défauts. La matrice de signatures est une table contenant n lignes représentant les résidus et m colonnes représentant les défauts à isoler. La signature d'un défaut est un vecteur binaire composée de « 0 » et de « 1 », avec un « 1 » lorsque le résidu est sensible au défaut, et un « 0 » dans l'autre cas. Afin d'isoler le défaut du système, les résidus sont comparés à cette matrice de signatures après une étape de seuillage.

Il existe plusieurs catégories de matrices de signatures :

- Les matrices de signatures non isolante sont les matrices dont plusieurs colonnes sont identiques, car plusieurs défauts partagent la même signature.
- Les matrices de signatures faiblement isolante permettent d'isoler correctement les défauts à moins qu'un résidu soit altéré, de 1 à 0. Dans ce cas, la matrice de signatures deviennent non isolante.
- Les matrices de signatures fortement isolante contiennent des signatures de défauts suffisamment différentes pour que l'altération d'une valeur ne conduit pas à une signature déjà existante.

Définition 12. L'identification de défaut désigne le fait de connaître le type du défaut, de déterminer son amplitude, et sa cause. Elle peut aussi être appelée *analyse* du défaut.

Ces différentes étapes sont combinées dans les systèmes de tolérance aux défauts, qui portent les acronymes suivants :

- FD : *Fault Detection*, uniquement la détection de la présence d'un défaut.
- FDI : *Fault Detection and Isolation*, spécifie en plus d'où viennent les défauts.

— FDII : *Fault Detection, Isolation and Identification*, spécifie en plus les caractéristiques des défauts, proposant la possibilité de les compenser. Ce type de système est aussi appelé FDIA pour *Fault Detection, Isolation and Analysis* (Ding 2013).

Une fois que les défauts ont été isolés, il est possible d'agir pour éviter qu'ils ne dégradent les performances du système (Goelles, Schlager et Muckenhuber 2020). Ces méthodes peuvent être désignées par l'acronyme FDIR ou FDIIR, avec le R signifiant *recovery*. Nous l'appellerons ici « récupération ». Il peut s'agir de réduire l'impact de la mesure défectueuse sans toutefois la supprimer, de corriger l'état du système en estimant, par exemple, l'amplitude du défaut, ou bien à exclure la mesure défaillante du système. Lorsqu'il n'y a que de l'exclusion, le système est qualifié de détection et exclusion de défauts (FDE pour *Fault Detection and Exclusion*) et repose sur la présence d'autres mesures redondantes pour maintenir le bon fonctionnement du système.

Les défauts peuvent être classés en trois catégories (Ding 2013, p. 28) :

1. Les défauts provenant des capteurs, c'est-à-dire leur partie mécanique et électronique, et non l'analyse qu'il en est fait. Un état de l'art orienté vers ce genre de tolérance aux défauts est présenté dans (Goelles, Schlager et Muckenhuber 2020), qui se focalise sur la localisation des défauts au niveau des composants et des sous-composants.
2. Les défauts provenant des processus. Il s'agit de défauts relevant d'un problème au niveau du traitement des données des capteurs, de l'environnement ou de la prise de décision.
3. Les défauts sur les actionneurs, c'est-à-dire sur l'application défectueuse des commandes envoyées.

Dans cette thèse, nous nous focaliserons sur les défauts processus provenant des capteurs. Dans notre cas, la définition de défaut est donc la suivante :

Définition 13: Défaut. Un défaut correspond à un biais de mesure ou une dérive qui peut avoir son origine du processus de traitement, de l'environnement ou d'un problème matériel au niveau du capteur. La définition du défaut sera encore étendue à un biais au niveau du positionnement des amers dans la carte.

Dans le cas des systèmes collaboratifs multi-robots, il convient de faire la distinction entre les différentes méthodes de détection de défauts (Graham Miller et Gandhi 2021). Il y a tout d'abord les méthodes permettant de détecter des défauts locaux au niveau de chaque robot, appelées endogènes. Puis, les méthodes qui permettent de détecter des défauts de manière externe, provenant des autres robots, appelées exogènes. Dans un système multi-robots, un défaut provenant d'un robot peut donc être détecté par un autre et partagé aux autres, ce qui permet une robustification du système global.

1.4.2. Applications à la localisation dans la littérature

Cette section aborde la tolérance aux défauts capteurs dans la littérature, dans le cadre de la localisation de robots mobiles.

Un algorithme très utilisé avec le GNSS est le RAIM. Il s'agit d'une méthode de vérification de l'intégrité d'un système de navigation GNSS via l'utilisation de signaux redondants pour détecter la présence de défauts. Le RAIM peut se réaliser (1) en comparant des pseudo-distances, (2) en se basant sur des résidus de moindre carrés ou (3) en projetant dans l'espace de parité (Brown 1992).

Dans (S. Roumeliotis, Sukhatme et Bekey 1998), une redondance logicielle est réalisée via l'utilisation d'un banc de quatre KF. Les filtres fonctionnent par paire (groupe de deux). Chaque filtre de la paire possède les mêmes équations cinématiques, mais ces équations diffèrent d'une paire à l'autre afin d'être capable de détecter un problème venant d'un changement dans les équations du modèle du robot. Ensuite, les deux filtres de la paire ont des paramètres de réglage différents. Chaque filtre produit un résidu, ils sont comparés et analysés dans un module de détection et d'isolation de défauts.

(Kellalib et al. 2021) utilise aussi un banc d'EKF pour du SLAM. Deux lidars et trois méthodes d'odométrie (gyroscope, codeurs et GNSS d'intérieur) sont utilisés dans des combinaisons différentes, de manière redondante. Il est ainsi possible d'isoler un défaut provenant d'un des capteurs ou d'un des filtres en comparant les sorties des différents EKF entre elles grâce à une matrice de signatures.

(Al Hage, E. El Najjar et Pomorski 2017) présente une méthode de détection et d'exclusion de défauts collaborative, utilisant un banc d'EIF et se reposant sur l'additivité du filtre informationnel pour calculer l'apport de chaque observation et ainsi isoler les défauts. Les résidus sont calculés grâce à la divergence de Kullback-Leibler (*Kullback-Leibler Divergence*, KLD), qui permet de comparer deux distributions plus finement qu'avec la seule distance de Mahalanobis.

Dans (Hamadi et al. 2022), une architecture résiliente aux défauts est présentée. Elle repose sur l'utilisation de deux EKF qui utilisent des données venant de capteurs différents dont les sorties sont introduites dans un système de vote. Ce système diffère des précédents utilisant un banc de filtre car il pondère toutes les sorties en fonction de leurs distances 2 à 2, suivant une combinaison de rampes pour un passage progressif de 1 (utilisée pleinement) à 0 (exclue).

(Marchand et al. 2008) présente une évaluation d'algorithmes d'intégrité appliquée à la localisation par GNSS pour des véhicules routiers dans un milieu urbain. La méthode de FDE utilisée combine plusieurs capteurs (GNSS, vitesse des roues, inertie, caméras, lidar, etc.) dans un filtre de Kalman afin de calculer des résidus avec l'innovation des observations des pseudo-distances.

1.5. Conclusion et positionnement de notre approche par rapport à la littérature

La localisation des robots avec une bonne précision et une bonne intégrité est toujours problématique dans les environnements complexes. La collaboration a une place centrale dans notre approche qui va permettre de tirer profit des différentes informations dans le système. Au niveau de l'estimation d'état, la collaboration s'effectue simultanément sur la localisation des robots, via le partage d'observations, et sur la cartographie, via la fusion des cartes. Dans la littérature, il a été observé qu'une grande partie du C-SLAM collabore seulement via la fusion des cartes et non via les observations. La collaboration via les observations est utilisée régulièrement dans la localisation collaborative, mais la carte, lorsqu'elle est présente, est généralement connue parfaitement. L'objectif ici est d'améliorer une carte existante avec les observations de tous les robots, tout en estimant la localisation de ces derniers. Nous utiliserons donc le filtre de Schmidt-Kalman (SKF) pour estimer la pose des robots et la carte de manière collaborative. Les observations d'amer en communs seront utilisées pour l'étape de correction, permettant une observation indirecte inter-robots. Ce type d'observation permet d'augmenter la portée de collaboration sans avoir besoin d'une ligne de vue directe entre les robots. De même, ces informations seront directement exploitables pour la mise à jour de la carte.

Notre approche repose sur l'utilisation d'un SKF suivant une architecture décentralisée afin de limiter les points de défaillance uniques, tout en évitant la mise en place d'un serveur de calcul central. Avec notre approche, les robots ne communiquent que lorsqu'ils observent un même amer, limitant ainsi les communications au minimum. De même, les robots peuvent continuer à fonctionner en cas de problèmes de communication. L'approche par filtrage permet d'évaluer au même moment la covariance de l'erreur, nécessaire pour l'étude de l'intégrité du système.

D'autre part, des défauts peuvent apparaître au niveau des mesures capteurs, une étape de détection et d'isolation de défauts est alors ajoutée afin d'améliorer la précision et l'intégrité de l'estimation d'état. La collaboration entre les robots est essentielle pour assurer une redondance nécessaire pour différencier entre des défauts d'observations et des défauts de la carte qui se caractérisent par des biais au niveau du positionnement initial de certains amers. Une étape de récupération est ensuite ajoutée afin d'assurer la continuité du fonctionnement du système.

2. Localisation et mise à jour de carte collaborative, décentralisée et tolérante aux défauts

Sommaire

2.1. Introduction

La localisation en robotique mobile est nécessaire pour beaucoup de tâches comme la planification, le contrôle d'exécution de même que la perception aidée par la carte. Il n'est pas toujours possible d'utiliser des systèmes de localisation par satellites (*Global Navigation Satellite System*, GNSS) tels que le GPS (*Global Positioning System*) américain,

le système européen Galileo, le système russe Glonass ou le système chinois Beidou. En effet, en intérieur, dans des zones couvertes (parkings par exemple), dans les tunnels, sous des arbres, la disponibilité et la précision du GNSS se détériorent et les performances ne répondent plus aux exigences requises pour les robots autonomes. L'absence de réception des signaux GNSS (en intérieur, sous terre, sous mer, etc.) posent de nombreux défis pour la localisation de robots.

Une alternative est d'utiliser des capteurs extéroceptifs (caméras, lidars, etc.) qui fournissent des mesures sur des éléments de l'environnement. Une bonne cartographie de l'environnement pour obtenir de faibles erreurs de localisation nécessite l'utilisation de matériel spécifique, généralement onéreux. Dans de nombreux environnements, il existe des cartes, mais leur précision n'est en général pas suffisante pour répondre aux besoins de localisation. Le SLAM (*Simultaneous Localization And Mapping*) avec une carte à priori répond à cette problématique en modifiant la carte au fur et à mesure pour améliorer sa précision et réduire les erreurs.

Lorsque des robots autonomes sont dans les mêmes zones d'évolution et qu'ils ont des capacités de communication, la problématique peut être abordée sous un angle différent. La localisation coopérative permet à un ensemble de robots de se localiser en s'entraidant, afin que chacun puisse obtenir une meilleure localisation via l'échange de données profitables à tous. Le SLAM collaboratif permet d'améliorer les estimations des poses et des cartes grâce à l'échange de données entre les robots. Ces deux tâches sont très liées, car l'amélioration de la carte profite à la localisation et inversement.

Le SLAM collaboratif peut se faire entre des robots qui se perçoivent entre eux directement, et qui mesurent donc leurs poses relatives, ou bien qui perçoivent en commun des amers de la carte. C'est cette dernière classe de problème qui nous intéresse particulièrement dans ces travaux. Ainsi, des robots peuvent collaborer dès l'instant qu'ils perçoivent en commun un élément de la carte, même s'ils sont loin les uns des autres et hors de portée directe des capteurs. Ces mesures indirectes permettent, au même moment, d'améliorer la carte et la localisation des deux robots.

Plusieurs architectures sont possibles pour le SLAM collaboratif. L'utilisation d'architectures centralisées, comme discuté en section 1.3.2, présentent plusieurs inconvénients. Le nœud central, qui effectue tous les calculs, constitue un point de défaillance unique (SPoF), rendant tous les robots dépendants de son bon fonctionnement. Une solution est de répliquer ce nœud en divers endroits du réseau, diluant ainsi ce problème. Cependant, pour qu'une architecture centralisée fonctionne, il faut que tous les robots communiquent en permanence avec le nœud central ou les nœuds centraux. Or, maintenir un réseau connexe entre tous les robots n'est pas toujours possible, que ce soit par la défaillance d'antennes relais, le dysfonctionnement du matériel de communication embarqué dans chaque robot, ou la portée limitée.

Ce chapitre s'attache à proposer d'abord une méthode permettant de localiser coopérativement des robots terrestres dans un environnement en utilisant une carte initiale imprécise. La localisation et la carte de chaque agent seront ensuite simultanément améliorées en s'appuyant sur la collaboration entre les robots et l'échange de cartes. La méthode proposée n'utilise pas les signaux GNSS pour couvrir les cas où ces signaux ne

seraient pas disponibles ou peu fiable. Cependant, elle pourra aisément être adaptée pour fusionner ces signaux-ci.

Afin de supprimer les SPoF liés à une architecture centralisée, nous avons plutôt choisi d'utiliser une architecture décentralisée qui permet une localisation continue des robots, même en l'absence de communication (basculement en mode isolé, ou *standalone*). Chaque robot peut communiquer avec les robots dans un certain rayon, ou échanger des informations en pair-à-pair.

Les mesures des capteurs peuvent parfois être erronées, ce qui affecte la précision et l'intégrité de la localisation et de la carte. Une étape de détection et d'isolation de défauts est donc réalisée, suivie de la récupération d'un état sans défaut.

Le filtre utilisé dans ces travaux pour la localisation et la mise à jour de la carte est le filtre de Schmidt-Kalman (SKF). Il est bien adapté à une architecture décentralisée avec une limitation des communications. Dans un premier temps, le principe du SKF est présenté, puis nous poserons le problème dans le cas d'une architecture centralisée pour exposer les modèles et le fonctionnement qu'aurait un filtre de Kalman étendu (EKF). Ensuite, les contraintes de l'architecture décentralisée seront décrites, puis le SKF sera exposé avec les adaptations mises en place. Enfin, le système de FDIR (détection et isolation de défauts, puis récupération) pour la tolérance aux défauts sera décrit.

2.2. Filtre de Schmidt-Kalman

Avant d'exposer notre approche sur l'utilisation du filtre de Schmidt-Kalman (SKF), aussi connu sous le nom de *consider filter*, cette section présente le fonctionnement de ce filtre peu connu dans la communauté robotique.

À l'origine, le but du SKF est de considérer l'effet des paramètres inconnus au niveau de l'état ou au niveau des mesures comme les biais des capteurs (Schmidt 1966). C'est-à-dire qu'il permet d'estimer qu'une partie de l'état, en prenant en compte l'effet de la partie non estimée. Il est fréquemment utilisé pour obtenir un gain de performance lorsque l'état est partiellement observable ou s'adapter à l'absence de disponibilité de certaines parties de l'état dont il faut estimer l'effet. Dans le cas présent, il permet de n'estimer qu'une partie des robots sans avoir à communiquer avec les autres.

L'état du système est ainsi divisé en deux parties. La première partie est estimée, elle est notée s. La deuxième partie n'est pas estimée, mais son impact sur le calcul de la covariance est pris en compte. C'est la partie des paramètres (ou *consider parameter*), notée p. L'état est représenté ainsi :

$$X = \begin{pmatrix} X_s \\ X_p \end{pmatrix}, \tag{2.1}$$

et la covariance associée :

$$P = \begin{pmatrix} P_{ss} & P_{sp} \\ P_{sp}^T & P_{pp} \end{pmatrix}. \tag{2.2}$$

L'intérêt du filtre de Schmidt-Kalman est que cette division de l'état n'est pas statique. Elle peut être différente à chaque étape, tout en gardant un état consistant. En effet, nous verrons plus tard que l'estimation de l'état complet, bien que non optimale, reste consistante.

2.2.1. Étape de prédiction

L'étape de prédiction ressemble en tout point à celle d'un EKF, où le modèle d'évolution f ne s'applique que sur une partie de l'état, la partie s. La partie p est considérée statique. Par conséquent, l'étape de prédiction est donnée par :

$$\begin{pmatrix} X_s \\ X_p \end{pmatrix}_{k+1|k} = \begin{pmatrix} f\left(X_{s,k|k}, u_k\right) \\ X_{p,k|k} \end{pmatrix}, \tag{2.3}$$

avec u_k le vecteur d'entrée pour le modèle d'évolution.

La matrice de covariance est prédite comme suit :

$$\begin{aligned} P_{k+1|k} = &\begin{pmatrix} F_{s,k} & \mathbb{0} \\ \mathbb{0} & I \end{pmatrix} \begin{pmatrix} P_{ss} & P_{sp} \\ P_{sp}^T & P_{pp} \end{pmatrix}_{k|k} \begin{pmatrix} F_{s,k} & \mathbb{0} \\ \mathbb{0} & I \end{pmatrix}^T \\ &+ \begin{pmatrix} G_{s,k} \\ \mathbb{0} \end{pmatrix} Q_u \begin{pmatrix} G_{s,k}^T & \mathbb{0} \end{pmatrix} + \begin{pmatrix} Q_s & \mathbb{0} \\ \mathbb{0} & Q_p \end{pmatrix}, \end{aligned} \tag{2.4}$$

où Q_s est la matrice de covariance du bruit de modèle d'évolution, et

$$F_{s,k} = \left.\frac{\partial f}{\partial X}\right|_{X_{s,k|k}}, \tag{2.5}$$

$$G_{s,k} = \left.\frac{\partial f}{\partial u}\right|_{u_k}. \tag{2.6}$$

En fonction du problème traité, Q_p, la covariance du bruit du modèle sur p, peut être nulle ou pas.

2.2.2. Étape de correction

L'étape de correction est un peu plus complexe. Afin de dériver les équations du SKF, les équations du filtre de Kalman étendu sont utilisées (Zanetti et D'Souza 2013). Par soucis de lisibilité, nous noterons temporairement l'état et la covariance prédite sans indice (X et P) et l'état et la covariance corrigée de la manière suivante : X^+ et P^+. Les équations de correction de l'EKF sont les suivantes, après développement de la forme de Joseph qui est valable quelque soit le gain K :

$$X^+ = X + K\left(Z - h\left(X\right)\right), \tag{2.7}$$

$$P^+ = P - KHP - PH^TK^T + KSK^T, \tag{2.8}$$

avec :

$$S = HPH^T + R. \tag{2.9}$$

En appliquant la décomposition en s et p, la matrice H s'exprime de la manière suivante :

$$H = \begin{pmatrix} H_s & H_p \end{pmatrix}, \tag{2.10}$$

où H_s correspond à la Jacobienne du modèle d'observation h par rapport à X_s et H_p par rapport à X_p. De même,

$$K = \begin{pmatrix} K_s \\ K_p \end{pmatrix}. \tag{2.11}$$

En utilisant la notation partitionnée, la correction de la covariance devient :

$$P_{ss}^+ = P_{ss} - [K_s H_s P_{ss} + K_s H_p P_{ps}] - \left[P_{ss} H_s^T K_s^T + P_{sp} H_p^T K_s^T\right] + \left[K_s S K_s^T\right], \tag{2.12}$$

$$P_{sp}^+ = (P_{ps}^+)^T = P_{sp} - [K_s H_s P_{sp} + K_s H_p P_{pp}] - \left[P_{ss} H_s^T K_p^T + P_{sp} H_p^T K_p^T\right] + \left[K_s S K_p^T\right], \tag{2.13}$$

$$P_{pp}^+ = P_{pp} - [K_p H_s P_{sp} + K_p H_p P_{pp}] - \left[P_{ps} H_s^T K_p^T + P_{pp} H_p^T K_p^T\right] + \left[K_p S K_p^T\right]. \tag{2.14}$$

Ces équations sont valides pour n'importe quel choix de K_s et K_p.

Dans un EKF, le gain optimal K_{opt} pour l'ensemble de l'état serait :

$$K_{opt} = PH^T S^{-1}, \tag{2.15}$$

lui aussi décomposable :

$$K_{opt} = \begin{pmatrix} K_{s,opt} \\ K_{p,opt} \end{pmatrix} = \begin{pmatrix} P_{ss} H_s^T + P_{sp} H_p^T \\ P_{sp}^T H_s^T + P_{pp} H_p^T \end{pmatrix} S^{-1}, \tag{2.16}$$

avec :

$$S = \begin{pmatrix} H_s & H_p \end{pmatrix} \begin{pmatrix} P_{ss} & P_{sp} \\ P_{sp}^T & P_{pp} \end{pmatrix} \begin{pmatrix} H_s^T \\ H_p^T \end{pmatrix} + R. \tag{2.17}$$

En remplaçant K_s par sa valeur optimale :

$$K_s = K_{s,opt} = \left(P_{ss} H_s^T + P_{sp} H_p^T\right) S^{-1}, \tag{2.18}$$

et en conservant K_p quelconque, les équations s'écrivent après calcul :

$$P_{ss}^{+} = P_{ss} - K_{s,opt} S K_{s,opt}^{T}, \tag{2.19}$$

$$P_{sp}^{+} = P_{sp} - K_{s,opt} H \begin{pmatrix} P_{sp} \\ P_{pp} \end{pmatrix}, \tag{2.20}$$

$$P_{pp}^{+} = P_{pp} - K_p H \begin{pmatrix} P_{sp} \\ P_{pp} \end{pmatrix} - \begin{pmatrix} P_{ps} & P_{pp} \end{pmatrix} H^T K_p^T + K_p S K_p^T. \tag{2.21}$$

Les corrections des covariances concernant la partie à estimer s (équations (2.19) et (2.20)) sont indépendantes de K_p. En d'autres termes, la valeur de K_p n'a pas d'influence sur les corrections de P_{ss} et P_{sp}. Cette propriété va nous simplifier le choix de K_p.

La correction de l'état peut s'écrire :

$$X^{+} = \begin{pmatrix} X_s \\ X_p \end{pmatrix} + \begin{pmatrix} K_s \\ K_p \end{pmatrix} (Z - h(X)). \tag{2.22}$$

Avec le filtre de Schmidt-Kalman, la partie X_p n'est pas mise à jour, ce qui peut être réalisé en choisissant $K_p = \mathbb{0}$. Ce choix est possible car P_{ss} et P_{sp} sont indépendants de K_p.

Avec $K_p = \mathbb{0}$, on obtient la correction de P_{pp} suivante :

$$P_{pp}^{+} = P_{pp} \tag{2.23}$$

La correction de l'état devient quant à elle :

$$X^{+} = \begin{pmatrix} X_s \\ X_p \end{pmatrix} + \begin{pmatrix} K_{s,opt} \\ \mathbb{0} \end{pmatrix} (Z - h(X)). \tag{2.24}$$

Le filtre de Schmidt-Kalman permet donc l'estimation d'une partie de l'état, tout en restant consistant. L'optimalité du filtre est uniquement sur la partie estimée s, il est donc moins optimal vis-à-vis de l'état global comparé au filtre de Kalman étendu.

Le SKF présenté ici a été, en partie, adapté à la forme informationnelle dans (Ke, Wu et Stergios I Roumeliotis 2019 ; Wu et Stergios I Roumeliotis 2016). Cependant, cette adaptation n'est pas suffisante pour notre problème.

2.3. Formalisation du problème d'estimation avec une architecture centralisée

Nous nous plaçons ici dans le cas de l'utilisation d'un EKF centralisé auquel chaque robot envoie ses observations brutes. On rappelle que ceci correspond à une architecture optimale du point de vue de la fusion de données.

L'objectif est donc de localiser en deux dimensions un nombre fixe de robots ainsi que de corriger les poses des amers constituant l'environnement (figure 2.1). Chaque amer possède un identifiant (ID) unique et les amers sont considérés discernables. Les observations

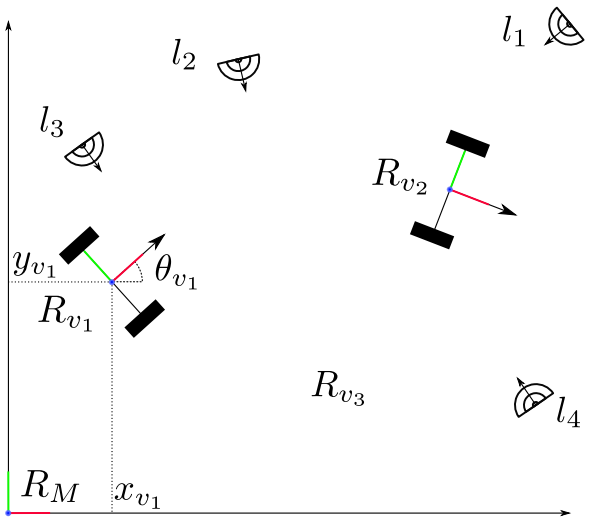


FIGURE 2.1. – Illustration des différents repères. Un repère global R_M est défini dans lequel se situent des amers fixes (l_1 à l_4) et des robots (v_1 et v_2). Les robots définissent leurs propres repères R_{v_1} et R_{v_2}.

spécifient l'amer observé. Ces amers sont initialement localisés avec incertitude. Le problème d'ajout et de suppression des amers n'est pas traité ici, ils sont donc en nombre fixe. Chaque robot est aussi identifié de manière unique.

2.3.1. Modélisation

L'état de l'EKF est constitué de la pose 2D des robots et des amers. Les robots sont représentés par leur position, (x_v, y_v), dans le repère global R_M et par leur orientation, θ_v, qui est l'angle entre l'axe longitudinal du robot et l'axe x de R_M (figure 2.1). La pose est donc $X_v = (x, y, \theta)_v^T$. L'ensemble des robots est noté V. Les amers sont considérés orientés et aussi représentés par une pose $X_l = (x, y, \theta)_l^T$ dans le repère R_M. L'ensemble des amers constituent la carte, notée L.

L'état X est constitué des poses des robots $X_V = (X_v)_{v \in V}$ et des amers $X_L = (X_l)_{l \in L}$:

$$X = \begin{pmatrix} X_V \\ X_L \end{pmatrix}. \tag{2.25}$$

La covariance estimée de l'erreur de cet état est donc décomposée en quatre parties :

$$P = \begin{pmatrix} P_V & P_{VL} \\ P_{VL}^T & P_L \end{pmatrix}. \tag{2.26}$$

Modèle d'évolution

Les robots sont représentés par un modèle unicycle (figure 2.2). Ils peuvent donc avancer de manière linéaire suivant l'axe longitudinal (x) et tourner sur eux-mêmes

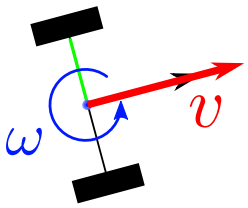


FIGURE 2.2. – Modèle unicycle utilisé pour représenter les robots.

suivant l'axe vertical z.

Pour un robot i à l'instant k, les mesures d'odométries composent le vecteur d'entrée $u_{i,k}$. Elles se présentent sous la forme de la vitesse longitudinale $v_{i,k}$ et de la vitesse angulaire $\omega_{i,k}$ du repère mobile R_v par rapport au repère de la carte R_M, réunies dans le vecteur $u_{i,k} = \left[\begin{array}{cc} v & \omega \end{array} \right]_{i,k}^T$ (figure 2.2). Ces mesures sont soumises à un bruit de mesure $\gamma_{i,k}$, supposé blanc centré gaussien, de covariance Q_u. On a $u_{i,k} = \bar{u}_{i,k} + \gamma_{i,k}$ où $\bar{u}_{i,k}$ est le vecteur d'entrée réel. Le modèle d'évolution du système est le suivant :

$$\left\{ \begin{array}{l} x_{i,k+1} = x_{i,k} + v_{i,k} dt \cos\left(\theta_{i,k} + \omega_{i,k} dt/2\right) + \alpha_{x,i,k} \\ y_{i,k+1} = y_{i,k} + v_{i,k} dt \sin\left(\theta_{i,k} + \omega_{i,k} dt/2\right) + \alpha_{y,i,k} \\ \theta_{i,k+1} = \theta_{i,k} + \omega_{i,k} dt + \alpha_{\theta,i,k} \end{array} \right. \tag{2.27}$$

$$\iff X_{i,k+1} = f(X_{i,k}, u_{i,k}) + \alpha_{i,k} \tag{2.28}$$

où dt est la période et $\alpha_{i,k}$ le bruit du modèle, supposé blanc centré gaussien avec une matrice de covariance Q.

Modèle d'observation

Chaque robot v_i définit un repère R_{v_i} dans lequel sont exprimées ses observations des amers. Il s'agit donc de mesures de pose relative entre le robot observant et l'amer observé (figure 2.3). Le vecteur d'observation, pour la version centralisée, est constitué de toutes les observations qui seront envoyées à l'unité centrale. La collaboration apparaît lorsque les robots observent des amers en commun, ce qui constitue ainsi des observations indirectes des poses relatives des robots. Il n'y a pas de traitement spécifique pour réaliser la collaboration.

On note ${}^{v_i}Z_{l_j}^{v_i}$ l'observation de l'amer l_j (en indice) par le robot v_i (en exposant à droite), dans le repère du robot R_{v_i} (en exposant à gauche). Afin de simplifier l'écriture, le repère des observations n'est pas noté lorsque l'observation est exprimée dans le repère du robot l'ayant effectuée. On l'écrit simplement $Z_{l_j}^{v_i}$ dans ce cas.

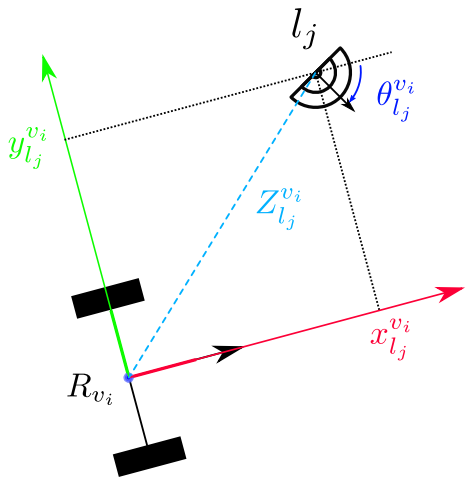


FIGURE 2.3. – Observation de la pose relative de l'amer l_j par le robot v_i, exprimée dans le repère du robot, R_{v_i}.

Le modèle d'observation est noté $h_{l_j}^{v_i}$, et s'exprime ainsi :

$$Z_{l_j}^{v_i} = \begin{pmatrix} x_{l_j}^{v_i} \\ y_{l_j}^{v_i} \\ \theta_{l_j}^{v_i} \end{pmatrix} + \beta_{l_j}^{v_i}, \tag{2.29}$$

$$Z_{l_j}^{v_i} = h(X_{v_i}, X_{l_j}) + \beta_{l_j}^{v_i} = h_{l_j}^{v_i}(X) + \beta_{l_j}^{v_i}, \tag{2.30}$$

$$Z_{l_j}^{v_i} = \underbrace{\begin{pmatrix} \cos\theta_{v_i} & \sin\theta_{v_i} & 0 \\ -\sin\theta_{v_i} & \cos\theta_{v_i} & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{\mathcal{R}_{v_i}^T} \begin{pmatrix} x_{l_j} - x_{v_i} \\ y_{l_j} - y_{v_i} \\ \theta_{l_j} - \theta_{v_i} \end{pmatrix} + \beta_{l_j}^{v_i}, \tag{2.31}$$

$$Z_{l_j}^{v_i} = \mathcal{R}_{v_i}^T (X_{l_j} - X_{v_i}) + \beta_{l_j}^{v_i}, \tag{2.32}$$

avec $\beta_{l_j}^{v_i}$ le bruit d'observation, supposé blanc centré gaussien, de covariance $R_{l_j}^{v_i}$. $\mathcal{R}_{v_i}$ correspond à la matrice de rotation entre le repère du robot R_{v_i} et la carte R_M. Elle devrait être écrite ${}^M\mathcal{R}_{v_i}$ mais pour des raisons de clarté, l'exposant M est retiré quand cette matrice de rotation est exprimée dans le repère de travail. Il en est de même pour les poses (X_{v_i} et X_{l_j}).

Le vecteur d'observation complet Z, utilisé par le filtre central à un moment donné, est la concaténation de toutes les observations $Z_{l_j}^{v_i}$ reçues d'une façon synchrone au même instant.

2.3.2. Étape de prédiction

L'étape de prédiction permet de prédire l'état, donc ici la pose des robots et des amers. Dans notre cas, elle est cadencée sur les mesures d'odométrie. La prédiction est donc calculée à chaque mesure reçue, indépendamment des observations, avec le modèle

d'évolution :

$$X_{i,k+1|k} = f(X_{i,k|k}, u_{i,k}) \tag{2.33}$$

$$\iff \begin{cases} x_{i,k+1|k} = x_{i,k|k} + v_{i,k}dt\cos\left(\theta_{i,k|k} + \omega_{i,k}dt/2\right) \\ y_{i,k+1|k} = y_{i,k|k} + v_{i,k}dt\sin\left(\theta_{i,k|k} + \omega_{i,k}dt/2\right) \\ \theta_{i,k+1|k} = \theta_{i,k|k} + \omega_{i,k}dt \end{cases} \tag{2.34}$$

La linéarisation du modèle d'évolution pour la prédiction de la covariance implique la définition des Jacobiennes suivantes :

$$F_{i,k} = \left.\frac{\partial f}{\partial X}\right|_{(X_{i,k|k}, u_{i,k})}, \tag{2.35}$$

$$F_{i,k} = \begin{pmatrix} 1 & 0 & -v_{i,k}dt\sin\left(\theta_{k|k} + \omega_{i,k}dt/2\right) \\ 0 & 1 & v_{i,k}dt\cos\left(\theta_{k|k} + \omega_{i,k}dt/2\right) \\ 0 & 0 & 1 \end{pmatrix}, \tag{2.36}$$

et

$$G_{i,k} = \left.\frac{\partial f}{\partial u}\right|_{(X_{i,k|k}, u_{i,k})}, \tag{2.37}$$

$$G_{i,k} = \begin{pmatrix} dt\cos\left(\theta_{i,k|k} + \omega_{i,k}dt/2\right) & -v_{i,k}\frac{dt^2}{2}\sin\left(\theta_{i,k|k} + \omega_{i,k}dt/2\right) \\ dt\sin\left(\theta_{i,k|k} + \omega_{i,k}dt/2\right) & v_{i,k}\frac{dt^2}{2}\cos\left(\theta_{i,k|k} + \omega_{i,k}dt/2\right) \\ 0 & dt \end{pmatrix}. \tag{2.38}$$

Ainsi, l'étape de prédiction se fait de la manière suivante, pour chaque robot i :

$$X_{i,k+1|k} = f(X_{i,k|k}, u_{i,k}) \tag{2.39}$$

$$P_{i,k+1|k} = F_{i,k}P_{i,k|k}F_{i,k}^T + G_{i,k}Q_uG_{i,k}^T + Q. \tag{2.40}$$

Les matrices de covariances Q et Q_u sont considérées constantes dans le temps et identiques pour tous les robots, et ne sont donc pas suffixées de i et de k.

Les amers ne bougent pas au cours du temps, la prédiction peut se faire d'une façon simple pour un amer l :

$$X_{l,k+1|k} = X_{l,k|k} \tag{2.41}$$

$$P_{l,k+1|k} = P_{l,k|k} \tag{2.42}$$

La prédiction des inter-covariances se fait naturellement

$$P_{ij,k+1|k} = F_{i,k}P_{ij,k|k}F_{j,k}^T \tag{2.43}$$

où $F_{i,k}$ (respectivement $F_{j,k}$) est remplacée par l'identité I si i (respectivement j) représente un amer.

2.3.3. Étape de correction

L'étape de correction permet de corriger la prédiction, via l'utilisation d'observations extéroceptives (provenant de caméras ou de lidars par exemple), et ainsi réduire la dérive engendrée par l'odométrie seule. Cette étape est effectuée à chaque réception d'une observation. L'observation est prédite avec la fonction d'observation :

$$Z_{l_j}^{v_i} = h_{l_j}^{v_i}(X_{k|k-1}) \tag{2.44}$$

$$\iff Z_{l_j}^{v_i} = \mathcal{R}_{v_i,k|k-1}^{T}(X_{l_j,k|k-1} - X_{v_i,k|k-1}). \tag{2.45}$$

Linéarisation

La linéarisation du modèle d'observation implique la définition de la matrice Jacobienne

$$H = \left.\frac{\partial h}{\partial X}\right|_{X_{k|k-1}}. \tag{2.46}$$

Cette matrice a une forme plus complexe que les Jacobiennes du modèle d'évolution. Tout d'abord, chaque ligne de H (ou ligne-bloc ici, de par le fait que les observations sont des vecteurs de trois variables) correspond à une observation, et chaque robot est capable d'observer plusieurs amers différents en même temps. La matrice H est décomposée en deux parties, une première partie (H_V) correspondant aux dérivées partielles sur la partie X_V de l'état, et une deuxième partie (H_L) correspondant aux dérivées partielles sur la carte X_L :

$$H = \begin{pmatrix} H_V & H_L \end{pmatrix}. \tag{2.47}$$

Pour une observation $Z_{l_j}^{v_i}$, à partir de l'équation (2.45), les éléments non nuls de la Jacobienne correspondent aux colonnes-bloc relatives à la pose du robot observateur X_{v_i} et à la pose de l'amer observé X_{l_j}. La Jacobienne, dans le cas d'une seule observation $Z_{l_j}^{v_i}$, peut s'écrire :

$$H = \begin{pmatrix} (H_V)_{l_j}^{v_i} & (H_L)_{l_j}^{v_i} \end{pmatrix}. \tag{2.48}$$

En développant l'équation (2.45) pour chaque composante de l'observation, on obtient, dans le repère de v_i :

$$Z_x = (x_{l_j} - x_{v_i})\cos\theta_{v_i} + (y_{l_j} - y_{v_i})\sin\theta_{v_i}, \tag{2.49}$$

$$Z_y = -(x_{l_j} - x_{v_i})\sin\theta_{v_i} + (y_{l_j} - y_{v_i})\cos\theta_{v_i}, \tag{2.50}$$

$$Z_\theta = \theta_{l_j} - \theta_{v_i}. \tag{2.51}$$

L'expression de la Jacobienne $(H_V)_{l_j}^{v_i}$ peut se factoriser :

$$(H_V)_{l_j}^{v_i} = \left.\frac{\partial h}{\partial X_{v_i}}\right|_{X_{k|k-1}} = \begin{pmatrix} -\cos\theta_{v_i} & -\sin\theta_{v_i} & -(x_{l_j}-x_{v_i})\sin\theta_{v_i} + (y_{l_j}-y_{v_i})\cos\theta_{v_i} \\ \sin\theta_{v_i} & -\cos\theta_{v_i} & (x_{l_j}-x_{v_i})\cos\theta_{v_i} - (y_{l_j}-y_{v_i})\sin\theta_{v_i} \\ 0 & 0 & -1 \end{pmatrix}, \tag{2.52}$$

$$= \underbrace{\begin{pmatrix} \cos\theta_{v_i} & \sin\theta_{v_i} & 0 \\ -\sin\theta_{v_i} & \cos\theta_{v_i} & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{\mathcal{R}_{v_i}^T} \begin{pmatrix} -1 & 0 & y_{l_j}-y_{v_i} \\ 0 & -1 & -(x_{l_j}-x_{v_i}) \\ 0 & 0 & -1 \end{pmatrix}. \tag{2.53}$$

Ce qui donne l'expression compacte :

$$(H_V)_{l_j}^{v_i} = \begin{cases} \mathcal{R}_{v_i}^T \Gamma_{l_j}^{v_i} & \text{si la colonne-bloc correspond à } v_i, \\ \mathbb{0}_{3\times 3} & \text{sinon,} \end{cases} \tag{2.54}$$

avec :

$$\Gamma_{l_j}^{v_i} = \begin{pmatrix} -1 & 0 & y_{l_j}-y_{v_i} \\ 0 & -1 & -(x_{l_j}-x_{v_i}) \\ 0 & 0 & -1 \end{pmatrix}. \tag{2.55}$$

Pour la partie des amers, on obtient :

$$(H_L)_{l_j}^{v_i} = \begin{cases} \mathcal{R}_{v_i}^T & \text{si la colonne-bloc correspond à } l_j, \\ \mathbb{0}_{3\times 3} & \text{sinon.} \end{cases} \tag{2.56}$$

Dans la suite, nous utiliserons $\mathbb{0}$ pour représenter des blocs de 3×3 composés de 0.

Exemple Soit trois robots v_1, v_2 et v_3 et deux amers l_1 et l_2. Si le robot v_1 observe les amers l_1 et l_2, et que le robot v_2 observe l'amer l_2, alors la matrice Jacobienne H à cet instant k est :

$$H_k = \begin{array}{c} \begin{matrix} v_1 & v_2 & v_3 & l_1 & l_2 \end{matrix} \\ \begin{pmatrix} \mathcal{R}_{v_1}^T \Gamma_{v_1}^{l_1} & \mathbb{0} & \mathbb{0} & \mathcal{R}_{v_1}^T & \mathbb{0} \\ \mathcal{R}_{v_1}^T \Gamma_{v_1}^{l_2} & \mathbb{0} & \mathbb{0} & \mathbb{0} & \mathcal{R}_{v_1}^T \\ \mathbb{0} & \mathcal{R}_{v_2}^T \Gamma_{v_2}^{l_2} & \mathbb{0} & \mathbb{0} & \mathcal{R}_{v_2}^T \end{pmatrix} \end{array} \begin{matrix} Z_{l_1}^{v_1} \\ Z_{l_2}^{v_1} \\ Z_{l_2}^{v_2} \end{matrix} \tag{2.57}$$

La colonne-bloc composée uniquement de $\mathbb{0}$ correspond au troisième robot v_3 qui n'a rien observé.

Équations de correction

Avec l'utilisation de la forme de Joseph pour des raisons de stabilité numérique, les équations de correction de l'EKF sont :

$$X_{k|k} = X_{k|k-1} + K_k \left(Z_k - h\left(X_{k|k-1} \right) \right), \tag{2.58}$$

$$P_{k|k} = \left(I - K_k H_k \right) P_{k|k-1} \left(I - K_k H_k \right)^T + K_k R K_k^T, \tag{2.59}$$

où K_k est le gain de Kalman défini à chaque instant k par :

$$K_k = P_{k|k-1} H_k^T \underbrace{(H_k P_{k|k-1} H_k^T + R)}_{S_k}{}^{-1}. \tag{2.60}$$

La matrice de covariance R est en général supposée être bloc diagonale et composée des covariances de chaque observation, notées $R_{l_j}^{v_i}$.

2.4. Adaptation à une architecture décentralisée

L'utilisation d'une architecture décentralisée implique l'absence d'un nœud de fusion central. Les robots échangent alors entre eux leurs observations et leurs états, grâce à des communications qui peuvent être intensives. Cette section présente la modélisation suivant une telle architecture, en accordant une importance majeure à la consistance des estimations tout en limitant les communications au maximum. Pour cela, le SKF sera appliqué à notre cas d'étude.

2.4.1. Phases de collaboration et conditions de communication

Avant de passer à l'estimation d'état, intéressons-nous à la phase de collaboration. Dans un premier temps, nous allons faire l'hypothèse que les observations des robots sont synchrones : elles sont réalisées au même instant et les communications sont sans délais.

L'une des contraintes principales de l'approche proposée est de limiter les communications au strict nécessaire, afin d'éviter de saturer un réseau de communication et de limiter la bande-passante utilisée. Les robots ne communiquent avec d'autres que lorsqu'ils sont en collaboration. Dans notre cas, la collaboration survient lorsque deux robots observent au moins un amer en commun, ce qui constitue une observation indirecte qui lie les deux robots.

Afin qu'ils puissent déterminer avec quels autres robots ils sont en collaboration, chaque robot diffuse les ID des amers observés à tous les autres situés à portée de communication (figure 2.4), tout en écoutant les émissions des autres robots. Ces messages sont très légers et nécessitent très peu de bande passante. Cela permet aux robots de déterminer les observations qu'ils vont utiliser.

Si un autre robot à portée a observé un amer en commun, alors la collaboration est enclenchée. Les robots échangent alors les informations nécessaires à la fusion collaborative via une connexion pair-à-pair (observations, état et covariance, carte comprise). Les

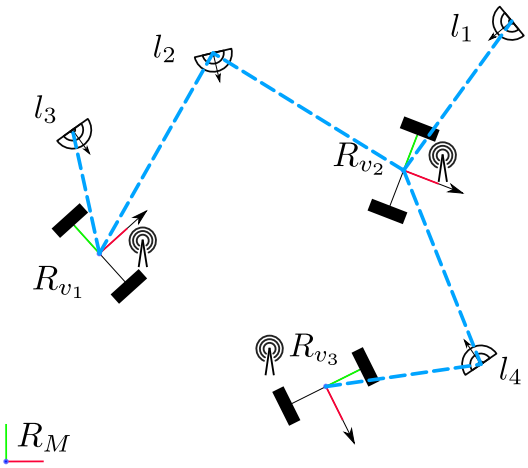


FIGURE 2.4. – Les robots observent des amers (traits bleus) et émettent à leur entourage les identifiants des amers observés. On suppose ici pour simplifier que les capteurs des robots sont synchronisés.

Algorithme 1 Mise en contact de robots collaborant avec v_i

`liste_obs` ← Acquérir les observations avec les capteurs propres.
`ids` ← IDs de `liste_obs`.
Envoyer `ids` aux robots voisins par diffusion (*broadcast*).
`ids_reçus` ← IDs envoyés par les voisins.
$V_i \leftarrow \{v_i\}$
pour `id` envoyé par v_j **dans** `id_reçus` **faire**
  si `id` **dans** `ids` **alors**
    $V_i \leftarrow V_i \cup \{v_j\}$
  fin si
fin pour
Établir connexion pair-à-pair avec les robots dans $V_i \setminus v_i$, c'est-à-dire les robots collaborant avec v_i mais sans v_i pour l'échange de données.

robots collaborant avec l'ego-robot v_i sont regroupés dans un ensemble V_i, avec $v_i \in V_i$. Ce procédé de mise en contact est présenté par l'algorithme 1. Après le bref échange d'informations, les robots ne communiquent plus jusqu'à la prochaine collaboration.

En dehors des phases de collaboration, les robots utilisent leurs propres observations pour estimer leurs poses et celle des amers dans la carte. Ils sont alors isolés.

2.4.2. Filtre de Schmidt-Kalman pour la collaboration

Le SKF est particulièrement bien adapté pour les problématiques de décentralisation, car il permet de n'estimer que la partie de l'état correspondante aux robots en collaboration (ESCOURROU, AL HAGE et BONNIFAIT 2022 ; LUFT et al. 2016).

Modélisation

Le vecteur d'état de l'équation (2.25) est réarrangé en deux parties comme suit :

$$X = \begin{pmatrix} X_s \\ X_p \end{pmatrix}, \tag{2.61}$$

avec :

— s est la partie qui comprend tous les éléments de la carte L, le robot v_i et les robots collaborant avec l'ego-robot v_i :

$$X_s = \begin{pmatrix} \left(X_{v_j}\right)_{v_j \in V_i} \\ \left(X_{l_k}\right)_{l_k \in L} \end{pmatrix}; \tag{2.62}$$

— p est la partie qui comprend les robots ne collaborant pas avec v_i, c'est-à-dire les robots n'appartenant pas à V_i :

$$X_p = \left(X_{v_j}\right)_{v_j \notin V_i}. \tag{2.63}$$

La figure 2.5 donne une illustration.

Il est à noter que le choix de mettre toute la carte dans s a été fait pour simplifier l'implémentation de la méthode. Dans la partie s, il est tout à fait possible de ne garder de la carte que les amers observés par v_i. De plus, initialement, le nombre de robots est inconnu, l'état sera donc étendu au fur et à mesure de la collaboration avec d'autres robots. La méthode utilisée pour augmenter la taille de l'état sera détaillée plus bas (section 2.4.3).

Étape de prédiction

L'étape de prédiction est la même que pour l'EKF centralisé, suivant les équations (2.39) et (2.40). La partie s correspond à l'ego-robot v_i ainsi qu'à toute la carte. La partie p contient les autres robots. Le modèle d'évolution n'utilise que le vecteur d'entrée concernant le robot v_i. En effet, les données d'odométrie des autres robots ne sont pas accessibles :

$$X_{v_i,k+1|k} = f\left(X_{v_i,k|k}, u_{v_i,k}\right). \tag{2.64}$$

La prédiction des covariances est similaire à l'équation (2.40) :

$$P_{v_i,k+1|k} = F_{i,k} P_{v_i,k|k} F_{i,k}^T + G_{i,k} Q_u G_{i,k}^T + Q_{v_i}, \tag{2.65}$$

$$P_{v_i j,k+1|k} = F_{i,k} P_{v_i j,k|k}, \tag{2.66}$$

avec j le reste des éléments, indissociablement un autre robot ou un amer, dont les modèles d'évolution sont statiques.

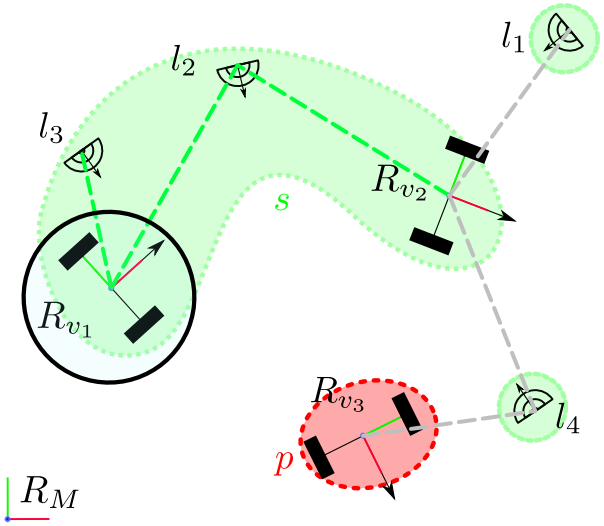


FIGURE 2.5. – Séparation de l'état en deux parties selon le point de vue du robot v_1. La partie s, en vert, sera estimée et l'impact sur l'inter-covariance de la partie p, en rouge, sera prise en compte.

On a donc, pour chaque amer l_j :

$$X_{l_j,k+1|k} = X_{l_j,k|k} \tag{2.67}$$
$$P_{l_j,k+1|k} = P_{l_j,k|k}, \tag{2.68}$$

et aucun bruit n'est ajouté, car il n'y a pas d'incertitude sur l'absence de mouvement des amers.

Au niveau des autres robots, on ne les estime pas, mais un modèle statique serait une grosse approximation. Nous utilisons donc une matrice de covariance, Q_p, dont le réglage sera expliqué plus tard :

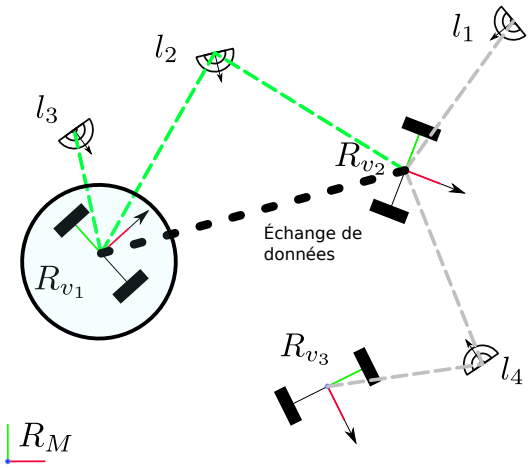
$$X_{v_j,k+1|k} = X_{v_j,k|k} \tag{2.69}$$
$$P_{v_j,k+1|k} = P_{v_j,k|k} + Q_p. \tag{2.70}$$

Étape de correction

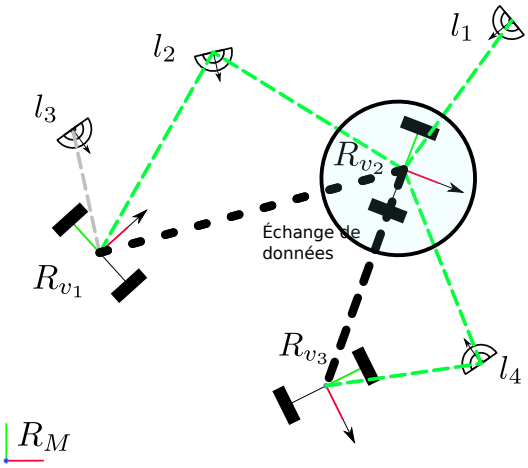
Pour l'étape de correction, considérons le cas d'un ego-robot v_i avec son groupe de robots V_i collaboratifs avec qui il a au moins un amer observé en commun.

Au début de l'étape de correction, chaque robot possède sa propre estimation de la carte et des autres robots. Les estimations des autres robots datent depuis la dernière fois où ils se sont rencontrés et ont collaboré. Nous verrons plus tard comment on obtient une même estimation pour chaque robot, afin d'effectuer la correction sur la même base.

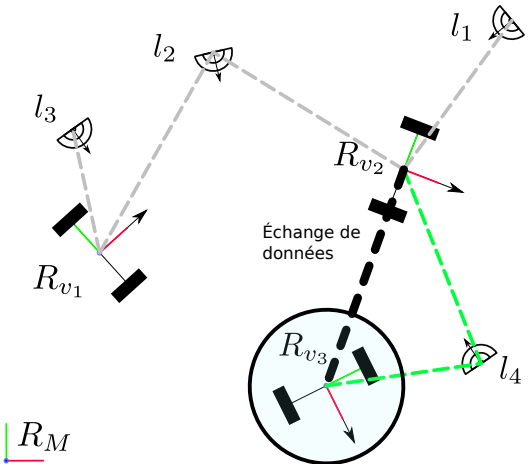
Pour un robot v_i, les observations utilisées sont les siennes et celles des amers observés par les autres robots et qui sont en commun avec v_i. Les autres observations des robots



(a) Le robot v_1 sélectionne ses observations et les observations d'autres robots ayant un amer en commun (v_2 dans le cas illustré, les observations utilisées sont représentées avec des traits verts), puis une communication pair-à-pair (représentée par le trait pointillé noir) est réalisée pour échanger les informations nécessaires à la fusion avec l'autre robot en collaboration.



(b) Cas du robot v_2.



(c) Cas du robot v_3.

FIGURE 2.6. – Illustration des observations sélectionnées par robot et des échanges d'informations.

dans V_i ne sont pas prises en compte. Autrement dit, nous utilisons les observations entre les robots dans V_i et les amers observés par v_i (traits en vert dans la figure 2.6a pour le robot v_1).

La sélection des observations fait que la partie p n'est pas observée, ce qui nous place dans un cas particulier du SKF, et son implémentation se simplifie.

Étant donné que toutes les observations utilisées sont faites entre un robot dans s et un amer aussi dans s, grâce aux équations (2.54) et (2.56), nous pouvons déterminer que H_p est nulle, avec H décomposée comme dans l'équation (2.10).

Exemple En reprenant l'exemple précédent, les robots v_1 et v_2 observent l'amer l_2 en commun. Le robot v_3 n'observe pas d'amer. La partie s est donc composée des robots v_1 et v_2, ainsi que des amers de la carte l_1 et l_2. La partie p est composée uniquement du robot v_3. La matrice Jacobienne H est donc réorganisée comme suit :

$$H_k = \begin{array}{c} \begin{array}{ccccc} v_1 & v_2 & l_1 & l_2 & v_3 \end{array} \\ \begin{pmatrix} \mathcal{R}_{v_1}^T \Gamma_{v_1}^{l_1} & \mathbb{0} & \mathcal{R}_{v_1}^T & \mathbb{0} & \mathbb{0} \\ \mathcal{R}_{v_1}^T \Gamma_{v_1}^{l_2} & \mathbb{0} & \mathbb{0} & \mathcal{R}_{v_1}^T & \mathbb{0} \\ \mathbb{0} & \mathcal{R}_{v_2}^T \Gamma_{v_2}^{l_2} & \mathbb{0} & \mathcal{R}_{v_2}^T & \mathbb{0} \end{pmatrix} \end{array} \begin{array}{c} Z_{l_1}^{v_1} \\ Z_{l_2}^{v_1} \\ Z_{l_2}^{v_2} \end{array}, \tag{2.71}$$

et H_p correspond à la dernière colonne-bloc composée uniquement de $\mathbb{0}$.

Par conséquent, avec $H_p = \mathbb{0}$, une autre expression plus compacte peut être trouvée pour K_s à partir de l'équation (2.18) :

$$K_{s,k} = P_{ss,k|k-1} H_s^T S^{-1}. \tag{2.72}$$

En reprenant les équations (2.19) et (2.20), elles se simplifient ainsi en remplaçant H_p et K_s par leurs nouvelles valeurs :

$$P_{ss,k|k} = (I - K_s H_s) P_{ss,k|k-1}, \tag{2.73}$$

$$P_{sp,k|k} = (P_{ps,k|k})^T = P_{sp,k|k-1} - K_s H_s P_{sp,k|k-1}, \tag{2.74}$$

$$= (I - K_s H_s)\, P_{sp,k|k-1}, \tag{2.75}$$

où S possède la forme suivante :

$$S_k = H_s P_{ss,k|k-1} H_s^T + R. \tag{2.76}$$

D'autre part, en partant de la forme développée de P_{ss} (équation (2.12)) et en rempla-

çant S par sa valeur, la forme de Joseph de l'équation (2.73) s'obtient :

$$P_{ss,k/k} = P_{ss,k|k-1} - K_s H_s P_{ss,k|k-1} - P_{ss,k|k-1} H_s^T K_s^T + K_s S_k K_s^T \tag{2.77}$$
$$= (I - K_s H_s) P_{ss,k|k-1} - P_{ss,k|k-1} H_s^T K_s^T + K_s H_s P_{ss,k|k-1} H^T K_s^T + K_s R K_s^T, \tag{2.78}$$
$$= (I - K_s H_s) P_{ss,k|k-1} - (I - K_s H_s) P_{ss,k|k-1} H^T K_s^T + K_s R K_s^T, \tag{2.79}$$
$$= (I - K_s H_s) P_{ss,k|k-1} (I - K_s H_s)^T + K_s R K_s^T. \tag{2.80}$$

Au final, en conservant la forme de Joseph (2.80) pour plus de stabilité numérique, les équations de correction de l'état sont les suivantes :

$$X_{s,k|k} = X_{s,k|k-1} + K_{s,k} \left(Z_k - h \left(X_{s,k|k-1} \right) \right), \tag{2.81}$$
$$X_{p,k|k} = X_{p,k|k-1}, \tag{2.82}$$
$$P_{ss,k|k} = (I - K_{s,k} H_{s,k}) P_{ss,k|k-1} (I - K_{s,k} H_{s,k})^T + K_{s,k} R K_{s,k}^T, \tag{2.83}$$
$$P_{sp,k|k} = (I - K_{s,k} H_{s,k}) P_{sp,k|k-1}, \tag{2.84}$$
$$P_{pp,k|k} = P_{pp,k|k-1}. \tag{2.85}$$

2.4.3. Fusion des différentes estimations avec la moyenne de Kullback-Leibler

La décentralisation engendre une multiplication des estimations des cartes, et plusieurs versions des inter-covariances robot-carte. En effet, chaque robot estime sa propre carte et lors des étapes de correction, il la corrige de façon collaborative avec les observations des autres robots avec qui il est en interaction. Cela entraine des différences dans les estimations des cartes ainsi que dans les inter-covariances avec les robots. Quelle version de la carte choisir lors de la correction en cas de collaboration ? Il n'est pas correct de sélectionner certains éléments d'un côté, et d'autres d'un autre, car cela détruirait la bonne forme de la covariance. De plus, pour que les observations provenant des autres robots aient un impact bénéfique, il faut que l'estimation de pose de l'autre robot soit consistante, et la meilleure possible. Pour n'avoir qu'une seule version de la carte, il faudrait centraliser la carte ou l'échanger en permanence entre les robots, ce qui n'est pas compatible avec ce que nous cherchons à faire.

Une première approche consisterait à décomposer les inter-covariances entre les robots, en s'inspirant des travaux de (Luft et al. 2016). Les inter-covariances robot-robot seraient décomposées de la manière suivante, où une partie est conservée par chaque robot :

$$P_{v_i,v_j} = \sigma_{v_i} \sigma_{v_j}^T, \tag{2.86}$$

avec σ_{v_i} et σ_{v_j} choisis de manière libre. Cette approche ne maintient pas la consistance de l'estimation, car dans notre cas, il est impossible de décomposer les inter-covariances robot-amer de la même façon. Cette première approche est détaillée en annexe A, mais elle introduit des inconsistances dans l'estimation des états.

L'approche que nous avons finalement retenue est de fusionner les estimations des différents robots en collaboration de manière conservative. Ainsi, les robots en collaboration se retrouvent avec des états consistants avant l'étape correction. Cette méthode est détaillée dans la suite.

Moyenne de Kullback-Leibler

Pour cette partie, nous introduisons une nouvelle notation. On note par un exposant le robot possédant la variable. Par exemple, l'état complet estimé par le robot v_1 est noté $X^{v_1} = \begin{pmatrix} X_V^{v_1} \\ X_L^{v_1} \end{pmatrix}$, et la covariance globale estimée par le robot v_1 est noté P^{v_1}. À l'intérieur de cette covariance, on peut retrouver $P_{v_2}^{v_1}$ qui est la covariance de v_2 estimée par le robot v_1.

Afin de fusionner les informations des différents robots, nous avons choisi d'utiliser la moyenne de Kullback-Leibler (*Kullback-Leibler Average*, KLA) exposée dans (Giorgio Battistelli et Luigi Chisci 2014). Il s'agit de trouver la distribution la plus proche de toutes celles que l'on veut fusionner, au sens de la divergence de Kullback-Leibler (KLD). La KLD est une mesure de dissimilarité entre deux distributions (Martins, Neto et Melo 2004). En d'autres termes, on cherche la distribution minimisant la moyenne des KLD avec toutes les autres :

$$\bar{p} = \arg \inf_{p \in \mathcal{P}} \frac{1}{m} \sum_{v_j \in V_i} D_{KL}(p||p^{v_j}), \tag{2.87}$$

où $D_{KL}(p||p^{v_j})$ est la KLD entre les distributions p et p^{v_j}, et v_j sont les robots en collaboration avec l'ego-robot v_i.

Dans le cas des distributions normales, la KLA se simplifie en la moyenne des vecteurs et matrices informationnels (Giorgio Battistelli et Luigi Chisci 2014) :

$$\bar{P}^{-1} = \frac{1}{|V_i|} \sum_{v_j \in V_i} (P^{v_j})^{-1}, \tag{2.88}$$

$$\bar{P}^{-1}\bar{X} = \frac{1}{|V_i|} \sum_{v_j \in V_i} (P^{v_j})^{-1} X^{v_j}. \tag{2.89}$$

avec $|V_i|$ est le cardinal de V_i (le nombre de robots en collaboration).

Les états complets des robots en collaboration, $X^{v_j} = \begin{pmatrix} X_V^{v_j} \\ X_L^{v_j} \end{pmatrix}$, sont donc fusionnés avant l'étape de correction du SKF en utilisant la KLA, comme illustré à la figure 2.7.

Il est à noter qu'une fusion utilisant l'intersection de covariance CI est aussi possible, mais peu souhaitable. En effet, elle a tendance à sélectionner l'une ou l'autre des estimations, avec peu de cas où un équilibre est fait entre les deux (Lima 2023). L'estimation d'un robot pourrait ainsi être modifiée en sélectionnant l'estimation faite par un autre robot, avec une erreur élevée. De même, le problème d'optimisation à résoudre avec le CI peut être coûteux dans le cas de fusion de n sources. Nous avons donc choisi d'utiliser la

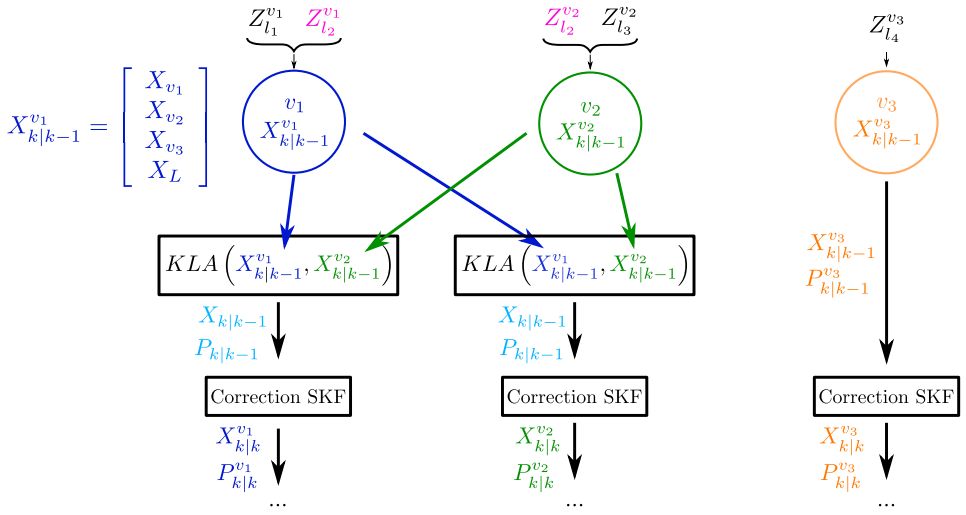


FIGURE 2.7. – Mise à jour collaborative, avec l'étape de KLA. Les robots v_1 et v_2 collaborent, donc échangent des données et réalisent la KLA avant de faire la correction avec leurs observations. Le robot v_3 ne collabore pas, il ne fait donc pas de KLA.

KLA, qui correspond à un CI où les poids sont équitablement répartis.

Adaptations nécessaires

Réaliser une fusion avec la KLA sur l'état complet nécessite que deux conditions soient remplies.

1. L'état doit être consistant avec une covariance estimée définie positive, c'est-à-dire que la covariance estimée doit être supérieure à la covariance réelle des erreurs d'estimation. Cela n'est pas le cas, car une partie de l'état (les autres robots) n'est pas prédite, et l'estimation n'a été mise à jour qu'à la dernière collaboration (qui peut être très ancienne).
2. Les tailles des états doivent être identiques car il est impossible de fusionner des états contenant un nombre différent d'éléments.

Afin de remplir le premier point, on peut revenir sur l'étape de prédiction du SKF, équations (2.69) et (2.70). L'étape de prédiction du robot v_i, pour chaque autre robot $v_j \neq v_i$, est :

$$X_{v_j,k+1|k}^{v_i} = X_{v_j,k|k}^{v_i}, \tag{2.90}$$

$$P_{v_j,k+1|k}^{v_i} = P_{v_j,k|k}^{v_i} + Q_p. \tag{2.91}$$

Le but étant d'avoir une estimation consistante des autres robots, sans utiliser leurs données d'odométrie, un bruit de modèle Q_p particulièrement élevé est appliqué sur les autres robots, à la manière d'un traqueur très simple. Q_p est une matrice de covariance

dimensionnée afin de prendre en compte le déplacement possible de v_j. De cette manière, l'estimation de $X_{v_j,k+1|k}$ par v_i reste consistante, bien que probablement erronée, et la KLA se chargera de la corriger.

Par rapport au second point, il faut d'abord dire que les robots n'ont pas connaissance des autres tant qu'ils n'ont pas collaboré avec eux. Il peut donc y avoir des incompatibilités d'état à fusionner lorsque des robots en collaboration n'ont pas rencontré les mêmes robots avant. Par exemple, un robot v_1 qui a rencontré un v_2 avant de collaborer avec v_3, aura une estimation de lui-même et de v_2 tandis que v_3 n'a pas d'estimation des autres robots. Il faut donc faire en sorte que les éléments présents dans les différents états complets à fusionner soient les mêmes.

Pour un ego-robot v_i, ces deux étapes sont réalisées comme suit :

— Nous ajoutons, dans l'état complet de chaque robot v_j collaborant avec v_i, les éléments présents dans l'état complet de v_i mais qui sont absents de l'état de v_j. Les nouveaux éléments sont initialisés avec l'estimation fournie par le robot v_i, et les inter-covariances sont mises à $\mathbb{0}$.

— Si c'est la première fois que v_i collabore avec v_j, l'état complet de v_i est augmenté pour ajouter l'estimation de v_j, fournie par lui-même, et avec des inter-covariances initialisées à $\mathbb{0}$.

Ces manipulations se font avec les états reçus par l'ego-robot v_i, l'état local à v_j n'est pas affecté, car on travaille sur des copies. Dans notre exemple, on ajoute v_1 et v_2 dans l'état de v_3 reçu par v_1, et on ajoute v_3 dans l'état de v_1 reçu par v_3. L'état local à v_3 ne comporte donc pas d'estimation de v_2.

Ensuite, seuls les éléments connus de l'ego-robot v_i sont sélectionnés dans l'état de v_j. Cela signifie que si v_j a collaboré avec un robot v_k inconnu de v_i, v_i ne l'ajoute pas dans son état, et v_k n'est pas sélectionné pour la KLA.

Ainsi, il est possible de faire la KLA, avec des états de même taille, et tous deux consistants.

Ces adaptations ont pour conséquence d'induire une sous-estimation des corrélations entre les robots et la carte lors de l'augmentation des états. Cela est moins optimal, mais reste tout de même consistant, et les matrices de covariances restent cohérentes et définies positives.

Le problème des angles

Lors de la KLA, une moyenne pondérée des états est réalisée. Lorsqu'on moyenne des états contenant un angle θ, le résultat de la moyenne dépend de la différence entre les angles. Par exemple, le résultat d'une moyenne entre un angle de 3,1 rad et un angle de −3,1 rad serait aux alentours de 0 rad. Or, on s'attend plutôt un résultat de π.

Afin de résoudre ce problème, tous les états des robots $v_j \neq v_i$ voient leurs angles reportés dans une fenêtre de $]-\pi;\pi]$ autour des angles estimés par v_i :

$$\forall v_j \in V_i \setminus v_i, \quad \forall k \in s, \qquad \theta_k^{v_j} \in]\theta_k^{v_i} - \pi; \theta_k^{v_i} + \pi] \tag{2.92}$$

Après la KLA, les angles en résultant sont reportés entre $-\pi$ et π.

2.5. Illustration du fonctionnement du filtre sur un exemple

Afin d'illustrer le fonctionnement du filtre, un exemple simple est étudié dans cette section, avec trois véhicules, v_1, v_2 et v_3, et deux amers, l_1 et l_2, composant la carte L.

Initialisation

On suppose que le filtre de chaque robot est initialisé avec une localisation très précise et une carte imprécise pour les amers. Chaque robot étant initialisé avec la même carte au départ, les estimations de l_1 et l_2 sont identiques. Pour le moment, les robots n'ont pas conscience de la présence des autres. L'état des robots au départ est donc :

$$X^{v_1} = \begin{pmatrix} X^{v_1}_{v_1} \\ X^{v_1}_{l_1} \\ X^{v_1}_{l_2} \end{pmatrix}, \qquad X^{v_2} = \begin{pmatrix} X^{v_2}_{v_2} \\ X^{v_2}_{l_1} \\ X^{v_2}_{l_2} \end{pmatrix}, \qquad X^{v_3} = \begin{pmatrix} X^{v_3}_{v_3} \\ X^{v_3}_{l_1} \\ X^{v_3}_{l_2} \end{pmatrix}.$$

P^{v_1}, P^{v_2} et P^{v_3}, les covariances correspondantes, sont initialisées avec des valeurs par défauts arbitraires, mais suffisamment conservatrices. Au départ, il n'y a pas d'inter-covariances : les matrices de covariances complètes sont donc diagonales.

Prédiction sans connaissance des autres robots

Pour le robot v_1, le vecteur d'état est défini par

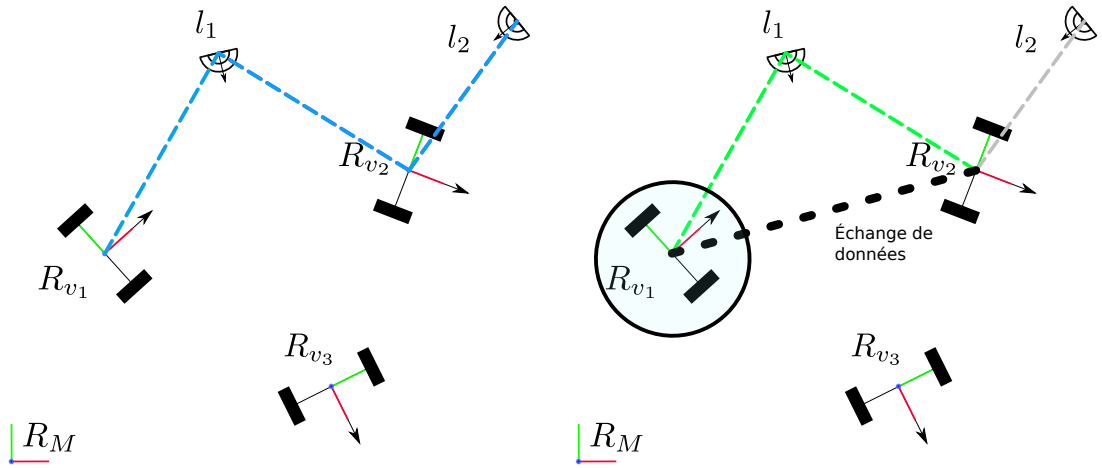
$$X^{v_1} = \begin{pmatrix} X_{v_1} \\ X_{l_1} \\ X_{l_2} \end{pmatrix}^{v_1}, \tag{2.93}$$

car v_1 n'a pas encore la connaissance de l'existence des autres robots. Chaque robot effectue son étape de prédiction grâce à ses données odométriques. Seul l'état du robot est affecté, ainsi que les inter-covariances entre ce robot et le reste.

L'étape de prédiction pour ce robot est alors :

$$X^{v_1}_{v_1,k+1|k} = f\left(X^{v_1}_{v_1,k|k}, u_{v_1,k}\right), \tag{2.94}$$

$$P^{v_1}_{k+1|k} = F^{v_1} P^{v_1}_{k+1|k} \left(F^{v_1}\right)^T + G^{v_1} Q_u \left(G^{v_1}\right)^T + Q, \tag{2.95}$$



(a) Les robots observent des amers. On suppose ici que les observations sont synchrones. Ils diffusent aux voisins la liste des amers observés. Dans cette figure, seuls v_1 et v_2 ont fait des observations sur des éléments de la carte.

(b) v_1 constate qu'il observe un amer en commun avec v_2. Il envoie donc les informations nécessaires à la fusion à v_2. Ce dernier arrive à la même conclusion et transmet ses informations à v_1. En vert, les observations utilisées par v_1.

FIGURE 2.8. – Situation initiale et échange de données pour une collaboration entre v_1 et v_2.

avec :

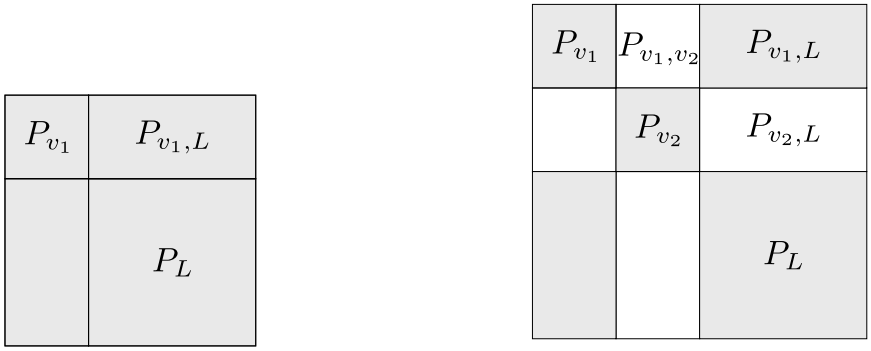
$$F^{v_1} = \begin{pmatrix} F_{v_1}^{v_1} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & I & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & I \end{pmatrix}, \tag{2.96}$$

$$G^{v_1} = \begin{pmatrix} G_{v_1}^{v_1} \\ \mathbb{0} \\ \mathbb{0} \end{pmatrix}. \tag{2.97}$$

Les I dans F^{v_1} correspondent au modèle d'évolution des amers l_1 et l_2 de la carte.

Correction et collaboration avec v_2

Le robot v_1 effectue maintenant une observation de l_1, $Z_{l_1}^{v_1}$. Il diffuse cette observation aux autres robots, et les autres robots réalisent la même procédure de leur côté. En écoutant ce que les autres ont observé, il reçoit l'information que v_2 a observé l_1 et l_2. Les deux robots entrent en collaboration car ils observent tous les deux l_1. v_1 envoie donc ses données à v_2, c'est-à-dire l'ensemble de son état estimé $X_{k|k-1}^{v_1}$, sa covariance $P_{k|k-1}^{v_1}$ et son observation $Z_{l_1}^{v_1}$. Le robot v_2 ayant le même raisonnement, au même moment, il envoie son état $X_{k|k-1}^{v_2}$, sa covariance $P_{k|k-1}^{v_2}$ et son observation de l'amer en commun $Z_{l_1}^{v_2}$ (figure 2.8).



(a) Covariance initiale du robot v_1 avant qu'il ne rencontre le reste des robots.

(b) Augmentation de la covariance lors de la rencontre avec le robot v_2. P_{v_1,v_2} et $P_{v_2,L}$ sont initialisées à $\mathbb{0}$ et P_{v_2} provient du robot v_2.

FIGURE 2.9. – Représentation de l'augmentation de la covariance lors de la rencontre entre v_1 et v_2. En gris sont représentés les blocs non nuls. Les exposants sont omis, nous nous plaçons du point de vue de v_1.

Première collaboration : augmentation de l'état Afin de pouvoir effectuer la correction, il faut que le robot v_1 ait une estimation de la localisation du robot v_2. Or, le robot v_1 n'a même pas v_2 dans son état (figure 2.9a). L'état $X^{v_1}_{k|k-1}$ et la covariance $P^{v_1}_{k|k-1}$ sont donc augmentés, en ajoutant un élément correspondant au robot v_2 (figure 2.9b). Cet élément est initialisé avec la pose fournie par v_2, $X^{v_2}_{v_2,k|k-1}$, et avec sa covariance $P^{v_2}_{v_2,k|k-1}$. Les inter-covariances entre le robot v_2 et le reste des éléments connus de v_1 sont initialisées comme nulles.

L'état de v_2 est aussi augmenté, de manière symétrique à l'augmentation de l'état de v_1. Si v_2 connaît déjà le robot v_3, seuls les éléments connus de v_1 sont extraits (sous forme de sous-matrices), puis réarrangées pour que l'ordre des éléments corresponde.

On a donc, du point de vue de v_1 :

$$P^{v_1} = \begin{pmatrix} P^{v_1}_{v_1} & \mathbb{0} & P^{v_1}_{v_1 l_1} & P^{v_1}_{v_1 l_2} \\ \mathbb{0} & P^{v_2}_{v_2} & \mathbb{0} & \mathbb{0} \\ P^{v_1}_{l_1 v_1} & \mathbb{0} & P^{v_1}_{l_1} & P^{v_1}_{l_1 l_2} \\ P^{v_1}_{l_2 v_1} & \mathbb{0} & P^{v_1}_{l_2 l_1} & P^{v_1}_{l_2} \end{pmatrix}. \tag{2.98}$$

En faisant le même raisonnement pour les données provenant de v_2, on obtient la matrice suivante, réarrangée pour que l'ordre des éléments corresponde à celui de v_1 :

$$P^{v_2} = \begin{pmatrix} P^{v_1}_{v_1} & \mathbb{0} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & P^{v_2}_{v_2} & P^{v_2}_{v_2 l_1} & P^{v_2}_{v_2 l_2} \\ \mathbb{0} & P^{v_2}_{l_1 v_2} & P^{v_2}_{l_1} & P^{v_2}_{l_1 l_2} \\ \mathbb{0} & P^{v_2}_{l_2 v_2} & P^{v_2}_{l_2 l_1} & P^{v_2}_{l_2} \end{pmatrix}. \tag{2.99}$$

Dans le robot v_2, l'ordre des éléments ne serait pas forcément le même.

Fusion des états avec la KLA Les estimations ne sont pas les mêmes entre les deux robots. Les équations (2.98) et (2.99) montrent que les matrices de covariance sont assez différentes. En particulier, ce sont les inter-covariances qui sont différentes, car on vient de les définir comme nulles entre v_2 et la carte, alors que ça n'est certainement pas le cas en réalité. Pour pouvoir faire une étape de correction avec le SKF, il est préférable d'avoir une estimation représentative des inter-covariances et des covariances de chaque robot. Une fusion des estimations des deux robots est donc réalisée avec la KLA :

$$\bar{P}^{v_1}_{k|k-1} = \left[\frac{1}{2}\left(P^{v_1}_{k|k-1}\right)^{-1} + \frac{1}{2}\left(P^{v_2}_{k|k-1}\right)^{-1}\right]^{-1} \tag{2.100}$$

$$\bar{X}^{v_1}_{k|k-1} = \bar{P}^{v_1}_{k|k-1}\left[\frac{1}{2}\left(P^{v_1}_{k|k-1}\right)^{-1} X^{v_1}_{k|k-1} + \frac{1}{2}\left(P^{v_2}_{k|k-1}\right)^{-1} X^{v_2}_{k|k-1}\right] \tag{2.101}$$

Détermination des parties s et p Le robot v_1 compose sa partie s de la manière suivante. Il s'ajoute bien évidemment dedans. Il ajoute ensuite les robots en collaboration (ici seulement v_2) puis tous les éléments de la carte (l_1 et l_2). On a donc $s = \{v_1, v_2, l_1, l_2\}$. N'ayant pas encore connaissance du robot v_3, la partie p est vide.

Correction avec le SKF Les équations du SKF présentées plus tôt sont utilisées :

$$X^{v_1}_{s,k|k} = \bar{X}^{v_1}_{s,k|k-1} + K^{v_1}_{s,k}\left(Z^{v_1}_k - h\left(\bar{X}^{v_1}_{s,k|k-1}\right)\right), \tag{2.102}$$

$$P^{v_1}_{ss,k|k} = \left(I - K^{v_1}_{s,k}H^{v_1}_{s,k}\right)\bar{P}^{v_1}_{ss,k|k-1}\left(I - K^{v_1}_{s,k}H^{v_1}_{s,k}\right)^T + K^{v_1}_{s,k}R^{v_1}\left(K^{v_1}_{s,k}\right)^T, \tag{2.103}$$

où les deux observations qui ont engendré la collaboration sont utilisées simultanément :

$$Z^{v_1}_k = \begin{pmatrix} Z^{v_1}_{l_1} \\ Z^{v_2}_{l_1} \end{pmatrix}, \tag{2.104}$$

$$K^{v_1}_{s,k} = \bar{P}^{v_1}_{ss,k|k-1}\left(H^{v_1}_{s,k}\right)^T\left(S^{v_1}_k\right)^{-1}, \tag{2.105}$$

$$S^{v_1}_k = H^{v_1}_{s,k}\bar{P}^{v_1}_{ss,k|k-1}\left(H^{v_1}_{s,k}\right)^T + R^{v_1}. \tag{2.106}$$

Dans le cas présent, la Jacobienne du modèle d'observation $H^{v_1}_{s,k}$ est :

$$H^{v_1}_{s,k} = \begin{array}{c} \begin{matrix} v_1 & v_2 & l_1 & l_2 \end{matrix} \\ \begin{pmatrix} \mathcal{R}^T_{v_1}\Gamma^{l_1}_{v_1} & \mathbb{0} & \mathcal{R}^T_{v_1} & \mathbb{0} \\ \mathbb{0} & \mathcal{R}^T_{v_2}\Gamma^{l_1}_{v_2} & \mathcal{R}^T_{v_2} & \mathbb{0} \end{pmatrix} \end{array} \begin{matrix} Z^{v_1}_{l_1} \\ Z^{v_2}_{l_1} \end{matrix}, \tag{2.107}$$

où $\mathcal{R}^{\mathcal{T}}$ et Γ sont définis aux équations (2.31) et (2.55). Il convient de noter que pour le calcul de $H^{v_1}_{s,k}$ et $K^{v_1}_{s,k}$, l'estimation utilisée est celle obtenue après la fusion avec la KLA (équation 2.101).

La correction au niveau de v_1 est terminée. Une procédure similaire est réalisée par les autres robots. Si v_2 n'observe aussi que l'amer l_1, alors il réalise les mêmes calculs qui vont conduire au même résultat. Par contre, s'il observe l_2 ou s'il collabore avec v_3, les calculs sont différents.

Prédiction après collaboration

Les robots réalisent leurs mesures d'odométrie. Cette fois-ci, l'étape de prédiction est différente car de l'incertitude est ajoutée sur l'estimation des autres robots via la covariance Q_p.

Dans notre exemple avec v_1, seule l'estimation de v_2 par v_1 est touchée pour le moment, car v_3 est encore inconnu.

On a donc :

$$P^{v_1}_{v_2,k+1|k} = P^{v_1}_{v_2,k|k} + Q_p, \tag{2.108}$$

avec Q_p dimensionné pour prendre en compte l'incertitude liée au déplacement possible de v_2.

Nouvelle collaboration mais avec v_3 uniquement

Le robot v_1 effectue encore une observation de l_1, $Z^{v_1}_{l_1}$, et collabore avec le robot v_3 avec l'observation du même amer $Z^{v_3}_{l_1}$. L'état du robot v_1 est donc de nouveau augmenté (figure 2.10a).

La KLA est effectuée de la même manière. Cependant, si v_3 n'a pas encore d'estimation de v_2, l'état que nous (le robot v_1) recevons doit être augmenté. Pour cela, notre propre estimation de v_2 est utilisée.

La partie s est définie de manière similaire, en sélectionnant le robot actuel, v_1, le robot en collaboration, v_3, ainsi que la carte l_1 et l_2, avec finalement $s = \{v_1, v_3, l_1, l_2\}$. Le robot v_2, qui ne collabore pas ici, est dans la partie p (figure 2.10b).

Les équations du SKF utilisées pour la correction sont les suivantes, tout en faisant attention que, cette fois-ci, la partie p n'est pas vide. Une correction de P_{sp} est alors nécessaire.

$$X^{v_1}_{s,k|k} = \bar{X}^{v_1}_{s,k|k-1} + K^{v_1}_{s,k}\left(Z^{v_1}_k - h\left(\bar{X}^{v_1}_{s,k|k-1}\right)\right), \tag{2.109}$$

$$P^{v_1}_{ss,k|k} = \left(I - K^{v_1}_{s,k}H^{v_1}_{s,k}\right)\bar{P}^{v_1}_{ss,k|k-1}\left(I - K^{v_1}_{s,k}H^{v_1}_{s,k}\right)^T + K^{v_1}_{s,k}R^{v_1}\left(K^{v_1}_{s,k}\right)^T, \tag{2.110}$$

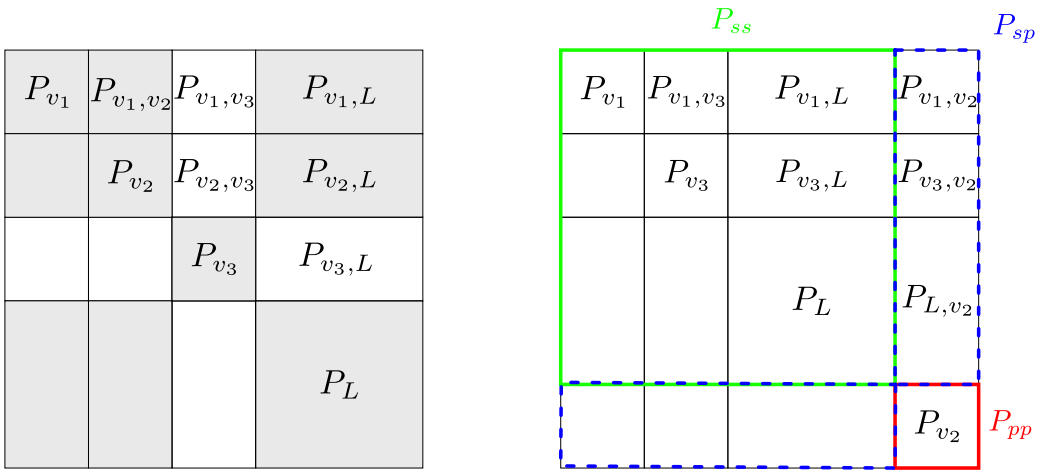
$$P^{v_1}_{sp,k|k} = \left(I - K^{v_1}_{s,k}H^{v_1}_{s,k}\right)\bar{P}^{v_1}_{sp,k|k-1}, \tag{2.111}$$

où $\bar{X}^{v_1}_{ss,k|k-1}$ et $\bar{P}^{v_1}_{ss,k|k-1}$ sont le résultat de la KLA entre l'état de v_1 et celui de v_3, et

$$Z^{v_1}_k = \begin{pmatrix} Z^{v_1}_{l_1} \\ Z^{v_3}_{l_1} \end{pmatrix}, \tag{2.112}$$

$$K^{v_1}_{s,k} = \bar{P}^{v_1}_{ss,k|k-1}\left(H^{v_1}_{s,k}\right)^T\left(S^{v_1}_k\right)^{-1}, \tag{2.113}$$

$$S^{v_1}_k = H^{v_1}_{s,k}\bar{P}^{v_1}_{ss,k|k-1}\left(H^{v_1}_{s,k}\right)^T + R^{v_1}. \tag{2.114}$$



(a) Augmentation de l'état de v_1 lors de la rencontre avec v_3. En gris sont représentés les blocs non nuls.

(b) Réarrangement des parties s et p, pour v_1.

FIGURE 2.10. – Représentation de l'augmentation de la covariance lors de la rencontre entre v_1 et v_3, ainsi que la sélection de s et p. v_2 ne collabore plus. Nous nous plaçons toujours du point de vue de v_1.

La Jacobienne $H^{v_1}_{s,k}$ est ici :

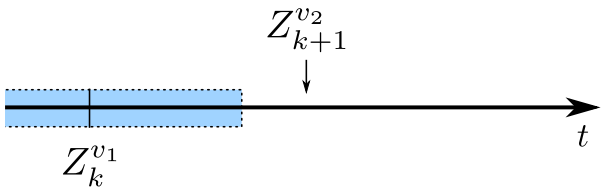
$$H^{v_1}_{s,k} = \begin{matrix} \\ v_1 \\ v_3 \end{matrix} \overset{\begin{matrix} v_1 & v_3 & l_1 & l_2 \end{matrix}}{\begin{pmatrix} \mathcal{R}^T_{v_1} \Gamma^{l_1}_{v_1} & \mathbb{0} & \mathcal{R}^T_{v_1} & \mathbb{0} \\ \mathbb{0} & \mathcal{R}^T_{v_3} \Gamma^{l_1}_{v_3} & \mathcal{R}^T_{v_3} & \mathbb{0} \end{pmatrix}} \begin{matrix} Z^{v_1}_{l_1} \\ Z^{v_3}_{l_1} \end{matrix} . \tag{2.115}$$

Cette Jacobienne est très similaire à celle calculée pour la collaboration entre v_1 et v_2, car elle porte sur la partie s, qui a subi une simple substitution de v_2 par v_3. La deuxième colonne-bloc correspond au robot v_3 et non plus au robot v_2 qui se retrouve dans la partie p.

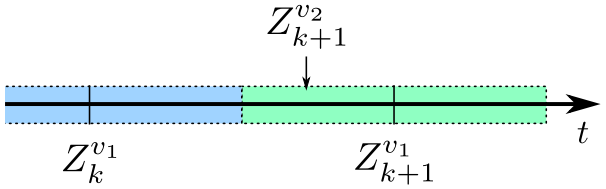
Les étapes de prédiction et de correction s'enchainent de nouveau.

2.6. Discussion sur la méthode proposée

La limitation du nombre de communications est un objectif que nous nous sommes donné. Cependant, les échanges actuels impliquent de transmettre tout l'état, ce qui n'est pas anodin. Il s'agit d'informations volumineuses, bien plus que ne le seraient les observations ou des contributions informationnelles. Une méthode reposant uniquement sur l'échange des observations, sans nécessiter la fusion des différents états, nécessite des échanges beaucoup plus fréquents, même en dehors des collaborations, pour maintenir ce niveau de précision (voir les méthodes utilisant le *channel filter*). L'échange des observations uniquement lors des collaborations est possible, mais donne des résultats



(a) Réception d'une observation d'un autre robot $Z_{k+1}^{v_2}$, hors de toute fenêtre temporelle existante (après la bleue).



(b) Obtention de l'observation locale $Z_{k+1}^{v_1}$, création de la nouvelle fenêtre d'association temporelle (vert), et association avec l'observation externe.

FIGURE 2.11. – Représentation de l'association temporelle des observations dans le cas de la réception d'une observation provenant d'un autre robot v_2 avant d'obtenir l'observation locale $Z_{k+1}^{v_1}$. Les fenêtres d'association sont représentées en bleu et vert.

moins bons sur l'ensemble des robots.

Ce problème de volumétrie de données découle de l'aspect décentralisé de la méthode, couplé à la limitation des communications, qui implique la mise en place d'une fusion des états estimés par chaque robot. Le maintien d'une estimation consistante nécessite l'échange et l'inversion des états complets, ce qui est coûteux et reste malgré tout moins optimal qu'une approche centralisée. Cependant, la méthode proposée est indépendante d'un serveur central, d'un système de communication centralisé et permet aux robots de continuer à se localiser en cas de rupture de communication avec les autres robots. Certaines optimisations pourraient être réalisées afin de ne pas échanger tout l'état à chaque collaboration, notamment si les deux robots ont collaboré peu de temps auparavant.

Une limitation de la quantité d'échanges peut se réaliser en supprimant la fusion avec la KLA. Dans ce cas, la méthode est coopérative dans le sens où chaque robot fusionne des observations reçues des autres robots avec sa version locale des estimations de poses et de carte. La mise à jour d'une carte globale n'est plus possible, ce qui réduit l'optimalité. Une nouvelle étude doit donc être menée pour déterminer les informations à échanger qui permettent de créer un compromis entre la méthode isolée et collaborative.

2.7. Résilience aux latences et aux problèmes de communication

Jusqu'à présent, les capteurs des différents robots ont été considérés comme étant synchronisés. Or, ce n'est pas nécessairement possible dans un cas réel. Si les capteurs ne sont pas synchronisés, négliger les décalages temporels peut avoir un impact sur l'estimation. Il se pose aussi le problème de l'association de données lorsqu'elles sont décalées temporellement, pour déclencher une collaboration.

Pour résoudre ce problème, les observations reçues par le robot i sont associées à ses observations les plus proches, dans une fenêtre de temps de la longueur de la période d'observation, centrée sur l'instant de ses observations. S'il n'y a pas encore d'observation dans la fenêtre, comme ce serait le cas si les capteurs ne sont pas synchronisés, les observations provenant des autres robots sont stockées en attendant au moins une observation du robot i (figure 2.11).

Le filtre de Schmidt-Kalman est un filtre récursif, chaque étape se basant sur l'état précédent du filtre, l'ordre dans lequel les données arrivent est très important. Des observations désordonnées ou avec de la latence par rapport à l'odométrie induisent des corrections d'un état présent alors qu'elles auraient dû être utilisées dans un état passé. Ces ruptures d'ordre et ces latences peuvent provenir de temps de traitements élevés, de latences sur le réseau de communication, etc. Ainsi, les données d'odométrie sont traitées rapidement par le filtre, car elles ne nécessitent pas de traitement en amont, tandis que les observations provenant de capteurs extéroceptifs plus complexes peuvent arriver avec du délai.

Ce problème est connu comme celui des *out-of-sequence measurements* (mesures hors-séquence, OOSM). Il s'agit d'un problème courant en pistage (*tracking*), car les données de capteurs provenant de différents robots peuvent arriver avec plus ou moins de délais, ce qui les désordonne temporellement. Plusieurs méthodes existent (Lima 2023 ; Muntzinger et al. 2010).

Le rejet Le procédé le plus simple est d'ignorer les mesures plus vieilles que la dernière utilisée. Si les mesures viennent de différentes sources, dont certaines très rapides (par exemple, les mesures locales), alors les mesures des sources les plus lentes seront systématiquement rejetées.

La mise en cache S'il n'y a pas de contrainte de temps réel, les mesures peuvent être stockées en cache en attendant qu'elles arrivent toutes. Une fois toutes les mesures obtenues, il suffit de les trier et d'exécuter les étapes du filtre. Cela conduit à de la latence dans la production du résultat final, correspondant à la latence du plus lent. Pour contrer cela, il est possible, en attendant, de prédire jusqu'au temps présent.

Prédiction de l'observation Une méthode présentée dans (Allig et Wanielik 2019) est de prédire ce qu'aurait été la OOSM à l'instant présent, puis de la fusionner. Le

problème est qu'il faut un modèle d'évolution de la mesure, qui peut être très bruité. Cette méthode est donc très légère en termes de calcul et de mémoire, mais elle est sous-optimale.

***Backward prediction* ou rétrodiction** L'état du filtre est prédit à rebours à l'instant correspondant à la OOSM. La mesure est intégrée puis l'état est de nouveau prédit jusqu'à l'instant présent. Il n'y a donc pas besoin de mettre en cache ou de recalculer des étapes, ce qui est très efficace d'un point de vue calculatoire, mais les bruits d'évolution aller/retour sont corrélés. Cela est fait de manière optimale grâce à la commutativité d'un IF dans (Nettleton et Hugh F. Durrant-Whyte 2001), ou par plusieurs méthodes sous-optimales utilisant le KF rappelées dans (Bar-Shalom 2002). Cette méthode ne peut fonctionner que sur des délais assez courts.

***Forward prediction and decorrelation* (FPFD)** Cette méthode, issue de (Rheaume et Benaskeur 2008), calcule deux prédictions à partir de l'instant juste avant l'OOSM jusqu'à l'instant présent. La première ne prend pas en compte la nouvelle mesure, et la deuxième fait la correction avec l'OOSM puis la prédit jusqu'à l'instant présent. Le gain d'information apporté par l'OOSM est calculé en comparant les deux prédictions, puis fusionné avec l'état actuel du filtre. Cette méthode est optimale pour de petits délais (une observation de retard) mais pas pour des délais plus longs. Elle nécessite de stocker l'état du filtre à chaque instant, mais nécessite moins de ressources calculatoires que le recalcul.

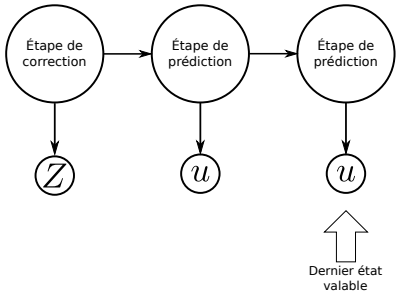
La mise en cache avec recalcul Cette méthode traite les mesures dès qu'elles arrivent, tout en les mettant en cache. Lorsqu'une OOSM arrive, il est donc possible de retraiter les mesures déjà présentes avec la nouvelle mesure (Tessier et al. 2006). Cela permet d'avoir la meilleure estimation possible avec les données disponibles, mais au prix d'une charge de calcul élevée et d'un besoin de beaucoup d'espace de stockage (il faut stocker les mesures, mais également les états à chaque instant).

Nous avons choisi d'utiliser la mise en cache avec recalcul, en représentant le SKF par une chaine de Markov. Chaque nœud est une étape du filtre, comme illustrée à la figure 2.12a. En gardant en mémoire l'état du filtre (état, covariance) à la fin de chaque étape, ainsi que les données utilisées, il est possible de rejouer l'enchainement des étapes à partir de n'importe quel instant. Chaque nœud est donc étiqueté temporellement par l'heure des données utilisées (heure de capture des données et non l'heure de réception).

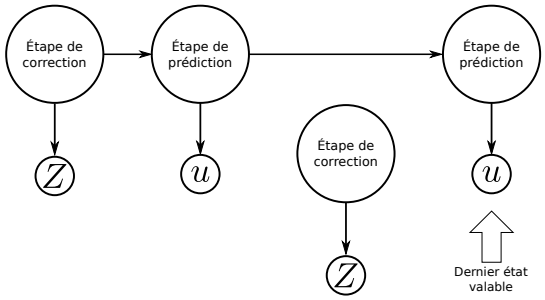
Dans le cas où une observation arriverait avec du délai, il suffit de replacer le filtre dans l'état de l'étape précédant immédiatement l'heure où le capteur a fourni les données de l'observation. L'observation est utilisée dans une étape de correction correspondante à un nouveau nœud inséré au bon instant dans la chaine de Markov (figure 2.12c). Enfin, toutes les étapes suivant cette nouvelle étape de correction sont recalculées pour avoir une estimation à l'instant présent qui prend en compte la nouvelle observation (figure 2.12d).

Le même principe est appliqué pour les observations provenant des autres robots. Lors de la réception des nouvelles observations, ces dernières sont ajoutées à la liste d'observation

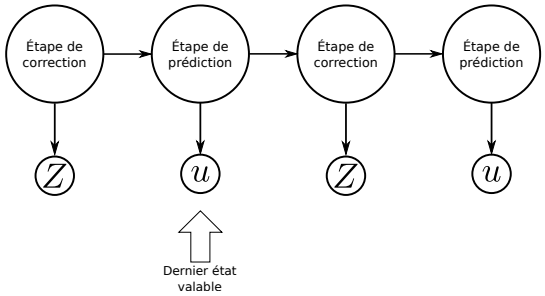
(a) Étape 1 : Initialement, la dernière étape de prédiction a été calculée et c'est l'état actuel du filtre.



(b) Étape 2 : Une observation est faite, et elle doit prendre place avant la dernière étape de prédiction.



(c) Étape 3 : Une nouvelle étape de correction est insérée, et on place le filtre dans l'état de sortie de l'étape précédente (ici l'étape de prédiction à gauche).



(d) Étape 4 : Enfin, on calcule les étapes nécessaires pour revenir au temps présent, donc la nouvelle étape de correction, puis l'étape de prédiction qui suit.

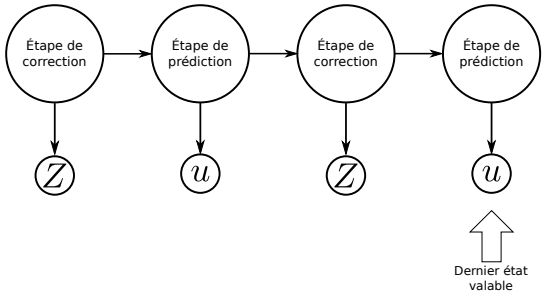


FIGURE 2.12. – Fonctionnement du processus de compensation de latence, par mise en cache avec recalcul, en quatre étapes.

du nœud de correction le plus proche (car les capteurs ne sont pas nécessairement synchrones), ou un nouveau nœud est créé s'il n'y en a pas d'assez proche (plus d'une demie-période d'observation de différence). Le filtre est placé dans l'état de sortie du nœud précédent. Toutes les étapes jusqu'à l'instant présent sont alors calculées, y compris l'étape de correction qui utilise maintenant de nouvelles observations.

Remarque. Ce mécanisme a ses limites et peut engendrer un effet boule de neige. Les délais d'observation nécessitent de recalculer des étapes déjà calculées, ce qui prend plus de temps de calcul et peut donc retarder le traitement des futures observations qui utilisent la même machine.

Ce procédé de tolérance aux latences permet donc de recomposer l'ordre dans lequel les données doivent être traitées, même si elles arrivent dans un ordre différent ou avec beaucoup de latence. Ainsi, le système peut correctement fonctionner, que ce soit quand les capteurs ne sont pas synchronisés, ou lorsque les observations des autres robots arrivent après celle de l'ego-robot.

2.8. Tolérance aux défauts

La tolérance aux défauts est l'aptitude d'un système à continuer de fonctionner en présence de défauts. Les défauts, ici, correspondent à un biais de mesures, une dérive, un blocage du capteur ou un biais au niveau du positionnement initial de quelques amers de la carte. Des défauts peuvent apparaitre, que ce soit au niveau des observations, de la carte initiale, des mesures d'odométrie, ou des différents traitements. Ces défauts auront un impact direct sur l'intégrité et la précision de l'estimation des poses des robots et des amers. Pour cela, une étape de détection et d'isolation de défauts, suivie d'une étape de récupération, sont ajoutées (FDIR). La détection de défauts se réalise au niveau de chaque robot en se basant sur un résidu global calculé dans l'espace des observations. Si un défaut est détecté, il sera isolé et une étape de récupération est lancée en fonction du type de défaut.

Dans cette thèse, nous avons considéré trois types de défauts illustrés à la figure 2.13.

Les défauts d'observation Les traitements utilisés pour extraire des observations des données brutes des capteurs peuvent produire de mauvais résultats, généralement à cause de données bruitées. Un défaut d'observation correspond au cas où le système n'observerait pas correctement l'amer, conduisant à une mesure qui diverge de la réalité. Il faut éviter d'inclure dans le filtre ces observations, car cela affecte la précision et l'intégrité de l'estimation d'état.

Les défauts de carte Lorsque les amers sont mal représentés dans la carte. Il s'agit d'amers loin de leurs poses réelles avec une covariance non représentative. Les amers ayant disparu de l'environnement ou qui viennent d'apparaitre ne sont pas considérés dans cette étude.

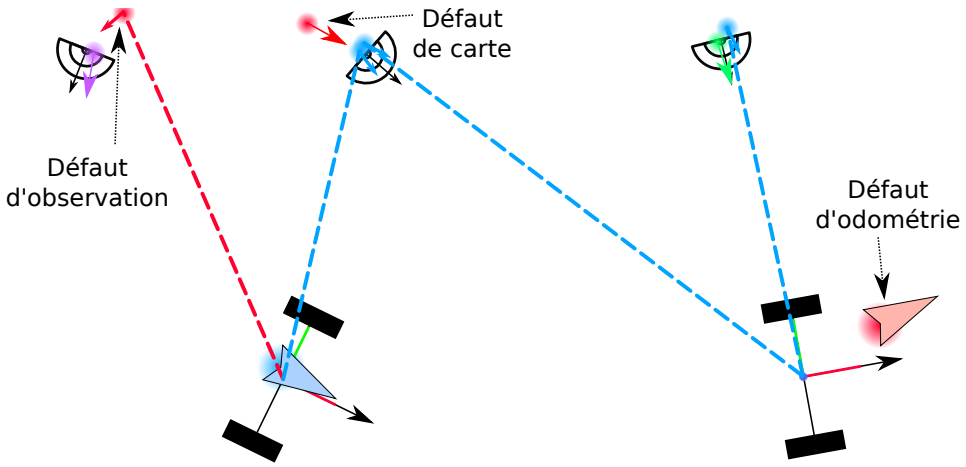


FIGURE 2.13. – Illustration des différents types de défauts isolables. L'observation rouge est erronée, elle sera donc exclue du vecteur d'observation. L'amer rouge est mal localisé dans la carte, mais comme il est observé par deux véhicules, il va pouvoir être corrigé. Enfin, le robot rouge a un défaut d'odométrie qui se traduit par une mauvaise estimation de sa pose. Les flèches violette et verte correspondent aux estimations correctes de la pose des amers.

Les défauts d'odométrie Ils correspondent au cas où l'estimation diverge à cause d'une erreur d'odométrie ou une erreur précédente non détectée.

Le système de FDIR mis en place se décompose en trois étapes : la détection de la présence d'un ou plusieurs défauts, l'isolation des défauts, puis la récupération d'un état sans défaut en excluant certaines données ou en corrigeant l'état. Dans la suite, les trois étapes du FDIR sont décrites, dans l'ordre d'exécution.

2.8.1. Détection de défauts

À chaque étape de correction du SKF, un résidu de détection est calculé afin de mettre en évidence la présence de défauts. Cette étape est réalisée au niveau de chaque robot en utilisant l'ensemble des observations utilisées pour l'estimation d'état.

Le résidu de détection est la distance de Mahalanobis entre le vecteur d'observation complet Z (comprenant ainsi les observations collaboratives venant d'autres robots) et sa prédiction $\hat{Z}$ calculée avec l'équation (2.45) :

$$r_d = \left(Z - \hat{Z}\right)^T S^{-1} \left(Z - \hat{Z}\right), \tag{2.116}$$

avec S la covariance de l'innovation, définie à l'équation (2.76).

Pour fixer le seuil de détection, la distribution de r_d doit être étudiée : il suit une loi du χ^2_n, avec n la dimension du vecteur d'observation Z.En pratique, la distribution théorique est rarement vérifiée à cause des bruits choisis et des hypothèses mal vérifiées.

Résidus	Amers		Observations de v_1		Odométrie
	l_1	l_2	$Z_{l_1}^{v_1}$	$Z_{l_2}^{v_1}$	odo_1
$r_{l_1}^{v_1}$	1	0	1	0	1
$r_{l_2}^{v_1}$	0	1	0	1	1

TABLE 2.1. – Matrice de signatures sans collaboration.

Si le résidu de détection r_d est au-dessus de son seuil, alors un défaut est détecté dans le système. La prochaine étape est de l'isoler.

2.8.2. Isolation de défauts

Lorsqu'un défaut a été détecté dans le système, il faut l'isoler, c'est-à-dire localiser la source de ce défaut. La collaboration joue un rôle important dans cette étape.

Pour cela, un ensemble de résidus basés sur la distance de Mahalanobis est calculé. Ils permettent, pour chaque observation, de calculer leurs distances au modèle estimé. Pour l'observation de l'amer l_j par le robot v_i, le résidu est le suivant :

$$r_{l_j}^{v_i} = \left(Z_{l_j}^{v_i} - \hat{Z}_{l_j}^{v_i}\right) \left(S_{l_j}^{v_i}\right)^{-1} \left(Z_{l_j}^{v_i} - \hat{Z}_{l_j}^{v_i}\right), \tag{2.117}$$

où :

$$\hat{Z}_{l_j}^{v_i} = \mathcal{R}_{v_i,k|k-1}^{T}(X_{l_j,k|k-1} - X_{v_i,k|k-1}), \tag{2.118}$$

$$S_{l_j}^{v_i} = H_{l_j}^{v_i} P_{ss,k|k-1} \left(H_{l_j}^{v_i}\right)^{T} + R_{l_j}^{v_i}. \tag{2.119}$$

Si les robots sont en collaboration et que la KLA est réalisée, la prédiction de l'état utilisée pour calculer le résidu utilise le résultat de la KLA. $H_{l_j}^{v_i}$ est la ligne-bloc de la matrice H correspondant à l'observation $Z_{l_j}^{v_i}$. De la même manière que pour le résidu de détection, ce résidu suit une loi du χ_3^2, car chaque observation est de taille 3. Ces résidus constituent nos **résidus d'observation**.

Le résidu $r_{l_j}^{v_i}$ présenté à l'équation (2.117) est sensible à un défaut de l'observation, $Z_{l_j}^{v_i}$, à un défaut au niveau de l'état de l'amer via $X_{l_j,k|k-1}$, et à un défaut de l'odométrie du robot à travers $X_{v_i,k|k-1}$.

Afin de déterminer l'origine du défaut, une matrice de signatures est utilisée (RAGOT 2021). Elle permet de caractériser l'isolabilité des défauts. La signature d'un défaut est un vecteur binaire, où les « 1 » indiquent la sensibilité du résidu au défaut et les « 0 » la non sensibilité.

La matrice illustrée à la table 2.1 montre un exemple de matrice de signatures, dans la situation où un robot v_1 observe deux amers l_1 et l_2. Des défauts peuvent être présents dans les observations $Z_{l_1}^{v_1}$ et $Z_{l_2}^{v_1}$, dans la carte avec les amers l_1 et l_2, ou bien dans l'odométrie (et autre), notée odo_1. Ce dernier type de défaut regroupe l'odométrie qui possède la même signature qu'un défaut du capteur, où toutes les observations du robot

Résidus \ Défauts	Amers		Observations de v_1		Observations de v_2		Odométrie	
	l_1	l_2	$Z_{l_1}^{v_1}$	$Z_{l_2}^{v_1}$	$Z_{l_1}^{v_2}$	$Z_{l_2}^{v_2}$	odo_1	odo_2
$r_{l_1}^{v_1}$	1	0	1	0	0	0	1	0
$r_{l_2}^{v_1}$	0	1	0	1	0	0	1	0
$r_{l_1}^{v_2}$	1	0	0	0	1	0	0	1
$r_{l_2}^{v_2}$	0	1	0	0	0	1	0	1

TABLE 2.2. – Matrice de signatures avec collaboration.

Résidus	Défaut unique								Défauts multiples			
	l_1	l_2	$Z_{l_1}^{v_1}$	$Z_{l_2}^{v_1}$	$Z_{l_1}^{v_2}$	$Z_{l_2}^{v_2}$	odo_1	odo_2	l_1 $Z_{l_1}^{v_1}$	l_1 $Z_{l_1}^{v_2}$	$Z_{l_1}^{v_1}$ $Z_{l_1}^{v_2}$	$Z_{l_1}^{v_1}$ $Z_{l_2}^{v_1}$
$r_{l_1}^{v_1}$	1	0	1	0	0	0	1	0	1	1	1	1
$r_{l_2}^{v_1}$	0	1	0	1	0	0	1	0	0	0	0	1
$r_{l_1}^{v_2}$	1	0	0	0	1	0	0	1	1	1	1	0
$r_{l_2}^{v_2}$	0	1	0	0	0	1	0	1	0	0	0	0

TABLE 2.3. – Matrice de signatures avec des défauts multiples. Seuls certains défauts multiples sont représentés afin de mettre en avant les problèmes de la matrice de signatures.

sont affectées. Ces deux types de défauts ne peuvent pas être différenciés dans notre approche et leur probabilité d'occurrence est supposée faible. Les lignes de la table correspondent aux résidus de chaque observation. Les éléments de la matrice sont remplis en prenant en compte la sensibilité des résidus.

On remarque que la matrice de signatures présentée à la table 2.1 possède des colonnes identiques, ce qui va empêcher l'isolation des défauts. Par exemple, si $r_{l_1}^{v_1}$ dépasse le seuil, la signature du défaut correspond à la fois à l_1 et à $Z_{l_1}^{v_1}$. Afin de pouvoir faire la distinction entre un défaut d'amer et un défaut d'observation, la redondance est nécessaire. C'est ici que la collaboration entre robots est très importante, lorsqu'un même amer peut être observé par plusieurs robots. En reprenant le même premier exemple, en rajoutant un robot v_2 qui collabore avec v_1 en observant aussi l_1 et l_2, la nouvelle matrice de signatures est maintenant présentée à la table 2.2. Dans ce cas, la signature d'un défaut d'amer l_1 est différente d'un défaut de l'observation $Z_{l_1}^{v_1}$. Par conséquent, l'isolation de défauts d'amer (carte) et d'observation est maintenant possible sous l'hypothèse d'un seul défaut à la fois, grâce à la collaboration.

La collaboration permet donc de rendre le système isolable en proposant la possibilité de faire la différence entre un défaut de carte ou un défaut d'observation. S'il n'y a pas de collaboration, le défaut d'observation sera privilégié.

Défauts multiples et résidu d'amer

Pour le moment, nous avons considéré qu'un seul défaut pouvait survenir au même instant. Or, cette hypothèse est trop forte pour notre application finale. Qu'en est-il du cas où il y aurait de multiples défauts ? Pour déterminer leurs signatures, il faut combiner les signatures des différents défauts (Ragot 2021) avec un OU logique. Cela crée de nouvelles signatures correspondant aux défauts combinés rajoutées à la matrice de signatures, montrée à la table 2.3. Par exemple, cette matrice n'est pas capable de différencier le défaut de carte l_1 du défaut combiné des deux observations de cet amer, $\{Z_{v_1}^{l_1}, Z_{v_2}^{l_1}\}$, car ils ont la même signature. Les résidus d'observation seuls ne sont donc pas suffisants.

Afin de lever l'ambiguïté et de rendre le système isolable jusqu'à deux défauts combinés, un nouveau résidu est introduit pour être indépendant de la carte. Ce résidu est calculé lorsque deux véhicules collaborent en observant le même amer permettant le calcul de deux estimations de pose différentes de cet amer. Chaque estimation est calculée en prenant compte une seule observation à la fois. Le raisonnement derrière est que si les deux estimations basées sur deux observations différentes sont cohérentes, mais que les observations sont toutes les deux loin de leur prédiction, alors le défaut provient de la carte (voir le défaut de carte sur la figure 2.13). Il est ensuite possible de distinguer le défaut multiple $\{Z_{v_1}^{l_1}, Z_{v_2}^{l_1}\}$ du défaut de l'amer l_1 (table 2.4).

Contrairement au premier type de résidu qui est calculé dans l'espace d'observation, ce nouveau résidu, dit d'amer, est calculé dans l'espace d'état. Il est obtenu à partir de la distance de Mahalanobis entre les estimations de poses du même amer dans le repère de la carte :

$$r_{l_j}^{v_i v_k} = \left({}^{M}Z_{l_j}^{v_i} - {}^{M}Z_{l_j}^{v_k}\right)^T \left({}^{M}R_{l_j}^{v_i} + {}^{M}R_{l_j}^{v_k}\right)^{-1} \left({}^{M}Z_{l_j}^{v_i} - {}^{M}Z_{l_j}^{v_k}\right) \tag{2.120}$$

où ${}^{M}Z_{l_j}^{v_i}$ et ${}^{M}Z_{l_j}^{v_k}$ sont les poses de l'amer l_j estimées par les robots v_i et v_k dans le repère de la carte R_M. ${}^{M}R_{l_j}^{v_i}$ et ${}^{M}R_{l_j}^{v_k}$ sont les matrices de covariances transformées dans le repère de la carte.

Ces poses sont obtenues à partir de la transformation des observations dans le repère de la carte via la pose du robot (figure 2.14) :

$$\begin{aligned} {}^{M}Z_{l_j}^{v_i} &= \left[{}^{M}x_{l_j}^{v_i}, {}^{M}y_{l_j}^{v_i}, {}^{M}\theta_{l_j}^{v_i}\right]^T && (2.121)\\ &= {}^{M}\mathcal{R}_{v_i} Z_{l_j}^{v_i} + {}^{M}\mathcal{T}_{v_i} && (2.122) \end{aligned}$$

avec ${}^{M}\mathcal{R}_{v_i}$ la matrice de rotation entre le repère du robot R_{v_i} et le repère de la carte R_M, et ${}^{M}\mathcal{T}_{v_i}$ la matrice de translation entre ces mêmes repères. ${}^{M}Z_{l_j}^{v_i}$ représente alors une pose de l'amer dans le repère global.

La covariance de l'observation est aussi transformée dans le repère de la carte (Smith,

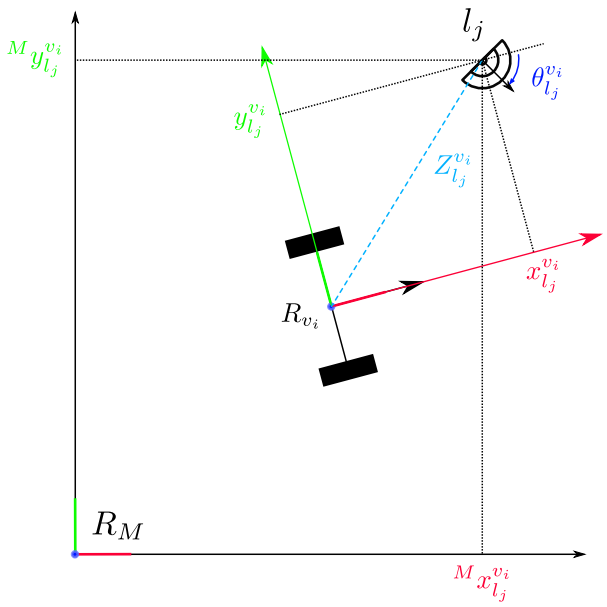


FIGURE 2.14. – Expression de l'observation de pose relative dans le repère de la carte.

SELF et CHEESEMAN 1990) :

$$ {}^M R_{l_j}^{v_i} = J \begin{pmatrix} P_{v_i} & O \\ O & R_{l_j}^{v_i} \end{pmatrix} J^T \tag{2.123} $$

où J est la matrice jacobienne de l'équation (2.122) :

$$ J = \frac{\partial({}^M Z_{l_j}^{v_i})}{\partial(X_{v_i}, Z_{l_j}^{v_i})} = \begin{pmatrix} 1 & 0 & -(y_{v_i} - {}^M y_{l_j}^{v_i}) & \cos\theta_{v_i} & -\sin\theta_{v_i} & 0 \\ 0 & 1 & (x_{v_i} - {}^M x_{l_j}^{v_i}) & \sin\theta_{v_i} & \cos\theta_{v_i} & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \tag{2.124} $$

Le résidu d'amer $r_{l_j}^{v_i v_k}$ est donc sensible à des défauts sur les deux observations ($Z_{l_j}^{v_i}$ et $Z_{l_j}^{v_k}$), et sur l'odométrie des deux robots. La carte n'intervenant pas dans le calcul du résidu, il n'est pas sensible à un défaut de carte. Un défaut combiné des deux observations n'est pas envisageable lorsque ce résidu est en dessous de son seuil, laissant donc seulement le défaut de carte compatible avec la signature. Le résidu d'amer suit aussi une loi du χ_3^2.

La table 2.4 présente le même cas que précédemment, mais en rajoutant les lignes correspondantes aux résidus d'amer. Il y a autant de résidus d'amer pour un amer qu'il y a de couples d'observations. Lorsque trois robots observent un même amer, il y a trois résidus calculés, avec des sensibilités aux défauts différentes.

Le système est maintenant isolable pour deux défauts simultanés si les hypothèses suivantes sont vérifiées :

1. Le défaut d'amer peut être isolé seulement lorsque au moins deux robots observent le même amer. La collaboration est donc essentielle, sous peine de se retrouver dans la situation de la table 2.1.

Résidus \ Défauts	Amers		Observations de v_1		Observations de v_2		Odométrie	
	l_1	l_2	$Z_{l_1}^{v_1}$	$Z_{l_2}^{v_1}$	$Z_{l_1}^{v_2}$	$Z_{l_2}^{v_2}$	odo_1	odo_2
$r_{l_1}^{v_1}$	1		1				1	
$r_{l_2}^{v_1}$		1		1			1	
$r_{l_1}^{v_2}$	1				1			1
$r_{l_2}^{v_2}$		1				1		1
$r_{l_1}^{v_1,v_2}$			1		1		1	1
$r_{l_2}^{v_1,v_2}$				1		1	1	1

TABLE 2.4. – Matrice de signatures avec les résidus d'inter-observation. Les « 0 » sont remplacés par du vide pour simplifier la lecture.

2. La signature d'un défaut d'odométrie est la même qu'un problème général au niveau du capteur de perception, ce qui affecte l'ensemble des observations du même robot. Pour cela, un certain niveau de redondance est nécessaire pour isoler un défaut odométrie. Ceci est réalisé dans le cas où un robot observe au moins trois amers. En effet, on considère que la probabilité d'avoir trois défauts d'observations simultanés est très faible.
3. Un défaut simultané d'une observation et de l'amer observé est non isolable, et il est considéré peu probable. L'hypothèse du défaut d'observation sera privilégiée.

Dans le reste, nous ferons l'hypothèse de deux défauts simultanés au maximum, et donc supposons que ces trois hypothèses sont remplies.

Cas ambigus

Lorsque les résidus donnent les résultats attendus, la matrice de signatures présentée ici fonctionne correctement. Cependant, en pratique, ce n'est pas toujours le cas. Nous avons évoqué plus tôt les niveaux d'isolation des matrices de signatures. La matrice de signatures présentée ici est faiblement isolante, puisqu'on peut retrouver la même signature pour des défauts multiples.

Dans ce cadre, pour isoler un défaut, une correspondance parfaite entre les signatures théoriques et la signature effective est recherchée. S'il n'y a pas une et une seule correspondance, le comportement par défaut est de désigner comme erronées les observations dont les résidus associés sont supérieurs à leur seuil.

2.8.3. Etape de récupération

Une fois les défauts isolés, l'objectif est de prendre la bonne action afin qu'ils n'affectent pas la précision et l'intégrité des estimations d'état. Différentes actions sont réalisées en fonction du type de défaut, à savoir, l'exclusion des observations erronées ou la correction de l'estimation de pose des amers erronées.

Exclusion de défauts

En cas de défaut d'une observation, une action simple consiste à exclure cette mesure de la fusion de données. Le nouveau vecteur d'observation est ainsi réduit. Les matrices H et R sont adaptées, c'est-à-dire que les lignes et colonnes correspondant à cette observation sont supprimées. S'il s'agit d'une observation locale et qu'elle fait l'objet d'une collaboration, l'observation de l'autre robot est conservée, autant garder les informations à disposition. Chaque robot réalise cette étape à son niveau.

Correction de la carte

Dans le cas d'un défaut d'amer, l'exclusion de défaut ne peut pas être réalisée. En effet, le défaut ne provient pas d'une mesure qu'on peut simplement supprimer, mais de l'état lui-même. Une première méthode consiste à utiliser les observations en espérant que leurs covariances soient suffisantes pour corriger la carte. Cette méthode peut être utilisée pour les petits défauts. Une deuxième méthode consiste à augmenter la contribution des observations par rapport à l'état de l'amer erroné en augmentant artificiellement la covariance de cet amer. Cette deuxième méthode est utilisée lorsque la covariance ne représente plus l'erreur de l'amer. Pour cela, une matrice de covariance A est ajoutée au bruit du modèle de l'amer correspondant, qui était nul. Le volume de cette matrice de covariance est définie en fonction de la distance d entre la moyenne des m observations de l'amer, $\bar{Z}_{l_i}$, et l'état de l'amer l_i :

$$\bar{Z}_{l_i} = \frac{1}{m} \sum_j Z_{l_i}^{v_j} \tag{2.125}$$

$$d = ||X_{l_i,k|k-1} - \bar{Z}_{l_i}|| \text{ en utilisant seulement } x \text{ et } y \tag{2.126}$$

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0.5 \end{bmatrix} * d^2 * \eta, \tag{2.127}$$

$$P^+_{l_i,k|k-1} = P_{l_i,k|k-1} + A, \tag{2.128}$$

où η est un facteur de taille, arbitrairement fixé ici à 9.

Après cette modification de la covariance, l'étape de correction se réalise en partant de cette valeur.

Réinitialisation du système

Lorsqu'un défaut d'odométrie est isolé, cela signifie que l'estimation actuelle du robot a divergé. Il faut donc envoyer un signal pour réinitialiser la pose via une méthode de localisation absolue (GNSS par exemple), ou bien prévenir l'utilisateur que l'estimation n'est pas utilisable.

2.9. Synthèse de l'approche proposée

Le fonctionnement de notre méthode suivant une architecture décentralisée avec des contraintes de communication est résumé à la figure 2.15.

Chaque robot, grâce à des capteurs (non spécifiés pour le moment) observe des amers discernables dans l'environnement. Les identifiants des amers observés sont envoyés aux robots avoisinants. S'il y a des amers observés en commun, les robots entrent en collaboration et échangent les données nécessaires au SKF (état, covariance et observations). Chaque robot se constitue une liste d'observations comprenant les siennes ainsi que celles d'amers déjà observés par d'autres robots.

Une fois la liste d'observation constituée, les estimations des robots en collaboration sont fusionnées avec la KLA. Ensuite, le modèle d'observation est utilisé pour prédire les observations. Avant de corriger l'état, une étape de FDIR est utilisée, pour détecter et isoler des défauts d'observation, de carte ou d'estimation du robot. Ces défauts sont ensuite pris en charge par l'exclusion d'observations, par la correction de la carte ou par la réinitialisation du filtre grâce au GNSS. L'isolation des défauts est largement aidée par la collaboration, ce qui permet d'améliorer l'intégrité de l'estimation. Le vecteur d'observation est constitué des observations restantes, puis est utilisé dans l'étape de correction du SKF.

Entre deux salves d'observations, l'odométrie est utilisée par chaque robot pour prédire l'état du robot. Les covariances des autres robots sont augmentées afin de garder une consistance dans leurs estimations.

2.10. Conclusion

La méthode présentée dans ce chapitre tire parti de la collaboration pour améliorer l'estimation de la carte et la localisation des robots simultanément à partir de mesures indirectes. Nous avons présenté une architecture décentralisée avec des contraintes de communication qui utilise le filtre de Schmidt-Kalman pour l'estimation d'état et la moyenne de Kullback-Leibler pour fusionner les estimations fournies par chaque robot.

Par ailleurs, nous avons cherché à réduire au minimum le nombre de communications. Ainsi, la méthode proposée repose sur l'échange de données entre les robots uniquement lorsqu'ils observent un amer en commun. Cela réduit grandement le nombre d'échanges comparé à une approche qui reposerait sur un serveur central, nécessitant un envoi des observations puis une réception de l'état corrigé. Cependant, les échanges restants sont assez volumineux, avec l'échange des observations et surtout de l'état complet, avec sa covariance. L'échange de ces données est nécessaire pour réaliser la KLA et ainsi donner toutes leurs forces aux observations collaboratives. La KLA est assez coûteuse en temps de calcul, car il faut inverser les matrices de covariance complètes. La méthode ne peut pas encore passer à l'échelle à ce niveau-là.

En contrepartie, la méthode proposée permet d'estimer à la fois la position du robot, mais aussi des amers de l'environnement, sans dépendre de la communication avec les autres robots ou avec un serveur central. En effet, chaque échange se suffit à lui-même

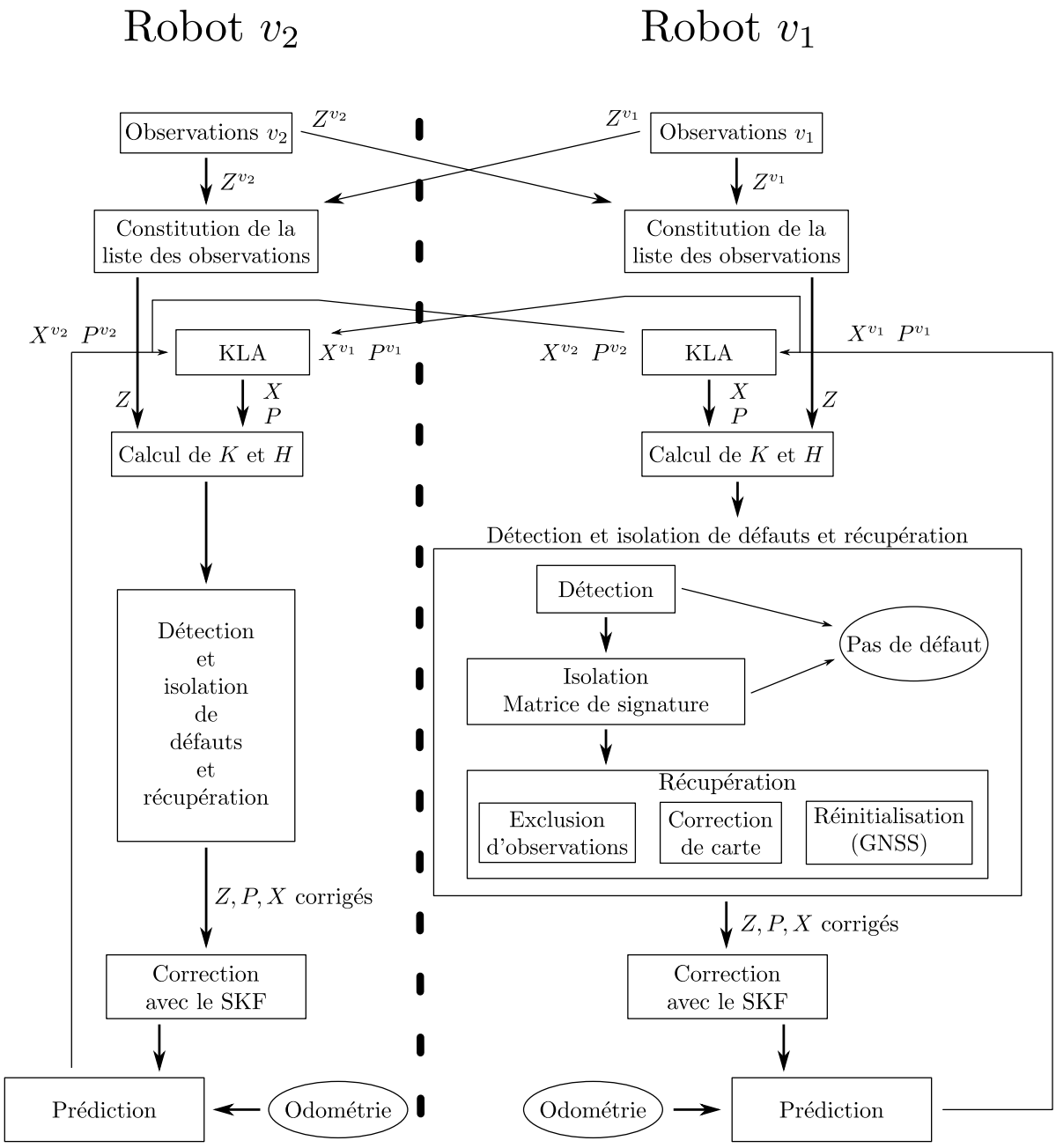


FIGURE 2.15. – Synthèse du fonctionnement de la méthode présentée.

et ne dépend pas d'une communication précédente ou des suivantes. Si un échange de donnée est perdu, le robot peut continuer à se localiser tout en restant consistant. De plus, même sans aucune communication, chaque robot peut réaliser son estimation locale. Cette estimation est renforcée par la collaboration avec d'autres robots lorsque c'est possible, à la fois par l'utilisation des observations collaboratives, mais aussi grâce à l'isolation de défaut plus performante.

Un exemple a été présenté pour illustrer le fonctionnement de la méthode sans la présence de défauts.

Quand ces derniers existent, nous avons montré que la collaboration permet aussi de mieux déterminer l'origine des défauts, grâce à la distinction rendue possible entre les défauts d'observation et les défauts de carte. Nous avons aussi proposé une méthode pour réagir à ces défauts et revenir à un état sain, par l'exclusion des observations, ou par la correction de la carte.

Dans le chapitre suivant, nous allons évaluer cette méthode dans un environnement de simulation avec plusieurs robots.

3. Évaluation de la méthode en simulation

Sommaire

3.1. Introduction

Avec l'aide d'un simulateur développé spécialement pour ces travaux, nous évaluons dans ce chapitre la méthode dans son ensemble et ainsi que l'effet de certains paramètres sur les résultats. Le module de détection et d'isolation de défauts puis de récupération (FDIR) sera évalué avec différents types de défauts afin de montrer la pertinence de la méthode. Tout ce chapitre utilise des paramètres de base communs qui seront explicités avec les premiers résultats

3.2. Présentation du simulateur développé

Afin d'évaluer la méthode proposée, un simulateur de robots a été développé. Il fonctionne dans le framework ROS (*Robot Operating System*) et comprend des modules interchangeables de contrôle, de simulation de physique et d'estimation d'état. Les communications inter-robots sont interceptées par le simulateur, permettant de rajouter des contraintes (latence, portée, etc.).

3.2.1. Robot Operating System

L'implémentation de la méthode proposée a été faite avec ROS [1], qui est un ensemble de programmes permettant d'abstraire les communications entre des nœuds. Ces nœuds sont des processus indépendants, exécutant différentes fonctions du système, et qui peuvent être présents sur plusieurs machines. Ils réalisent leurs tâches en fonction de données auxquelles ils sont abonnés (émises par d'autres nœuds) et produisent de nouvelles données qu'ils peuvent envoyer aux nœuds suivants.

Le système de communication se fait via des *topics*, des canaux dans lesquels plusieurs nœuds peuvent lire et écrire. Il n'est donc pas nécessaire de connaitre l'identité des nœuds produisant les données voulues, ni les nœuds utilisant les nôtres, tout cela est régi par ROS. Afin de permettre la communication entre les nœuds, potentiellement écrits dans des langages différents (C++, Python, etc.), un système de messages propres à ROS est utilisé, qui peut être sérialisé par un langage et désérialisé par un autre.

3.2.2. Simulateur multi-robots

Le simulateur développé dans le cadre de cette thèse est composé de plusieurs modules : un module de simulation physique, un module de contrôle et un module d'estimation d'état. L'idée étant que chaque module peut être changé. Chaque module possédant une version d'externalisation, il serait possible d'utiliser un robot réel, de type turtlebot, pour remplacer la simulation physique ou un algorithme spécifique pour le contrôle et l'estimation d'état. La configuration des différents modules utilisée pour évaluer les résultats est présentée ci-après.

Simulation physique Chaque robot est représenté par un modèle unicycle, avec deux roues contrôlées, de la même façon qu'un turtlebot [2]. La représentation interne du robot est composée d'une pose et des vitesses des deux roues. Lorsqu'une commande, en vitesse, est reçue, elle est appliquée au robot. Lors du prochain pas de simulation, la pose sera mise à jour avec les nouvelles vitesses de roue. Du bruit peut être ajouté sur l'actionneur, mais ce n'est pas le cas ici.

Contrôleur Le contrôle de chaque robot se fait via un contrôleur « proportionnel-dérivée » (PD). Un chemin de consigne est fourni sous forme d'une liste de points, puis l'estimation de la pose du robot est projetée sur la polyligne avec une avance de phase. Les erreurs en angle et en vitesse sont ensuite calculées, puis la commande, sous forme d'une vitesse longitudinale et d'une vitesse de rotation, est déterminée. Enfin, cette commande est convertie en vitesse de roues pour être envoyée au module de simulation physique.

1. `https://www.ros.org/`
2. `https://www.turtlebot.com/`

Estimation d'état Ce module est très simple, car il utilise la vérité terrain, grâce à la pose du module physique directement.

Les capteurs sont simulés avec la simulation physique. Pour répondre à notre cas d'étude, deux capteurs sont simulés : un capteur d'odométrie et un capteur de pose relative d'amer qui peut être obtenu réellement à partir d'un lidar ou d'une caméra (le traitement des données capteurs est simulé).

Odométrie Les vitesses de roues présentes dans le module physique sont récupérées par le capteur d'odométrie. Ces mesures sont ensuite bruitées avec un générateur de nombre pseudo-aléatoire suivant une distribution normale, centrée et avec une variance donnée.

Pose relative d'amer Afin de générer des observations des amers sous forme de poses, la pose provenant de la simulation physique est utilisée. Le même modèle d'observation que le filtre est utilisé, avec la pose réelle du robot et une carte donnée. La carte est parfaite, mais les observations sont bruitées pour simuler le bruit de capteur, en temps-réel, suivant une loi normale.

Une surcouche de communication a été rajoutée par-dessus celle de ROS. Il est ainsi possible de diffuser un message à tous les robots à portée de communication, ROS ne supportant pas la limitation de portée des messages. Il en est de même pour les communications en pair-à-pair, limitées à deux robots, ce qui n'est pas le cas des *topics* de ROS. Il est aussi possible de rajouter du délai à l'émission ou à la réception, mais finalement, aucune latence ni limite de portée ne sera utilisée dans la suite.

3.3. Résultats de localisation et mise à jour de carte collaborative

Dans un premier temps, la méthode est évaluée dans un environnement sans injection de défauts et sans l'étape de FDIR. Dans cette section, l'impact de la collaboration est étudié par rapport à son absence (mode isolé). L'impact de différents paramètres est ensuite évalué.

Le protocole présenté ici est détaillé et les résultats suivants ne seront que des variations sur ce protocole.

Protocole

Trois robots sont simulés et chacun suit sa propre trajectoire, à des vitesses différentes. Quatre amers sont disposés sur un carré de 10 m de côté. L'amer 1 est parfaitement localisé dans la carte initiale, et sa petite covariance initiale représente le fait qu'il est parfaitement localisé. La figure 3.1 montre le départ du scénario, avec les poses et les ellipses de covariances (2σ) initiales des robots et des amers.

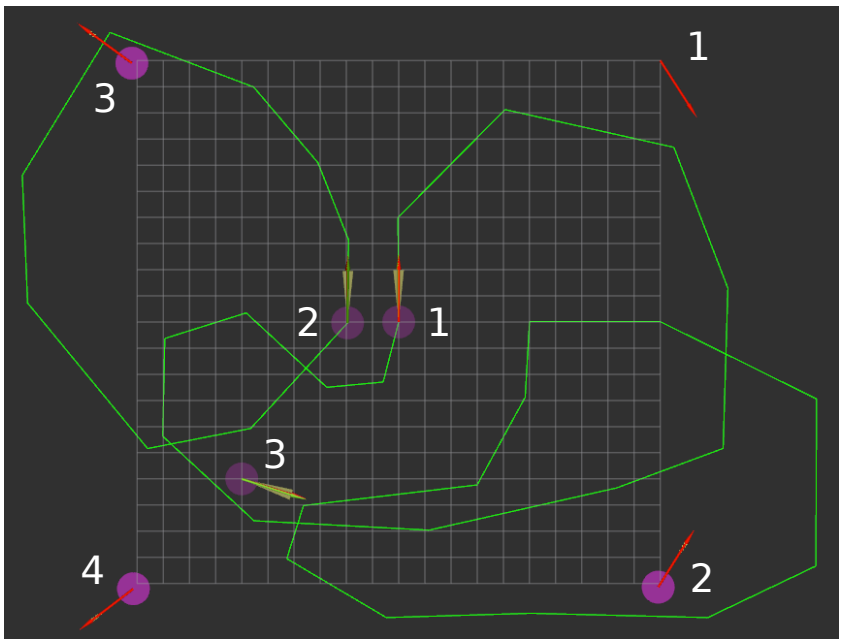


FIGURE 3.1. – Initialisation du scénario utilisé. Chacun des trois robots est asservi à suivre son propre chemin à une vitesse donnée. Les amers sont positionnés aux quatre coins de la grille, espacés de 10 m, et l'amer positionné en haut à droite est parfaitement localisé.

Les robots peuvent observer les amers dans un rayon de 6 m, à une fréquence de 5 Hz. Les mesures d'odométrie ont une fréquence de 25 Hz. Les variances des bruits d'observations et d'odométrie sont de 10^{-3} m^2 et 10^{-3} m^2/s^2, respectivement. La covariance du bruit du modèle associé à la partie p, Q_p, est choisie avec une valeur conservatrice et les amers imprécis ont une variance de 0,1 m^2.

Dans ce chapitre, nous ne nous focaliserons pas sur les valeurs moyennes des erreurs en tant que telles, mais sur les comparaisons entre les méthodes pour évaluer l'apport de la collaboration pour la localisation et pour l'estimation de la carte, par rapport à une méthode isolée, où chaque robot utilise seulement ses propres observations locales. Il convient de noter que les erreurs sont plus ou moins grandes suivant des réglages de la simulation.

Afin de ne pas dépendre du tirage aléatoire des bruits de mesure, les résultats présentés dans cette section sont une moyenne de 100 exécutions. À chaque pas de temps, l'erreur moyenne des 100 exécutions est calculée, ainsi que la courbe de covariance moyenne.

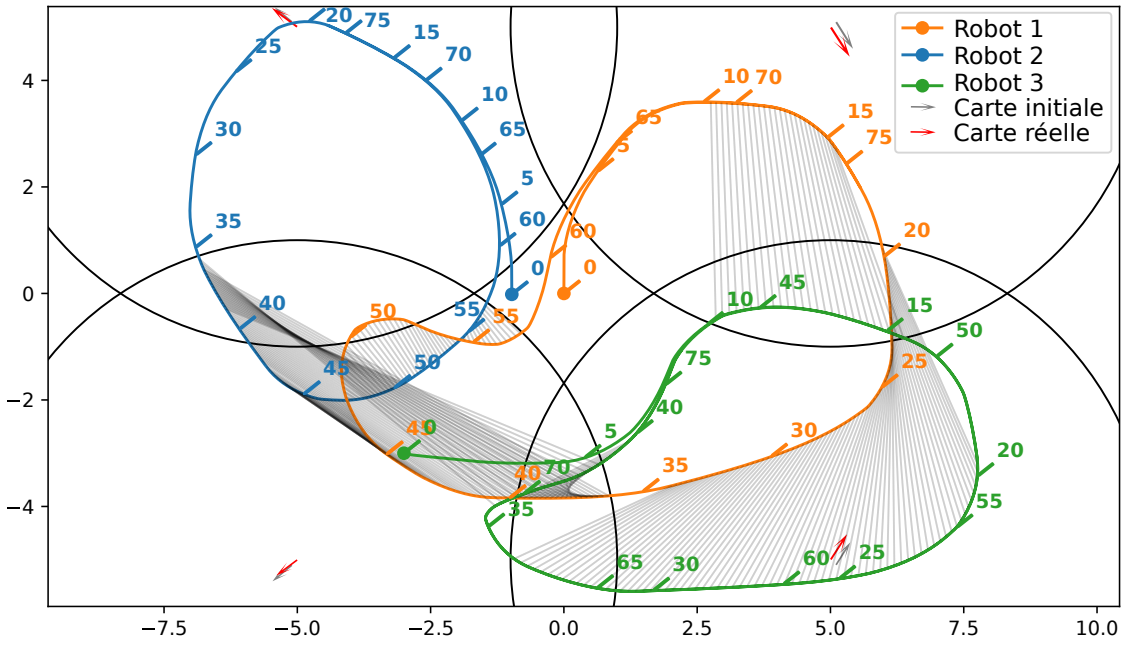


FIGURE 3.2. – Différentes situations se produisant avec les trajectoires choisies. Les positions initiales des robots sont représentées par les disques. Les flèches représentent les amers et le cercle noir montre la portée à laquelle ils peuvent être détectés. La carte représentée ici est celle sans amers parfaitement localisés, utilisée à la section 3.3.2. Les points des trajectoires sont reliés par un trait noir lorsqu'ils peuvent collaborer.

3.3.1. Impact de la collaboration dans le cas d'un amer parfaitement localisé

Scénario

Les trajectoires d'une durée de 75 s permettent d'avoir plusieurs situations de collaboration. La figure 3.2 illustre les situations de collaboration à 2 ou 3 robots, celles quand les robots sont seuls à observer un amer, ainsi que les phases d'odométrie seule. Cela permet de mettre en avant le fonctionnement de la collaboration, notamment pour la mise à jour de la carte alors que certains robots n'observent pas tous les amers.

Au départ, les robots 1 et 2 se localisent avec leur odométrie tandis que le robot 3 commence par des observations de l'amer 4 jusqu'à $t = 5{,}5$ s. Le robot 2 commence à observer l'amer 3, seul, à partir de $t = 1{,}9$ s jusqu'à $t = 40$ s. Le robot 1 observe l'amer 1 qui est parfaitement positionné, à partir de $t = 3$ s jusqu'à $t = 23$ s. Il est rejoint par le robot 3 à $t = 10$ s et les deux collaborent en observant l'amer parfait jusqu'à $t = 15{,}5$ s où le robot 1 ne l'observe plus. Pendant ce temps, le robot 3 observe l'amer 2, depuis $t = 2$ s, et il continue de l'observer jusqu'à $t = 34$ s. Le robot 3 observe donc à la fois l'amer parfait et l'amer 2 de $t = 10$ s à $t = 15{,}5$ s.

Peu de temps après, à $t = 19$ s, le robot 1 observe aussi l'amer 2 et reprend sa collaboration avec le robot 3 à travers cet amer, jusqu'à ce que le robot 3 ne l'observe plus (à $t = 34$ s). Le robot 3 recommence à observer l'amer 4 à partir de $t = 31$ s, tout en continuant d'observer l'amer 2.

À $t = 35{,}5$ s, le robot 2, qui reste pour l'instant de son côté, observe l'amer 4 en même temps que le robot 3 et entre donc en collaboration avec ce robot. À $t = 36$ s, le robot 1 rejoint les deux autres robots en observant l'amer 4 et les trois robots entrent donc en collaboration, jusqu'à ce que le robot 3 s'écarte et n'observe plus l'amer 4, à $t = 39{,}5$ s. Les robots 1 et 2 continuent de collaborer en observant l'amer 3 en plus de l'amer 4 à partir de $t = 50$ s. Le robot 1 observe cet amer jusqu'à $t = 56{,}5$ s, puis se retrouve en odométrie, avant d'observer de nouveau l'amer 1, parfait, à partir de $t = 61{,}5$ s. À ce moment-là, le robot 3 aura déjà observé de nouveau l'amer 2 de $t = 37$ s à environ $t = 67$ s, et l'amer parfait de $t = 44$ s à $t = 49$ s.

Le scénario est le même en mode isolé mais sans phase de collaboration. Les robots ne communiquent pas. Ils n'utilisent donc que leurs observations locales.

Résultats

Les erreurs des robots sont présentées sur la figure 3.3.

Tout d'abord, avant la première phase de collaboration à $t = 10$ s, les deux méthodes se comportent de la même façon, ce qui est attendu, car la méthode collaborative n'a pas de données supplémentaires à utiliser par rapport au mode isolé.

Pour les robots 1 et 3, les résultats sont similaires entre la méthode collaborative et la méthode isolée, avec plus de dynamique au niveau de la méthode collaborative. Cela est dû au fait que ces deux robots observent l'amer parfaitement localisé (le 1) dès le début

Méthodes \ Robots	Robot 1	Robot 2	Robot 3	Tous
Collaborative	2,26	**4,73**	2,42	**3,14**
Isolée	**2,20**	7,64	**2,27**	4,04

TABLE 3.1. – Erreurs (cm) des robots en fonction de la méthode, avec l'amer 1 parfaitement positionné.

(à $t = 3$ s pour le robot 1 et à $t = 10$ s pour le robot 3), ce qui se propage à l'estimation de la carte et à la localisation de ces robots.

Cependant, une différence très significative peut être observée pour le robot 2. L'intérêt de la collaboration est largement visible pour ce robot, qui n'observe pas l'amer parfait. L'utilisation de la collaboration a permis à ce robot de profiter de l'information de cet amer d'une façon indirecte, dès le moment où il collabore avec le robot 3, à partir de $t = 35{,}5$ secondes, ce qui n'est pas le cas en mode isolé. La légère augmentation de l'erreur en mode isolé est due à l'erreur de localisation de l'amer 4.

Ces résultats sont confirmés à travers les erreurs moyennes dans la table 3.1 : les erreurs moyennes des robots 1 et 3 ne diffèrent pas entre modes collaboratif et isolé. Par contre, la collaboration fait passer l'erreur de 7,64 cm en isolé à 4,73 cm en collaboratif pour le robot 2. À noter que l'amélioration au niveau du robot 2 apparait au moment de la collaboration et donc ne concerne pas toute la trajectoire, ce qui explique une erreur moyenne plus grande pour ce robot en comparant avec les deux autres.

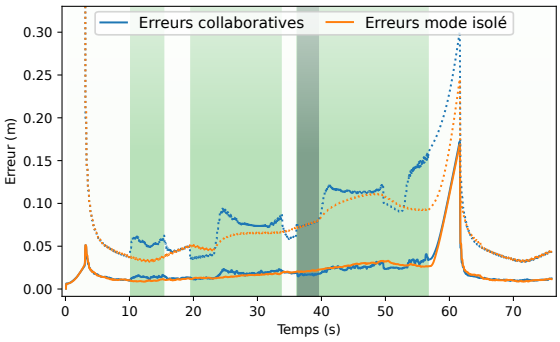
Les résultats de la mise à jour de la carte sont présentés aux figures 3.4 et 3.5, en sélectionnant les courbes les plus informatives. Le positionnement de l'amer 2 est amélioré assez rapidement en mode isolé et collaboratif, avec des résultats assez similaires, du fait de son placement (figure 3.4a et 3.4c). Cette grande similarité s'explique par le fait que l'amer 2 est observé en même temps que l'amer 1 parfait par les robots 1 et 3. Cela propage la très bonne précision de l'amer 1 à l'amer 2, que ce soit en collaboratif ou en mode isolé.

En mode isolé, lorsqu'un robot n'observe pas certains amers, il ne peut pas les corriger dans sa carte. La méthode collaborative quant à elle peut les améliorer grâce aux autres robots, ce qui lui donne un avantage immédiat de ce point de vue. C'est le cas pour l'amer 2 pour le robot 2 et l'amer 3 pour le robot 3 qui sont trop éloignés pour être détectés par ces robots. L'exemple de l'amer 2 du robot 2 est montré au niveau de la figure 3.4b.

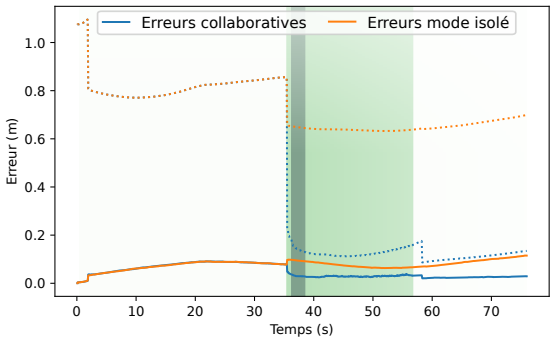
Sur toutes les courbes, l'estimation des amers par la méthode collaborative termine avec une erreur moindre par rapport à la méthode isolée. De plus, les erreurs d'estimation des amers diminuent jusqu'à converger vers une erreur moyenne finale de 1,6 cm en collaboratif (contre 4,1 cm en mode isolé). En mode isolé, certains amers convergent, notamment pour les robots 1 et 3, mais le robot 2 ne parvient pas à diminuer l'erreur de ces différents amers.

La table 3.2 montre les erreurs moyennes des amers. La méthode collaborative permet d'avoir une meilleure estimation de la position de tous les amers, individuellement et

(a) Robot 1.



(b) Robot 2.



(c) Robot 3.

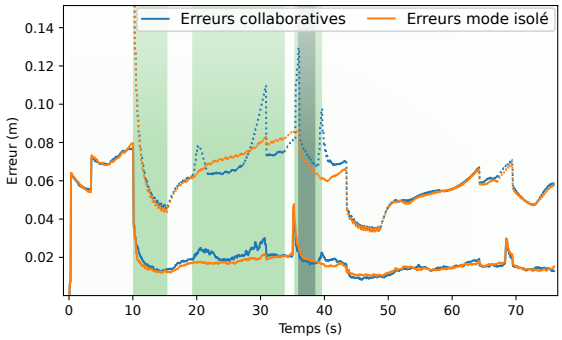
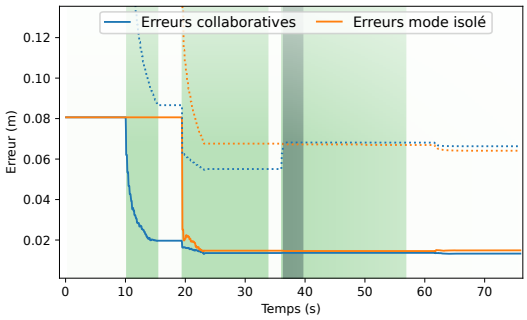
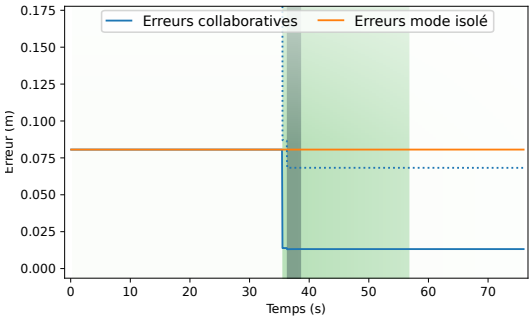


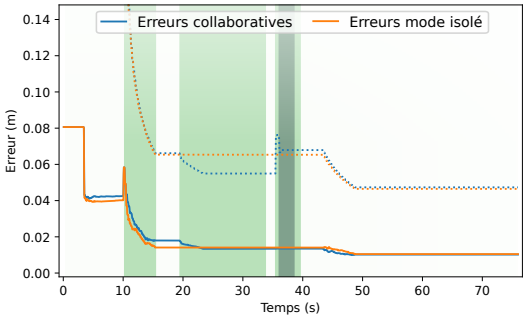
FIGURE 3.3. – Erreurs euclidiennes de position des trois robots, moyennées sur 100 exécutions et avec la représentation de leur covariance. Les courbes de covariance sont représentées en pointillé, de la même couleur, à 4σ, où σ est la plus grande valeur propre de la matrice de covariance. La distance euclidienne suivant ici une distribution de Rayleigh, cela correspond à une incertitude à 99,97 %. Les bandes vertes représentent les moments de collaboration (pour le cas collaboratif) du robot représenté. Le vert est clair lorsque le robot représenté collabore avec un autre robot, et il devient foncé lorsqu'il collabore avec les deux autres.



(a) Robot 1.



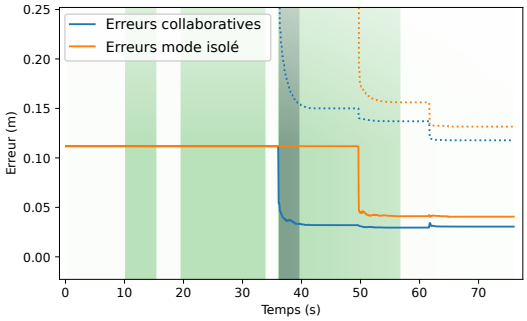
(b) Robot 2.



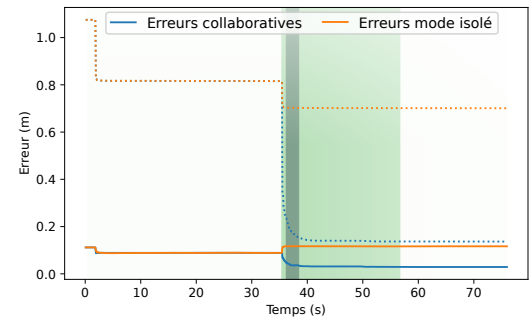
(c) Robot 3.

FIGURE 3.4. – Erreurs d'estimation de l'amer 2 (en trait plein), dans le cas d'un amer parfaitement localisé, régions d'incertitude à 99,97 % (en pointillé).

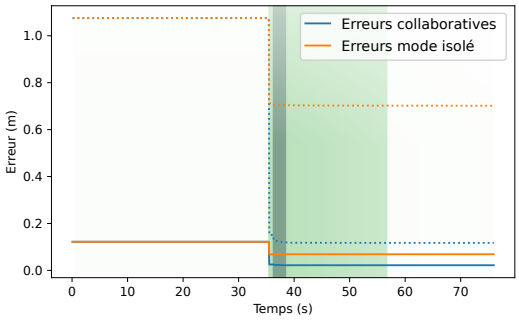
(a) Amer 3 – Robot 1.



(b) Amer 3 – Robot 2.



(c) Amer 4 – Robot 2.



(d) Amer 4 – Robot 3.

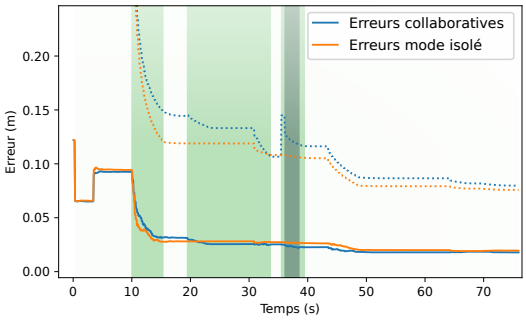


FIGURE 3.5. – Erreurs de position des amers 3 et 4 avec l'amer 1 parfaitement localisé.

Méthodes \ Amers	Amer 2	Amer 3	Amer 4	Tous (hors 1)
Collaborative	**2,99**	**6,65**	**4,73**	**4,79**
Isolée	4,37	10,10	6,66	7,04

TABLE 3.2. – Erreurs (cm) des amers en fonction de la méthode, avec un amer (le 1) parfaitement localisé dans la carte initiale.

collectivement. Ces moyennes considèrent tous les robots, ce qui avantage naturellement la méthode collaborative puisque des amers ne sont pas observés par certains robots, et conservent donc leur erreur initiale.

Du point de vue de l'estimation de la covariance, toutes les méthodes et tous les robots et amers calculent une covariance qui englobe correctement l'erreur. On peut noter que les covariances sont surévaluées car 100 % des erreurs sont dans l'enveloppe de la covariance à 99 %. Les comparaisons faites au niveau de l'erreur se retrouvent au niveau de la covariance : les robots 1 et 3 n'apportent pas une amélioration au niveau de l'estimation de la covariance par rapport au mode isolé, à cause de l'observation de l'amer parfait qui permet de réduire les valeurs de covariances (figures 3.3a et 3.3c). Cette absence d'amélioration se note aussi au niveau de l'amer 2 pour les robots 1 et 3 (figures 3.4a et 3.4c), dont le seul avantage de la collaboration est une convergence plus rapide. Cependant, la réduction de la covariance du robot 2 (figure 3.3b) est bien claire tout en conservant la consistance, phénomène qu'on observe aussi au niveau de son estimation de l'amer 2 (figure 3.4b).

Cette réduction montre l'intérêt de la collaboration au niveau du calcul des régions d'incertitude qui permet de réduire le pessimisme par rapport au mode isolé, en fusionnant plus de données. Les petites augmentations observées au niveau des covariances des robots 1 et 3 en mode collaboratif par rapport à la méthode isolée peuvent s'expliquer par l'utilisation de la KLA qui peut conduire à une certaine perte d'information pour ces robots. C'est aussi le cas pour l'amer 4 estimé par le robot 3 (figure 3.5d). Dans une situation où la collaboration apporte déjà peu, la KLA peut rendre l'estimation plus pessimiste.

Le scénario a été prolongé pendant une dizaine de minutes, avec les robots continuant les boucles. L'estimation est restée consistante, mais les situations intéressantes étaient moins présentes, car le scénario n'a pas été étudié pour proposer des situations intéressantes aussi longtemps.

Le scénario a aussi été testé en odométrie seule et les estimations des différents robots dérivent logiquement avec une perte de consistance pour les robots 2 et 3. Cela peut s'expliquer par la linéarisation du modèle avec des trajectoires comportant des virages serrés, ce qui conduit à une augmentation des erreurs.

Méthodes \ Robots	Robot 1	Robot 2	Robot 3	Tous
Collaborative	**4,10**	**9,84**	**5,21**	**6,39**
Isolée	7,58	14,12	7,17	9,62

TABLE 3.3. – Erreurs (cm) des robots en fonction de la méthode, sans amers parfaitement localisés.

	Amer 1	Amer 2	Amer 3	Amer 4	Tous	Tous final	Biais moyen	Biais final
Collaborative	**4,89**	**5,06**	**7,33**	**7,22**	**6,12**	**1,03**	**0,20**	**0,035**
Isolée	9,00	9,49	11,68	12,02	10,54	7,99	0,32	0,205

TABLE 3.4. – Erreurs (cm) moyennes des amers en fonction de la méthode, sans amers parfaitement localisés. La table montre aussi les erreurs moyennes des amers ainsi que leur biais au denier instant (final).

Conclusion

L'intérêt de la collaboration a été mis en évidence à travers cette première simulation. Les résultats de la collaboration sur chaque robot ne sont pas forcément améliorés quel que soit le scénario considéré. Cependant, la collaboration apporte une amélioration au niveau du système global. Par exemple, l'effet de la collaboration est très clair dans notre scénario sur le robot 2 qui est dans une situation moins avantageuse par rapport aux autres.

Ces résultats montrent aussi que la collaboration permet à la carte de converger vers une estimation correcte et ainsi, proposer une meilleure estimation des amers observés pour améliorer la localisation future des robots.

3.3.2. Cas d'une carte sans amers parfaitement localisés

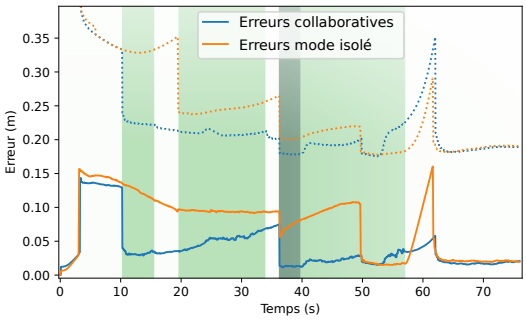
Après avoir étudié l'apport de la collaboration dans le cas d'un amer parfaitement localisé, permettant de faire converger la carte et d'avoir globalement une bonne précision à la fin de la simulation, nous allons ici évaluer l'impact d'une carte initiale, incertaine et avec un moyen biais nul sur la localisation et sur la correction de la carte.

Les résultats sont montrés sur le même scénario et les mêmes paramètres que dans la section 3.3.1, mais la carte initiale contient des amers positionnés de manière erronée, avec une moyenne d'erreurs vectorielles nulle. Les covariances initiales des amers représentent correctement leurs erreurs. Les résultats sont toujours moyennés sur 100 exécutions.

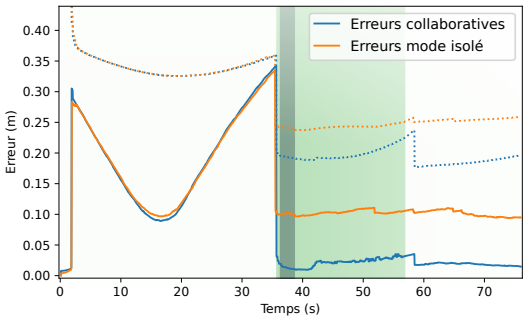
Résultats

Les erreurs des robots sont montrées au niveau de la figure 3.6. La différence entre la méthode collaborative et le mode isolé est bien visible, notamment au niveau des

(a) Robot 1.



(b) Robot 2.



(c) Robot 3.

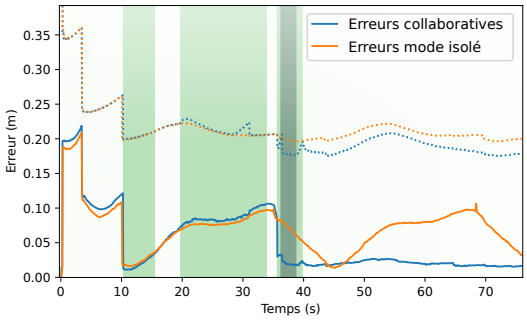


FIGURE 3.6. – Erreurs euclidiennes de position des robots sans amers parfaitement localisés.

robots 1 et 2. Le robot 3 montre un comportement similaire entre les deux méthodes jusqu'à $t = 35{,}5$ secondes, alors qu'il collabore avec le robot 1. Cela s'explique par ses observations. En effet, le robot 3 observe les amers 1 et 2 simultanément, ce qui donne déjà beaucoup d'informations, laissant peu d'intérêt aux observations collaboratives supplémentaires. Sur la fin de l'exécution, les robots 2 et 3 montrent une meilleure localisation avec la collaboration, ainsi qu'une incertitude plus faible. Le robot 1 présente une fin similaire entre les deux modes, ce qui s'explique par son observation de tous les amers, le mode isolé étant aussi capable de corriger l'ensemble de la carte.

La table 3.3 fournit les erreurs moyennes des robots où une réduction est constatée après la collaboration. L'erreur est plus importante qu'à la section 3.3.1 du fait de l'absence de l'amer parfaitement localisé.

Les figures 3.7 et 3.8 présentent les erreurs d'estimation des amers 1 à 4 sous forme d'erreur euclidienne sur la position, par les trois robots, avec les méthodes collaborative et isolée. Comme dans le premier exemple, lorsque le robot n'observe pas directement l'amer, il n'est pas mis à jour avec la méthode isolée (figure 3.7d). L'absence d'un amer parfaitement localisé conduit à des résultats avec des erreurs plus grandes que celles présentées dans la section 3.3.1. En effet, aucun point de repère parfait n'existe permettant de corriger les incertitudes des autres éléments.

La table 3.4 montre les erreurs moyennes et finales des amers. L'erreur finale de la carte passe de 8,0 cm en mode isolé à 1,0 cm en mode collaboratif, tandis que la valeur

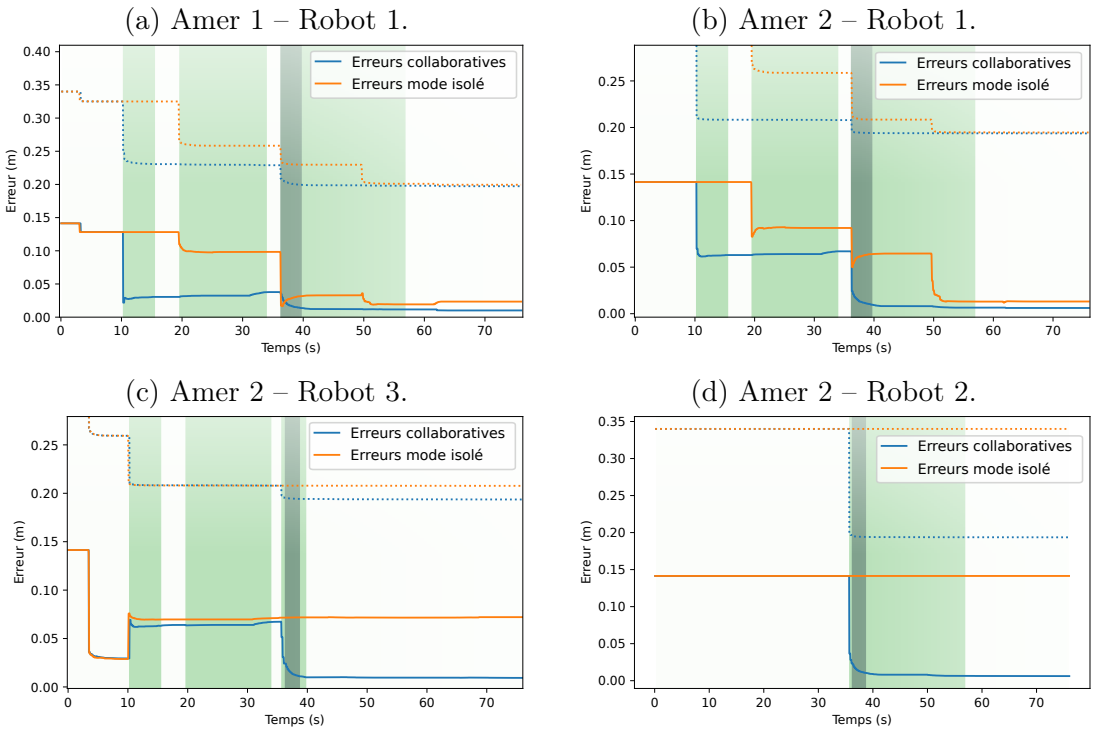


FIGURE 3.7. – Erreurs de position des amers 1 et 2 avec une carte sans amers parfaitement localisés.

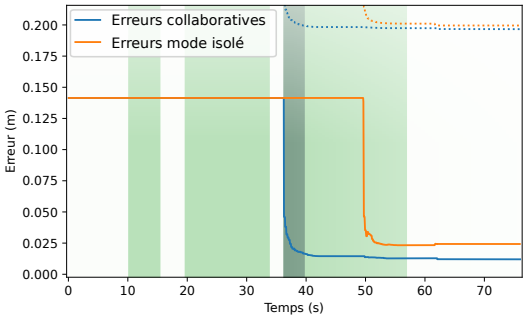
moyenne passe de 10,54 cm à 6,12 cm. Cela confirme les observations effectuées sur les courbes.

La collaboration joue un rôle primordial dans ce scénario pour corriger la carte à plus grande échelle, pour tous les robots. Par exemple, les robots 2 et 3 n'observant pas l'ensemble des amers, ils ne peuvent pas faire converger la carte vers une faible erreur en mode isolé. Cet effet est particulièrement visible au niveau du robot 2 car l'erreur finale reste particulièrement élevée (figure 3.8b).

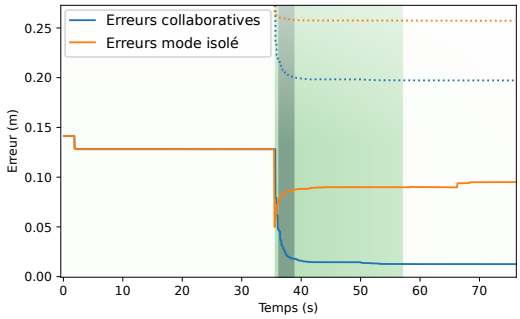
La mise à jour de la carte avec la méthode collaborative conduit à un biais final de 0,4 mm qui est beaucoup plus faible que la méthode isolée qui conduit à une valeur de 2,1 mm, sachant que le biais initial de la carte est nul. Pour le calcul du biais, la moyenne des erreurs de la carte suivant x et y est déterminée à chaque instant. Sa norme euclidienne est ensuite calculée, ce qui donne le biais de la carte à cet instant. Le biais moyen est la moyenne de ces biais sur l'ensemble de la trajectoire. L'amélioration est significative et elle de l'ordre de -83%.

Concernant les régions d'incertitude, comme pour l'exemple précédent, la collaboration permet de réduire les covariances par rapport au mode isolé tout en conservant la consistance.

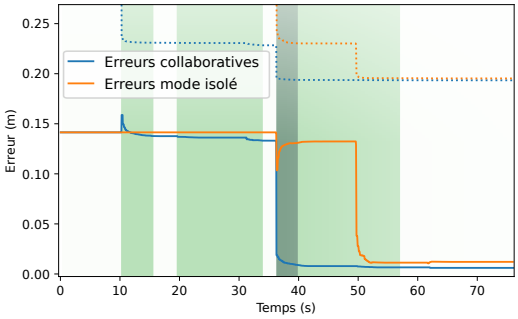
(a) Amer 3 – Robot 1.



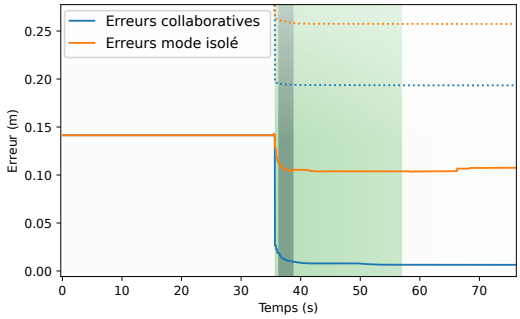
(b) Amer 3 – Robot 2.



(c) Amer 4 – Robot 1.



(d) Amer 4 – Robot 2.



(e) Amer 4 – Robot 3.

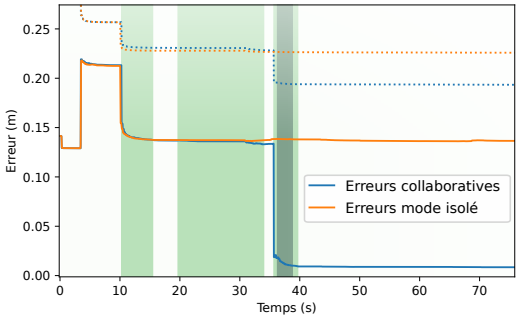


FIGURE 3.8. – Erreurs de position des amers 3 et 4 avec une carte sans amers parfaitement localisés.

Paramètres \ Robots	Robot 1	Robot 2	Robot 3	Tous
$Q_{p,1}$	2,36	4,92	2,48	3,26
$Q_{p,2}$	**2,12**	**4,80**	**2,33**	**3,08**
$Q_{p,3}$	29,32	38,52	52,39	40,08

TABLE 3.5. – Erreurs des robots (en cm) en fonction du Q_p utilisé.

Conclusion

Avec l'utilisation d'une carte imprécise, nous avons montré que la collaboration est encore très pertinente pour améliorer la localisation des robots et des amers. Cela permet d'estimer correctement les positions relatives des amers et ainsi de converger vers le biais de la carte initiale. Dans une utilisation plus complète couplée à des mesures absolues, type GNSS par exemple, il serait ainsi possible de corriger le biais de la carte.

3.3.3. Impact du paramètre Q_p

Dans cette section, nous étudions l'effet du paramètre Q_p (covariance de l'erreur de modèle des robots dans la partie p) sur les résultats. Théoriquement, une valeur de Q_p qui s'approche de $\mathbb{0}$ peut conduire à une inconsistance et donc à une dégradation de la performance de la KLA. En effet, si le robot en collaboration possède une estimation erronée de la pose de l'ego-robot avec une petite covariance, la KLA va conduire à un mauvais résultat.

Nous reprenons ici les paramètres du scénario dans le cas d'un amer parfaitement localisé. Seules 10 exécutions sont réalisées avec trois valeurs de Q_p différentes.

$$Q_{p,1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0{,}5 \end{pmatrix}$$
$$Q_{p,2} = \begin{pmatrix} 0{,}3 & 0 & 0 \\ 0 & 0{,}3 & 0 \\ 0 & 0 & 0{,}1 \end{pmatrix}$$
$$Q_{p,3} = \mathbb{0}$$

$Q_{p,1}$ correspond à la valeur utilisée dans les résultats précédents, volontairement exagérée et $Q_{p,3}$ est le cas où aucun bruit Q_p n'est considéré. $Q_{p,2}$ est une valeur arbitraire entre les deux.

Seules des exécutions en mode collaboratif sont considérées comme l'effet de Q_p n'intervient pas dans les autres méthodes. Les résultats sont moyennés cette fois-ci, sur 10 exécutions.

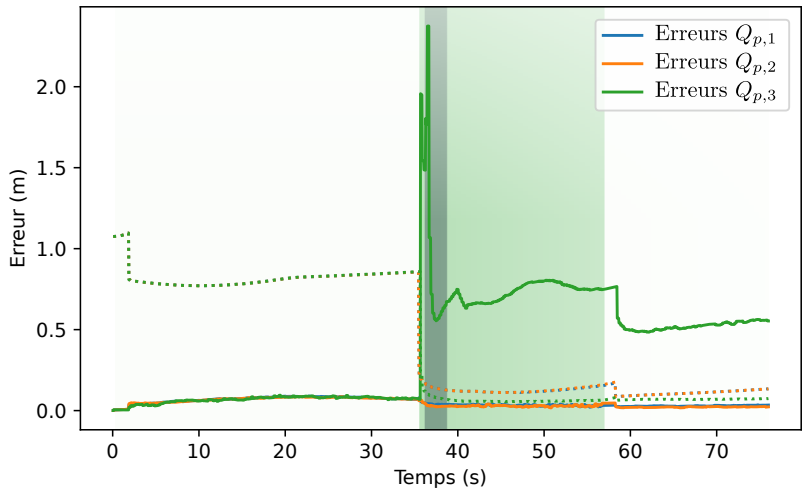


FIGURE 3.9. – Erreur de position du robot 2 avec les différentes valeurs de Q_p (traits pleins), covariance (en pointillé).

Les erreurs des robots présentées à la table 3.5 ne montrent pas de différences notables entre les différentes valeurs de Q_p, lorsque celui-ci est différent de $\mathbb{0}$. En effet, comme montré à la figure 3.9, l'absence de Q_p lors de la collaboration provoque une augmentation très importante de l'erreur de localisation. Cela montre l'importance de l'utilisation de Q_p pour conserver un état consistant avant la fusion avec la KLA. Pire encore, la covariance estimée sans Q_p sous-estime largement l'erreur, entrainant ainsi une perte d'intégrité de l'estimation.

On observe une faible différence entre $Q_{p,1}$ et $Q_{p,2}$, ce qui montre que, bien qu'il soit nécessaire, il n'y a pas de grande perte de performance lorsque Q_p est réglé très grand. Ce n'est donc pas un paramètre très sensible. Même au niveau des covariances, la KLA réagit correctement pour qu'un grand Q_p n'ait pas d'effet significatif.

3.4. Résultats avec l'étape de détection et d'isolation des défauts

Le même scénario que dans la section 3.3.2 est considéré. Tous les résultats de cette section sont présentés avec une seule exécution. La valeur initiale du générateur aléatoire des bruits a été fixée.

3.4.1. Évaluation avec des défauts observations

Afin de pouvoir évaluer notre méthode de FDIR, des défauts simulés sont injectés au niveau des observations au moment de la réception de l'observation du capteur par le robot. Les défauts sont injectés d'une façon aléatoire dans la plage 10–15 s et 20–40 s pour les trois robots, avec une probabilité d'apparition de 30 %. Les défauts sont injectés

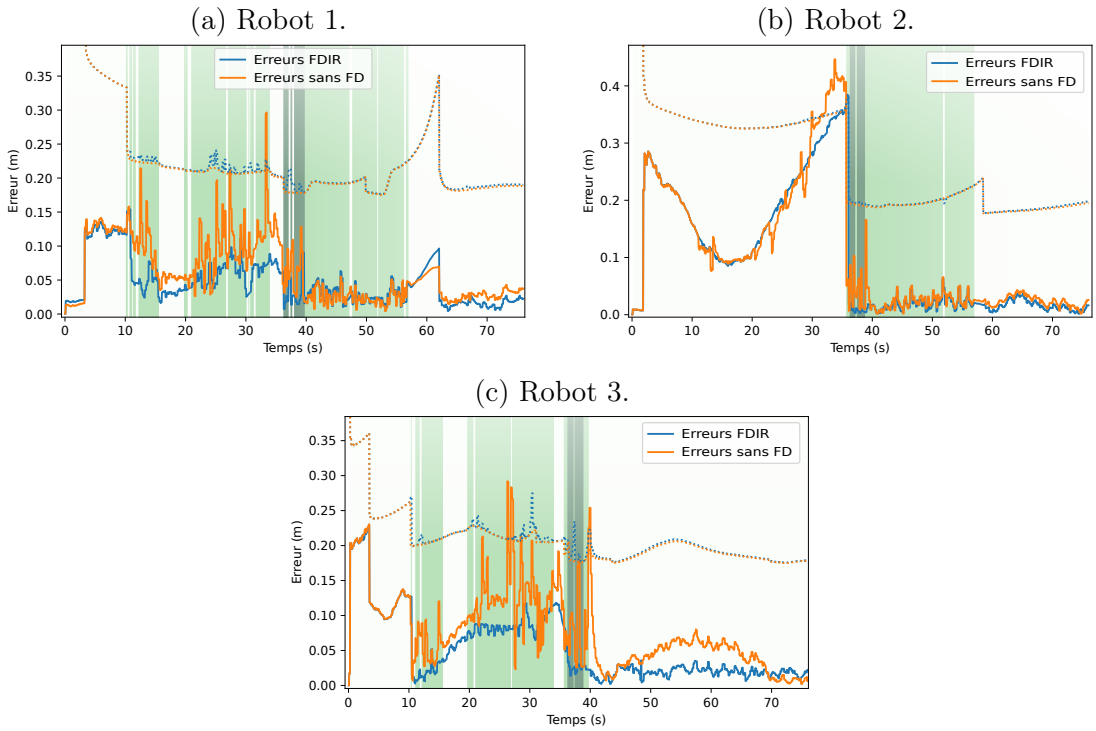


FIGURE 3.10. – Erreurs de position des robots lors de l'injection de défauts d'observation.

avec une valeur moyenne de biais de 0,2 m et de variance de 0,05 m^2 tout en vérifiant notre hypothèse : deux défauts simultanés au maximum.

Les défauts injectés sont enregistrés pour pouvoir être rejoués plusieurs fois, si besoin. De plus, certaines configurations ont été manuellement ajoutées : un défaut continu a été ajouté au robot 3 dans la plage de 11 à 12 secondes.

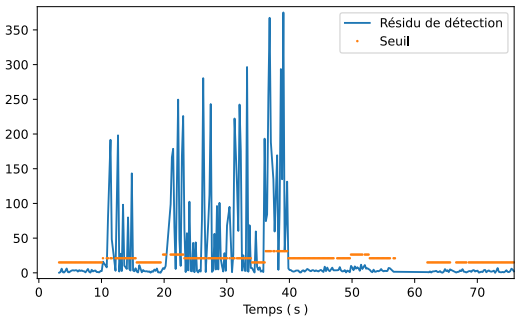
On rappelle quelques conditions et hypothèses formulées pour notre méthode de FDIR : le défaut d'amer peut être isolé lorsque au moins deux robots l'observent, et un défaut simultané d'une observation et de l'amer observé est peu probable. Dans le cas contraire, le défaut d'observation est privilégié.

Les erreurs des robots sont montrées à la figure 3.10 avec et sans détection de défauts. L'amélioration au niveau des erreurs avec la détection et l'exclusion des défauts est visible sur ces figures. L'absence de module de détection de défaut rend le système beaucoup moins précis et conduit à des augmentations ponctuelles des erreurs. Ces erreurs dépassent alors fréquemment la covariance théorique qui ne reflète plus alors la réalité physique.

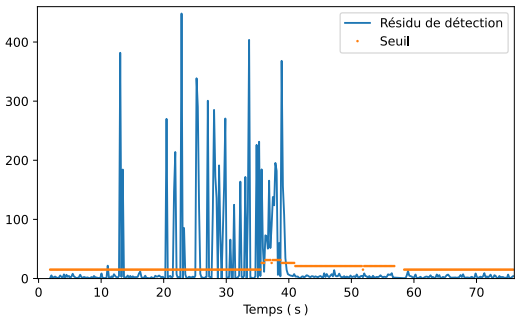
La table 3.6 montre la valeur moyenne des erreurs pour les robots et les amers. On observe que la méthode avec FDIR produit de meilleurs résultats pour tous les robots et pour tous les amers.

Les figures 3.11 montrent les résidus utilisés pour la détection de défauts au niveau de chaque robot. Lorsque le résidu de détection au niveau d'un robot i dépasse le seuil, les résidus d'observations et d'amers sont calculés pour déterminer l'origine du défaut. Le seuil est déterminé à partir d'une distribution de χ^2_n où n est le degré de liberté choisi à

(a) Robot 1.



(b) Robot 2.



(c) Robot 3.

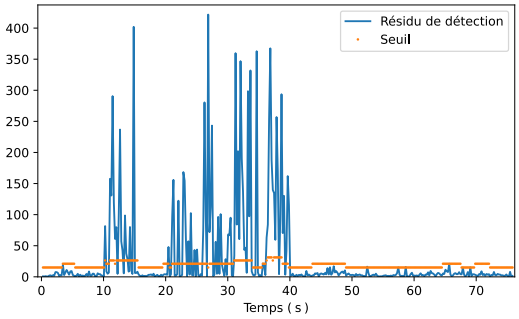


FIGURE 3.11. – Résidus de détection lors de l'injection de défauts d'observation. Le seuil à chaque instant est représenté par les points orange, variable en fonction des observations.

	FDIR	Sans FD
Robot 1	4,61	6,19
Robot 2	9,97	10,48
Robot 3	5,28	7,82
Tous les robots	6,62	8,16
Amer 1	4,60	7,65
Amer 2	4,98	6,96
Amer 3	7,06	7,49
Amer 4	6,98	8,22
Tous les amers	5,91	7,58

TABLE 3.6. – Erreurs (cm) des robots et des amers, sans amer parfaitement positionné, lors de l'injection de défauts d'observation. Nous comparons les erreurs avec toute l'étape de FDIR et sans aucune détection de défaut (FD).

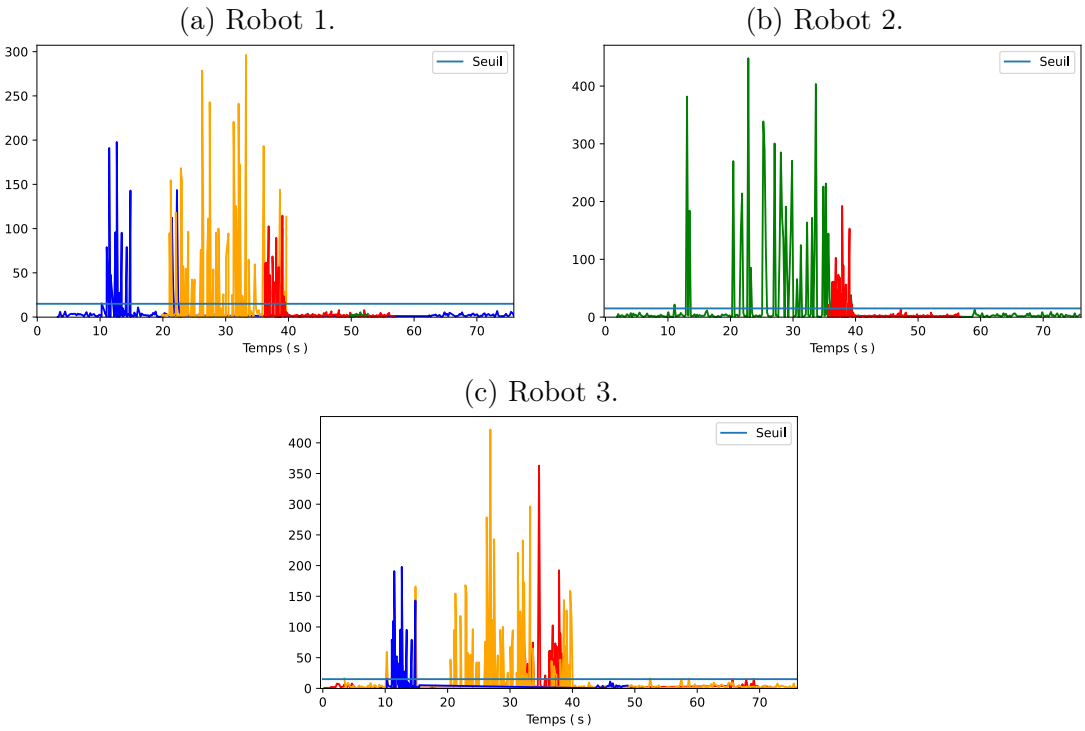


FIGURE 3.12. – Résidus d'observation des robots lors de l'injection de défauts d'observation.

partir du rang de la matrice d'observation H, $n = \text{rang}(H)$, et avec une probabilité de fausse alarme de 0,2 %. L'utilisation du rang(H) pour le degré de liberté conduit aux meilleurs résultats. Cette procédure de détection de défaut est décentralisée et se réalise au niveau de chaque robot indépendamment des autres.

Les résidus d'observations sont montrés à la figure 3.12, avec le seuil de détection. Chaque robot calcule individuellement ses résidus d'observation. En parallèle, lorsque des robots collaborent, chacun calcule les résidus des observations reçues et d'amers. Un vecteur binaire des résidus est déterminé après une étape de seuillage, en utilisant une loi de χ^2 à 3 degrés de liberté avec la même probabilité de fausse alarme de 0,2 %. Ce vecteur est ensuite comparé à la matrice de signatures pour déterminer l'origine du défaut. Ici, seuls des défauts d'observations sont isolés, et la grande majorité des défauts injectés ont été isolés et exclus.

Pour étudier le comportement de notre méthode en présence de deux défauts, la simulation comporte des cas avec des défauts simultanés injectés. Pour les défauts d'observation, nous pouvons étudier deux cas. Le premier cas correspond à deux défauts d'observation simultanés provenant du même robot, donc des observations de différents amers. La figure 3.13 montre un exemple de ce cas qui a été produit à $t = 33{,}7$ s lorsque deux défauts sont injectés au niveau des observations des amers 2 et 4 du robot 3. Les deux résidus d'observation ont réussi à détecter la présence d'un problème en dépassant leur seuil. Via la matrice de signatures (table 3.7), le défaut correspondant est celui

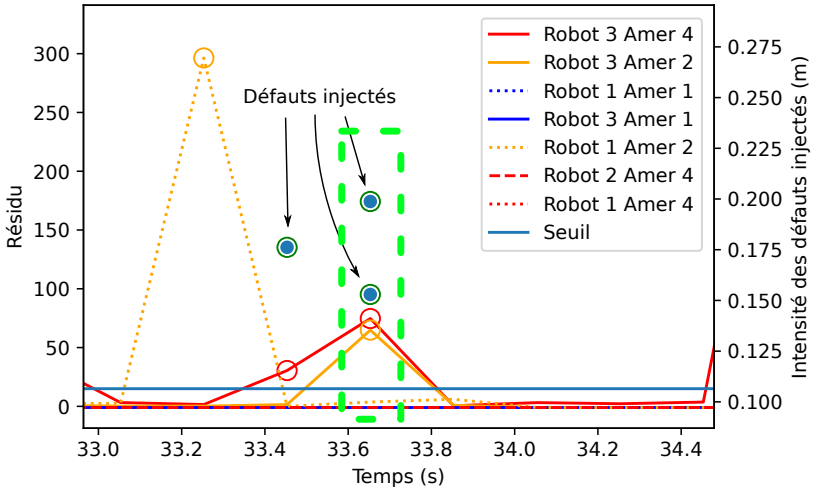


FIGURE 3.13. – Zoom sur les résidus lorsque deux défauts d'observations apparaissent simultanément sur le robot 3, encadrés en vert. Les disques bleus représentent l'intensité des défauts injectés.

Résidus \ Défauts	Amers		Observations de v_3	
	l_2	l_4	$Z_{l_2}^{v_3}$	$Z_{l_4}^{v_3}$
$r_{l_2}^{v_3} = 1$	1	0	1	0
$r_{l_4}^{v_3} = 1$	0	1	0	1

TABLE 3.7. – Matrice de signatures lorsque deux défauts d'observations sont présents sur un même robot. Les défauts d'amers ne sont pas pris en compte lorsqu'il n'y a pas plusieurs observations du même amer, possible uniquement en collaboration.

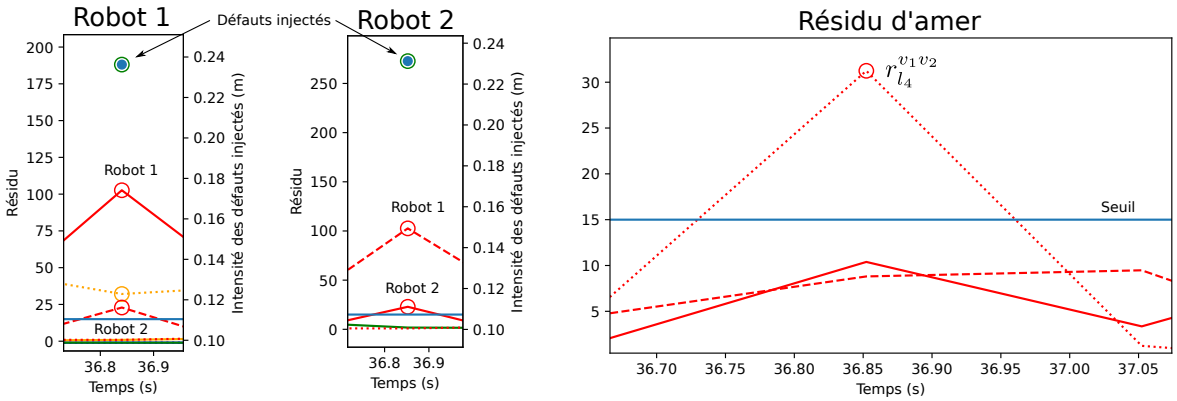


FIGURE 3.14. – Zoom sur les résidus lorsque deux défauts d'observation apparaissent simultanément sur l'amer 4, par les robots 1 et 2. De gauche à droite, il y a les résidus d'observation du robot 1, les résidus d'observations du robot 2 et le résidu d'amer (par le robot 1).

Résidus \ Défauts	Amers		Observations			Défaut multiple
	l_2	l_4	$Z_{l_4}^{v_1}$	$Z_{l_4}^{v_2}$	$Z_{l_2}^{v_3}$	$Z_{l_4}^{v_1}$ et $Z_{l_4}^{v_2}$
$r_{l_4}^{v_1} = 1$	0	1	1	0	0	1
$r_{l_4}^{v_2} = 1$	0	1	0	1	0	1
$r_{l_2}^{v_3} = 0$	1	0	0	0	1	0
$r_{l_4}^{v_1v_2} = 1$	0	0	1	1	0	1

TABLE 3.8. – Matrice de signatures lorsque deux défauts d'observations sont présents pour un même amer, vue par le robot 1.

combinant les défauts des deux observations locales qui seront alors exclues. En absence de collaboration, la détection des défauts amers n'est pas possible et donc les signatures des défauts amers qui sont les mêmes que les défauts observations ne sont pas prises en compte.

Un deuxième type de défauts simultané correspond à deux défauts d'observations d'un même amer par deux robots différents. Ce cas s'est produit à $t = 36{,}85$ s et est présenté à la figure 3.14. Les deux résidus d'observation de l'amer 4 par les deux robots 1 et 2 dépassent leur seuil, et le résidu d'amer 4, comparant les deux poses de l'amer 4 comme calculées par les deux robots, dépasse aussi son seuil. En comparant avec la matrice de signatures (figure 3.8), ce cas correspond aux défauts observations qui seront toutes les deux exclues. Nous rappelons qu'un défaut simultané d'une observation et de l'amer observé est considéré peu probable. Notre méthode a donc bien réagi dans le cas des défauts observations.

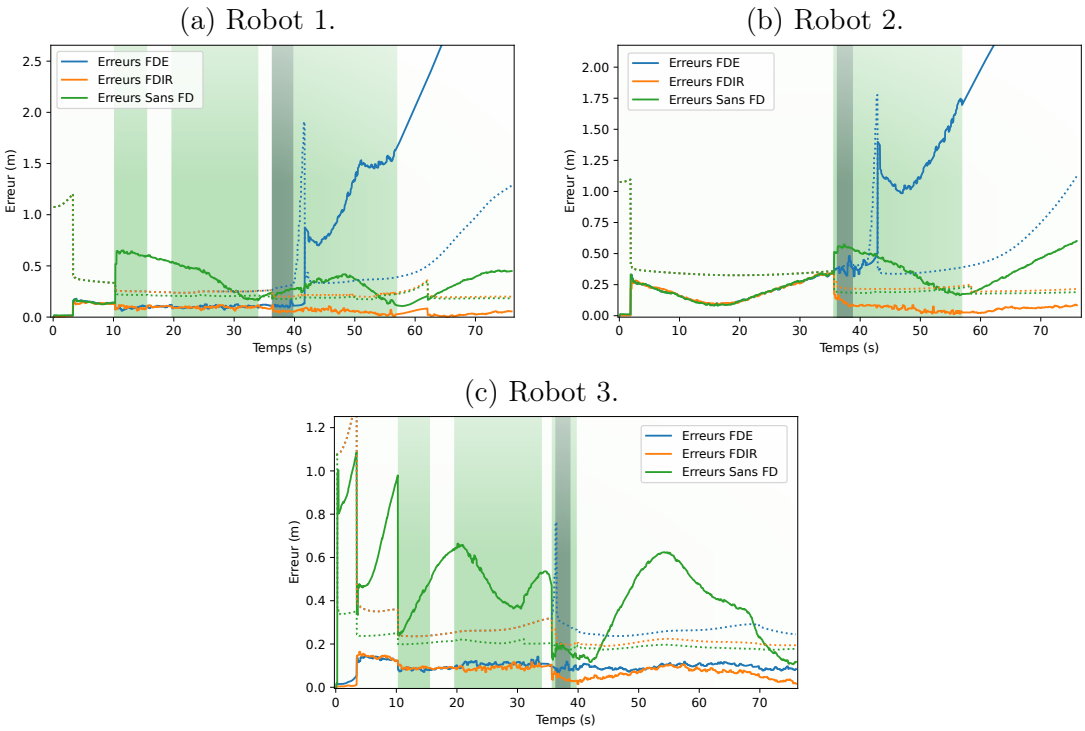


FIGURE 3.15. – Erreurs de position des robots lors de l'injection d'un défaut de carte, avec l'isolation de défaut de carte, avec uniquement l'exclusion d'observations ou sans méthode de détection de défaut.

3.4.2. Évaluation avec un défaut carte

Afin de tester la capacité de la méthode à détecter des défauts cartes, un biais est simulé au niveau de la position de l'amer 4 de la carte, de $-0{,}3$ m suivant x et y. La variance initiale de cet amer est de 0,01 m^2 qui ne représente pas ce biais. Une erreur d'angle est aussi ajoutée de 0,25 rad.

La figure 3.15 montre les erreurs des robots, dans ce cas, avec différentes méthodes de gestion des défauts. Le module complet de FDIR (donc avec la correction de carte) permet de réduire les erreurs tout en gardant une bonne consistance grâce la correction de la carte. Les figures 3.17 et 3.18 montrent les résidus d'observation et d'amer. Au début et jusqu'à $t = 5{,}5$ s, le robot 3 opère en mode isolé et observe l'amer 4, ce qui conduit à une exclusion de son observation de la procédure de fusion, il utilise ainsi uniquement l'odométrie. En effet, en absence de collaboration, comme déjà discuté en chapitre 2, le robot n'est pas capable de détecter un défaut de carte. À partir de $t = 35{,}5$ secondes, les robots 2 et 3 collaborent en observant l'amer 4, ce qui conduit à la correction du défaut (figure 3.16). En effet, en observant les résidus, on constate que les résidus d'observation de l'amer 4 par les robots dépassent le seuil mais pas le résidu d'amer (figure 3.19).

Quelques instants plus tard, à $t = 36$ s, le robot 1 collabore avec les deux autres qui ont déjà corrigé la carte. La KLA va aider ce robot à corriger sa version de la carte sans que la procédure de correction de la carte à travers la FDIR soit enclenchée. On observe,

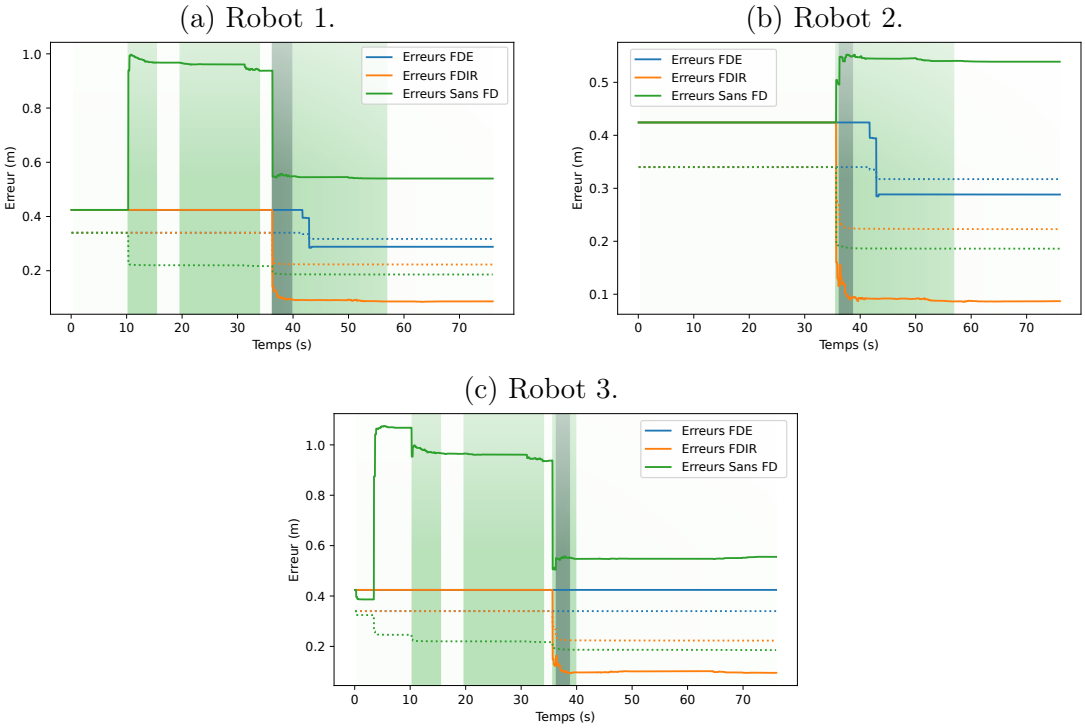


FIGURE 3.16. – Erreurs de position de l'amer 4, initialement erroné.

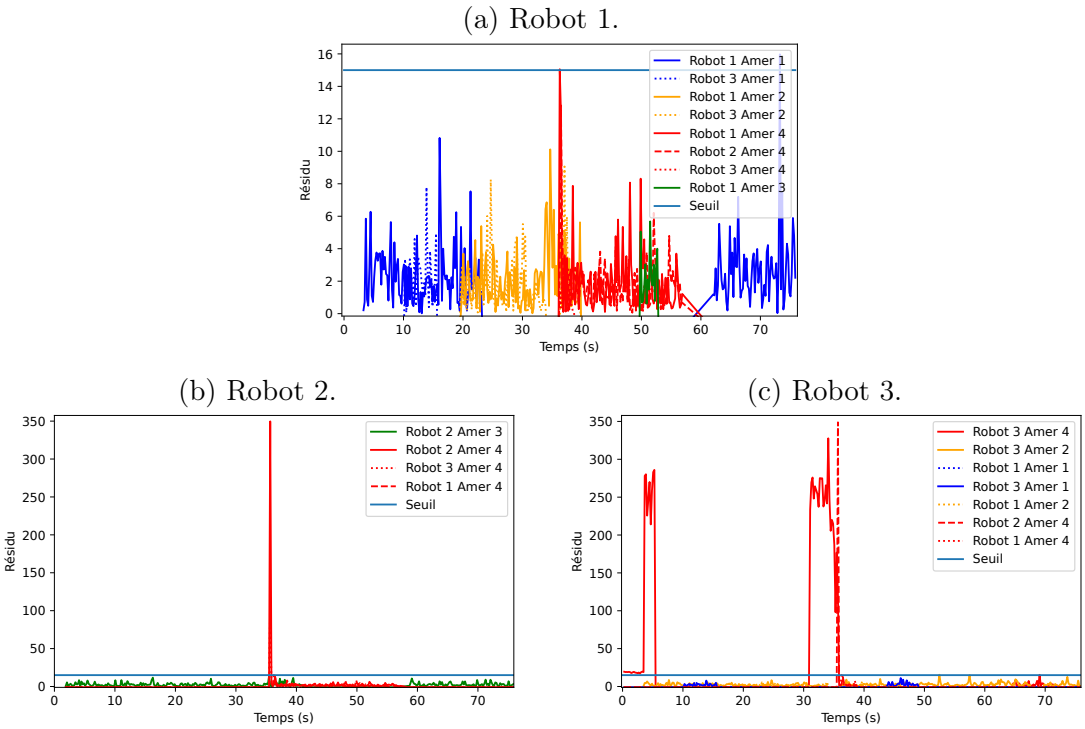


FIGURE 3.17. – Résidus d'observation des robots lors de l'injection d'un défaut de carte.

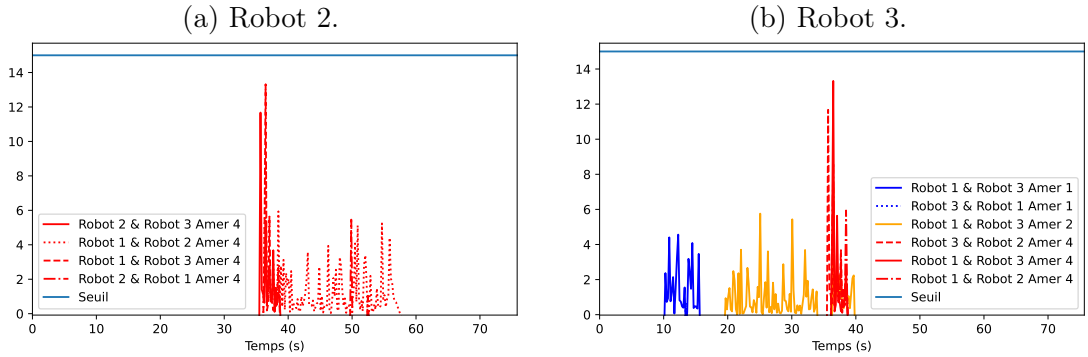


FIGURE 3.18. – Résidus d'amer des robots lors de l'injection d'un défaut de carte. Ici, aucun résidu ne dépasse son seuil.

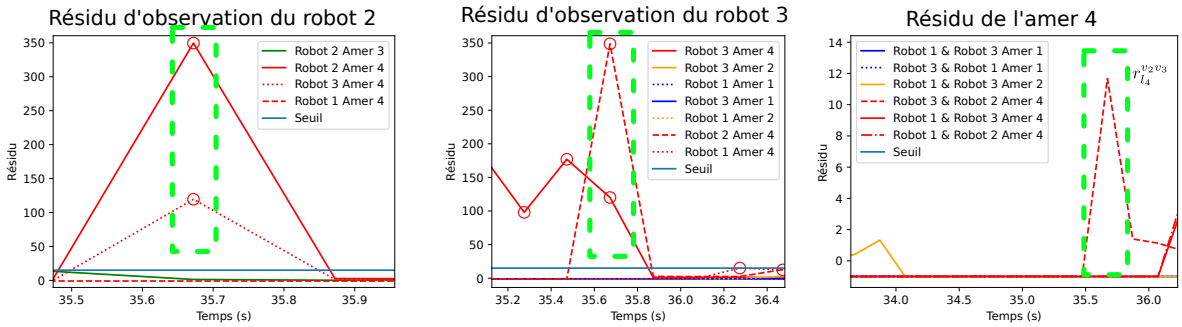


FIGURE 3.19. – Zoom sur les résidus concernés par la correction de l'amer n°4. Le résidu d'amer, à droite, ne dépasse pas son seuil, qui est à 15.

	FDIR	FDE	Sans FD
Robot 1	6,98	99,44	31,55
Robot 2	12,02	96,60	28,79
Robot 3	7,63	9,66	43,98
Tous les robots	8,88	68,57	34,78
Amer 1	8,60	10,19	49,77
Amer 2	9,43	8,91	40,46
Amer 3	11,39	13,60	39,01
Amer 4	24,95	38,44	62,73
Tous les amers	13,59	17,79	48,00

TABLE 3.9. – Erreurs (cm) des robots et des amers, sans amer parfaitement positionné, lors de l'injection d'un défaut de carte. Nous comparons les erreurs avec toute l'étape de FDIR, avec seulement l'exclusion des observations (FDE) et sans aucune détection de défaut (FD).

en effet, que son amer est corrigé (figure 3.16a) sans que son résidu d'observation ne soit très élevé (figure 3.17a). Cependant, son résidu est très légèrement au-dessus de son seuil, ce qui peut s'expliquer par une observation particulièrement bruitée.

La méthode qui exclue simplement les observations erronées conduit à une dérive, pour les robots 1 et 2. En effet, à $t = 36$ s, le robot 1 observe l'amer erroné et il collabore avec les deux autres robots 2 et 3, qui observent le même amer. Les observations de l'amer erroné sont toutes exclues. Comme le robot 1 observe uniquement cet amer-ci, il se retrouve sans observations et continue avec son odométrie, ce qui conduit à une augmentation rapide de sa covariance. Ainsi, la valeur du résidu d'observation diminue, jusqu'à passer sous son seuil, n'excluant plus l'observation. La covariance du robot 1 étant élevée, l'observation est utilisée pour corriger la pose du robot vers l'amer erroné. L'amer est aussi légèrement corrigé.

Sans la détection de défaut, les erreurs ont des valeurs relativement grandes suivant les robots. Toutes les observations sont acceptées, on observe donc une grande augmentation de l'erreur dès le début pour le robot 3 qui observe l'amer erroné (figure 3.15c). Pour le robot 1, à $t = 10$ s, il collabore avec le robot 3. Or, ce dernier a une erreur très importante, ce qui contamine le robot 1 et augmente fortement son erreur. À $t = 36$ s, lorsque le robot 1 observe l'amer erroné, son erreur de localisation est déjà équivalente à l'erreur de l'amer, c'est pourquoi on ne constate pas de grand changement.

La localisation en mode isolé se comporte comme la méthode utilisant uniquement l'exclusion des observations (FDE). En effet, en mode isolé, il est impossible d'isoler un défaut de carte, seules les observations sont rejetées. L'intérêt de la collaboration se porte particulièrement au niveau de la possibilité d'isoler un défaut de carte et de le corriger.

Les résultats de la table 3.9 montrent les valeurs moyennes d'erreurs des différentes méthodes testées. Ces erreurs confirment les effets déjà observés plus tôt au niveau des robots. Cependant, nous pouvons noter que l'erreur de l'amer 2 est plus petite avec la méthode excluant les observations comparée à la méthode FDIR. L'explication de ce

	FDIR	FDE	Sans FD
Robot 1	8,83	103,26	39,39
Robot 2	11,78	97,47	32,19
Robot 3	11,10	15,52	53,81
Tous les robots	10,57	72,09	41,79
Amer 1	11,33	11,28	64,43
Amer 2	13,26	14,64	50,38
Amer 3	11,94	13,56	43,58
Amer 4	23,72	38,41	72,82
Tous les amers	15,06	19,47	57,81

TABLE 3.10. – Erreurs (cm) des robots et des amers, sans amer parfaitement positionné, lors de l'injection d'un défaut de carte et de défauts d'observation.

phénomène surprenant est la suivante. Lorsque les robots 1 et 2 divergent, la propagation des informations via les inter-covariances modifie l'estimation de l'amer 2, observé plus tôt par le robot 1. Cette modification se fait dans le sens de la réduction de l'erreur de l'amer 2, ce qui ne serait pas arrivé avec une disposition de carte différente, ou avec un défaut différent injecté au niveau de l'amer 4.

Au niveau du positionnement des amers, l'amer 4 est mal localisé en moyenne. Ce comportement est normal, car il ne peut être corrigé que vers le milieu du scénario, lorsque les robots collaborent. On observe tout de même une large amélioration de l'estimation de carte avec l'utilisation du module complet de FDIR.

3.4.3. Combinaison des défauts

Les défauts simulés en sections 3.4.1 et 3.4.2 sont maintenant combinés.

Les résultats présentés à la table 3.10 montrent qu'en cas de combinaison de défauts, le système de détection de défaut corrige correctement la carte et la localisation des robots. Cependant, l'erreur est encore en moyenne grande au niveau des amers, ce qui est dû au temps de convergence. En fin de scénario, l'erreur est beaucoup plus faible avec la méthode avec FDIR qu'avec les autres.

Le bon fonctionnement de la méthode lors de la combinaison des différents types de défauts montre que notre système de FDIR est efficace. Il permet de détecter et d'isoler efficacement les défauts afin de maintenir les estimations de la localisation des robots et de la carte correctes. L'apport de la collaboration est important ici puisque sans collaboration, seule la FDE serait possible et il serait impossible de corriger la carte.

Le module de FDIR réduit l'effet des défauts mais sans non plus les effacer complètement (surtout dans le cas des défauts cartes). En effet, l'exclusion des observations peut provoquer de la perte d'information, et avant de pouvoir corriger la carte après une collaboration, toutes les observations de cet amer sont considérées erronées et par conséquent exclues.

3.5. Conclusion

Dans ce chapitre, nous avons présenté l'évaluation de la méthode dans un environnement de simulation.

L'implémentation que nous avons faite de la méthode présentée dans le chapitre précédent avec une architecture décentralisée indique que le système collaboratif fonctionne correctement. Le filtre de Schmidt-Kalman est donc une méthode d'estimation bien adaptée à notre problème. Il est de même pour la moyenne de Kullback-Leibler pour faire la fusion des états des robots.

En termes d'évaluation métrique, les résultats présentés montrent que la collaboration améliore grandement à la fois la localisation des robots, l'estimation de la carte et la détection de défauts. L'amélioration de la localisation des robots dépend du scénario, de la qualité des observations et des amers observés. Certains robots se retrouvent à bénéficier de peu d'améliorations mais font profiter à d'autres robots de leurs estimations en leur partageant des informations. La collaboration rend possible l'isolation des défauts d'amer et ceci améliore de manière significative la qualité de l'estimation de la localisation des robots, permettant ainsi la correction de la carte, et non uniquement l'exclusion des observations, cette dernière pouvant conduire à une dérive odométrique. L'étape de détection et d'isolation de défauts réagit correctement même dans des scénarios complexes avec des défauts multiples.

Même si dans un environnement de simulation il est très difficile de reproduire les erreurs qu'on pourrait avoir dans un environnement réel, ces résultats permettent de valider globalement le fonctionnement du système collaboratif décentralisé avant qu'il ne soit mis en œuvre sur des données réelles dans la prochaine partie.

Deuxième partie

Application aux véhicules intelligents et résultats expérimentaux

Introduction

Cette seconde partie de la thèse porte sur l'application de la méthode proposée dans le cas de véhicules intelligents évoluant dans un environnement urbain défavorable au GNSS. Pour cela, une méthode d'extraction d'observation sous forme de pose relative est proposée, utilisant un lidar 3D et 360° afin de détecter les centres des façades des bâtiments. Le choix des bâtiments permet d'avoir des amers stables dans le temps. De la segmentation sémantique et géométrique, de l'alignement de nuages de points et de l'association point-à-façade sont utilisés afin de préparer le nuage de points pour permettre l'extraction des observations. L'observation utilisée est le milieu des façades qui doivent être observées entièrement. L'orientation de l'observation suit la normale à la façade. La méthode globale est ensuite évaluée sur des données réelles, collectées grâce à trois véhicules expérimentaux spécialement pour cette thèse.

4. Génération d'observations de façades à partir de nuages de points lidar

Sommaire

4.1. Introduction

La mise en place de la méthode présentée en première partie avec des données réelles nécessite le traitement des données brutes provenant du lidar pour en extraire des

observations compatibles avec le filtre. Les observations choisies sont construites à partir des façades des bâtiments. Ces dernières ont l'avantage d'être stables au fil des saisons et de subir peu de changement avec le temps. De plus, il est possible d'avoir facilement une carte de ces façades grâce au cadastre (FINANCES 2013) et à OpenStreetMap [1] (OSM). En France, les bâtiments présents dans OSM sont essentiellement issus du cadastre.

Ce chapitre présente l'approche utilisée pour générer les observations utilisées par le SKF, après un état de l'art concernant la génération d'observation à partir de nuages de points lidar.

4.2. Capteurs extéroceptifs

Dans cette section, les principaux types de capteurs extéroceptifs utilisés pour de la perception sont décrits. Il s'agit des caméras et des lidars. D'autres capteurs existent (radar, sonar, ultrasons, etc.) mais ne feront pas l'objet de cette section, car ils sont moins utilisés pour les tâches de SLAM avec des véhicules terrestres.

4.2.1. Caméras

La caméra est le capteur le plus répandu pour observer son environnement. Elle est constituée d'un bloc optique et d'un capteur photosensible. Le bloc optique permet de focaliser la lumière sur le capteur, et ainsi permettre une focale, une mise au point et une ouverture adaptées à la situation.

Il existe l'habituelle caméra monoculaire, composée d'une seule caméra. Cette caméra peut être en noir et blanc (monochrome), ou en couleur (appelée alors RGB pour *Red Green Blue* en anglais). L'avantage de cette caméra est qu'elle est simple à mettre en place, bon marché, et les informations produites sont parfaitement adaptées aux réseaux de neurones permettant l'extraction de *features*. Cependant, elle ne fournit aucune information de profondeur nativement et fonctionne mal en conditions de faible luminosité.

Lorsque deux caméras monoculaires sont solidaires, il est possible de faire de la vision stéréoscopique et ainsi construire des informations de profondeur grâce à la parallaxe. Une autre manière d'obtenir des informations de profondeur est d'utiliser une caméra RGB-D (D pour *Depth* en anglais, profondeur), composée d'une caméra RGB standard couplée à un capteur de profondeur. Il y a deux technologies pour ce capteur. Soit, il utilise la projection d'un motif dans le spectre infrarouge et une caméra infrarouge repère les déformations, comme le fait la Kinect (Microsoft). Soit, elle utilise la projection de faisceaux infrarouges et calcule le temps d'aller-retour (*time of flight*). L'inconvénient de ces deux méthodes est qu'elles nécessitent un projecteur suffisamment puissant pour atteindre les surfaces observées, ce qui réduit leur portée.

Enfin, pour terminer, il existe les caméras à évènements (GALLEGO et al. 2022). Ces caméras sont aussi composées d'un bloc optique et d'un capteur. En revanche, au lieu d'enregistrer toute l'image de manière périodique comme le font les caméras RGB, chaque

1. `https://www.openstreetmap.org/about`

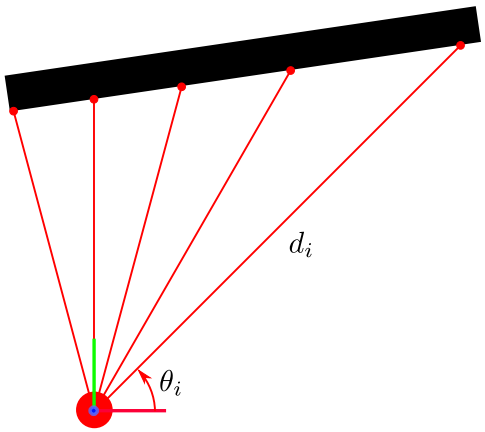


FIGURE 4.1. – Illustration d'un lidar rotatif. Un faisceau rotatif tire plusieurs rayons (lignes rouges) qui sont orientés d'un angle différent θ_i et qui permettent de mesurer une distance différente d_i. Ces rayons ne sont pas tirés au même moment, ce qui peut entrainer des distorsions. Les données brutes du lidar sont donc exprimées en coordonnées polaires.

pixel de ces caméras envoie un évènement à chaque changement de luminosité perçue. Cela permet d'avoir des informations à très haute fréquence, mais également de facilement détourer les objets et connaître la direction de leur mouvement. Ces caméras possèdent aussi une plus grande plage dynamique. Cependant, elles sont encore peu utilisées et les méthodes pour traiter ces informations asynchrones sont encore sommaires.

Les caméras sont des capteurs très bons pour fournir du contexte et très utiles pour la classification. Elles sont aussi bon marché et il existe beaucoup de méthodes très performantes pour traiter leurs informations et en extraire des *features*. Cependant, à moins de les multiplier, elles ont un champ de vue limité et sont assez mauvaises en conditions de basse luminosité.

4.2.2. Lidars

La technologie lidar, un acronyme signifiant *light detection and ranging* (soit détection et mesure de distance par la lumière), repose sur l'émission d'un rayon laser dans l'infrarouge et la mesure du temps d'aller-retour. Cela permet de calculer une distance en connaissant la vitesse de la lumière. La mesure du temps est généralement couplée à une analyse du décalage de phase entre l'émission et la réception, affinant ainsi la mesure de distance.

Avec un seul rayon laser, on ne peut mesurer la distance que d'un seul point. Afin de constituer un nuage de points, deux technologies existent. La première, la plus ancienne, est le lidar rotatif. L'émetteur laser est monté sur une plateforme tournante, et peut donc mesurer en permanence le point devant lui. En connaissant l'orientation θ_i de l'émetteur et la distance d_i du point mesuré, un nuage de points 2D peut donc être constitué, en

coordonnées polaires (figure 4.1) :

$$x_i = d_i \cos\theta_i, \tag{4.1}$$
$$y_i = d_i \sin\theta_i. \tag{4.2}$$

Pour passer à la 3D, plusieurs faisceaux sont superposés, avec des angles d'élévation différents. Chaque faisceau produit une nappe (ensemble des points avec un angle d'élévation particulier), et l'ensemble des nappes constitue un nuage de points 3D. Certains lidars montent jusqu'à 128 nappes. Un des inconvénients majeurs des lidars rotatifs est que leur partie rotative induit un effet gyroscopique, car elle tourne à haute vitesse (10 tours par secondes au moins). Lorsque le lidar est utilisé en robotique mobile, le déplacement du lidar va à l'encontre de cet effet gyroscopique, ce qui provoque une usure accélérée du capteur. De plus, à cause de la rotation permanente des faisceaux, les points d'un scan lidar ne sont pas tous mesurés au même moment (il y a du délai entre le premier point mesuré et le dernier), ce qui induit de la déformation, particulièrement visible à haute vitesse (Renzler et al. 2020).

Afin de résoudre les inconvénients du lidar rotatifs, les lidars à éléments fixes (*solid states*) commencent à émerger et à fournir des performances intéressantes (N. Li et al. 2022). Ces lidars n'ont plus de parties mobiles, des faisceaux avec différentes phases sont tirés dans des directions différentes, contrôlées électroniquement. Ils produisent un nuage de points très dense dans le champ de vue du lidar. Les lidars *solid states* ont une durée de vie bien plus grande que les lidars rotatifs. Il est aussi possible de changer l'orientation des faisceaux sans démonter le capteur, et donc de focaliser les mesures dans une zone particulière. Cependant, pour le moment, ils ont un champ de vue très réduit, contrairement aux lidars rotatifs qui peuvent observer à 360°.

Le gros avantage des lidars 3D rotatifs est leur champ de vue à 360° et leurs informations de profondeur fiables. Ils sont aussi performants sous n'importe quelles conditions de luminosité et certains lidars se comportent aussi bien sous la pluie. Leurs gros inconvénients sont leur prix et leur fragilité.

Dans nos travaux, nous cherchons à observer des façades de bâtiments en ville, à partir d'un véhicule. Nous sommes donc davantage intéressés par un large champ de vue afin d'observer des façades dans toutes les directions. De plus, nous utilisons des observations nécessitant une bonne mesure de distance. C'est pourquoi nous avons choisi d'utiliser un lidar plutôt qu'une ou plusieurs caméras. L'état de l'art qui suit se focalisera donc sur les méthodes applicables au lidar.

4.3. État de l'art sur le traitement de nuages de points lidar pour l'obtention d'observations

Afin de générer des observations à partir d'un nuage de points lidar, plusieurs méthodes existent, reposant sur un ensemble de traitements effectués sur le nuage de points. Un état de l'art sur ces traitements et sur l'obtention d'observation est présenté dans cette section.

Tout d'abord, les méthodes de segmentation de nuages de points sont abordées. Ensuite, les méthodes permettant l'extraction de *features* de nuages de points sont présentées, suivies des méthodes d'alignement permettant de calculer des transformations entre des nuages de points. Une combinaison de ces différentes méthodes permettra de générer les observations utilisables dans notre situation.

Nous appelons ici « observation » une mesure pouvant être utilisée dans un système de localisation comme un filtre. Il peut s'agir d'une transformation, d'un descripteur, d'une *feature*, d'un point construit géométriquement, etc.

4.3.1. Jeux de données

Dans cet état de l'art, nous aborderons plusieurs méthodes utilisant des réseaux de neurones. Ces méthodes nécessitent une grande quantité de données pour entrainer les modèles. Ces données doivent être annotées, c'est-à-dire que le résultat attendu du réseau de neurone doit être attaché à chaque donnée. Cela rend ces données compliquées à produire, tout en demandant beaucoup de temps, car les annotations sont finalisées à la main. Il existe donc des jeux de données reconnus pour leur qualité, utilisés par la communauté afin d'entrainer, puis d'évaluer leurs méthodes.

Plusieurs jeux de données publics spécifiques aux lidars existent et mettent en place un classement des meilleures méthodes. On peut citer SemanticKITTI (Behley et al. 2019), qui est un ensemble de 22 séquences de roulage d'un véhicule équipé de lidar, dont chaque point lidar a été annoté dans une des 28 classes réparties dans les catégories suivantes : sol, structure, véhicule, nature, humain, objet et inconnu. Parmi ces classes, certaines font la différence entre les objets mobiles et immobiles (au moment de l'observation).

Un deuxième jeu de données notable pour la segmentation de nuages de points est NuScenes, développé par Motional (Fong et al. 2021). Il s'agit d'un ensemble de 1000 séquences lidar, d'une vingtaine de secondes, sélectionnées pour montrer une diversité de condition de conduite (manœuvres, trafic, comportements inattendus).

4.3.2. Segmentation sémantique de nuages de points

La segmentation d'un nuage de points consiste à assigner une classe à chaque point. Les classes peuvent varier en fonction des besoins. La segmentation sémantique est très largement dominée par les réseaux de neurones et permet de déterminer la nature des objets détectés (voiture, mur, végétation, etc.). Pour chaque classe, le modèle la représentant est très complexe, utilisant un grand nombre de paramètres.

Des méthodes géométriques existent, mais elles sont mal adaptées pour cette tâche. En effet, les méthodes géométriques ont beaucoup plus de difficultés à généraliser leur modèle à tout type d'objet appartenant à une certaine classe. Cela fait longtemps qu'elles ont été dépassées par les méthodes d'apprentissage dans les classements établis par les différents jeux de données, et c'est pourquoi cet état de l'art se focalisera sur les méthodes utilisant des réseaux de neurones.

(X. Zhu et al. 2021) propose Cylinder3D, utilisant un réseau de neurones MLP pour

définir des *features* puis un encodeur-décodeur pour les attribuer une classe. La spécificité de Cylinder3D est qu'il utilise une grille cylindrique afin de s'adapter à la densité décroissante du nuage de points lidar avec la distance.

Point-to-VoxelKD s'efforce d'utiliser au maximum la connaissance présente à tous les niveaux du réseau de neurones pour améliorer la segmentation (Hou et al. 2022). Pour cela, ils utilisent la distillation de connaissance, qui permet de compresser le modèle en transférant la connaissance d'un grand modèle vers un réseau plus compact. Cela permet une meilleure exploitation des informations structurelles et donc une meilleure segmentation.

2DPASS (Yan et al. 2022) est, en octobre 2023, premier au classement de SemanticKITTI. Ils utilisent des données de caméra et de lidar pour améliorer l'entrainement, puis seul le lidar est nécessaire. Il se base sur le même fonctionnement que Point-to-VoxelKD, tout en tirant parti des deux types de capteurs pour enrichir l'information sémantique et structurelle lors de l'entrainement, ce qui produit de meilleurs résultats lors de l'utilisation sur de nouvelles données.

4.3.3. Segmentation géométrique de nuages de points

La segmentation sémantique permettant d'extraire des objets sémantiques du nuage de points, il est aussi possible d'extraire du nuage de points des formes géométriques (plan, sphère, tore, poteau, etc.). Pour cela, les méthodes géométriques sont très efficaces (Xia et al. 2020).

Une première méthode permettant de segmenter les points en fonction de modèles géométriques déterminés est le *Random Sampling Consensus* (RANSAC), introduit pour la première fois par (Fischler et Bolles 1981), où un modèle géométrique avec un certain nombre de paramètres est recherché. Par exemple, dans le cas d'une ligne, ce modèle nécessite deux points pour avoir une instance du modèle (en vert sur la figure 4.2).

Pour un modèle défini à partir de m points, un groupe de m points est sélectionné aléatoirement dans le nuage de points. Pour chaque point du nuage complet, sa distance avec la forme définie par les m points sélectionnés est calculée. Le nombre de points dont la distance est sous un certain seuil fixé (représentée par la zone bleue, figure 4.2) est compté, et cela constitue le score de l'échantillon. Ces points sont appelés *inliers*. L'opération est répétée un nombre fixe de fois. Lorsque toutes les itérations sont effectuées, l'échantillon avec le meilleur score est sélectionné et forme le résultat du RANSAC.

(Schnabel, Wahl et Klein 2007) propose une méthode qui permet au RANSAC de chercher plusieurs types de modèles (lignes, cercles, etc.) en parallèle dans un nuage de points.

Le RANSAC *a contrario*, aussi appelé RANSAC optimisé (ORSA, *Optimized RANSAC*), permet une plus grande flexibilité sur le réglage de l'algorithme (Moisan, Moulon et Monasse 2012). Le critère d'*a contrario* évite de fixer les seuils d'inclusion et d'exclusion. Il permet aussi de quantifier la qualité de la forme et de potentiellement déclarer l'absence de la forme recherchée dans le nuage de points, tandis qu'un RANSAC standard trouvera

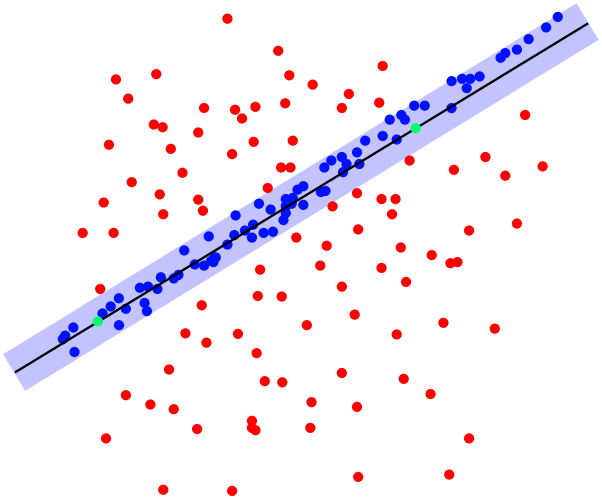


FIGURE 4.2. – Illustration du RANSAC. Pour chercher une ligne, deux points sont sélectionnés aléatoirement (en vert). Les *inliers* sont en bleu, dans la zone de tolérance bleue.

toujours une forme, même s'il y a très peu d'*inliers* dessus. Ce type de RANSAC révèle son avantage lorsque le nuage de points est de taille importante et lorsque le modèle recherché est incertain.

Une méthode pour extraire des formes d'un ensemble de données, de manière exhaustive, est la transformée de Hough (HOUGH 1962). La méthode consiste à définir un modèle à partir de m de paramètres (coefficients d'une droite par exemple). Ensuite, chaque point du nuage de points est converti dans l'espace des paramètres en une courbe représentant tous les paramètres possibles pour ce point (lignes rouges à droite de la figure 4.3). Pour choisir le meilleur ensemble de paramètres, l'intersection des courbes est sélectionnée (figure 4.3).

Cette méthode est très adaptée pour le traitement d'images, mais trouve ses limites pour les nuages de points 3D, car la quantité de données et le nombre de paramètres du modèle sont beaucoup plus importants. En effet, la méthode utilise tous les points et la phase de recherche du meilleur paramètre est aussi très coûteuse en temps de calcul si une bonne précision est recherchée.

Une autre méthode permettant de segmenter un nuage de points est la segmentation par facettes (LIN et al. 2017). Elle est plutôt adaptée pour les nuages de points denses. Un point est choisi aléatoirement dans le nuage de points, puis, la forme choisie est construite en utilisant les voisins. Ensuite, chaque point voisin au groupe y est ajouté ou non en fonction de son degré de correspondance à la forme définie par le groupe. Ce procédé est répété avec plusieurs points de départ. Cette méthode est utilisée en photogrammétrie (XIA et al. 2020), sur des nuages de points très denses structurés, et n'est pas forcément adaptée à du traitement en temps réel sur des nuages de points épars.

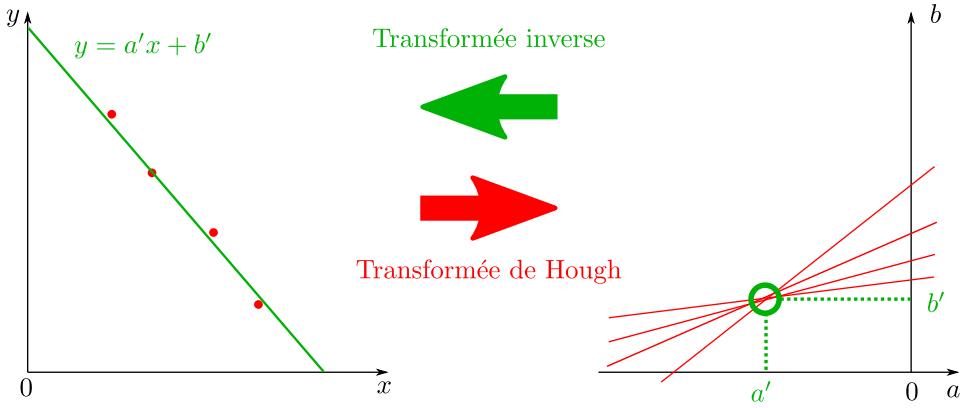


FIGURE 4.3. – Transformée de Hough. Chaque point dans l'espace des coordonnées est converti en une ligne dans l'espace des paramètres. L'intersection des lignes définie les paramètres majoritaires et permettent de définir une ligne dans l'espace des coordonnées. Attention, la courbe dans l'espace des paramètres n'est pas nécessairement une ligne droite, cela dépend du modèle.

4.3.4. Méthodes d'extraction de *features*

La segmentation des nuages de points ne suffit pas toujours pour obtenir des observations utilisables par un système de SLAM. L'extraction de *features* est parfois nécessaire pour constituer des observations ponctuelles ou avec plus de contexte. Le terme *feature* regroupe à la fois des points caractéristiques (coins par exemple), mais aussi des points résumant un objet, moins soumis au bruit de mesure, comme le centre de masse d'une voiture.

(J. ZHANG et SINGH 2014) ont présenté le LOAM (*lidar odometry and mapping*). Afin d'extraire des *features* du nuage de points 3D, un indice de lisseur (*smoothness*) est calculé avec les points consécutifs d'une même nappe. Cet indice permet d'extraire deux types de points : les points-coin appartenant à un coin et les points-surface appartenant à une surface plane. Dans un même nuage de points, ces points sont ensuite groupés pour construire une ligne pour les coins et un patch planaire pour les surfaces planes. Les centres géométriques de ces groupes sont ensuite utilisés comme *features*. Afin de répartir les *features* dans tout le nuage, ce dernier est divisé en quatre cadrants, et chaque cadrant ne peut fournir que deux points-coin et quatre points-surface au maximum. Ces *features* sont utilisées pour de l'odométrie lidar et de la cartographie. Les points-coin sont très stables, mais les points-surface sont susceptibles de changer d'un scan à l'autre, notamment sur de longues surfaces. (GUO, J. ZHU et Yunping CHEN 2022) propose le E-LOAM, qui utilise le LOAM et l'étend en le couplant à la NDT pour améliorer la représentation de la structure de l'environnement.

Afin d'extraire des *features* d'un nuage de points 2D, la méthode FLIRT est particulièrement adaptée, car elle produit des *features* indépendantes de l'échelle (TIPALDI et

Arras 2010). La méthode utilise un détecteur sélectionnant les points à partir de leur courbure locale, sur plusieurs échelles, c'est-à-dire que plusieurs courbures sont calculées en prenant un nombre différent de points contigus sur la nappe. Ce détecteur est combiné à un descripteur sous forme d'une grille d'occupation polaire probabiliste. L'ensemble des probabilités d'occupation sur la grille, avec leurs variances, définit le descripteur.

Utilisant OSM, (Cho et al. 2022) génère sur les routes de la carte un descripteur tous les 1 m. Ce descripteur correspond au nuage de points que devrait observer le robot s'il se trouvait à l'emplacement du descripteur, via la simulation d'un lidar observant uniquement les façades de la carte. Le descripteur se présente sous la forme d'un vecteur de distance chacune associée à un angle, ce qui correspond à la représentation polaire d'un nuage de points 2D. Chaque descripteur a aussi une version invariante en rotation, qui est calculée en sommant le nombre de points par tranche de distance (nombre de points dans des disques concentriques).

Des *features* sous forme de descripteurs sont proposées dans SegMatch (Dube et al. 2017). Le nuage de points est d'abord segmenté sémantiquement en plusieurs objets (arbres, véhicules, bâtiment...). Ensuite, deux types de descripteurs sont extraits analytiquement de chaque groupe de points. Un premier type de descripteur représente des caractéristiques géométriques et un second type est composé d'histogrammes représentant des fonctions de forme (de dimension 640).

Les descripteurs peuvent être aussi obtenus via des réseaux de neurones. Dans (Dubé et al. 2018), les auteurs utilisent un encodeur afin de réduire les éléments segmentés du nuage de points à un descripteur de dimension 64.

4.3.5. Alignement de nuages de points

Les méthodes d'alignement sont très souvent utilisées pour faire de l'odométrie visuelle ou du SLAM, en utilisant les transformations entre les nuages de points consécutifs comme mesure du déplacement. Dans ces cas-là, l'alignement (*registration* en anglais) de deux nuages de points peut remplacer l'extraction de *features*. En SLAM, l'alignement est aussi utilisé pour la fermeture de boucle, c'est-à-dire l'alignement du nuage de points le plus récent avec la carte (Mendes, Koch et Lacroix 2016).

Dans cette section, nous appellerons le nuage de points « cible » celui qui est fixe durant l'alignement (le nuage de points de l'instant précédent, ou la carte, par exemple), et le nuage de points « source » celui sur lequel est appliquée la transformation.

Iterative Closest Point

La méthode la plus connue pour cette tâche est l'*Iterative Closest Point* (ICP). Dans sa version originale, l'ICP cherche la transformation qui minimise la somme des erreurs quadratiques de chaque point du nuage source par rapport au point le plus proche dans le nuage cible (Besl et McKay 1992). À chaque itération, chaque point source est associé à un point cible, puis une erreur est calculée. Le nuage de points source est transformé dans

la direction qui minimise la somme des erreurs quadratiques. La descente de gradient peut être utilisée pour la minimisation. Pour cela, il faut que le point d'initialisation ne soit pas trop loin de la solution réelle, pour éviter de converger vers un minimum local.

L'ICP Généralisé a été proposé dans (Segal, Haehnel et Thrun 2009) pour améliorer la rapidité et la robustesse de l'alignement. Les auteurs considèrent l'existence de deux ensembles de distributions normales (ND), un ensemble pour le nuage source et un ensemble pour le nuage cible. Chaque distribution produit un seul point, il y a donc autant de ND que de points. La transformation optimale est obtenue en maximisant une fonction de vraisemblance entre les deux ensembles de distributions normales. Cette approche correspond à la minimisation d'une erreur quadratique moyenne où chaque erreur est pondérée par la variance de cette erreur, sous la forme d'une distance de Mahalanobis. Les expérimentations menées dans (Segal, Haehnel et Thrun 2009) montrent que la méthode est plus précise que l'ICP standard, tout en maintenant sa vitesse. De même, le modèle probabiliste permet de réduire les effets d'associations incorrectes.

L'ICP peut aussi utiliser des correspondances point-à-ligne (Yang Chen et Medioni 1992). Il s'agira de minimiser les distances entre chaque point et sa projection sur la ligne la plus proche. Cette approche permet d'aligner un nuage de points sur un ensemble de lignes (ou de plans en 3D) et donc avec une carte vectorielle, par exemple.

(Agamennoni et al. 2016) présente une extension de l'ICP, sous le nom d'association de données probabiliste. Cette méthode est très semblable à l'ICP mais chaque point du nuage source est associé à plusieurs points du nuage cible. Chaque association est pondérée par une probabilité d'association, suivant une loi de Student. L'association est faite une seule fois, puis, au fil des itérations, les poids sont changés sans chercher de nouveau de nouvelles associations. L'alignement est fait via un algorithme d'espérance-maximisation (*Expectation-Maximization*, EM). Cette extension rend l'alignement moins sensible au bruit.

Normal Distributions Transform

Une autre méthode pour aligner des nuages de points est la *Normal Distributions Transform* (transformation via distributions normales, NDT), proposée initialement en deux dimensions dans (Biber et Strasser 2003), puis en 3D dans (Takeuchi et Tsubouchi 2006). Le principe de la NDT est de subdiviser l'environnement en cellules dans lesquelles une distribution normale (ND) est placée. Cette distribution normale multivariée (2D ou 3D) représente la distribution des points lidar dans la cellule, et donc la probabilité d'en observer en chaque endroit. Afin de déterminer la transformation entre le nuage de points source et la grille de ND, la vraisemblance est maximisée. Comme l'ICP, la NDT est très sensible à l'initialisation.

Cette façon de répartir les points dans des cellules crée des effets de seuils aux bords des cellules, le point pouvant changer de cellule d'une itération à l'autre. (Biber et Strasser 2003) propose d'utiliser plusieurs grilles de taille identiques décalées d'une demi-cellule dans chaque direction, mais sans le mettre en place.

Le choix de la taille des cellules affecte les résultats de deux manières. En augmentant

la taille des cellules, la convergence vers le minimum global est plus probable, mais en diminuant la taille, la précision est meilleure. Afin d'avoir le meilleur des deux, (Ulas et Temeltas 2011) propose de faire plusieurs alignements NDT, en commençant par des cellules de grande taille, pour aller vers de plus petites cellules afin d'affiner la transformation.

Dans des nuages de points de très grande taille, avec beaucoup de vide, la construction d'une grille dense n'est pas toujours adaptée. (Das et Waslander 2012) propose donc de diviser le nuage de points en N groupes. Une seule ND est associée à chaque groupe. L'alignement se fait avec ces ND, et la vraisemblance de chaque point est évaluée par rapport à toutes les ND. Une fois l'alignement réalisé avec un nombre N_k de groupes, l'opération est renouvelée pour un nombre N_{k+1} de groupes plus important, en conservant la transformation estimée à l'échelle précédente.

Dans (Schulz et Zell 2020), les auteurs ajoutent une probabilité d'existence aux cellules de la NDT, transformant la grille en grille d'occupation. Cela permet de pondérer la vraisemblance de chaque cellule par sa probabilité d'existence et ainsi robustifier l'alignement, notamment lorsqu'il y a peu d'éléments dans l'environnement.

Les méthodes de NDT évoquées jusqu'à présent associent des points à des distributions, dans ce qu'on peut appeler de la « NDT point-à-distribution » (P2D-NDT). Dans (Stoyanov et al. 2012), ce sont des distributions qui sont comparées (D2D-NDT). Les deux nuages de points sont transformés en grilles de ND. Ensuite, une fonction de coût basée sur la distance de Mahalanobis entre les distributions est minimisée, où les distances sont définies par toutes les paires de ND possibles entre les deux nuages de points. Cette méthode est plus rapide que la version point-à-distribution, avec une précision similaire ou meilleure en fonction des conditions. La méthode est proche de l'ICP généralisé introduit par (Segal, Haehnel et Thrun 2009). La différence réside dans le fait que la D2D-NDT calcule les distances entre des distributions représentant plusieurs points, tandis que l'ICP généralisé considère une ND pour chaque point et les associe deux-à-deux (point le plus proche).

(Shi et al. 2019) propose une combinaison de la NDT avec l'ICP. Un alignement grossier est fait avec la NDT, puis il est affiné par l'ICP. Le but de cette méthode est d'améliorer la vitesse de convergence grâce à la NDT et de produire un meilleur alignement avec l'ICP. Cette méthode part du principe que l'ICP a une meilleure précision que la NDT.

Comparaison

La P2D-NDT, la D2D-NDT, l'ICP point-à-point et l'ICP point-à-ligne ont été comparés dans (Magnusson et al. 2015). Les deux méthodes d'ICP possèdent de bonnes performances dans des environnements structurés, mais ont des lacunes dans des environnements naturels et lorsqu'il y a peu de recouvrement ($\leq 50\%$) entre les nuages de points (Magnusson et al. 2015). Cependant, l'ICP présente l'avantage de nécessiter moins de réglages que la NDT (taille de cellule).

Les méthodes basées sur la NDT donnent des résultats plus robustes. La D2D-NDT a l'avantage d'être la plus rapide, tout en étant la moins sensible à une erreur sur l'initialisation. Dans le cas des environnements peu structurés ou avec peu de chevauchement

entre les nuages de points, la P2D-NDT donne la meilleure précision.

4.4. Approche proposée et architecture

L'objectif est de localiser des véhicules dans un environnement urbain doté d'une carte à priori, en profitant de la collaboration entre les véhicules. Cette collaboration se réalise à travers des mesures indirectes, en observant des façades des bâtiments, qui sont les composantes les plus stables de l'environnement. Les mesures indirectes permettent aussi d'augmenter la portée de la collaboration car les véhicules peuvent collaborer sans qu'ils s'observent directement.

Un capteur lidar 3D 360° est très bien adapté à notre problématique, car il allie un grand champ de vue avec une mesure précise des distances. De même, ce capteur est moins sensible aux conditions de luminosité.

L'architecture décentralisée pour la localisation et la mise à jour de carte proposée au chapitre 2 sera adaptée au cas réel. Pour rappel, le SKF utilise les poses relatives des amers comme observations.

Comme nous l'avons déjà vu, l'observation de l'amer l_j par le véhicule v_i est :

$$Z_{l_j}^{v_i} = \begin{pmatrix} \cos\theta_{v_i} & \sin\theta_{v_i} & 0 \\ -\sin\theta_{v_i} & \cos\theta_{v_i} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{l_j} - x_{v_i} \\ y_{l_j} - y_{v_i} \\ \theta_{l_j} - \theta_{v_i} \end{pmatrix}, \tag{4.3}$$

où $(x, y, \theta)_{l_j}$ est la pose de l'amer dans le repère carte et $(x, y, \theta)_{v_i}$ la pose du véhicule.

Le but est donc de proposer une approche permettant de générer ce type d'observations pour chaque façade de bâtiment présente dans le nuage de points lidar.

Ces observations doivent satisfaire plusieurs exigences. Tout d'abord, elles doivent être conditionnellement indépendantes de l'estimation de localisation du véhicule, ainsi que de l'estimation de l'amer dans la carte. Chaque observation qui correspond à un certain amer associé à une façade est identifiée, grâce à l'association point-à-façade, via un identifiant unique (ID) de l'amer stocké dans la carte. Ensuite, les observations provenant de différents véhicules doivent être comparables, c'est-à-dire que les observations ne doivent pas changer suivant l'angle de vision ou selon le véhicule. Une observation est alors la pose du milieu de la façade obtenue à partir du nuage de points. Ce choix est motivé par la possibilité de générer une observation à partir d'une vision partielle de la façade.

Comme déjà discuté, l'approche se base sur l'utilisation d'une carte à priori. Notre carte se présente sous trois formes. La première représentation est un ensemble de polygones 2D représentant l'empreinte au sol des bâtiments, provenant directement d'OpenStreetMap (OSM). Les milieux des segments constituant les polygones sont ensuite calculés et orientés pour définir les poses utilisables par le filtre. Ceci constitue la deuxième forme 2D de la carte. Enfin, une représentation 3D des façades est ajoutée, afin de permettre l'utilisation de méthodes d'alignement en 3D. Ces représentations seront détaillées dans la section 4.5.

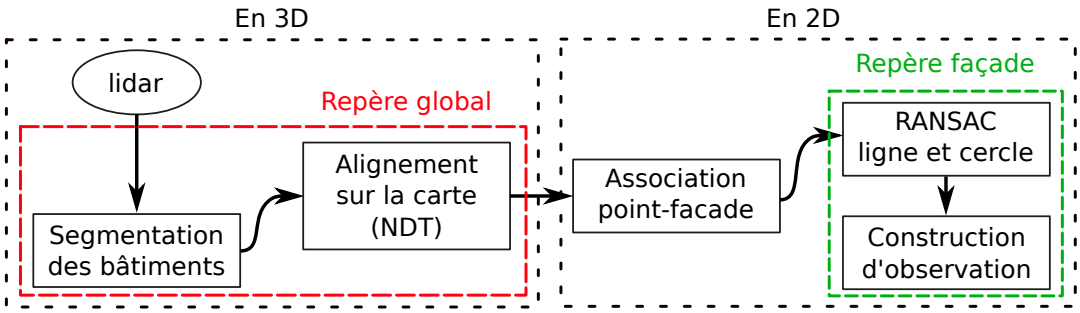


FIGURE 4.4. – Architecture proposée pour la génération d'observations.

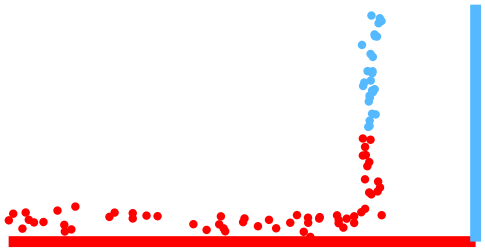


FIGURE 4.5. – Exemple de mauvaise association point-à-façade lors d'un mauvais alignement entre le nuage de points et la carte.

Pour extraire les observations à partir du nuage de points, plusieurs étapes de traitement sont nécessaires. Une grande partie des traitements effectués a pour but de retirer un maximum de points aberrants n'appartenant pas à des façades. L'approche est synthétisée par la figure 4.4 :

1. La première étape est la segmentation sémantique du nuage de points qui permet d'extraire uniquement des points appartenant à des bâtiments.
2. La deuxième étape est l'alignement à l'aide de la NDT. Il convient de noter que la transformation obtenue n'est pas utilisée pour la localisation.
3. Seuls les points proches des façades dans la carte sont sélectionnés et groupés par façade, ce qui constitue l'étape association point-façade. Cela permet de retirer les points mal segmentés ou appartenant à des façades inconnues. On peut noter que cette étape n'est pas adaptée à l'ajout des façades dans la carte, car la détection d'une façade repose sur sa présence dans la carte.
4. Une dernière étape de retrait de points aberrants est effectuée dans chaque groupe de points à l'aide d'un RANSAC. Les points restants dans chaque groupe de points correspondent à un modèle en ligne ou en arc de la façade.
5. Il suffit ensuite de construire géométriquement une observation sous forme de pose à partir de ces points.

Le groupement des points par façade, effectué en associant chaque point au segment de la carte le plus proche, est très sensible à un décalage entre la carte et le nuage de points. De mauvaises associations peuvent être réalisées, en particulier dans les coins (figure 4.5). Afin de limiter ces mauvaises associations, un alignement est fait avant de

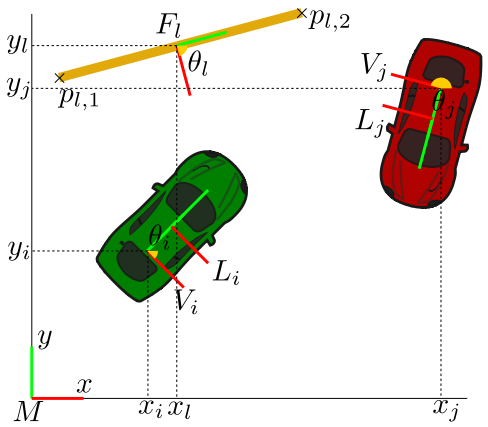


FIGURE 4.6. – Repères.

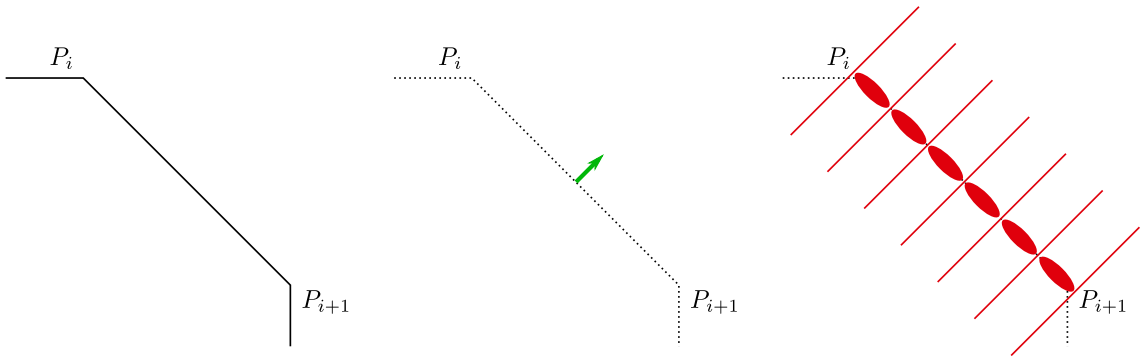
grouper les points par façades. Cet alignement est réalisé avec la NDT, entre le nuage de points et la carte. Pour cela, la carte constituée de polygones 2D est convertie en un ensemble de distributions normales 3D (détaillé en section 4.5), et la NDT utilise ces distributions normales pour son alignement à un niveau global.

La NDT présente ici l'avantage de pouvoir représenter plus finement les façades que l'ICP. Avec cette représentation, les façades peuvent être affinées au fur et à mesure des passages dans une même zone (même si ceci n'a pas été implémenté dans ces travaux). C'est la raison principale qui a guidé ce choix. Notons que la NDT ne sert qu'à aligner le nuage de points pour le groupement par façade. Son résultat n'est pas utilisé dans le filtre afin de ne pas créer de dépendance entre la pose observée du véhicule et les futures observations des façades.

Repères Les repères utilisés dans ce chapitre sont représentés à la figure 4.6. Le nuage de points est capturé par le lidar, dans le repère L_i. Il est instantanément transformé dans le repère du véhicule V_i pour la suite des traitements via une transformation statique. Ce dernier est le repère mobile, dont la transformation avec le repère global de la carte, noté M, est estimée par la pose. Le repère de la carte est le repère de travail. Chaque façade l définit un repère F_l à partir de sa représentation sous forme de pose. Ce repère est placé au centre de la façade.

4.5. Représentations de la carte

Avant de détailler l'approche résumée en figure 4.4, les différentes représentations de la carte sont montrées sur la figure 4.7. Ces représentations sont directement liées entre elles.



(a) Façade sous forme de polygone. (b) Façade sous forme de pose. (c) Façade sous forme de grille de ND.

FIGURE 4.7. – Différentes représentations d'une façade dans la carte. En noir est représenté le segment issu d'OSM. En vert, c'est la pose, positionnée au milieu du segment et qui est utilisée par le SKF. Enfin, la grille de distributions normales est représentée en rouge. La grille est verticale et en 3D, mais elle est représentée ici en vue de dessus, par simplification.

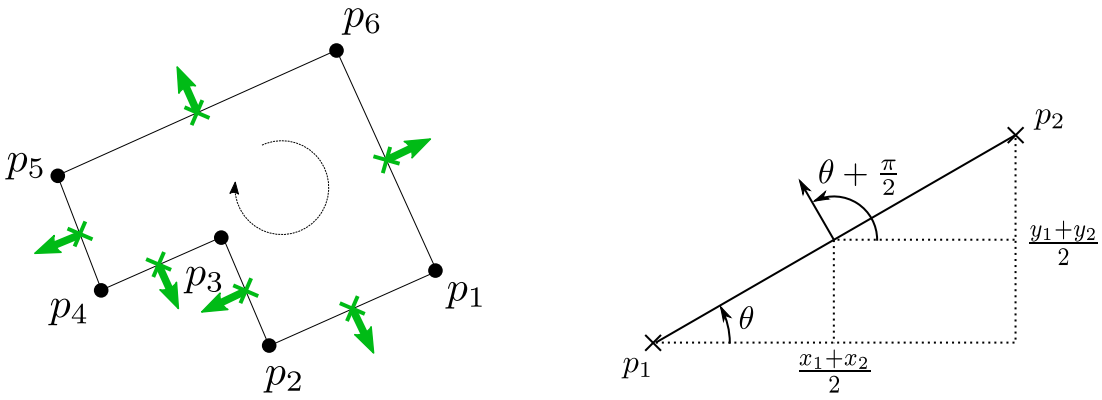
4.5.1. Une carte de polygones

La représentation première de la carte est sous forme de polygones 2D qui représentent les empreintes au sol des bâtiments (figure 4.7a). En effet, la carte est issue d'OSM, et donc en grande partie du cadastre français. D'après ses spécifications, le cadastre doit avoir une précision de 10 cm dans les échelles les plus précises (FINANCES 2013). Les bâtiments issus d'OSM se présentent directement sous forme de polygones, mais parfois, des façades rectilignes sont séparées en plusieurs segments. Un traitement manuel est donc réalisé pour que la carte de polygone suive la règle suivante : une façade est délimitée par deux coins. Dans le cas des façades courbes, un découpage est fait pour que la façade réelle ne soit jamais trop loin du segment.

Afin que les façades puissent être manipulées individuellement (déplacements et rotations), chaque segment du polygone est détaché de ses voisins. Cela permet une modification de cette représentation de la carte après des corrections apportées par le filtre. Cependant, on perd le lien entre les façades, pouvant provoquer l'apparition de trous entre deux façades.

4.5.2. Une carte de poses

Une représentation de façades sous forme de poses 2D est utilisée pour l'estimation d'état à travers le SKF. Cette représentation a été choisie pour que la collaboration entre les véhicules se fasse via l'échange d'informations légères, tout en conservant suffisamment d'informations pour permettre la localisation des véhicules. Pour cela, le milieu des segments de la représentation par polygones a été choisi, avec comme orientation la normale du segment. Elle est représentée en vert sur la figure 4.7b. Cette représentation est la plus compacte de toutes, avec 3 paramètres contre 4 pour le segment avec ses deux



(a) Afin de déterminer les poses à partir du polygone du bâtiment, les points sont parcourus dans le sens horaire et le milieu de chaque segment est sélectionné.

(b) Construction de la pose au milieu d'un segment.

FIGURE 4.8. – Construction de la carte sous forme de poses.

points. Ce choix a aussi été motivé par la possibilité d'avoir des observations incomplètes de façades (car la pose du centre est détectable même s'il existe des trous dans le nuage de points à cause d'occultations partielles par exemple).

Afin de déterminer cette pose dans la carte, les points des polygones sont parcourus dans le sens horaire, comme illustré en figure 4.8a. À partir de chaque segment $[P_iP_{i+1}]$ le milieu et la normale sont déterminés ainsi (figure 4.8b) :

$$x_l = \frac{x_i + x_{i+1}}{2}, \tag{4.4}$$

$$y_l = \frac{y_i + y_{i+1}}{2}, \tag{4.5}$$

$$\theta_l = \text{atan2}\left(y_{i+1} - y_i, x_{i+1} - x_i\right) + \frac{\pi}{2}. \tag{4.6}$$

Si les points du polygone sont parcourus dans le sens anti-horaire, les normales seront tournées vers l'intérieur du bâtiment. Cela ne pose pas de problème, tant que la représentation de chaque façade est cohérente à travers tout le système.

4.5.3. Une carte de grilles de ND

Afin d'utiliser la NDT pour l'alignement, nous avons besoin d'une représentation sous forme de grille de ND. Pour cela, une grille verticale est placée sur chaque segment de la carte de polygones et elle est remplie de distributions normales 3D (figure 4.9). La grille est délimitée par les deux points du segment $[P_1P_2]$ qui se trouvent au niveau du sol et avec une hauteur h arbitraire de telle sorte qu'elle couvre tout le bâtiment. Bien que les ND soient en 3D, la grille ne comporte qu'une seule couche de cellule, dont la taille est fixée et la profondeur est choisie arbitrairement pour englober tous les points possibles. La grille peut donc légèrement dépasser les points la délimitant, ce qui peut poser des

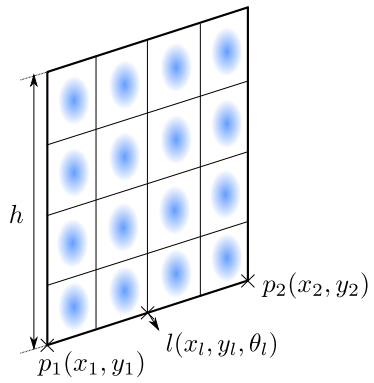


FIGURE 4.9. – Définition de la grille de distributions normales associée au segment $[P_1P_2]$.

problèmes si le dépassement est trop important, ce qui nécessitera des réglages. Les ND sont exprimées par rapport au repère de la façade.

Chaque cellule de la grille contient une ND dont les axes x et y suivent le plan de la façade. Nous avons choisi d'initialiser P_x et P_y de telle sorte que l'ellipse à 2σ soit inscrite dans la cellule. Cela se traduit par :

$$2\sigma = \frac{c}{2}, \tag{4.7}$$

$$\sigma = \frac{c}{4}, \tag{4.8}$$

$$P_x = P_y = \frac{c^2}{16}. \tag{4.9}$$

avec c la taille du côté de la cellule. P_z, la variance suivant l'axe de la normale à la façade, est un paramètre à définir pour chaque façade, en fonction de sa forme (plate, courbe) et de la répartition de ses points (fenêtres présentes, balcon, etc.).

4.6. Génération des observations

Cette section présente les différentes étapes nécessaires à la génération des observations issues du lidar (voir figure 4.4).

4.6.1. Segmentation sémantique du nuage de points

La première étape de l'approche consiste à extraire les bâtiments. Pour cela, Cylinder3D est utilisé. Cette méthode, présentée en figure 4.10, a été choisie car elle figure parmi les meilleures du classement de SemanticKITTI. L'architecture de Cylinder3D est composée d'un réseau de neurones à convolution (CNN) combiné à des MLP (*Multi-Layer Perceptron*, perceptron multi-couches) avec une grille radiale pour représenter les scans lidars (X. ZHU et al. 2021). Des *features* sont obtenues à partir d'un groupe de trois MLP. Une fois les *features* produites par le MLP, elles sont insérées dans une grille cylindrique en fonction

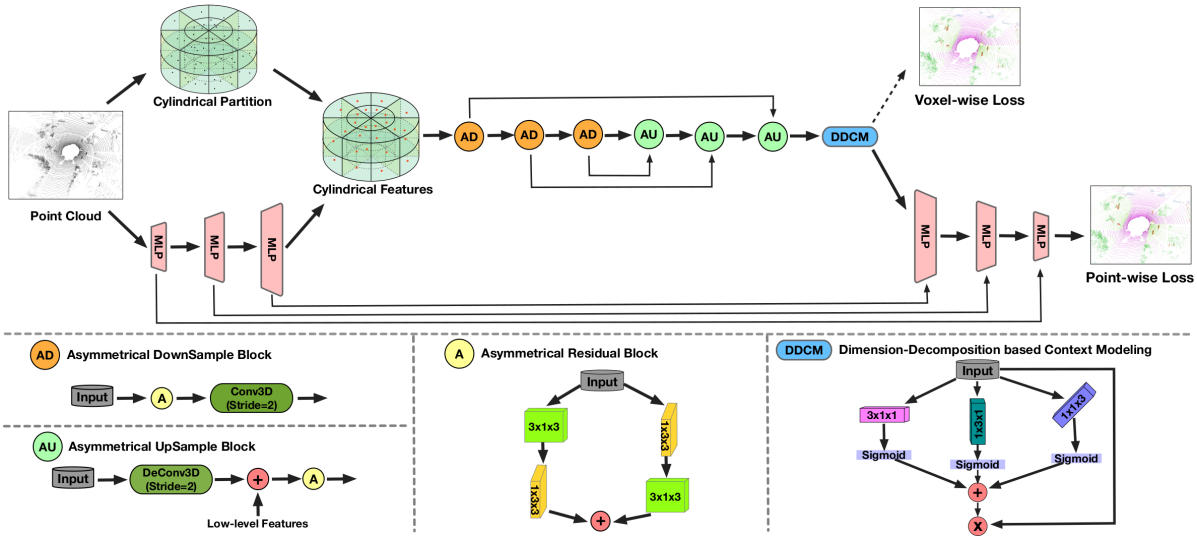


FIGURE 4.10. – Architecture de Cylinder3D. Illustration de (X. ZHU et al. 2021).

de leur emplacement spatial. Une représentation sous forme de grille cylindrique permet de s'adapter à la diminution de la densité des points avec la distance, inhérente aux lidars.

La grille cylindrique remplie de *features* est ensuite servie en entrée de trois blocs de convolution asymétriques, puis trois blocs de déconvolution, permettant d'abstraire l'ensemble de *features*, et ainsi les regrouper en classes. Une décomposition par dimension est ensuite utilisée pour obtenir des classes par voxel. Enfin, le MLP est renversé pour reproduire le nuage de points, avec les classes associées.

4.6.2. Alignement avec la NDT

La *Normal Distributions Transform* (NDT) est utilisée pour réaliser un alignement global du nuage de points contenant uniquement les bâtiments sur la carte. Cet alignement, en 3D, permet de limiter les erreurs d'association pour l'étape de groupement des points par façade, notamment au niveau des coins. La transformation issue de la NDT n'est pas utilisée comme observation de pose du véhicule dans le filtre, elle ne sert que pour améliorer l'association des points lidars aux façades.

La NDT est une méthode d'alignement de nuages de points permettant de trouver la transformation d'un nuage *source* vers un nuage *cible*. Dans notre cas, la cible est directement notre carte sous forme de ND où chaque cellule k contient une distribution normale définie par une moyenne ${}^{F_l}\mu_k$ et une covariance ${}^{F_l}P_k$ dans le repère de la façade F_l. Pour cela, les ND contenues dans chaque cellule sont d'abord transformées dans le repère global M. La moyenne de la distribution normale dans le repère de la carte est alors :

$$ {}^{M}\mu_k = \underbrace{\mathcal{R}_y\mathcal{R}_x\mathcal{R}_z}_{{}^{M}\mathcal{R}_{F_l}} {}^{F_l}\mu_k + {}^{M}\mathcal{T}_{F_l}, \tag{4.10} $$

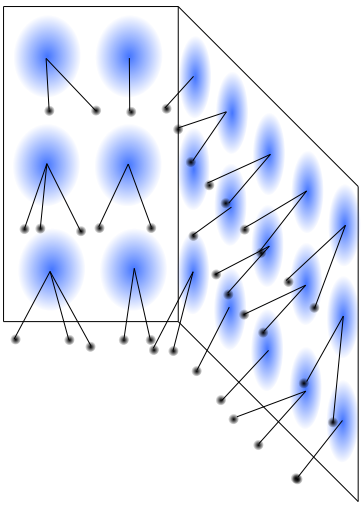


FIGURE 4.11. – Association de points lidar (en noir) à la distribution normale la plus proche.

où ${}^{M}\mathcal{R}_{F_l}$ est la rotation entre le repère de la façade et le repère carte avec $\mathcal{R}_x$ une rotation de $\frac{\pi}{2}$ autour de l'axe x, $\mathcal{R}_y$ une rotation de $\frac{\pi}{2}$ autour de l'axe y et $\mathcal{R}_z$ une rotation de θ_l autour de l'axe z, avec θ_l l'angle de la normale à la façade (l'orientation de la pose). ${}^{M}\mathcal{T}_{F_l}$ est la translation entre le repère de la façade F_l et le repère de la carte M, ce qui correspond à la position du milieu de la façade dans le repère de la carte.

La covariance de la ND de la cellule k dans le repère de la carte s'écrit :

$$ {}^{M}P_k = {}^{M}\mathcal{R}_{F_l}\, {}^{F_l}P_k\, {}^{M}\mathcal{R}_{F_l}^{T}. \tag{4.11} $$

Une fois les ND transformées dans le repère global, elles sont indexées dans un kd-tree en utilisant leur centre (μ_k), afin de pouvoir être retrouvée rapidement avec un algorithme du plus proche voisin, comme illustré à la figure 4.11 (JAVANMARDI et al. 2019).

L'objectif de la NDT est de trouver la transformation à appliquer au nuage source (X) issu du lidar pour qu'il se positionne au mieux sur la carte. Le nuage de points transformé est noté (X').

Afin d'améliorer la robustesse, la probabilité de mesurer un point X'_i par rapport à la ND la plus proche $\mathcal{N}(\mu_k, P_k)$ est représentée par une mixture entre une loi normale et une loi uniforme comme proposé dans (MAGNUSSON 2009) :

$$ p(X'_i) = \xi_1 \exp\left(-\frac{(X'_i - \mu_k)^T P_k^{-1} (X'_i - \mu_k)}{2}\right) + \xi_2 p_0, \tag{4.12} $$

où p_0 est la proportion des valeurs aberrantes, et ξ_1 et ξ_2 sont des paramètres fixés pour ajuster la répartition de la mixture.

La NDT consiste donc à trouver la transformation qui maximise la fonction de vrai-

semblance suivante :

$$L = \prod_{i=1}^{N_X} p(X'_i), \tag{4.13}$$

où N_X est le nombre de points du nuage de points source suffisamment proche d'une ND, suivant un seuil à fixer.

Afin de simplifier le processus de maximisation, la log-vraisemblance négative est utilisée :

$$-\ln L = -\sum_{i=1}^{N_X} \ln\left(p(X'_i)\right) = -\sum_{i=1}^{N_X} l(X'_i). \tag{4.14}$$

La log-vraisemblance négative pour un point s'écrit :

$$-l(X'_i) = -\ln\left(\xi_1 \exp\left(-\frac{(X'_i-\mu_k)^T P_k^{-1}(X'_i-\mu_k)}{2}\right) + \xi_2 p_0\right). \tag{4.15}$$

L'optimisation de cette fonction, à travers l'algorithme de Newton, nécessite de pouvoir la dériver deux fois, ce qui n'est pas facile à faire. (Magnusson 2009) propose l'utilisation d'une approximation gaussienne :

$$-\tilde{l}(X'_i) = d_1 \exp\left(-\frac{d_2}{2}(X'_i-\mu_k)^T P_k^{-1}(X'_i-\mu_k)\right) + d_3 \tag{4.16}$$

avec d_1, d_2 et d_3 des paramètres réglés de telle sorte que $-\tilde{l}(X)$ se comporte comme $-l(X)$ pour les points $X-\mu_k = 0$, $X-\mu_k = \sqrt{|P|}$ et $X-\mu_k \to \infty$. Ces paramètres sont définis comme suit :

$$d_3 = -\ln(\xi_2 p_0), \tag{4.17}$$

$$d_1 = -\ln(\xi_1 + \xi_2 p_0) - d_3, \tag{4.18}$$

$$d_2 = -2\ln\left(\frac{-\ln(\xi_1 \exp(-1/2) + \xi_2 p_0) - d_3}{d_1}\right). \tag{4.19}$$

Par conséquent, la fonction à optimiser s'écrit sous la forme d'une somme des vraisemblances approximées $-\tilde{l}(X'_i)$:

$$\text{score}(\mathbf{q}) = \sum_{i=1}^{N_X} d_1 \exp\left(-\frac{d_2}{2}(X'_i-\mu_k)^T P_k^{-1}(X'_i-\mu_k)\right) + d_3, \tag{4.20}$$

où X'_i est le point X_i transformé avec la transformation issue de $\mathbf{q}$ qui représente six paramètres, soit trois translations t_x, t_y et t_z, et trois rotations, r_x, r_y et r_z. Cela correspond à notre pose dans le repère global. Seuls les points suffisamment proches d'une ND sont pris en compte, les autres n'entrent pas dans le calcul.

L'optimisation avec l'algorithme de Newton fonctionne normalement sur des fonctions convexes qui ne contiennent qu'un seul minimum. Cependant, ce n'est pas nécessairement le cas avec la NDT. Pour cela, il faut s'assurer que l'initialisation de l'algorithme de

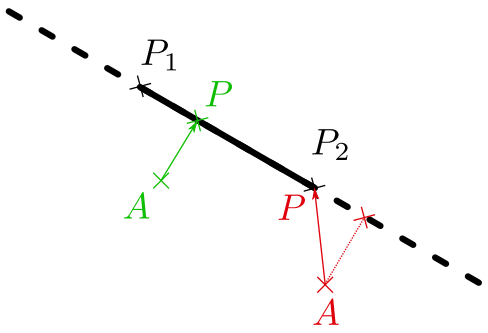


FIGURE 4.12. – Différents cas de projection d'un point sur un segment. En vert, le point projeté est sur le segment, il est donc conservé tel quel. En rouge, le point projeté n'est pas sur le segment, on prend donc le point du segment le plus proche comme projection finale.

Newton soit suffisamment proche du minimum global pour une bonne convergence. Nous utilisons donc l'estimation de notre pose par le SKF pour initialiser la NDT.

En pratique, le nombre d'itérations de l'algorithme de Newton est limité et une non-convergence est déclarée si cette limite est atteinte. Une fois le minimum trouvé, le paramètre $\mathbf{q}$ correspond à la transformation recherchée.

Même si la NDT est réalisée en 3D afin d'exploiter toutes les informations possibles, en réalité, le véhicule est supposé avoir une assiette horizontale, et donc la transformation peut se limiter à deux dimensions. Pour cela, le gradient projeté est utilisé (CALAMAI et MORÉ 1987) afin de contraindre la descente de gradient à fournir une transformation 3D compatible avec une transformation 2D, c'est-à-dire que la translation sur l'axe Z, et les rotations sur les axes X et Y sont mises à zéro ($t_z = 0$, $r_x = 0$ et $r_y = 0$). Lors de chaque itération, après avoir calculé les nouveaux paramètres $\mathbf{q}_{k+1}$, ces derniers sont projetés sur l'espace acceptable.

Afin d'accélérer le processus, le nuage de points aligné par la NDT est préalablement sous-échantillonné. La transformation résultant de l'alignement est ensuite appliquée au nuage de points complet pour l'étape de groupement par façade.

Comme déjà évoqué, la transformation obtenue dans cette étape est seulement utilisée pour l'association point-à-façade.

4.6.3. Association point-à-façade

Chaque point du nuage de points aligné sur la carte doit être associé à la façade la plus proche, dans la limite d'une certaine distance. Les façades étant verticales, et afin de simplifier les calculs, l'ensemble de l'opération est effectué en 2D. Le nuage de points est donc projeté sur le plan horizontal $z = 0$.

Afin de déterminer la projection d'un point A sur une façade représentée par son segment $[P_1P_2]$, le projeté P de A sur le segment $[P_1P_2]$ est déterminée et la distance euclidienne

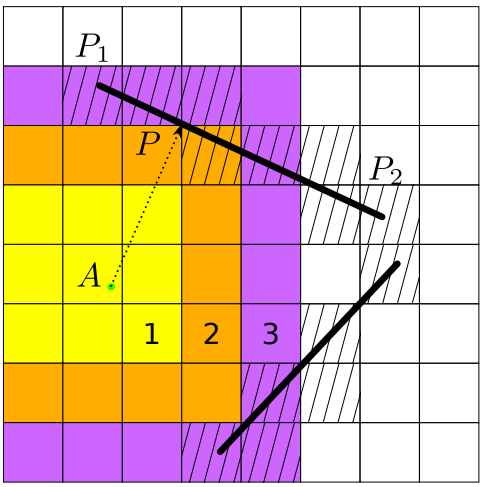


FIGURE 4.13. – Indexation puis recherche de la façade la plus proche. Les cellules traversées par une façade sont hachurées. Pour trouver la façade la plus proche du point vert, la première zone de recherche est en jaune. Comme il n'y a aucune façade dans cette zone, on étend à la zone orange. Une façade est trouvée dans la zone orange, on calcule donc sa distance. Afin d'être sûr qu'une façade plus proche n'a pas été oubliée dans une cellule adjacente, on étend une dernière fois à la zone violette. La dernière façade est plus loin, donc on garde la première façade.

entre A et P est calculée. Si le projeté est en dehors du segment, le point délimitant le segment le plus proche est sélectionné à la place du point projeté (figure 4.12) :

$$\alpha = \frac{\overrightarrow{P_1A} \,.\, \overrightarrow{P_1P_2}}{||\overrightarrow{P_1P_2}||^2} \tag{4.21}$$

$$P = \begin{cases} P_1 & \text{si } \alpha < 0, \\ P_2 & \text{si } \alpha > 1, \\ P_1 + \alpha \, \overrightarrow{P_1P_2} & \text{sinon.} \end{cases} \tag{4.22}$$

Pour déterminer la façade la plus proche, il est possible de calculer les projections sur toutes les façades. Afin de limiter le coût en calcul, une grille est utilisée pour indexer spatialement les façades. La façade la plus proche est sélectionnée en parcourant les cellules voisines de celle contenant le point (figure 4.13). Il convient de noter que cette grille est différente de celle de la carte de ND. Ici, la grille est dans le repère global, horizontale, et couvre toute la surface de la carte. Elle sert uniquement d'index.

La première étape est d'indexer chaque façade. Toutes les cellules de la grille traversées par une façade ajoutent à leur liste interne une référence vers cette façade (cellules hachurées dans la figure 4.13).

Par exemple, afin de déterminer les cellules traversées par un segment $[P_1P_2]$, il faut tout d'abord se placer dans la cellule contenant P_1, puis l'identifiant de la façade est ajouté à la liste contenue dans la cellule. Afin de savoir dans quelle cellule se déplacer, les points

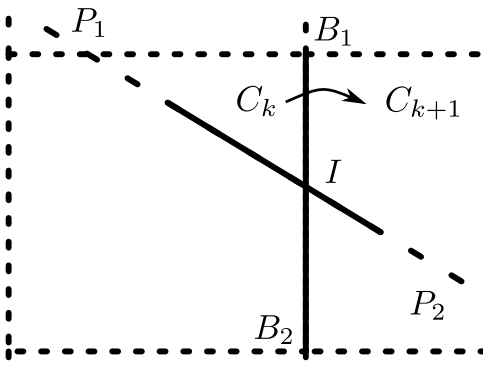


FIGURE 4.14. – Intersection entre le segment $[P_1P_2]$ et la bordure $[B_1B_2]$, puis déplacement dans la nouvelle cellule C_{k+1}.

d'intersection entre le segment et les bordures de la cellule sont calculés. L'intersection I entre le segment et une bordure $[B_1B_2]$ est obtenue comme suit (figure 4.14) :

$$I = \frac{(P_{1,x}P_{2,y} - P_{1,y}P_{2,x})\,\overrightarrow{B_2B_1} - \overrightarrow{P_2P_1}\,(B_{1,x}B_{2,y} - B_{1,y}B_{2,x})}{(P_{1,x} - P_{2,x})\,(B_{1,y} - B_{2,y}) - (P_{1,y} - P_{2,y})\,(B_{1,x} - B_{2,x})}. \tag{4.23}$$

Afin de déterminer si l'intersection se trouve sur le segment $[P_1P_2]$, le même procédé que pour la projection est utilisé :

$$\alpha = \frac{\overrightarrow{P_1P_2} \,.\, \overrightarrow{P_1I}}{||\overrightarrow{P_1P_2}||^2}. \tag{4.24}$$

L'intersection I est sur le segment $[P_1P_2]$ si $0 \leq \alpha \leq 1$. La même procédure est appliquée pour déterminer si I est sur la bordure $[B_1B_2]$. Si le point d'intersection I se trouve à la fois sur le segment $[P_1P_2]$ et sur la bordure $[B_1B_2]$, alors le segment possède une intersection avec la bordure, et le déplacement vers la cellule partageant la bordure est possible. L'identifiant de la façade est ajouté dans la liste de cette nouvelle cellule, puis les intersections sont calculées de nouveau. La procédure est répétée jusqu'à atteindre la cellule contenant P_2.

Cette méthode d'indexation est présentée dans l'algorithme 2.

Ensuite, lors des associations point-à-façade, les cellules sont parcourues par zones. La zone de départ est constituée de la cellule du point à associer, A, et des cellules voisines (zone jaune sur la figure 4.13). S'il n'y a pas de façade dans ces premières cellules, on étend la recherche aux cellules voisines à la zone (zone orange sur la figure 4.13), et ainsi de suite. Cela permet de rechercher localement la façade la plus proche. Dès qu'il y a au moins une façade dans la zone de recherche, les projections du point A sur les segments des façades sont calculées avec l'équation (4.22), ainsi que les distances entre ces points projetés et le point A. Lorsqu'une façade a été trouvée, la zone est tout de même étendue une dernière fois (zone violette sur la figure 4.13) pour couvrir des configurations particulières de façades.

La façade retenue à la fin est celle dont la distance est la plus faible.

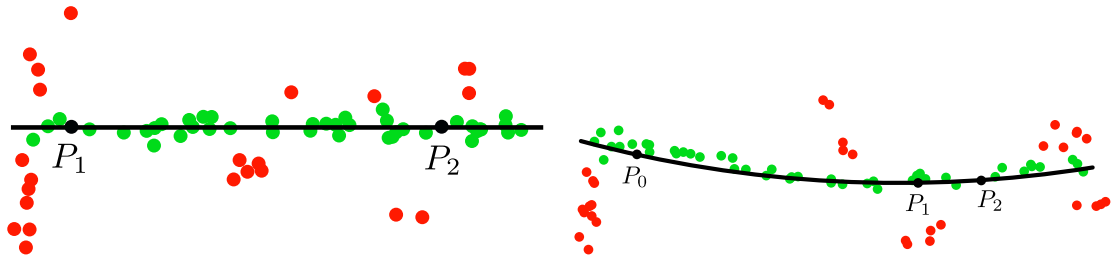
Algorithme 2 Indexation de la façade i représentée par le segment (P_1, P_2)

$C_k \leftarrow$ Cellule de P_1.
$C_{k-1} \leftarrow C_k$
Ajouter i à la liste de C_k.
tant que $C \neq$ Cellule de P_2 **faire**
$B_1, B_2, B_3, B_4 \leftarrow$ coins de la cellule C_k, dans le sens horaire en partant d'en haut à gauche.
$I_{12} \leftarrow$ intersection entre le segment (P_1, P_2) et la bordure (B_1, B_2).
$I_{23} \leftarrow$ intersection entre le segment (P_1, P_2) et la bordure (B_2, B_3).
$I_{34} \leftarrow$ intersection entre le segment (P_1, P_2) et la bordure (B_3, B_4).
$I_{41} \leftarrow$ intersection entre le segment (P_1, P_2) et la bordure (B_4, B_1).
/* *Déplacement dans la cellule partageant la bordure avec intersection :* */
si I_{12} existe et CelluleEnHaut $\neq C_{k-1}$ **alors**
$C_{k-1} \leftarrow C_k$.
$C_k \leftarrow$ CelluleEnHaut.
sinon si I_{23} existe et CelluleADroite $\neq C_{k-1}$ **alors**
$C_{k-1} \leftarrow C_k$.
$C_k \leftarrow$ CelluleADroite.
sinon si I_{34} existe et CelluleEnBas $\neq C_{k-1}$ **alors**
$C_{k-1} \leftarrow C_k$.
$C_k \leftarrow$ CelluleEnBas.
sinon si I_{41} existe et CelluleAGauche $\neq C_{k-1}$ **alors**
$C_{k-1} \leftarrow C_k$.
$C_k \leftarrow$ CelluleAGauche.
fin si
Ajouter i à la liste de C_k.
fin tant que

4.6.4. RANSAC

Après l'association des points aux façades, un RANSAC est appliqué sur le groupe de points correspondant à une façade afin d'éliminer un maximum de points aberrants pour pouvoir extraire l'observation correctement. Pour cela, la première étape consiste à transformer les points dans le repère du véhicule. Le RANSAC recherche un modèle dans ce nuage de points, puis fournit à la fois le modèle et les points appartenant à ce modèle (les *inliers*), comme illustré à la figure 4.15.

Afin de prendre en compte la possibilité que la façade soit courbée, deux modèles sont recherchés par le RANSAC : une ligne droite et un cercle. La forme regroupant le plus d'*inliers* est sélectionnée.



(a) RANSAC utilisant un modèle de ligne. (b) RANSAC utilisant un modèle de cercle.

FIGURE 4.15. – Sélection d'*inliers* avec deux modèles de RANSAC, le modèle recueillant le plus d'*inliers* est sélectionné à la fin. Les *inliers* sont représentés en vert et l'échantillon définissant la forme est en noir.

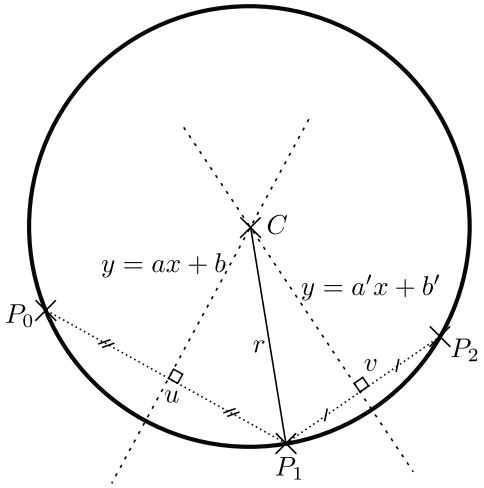


FIGURE 4.16. – Construction d'un cercle à partir de trois points P_0, P_1 et P_2.

Modèle de ligne

Le modèle de la ligne est défini par deux points P_1 et P_2 choisis aléatoirement (figure 4.15a). La distance d entre un point P_i et la ligne est déterminée en projetant P_i sur la droite (P_1P_2), puis en calculant la distance entre le point projeté et P_i.

Tous les points P_i dont la distance à la ligne est inférieure à un seuil sont sélectionnés comme *inliers*. Puis, la procédure est répétée avec une nouvelle paire aléatoire (P_1, P_2). Le couple de points ayant le plus grand nombre d'*inliers* est retenu comme résultat du RANSAC ligne.

Modèle de cercle

Le modèle du cercle est défini par un point central C et un rayon r, définis à partir de trois points, P_0, P_1 et P_2 (figure 4.15b). Le centre est situé à l'intersection des médiatrices des segments $[P_0P_1]$ et $[P_1P_2]$ (figure 4.16).

Pour déterminer l'intersection $C = \begin{pmatrix} x_c & y_c \end{pmatrix}$ de la médiatrice de $[P_0P_1]$, $y = ax + b$,

et de la médiatrice de $[P_1P_2]$, $y = a'x + b'$, on procède ainsi :

$$\begin{cases} y_c = ax_c + b \\ y_c = a'x_c + b' \end{cases} \tag{4.25}$$

$$\iff \begin{cases} y_c = ax_c + b \\ x_c = \frac{b'-b}{a-a'} \end{cases} \tag{4.26}$$

La pente a de la médiatrice de $[P_0P_1]$ est (figure 4.16) :

$$a = -\frac{x_1 - x_0}{y_1 - y_0}. \tag{4.27}$$

Son ordonnée à l'origine se calcule de telle sorte que la médiatrice passe par le milieu de $[P_0P_1]$, noté u :

$$b = -ax_u + y_u, \tag{4.28}$$

$$b = \left(\frac{x_1 - x_0}{y_1 - y_0}\right)\left(\frac{x_0 + x_1}{2}\right) + \frac{y_0 + y_1}{2}. \tag{4.29}$$

La même procédure est appliquée pour la détermination de a' et b'.

Le centre C peut être maintenant calculé. Pour le rayon r, son calcul se réalise de manière triviale via la distance entre le centre et un des trois points.

Pour pouvoir être utilisé avec le RANSAC, une distance d entre un point P_i et le modèle de cercle doit être calculée :

$$d = |\,(||P_i - C||) - r|. \tag{4.30}$$

Si cette distance est inférieure à un seuil, alors le point est un *inlier*.

Le modèle de cercle avec le plus d'inliers est sélectionné.

Critère d'arrêt

Pour fixer le nombre optimal d'itérations, la méthode proposée dans (Schnabel, Wahl et Klein 2007) est utilisée : l'objectif est que la probabilité p_t qui représente la probabilité désirée de trouver la forme optimale soit suffisamment grande. La probabilité de détecter une forme définie par m points, dans un nuage de points de taille N est définie par :

$$p(n) = \frac{\binom{n}{m}}{\binom{N}{m}} \approx \left(\frac{n}{N}\right)^m, \tag{4.31}$$

où n est le nombre de points appartenant à la forme (les *inliers*) et $\binom{N}{m}$ est le coefficient binomial qui représente le nombre de combinaisons de m éléments parmi N.

La probabilité de trouver la forme optimale après k itérations est définie par (Schnabel, Wahl et Klein 2007) :

$$p(n,k) = 1 - (1 - p(n))^k. \tag{4.32}$$

Le nombre minimal d'itérations $k_{\min}$ requis pour que $p(n, k_{\min}) \geq p_t$ est :

$$k_{\min} \geq \frac{\ln(1-p_t)}{\ln(1-p(n))}. \tag{4.33}$$

n étant généralement petit devant N, $p(n)$ est faible, donc le développement de Taylor peut être appliqué :

$$k_{\min} \approx -\frac{\ln(1-p_t)}{p(n)}. \tag{4.34}$$

Sélection finale du modèle et des *inliers*

Lorsque les deux modèles de RANSAC ont effectué leurs itérations, ils proposent tous les deux un ensemble d'*inliers* et une forme. Le modèle ayant le plus grand nombre d'*inliers* est sélectionné, puis seuls ces points-là sont gardés dans le groupe de points associés à la façade.

4.6.5. Construction des observations de façades

Maintenant que les points ont été répartis en groupes par façade en enlevant le plus de points aberrants, l'objectif est d'extraire pour chaque groupe une observation sous forme de la pose du centre de la façade pour être utilisée dans le SKF (figure 4.17).

Dans un couloir urbain, le calcul du positionnement dans la direction longitudinale est particulièrement compliqué. Afin de limiter les erreurs des observations dans le plan de la façade, nous avons choisi de garder uniquement les observations de façade complètes. Il est à noter que nous avons essayé de construire des observations partielles de façades. Pour cela, nous avons d'abord utilisé une NDT dont le but était d'associer les groupes de points aux façades de la carte et d'obtenir une transformation partielle par façade (voir l'annexe B). Cependant, cette méthode créait une dépendance forte entre les observations et l'estimation de la pose du véhicule.

Pour obtenir une observation (complète donc) de façade, parmi les *inliers* du RANSAC, les deux points extrêmes sont sélectionnés. Pour cela, chaque point classifié comme *inlier* par le RANSAC est projeté sur le modèle (ligne ou cercle), puis les distances entre les points projetés sont calculées, en les comparant deux-à-deux. Les deux points les plus éloignés l'un de l'autre sont sélectionnés (figure 4.17b). Tout ceci est réalisé dans le repère du véhicule, dans lequel le nuage de points a été transformé avant le RANSAC.

Une fois les deux points extrêmes sélectionnés (E_1 et E_2 sur la figure 4.17c), l'observation Z_l^v est construite comme étant le point central entre les deux, de la même façon que la

(a) Groupe de points correspondant à la façade traitée.

(b) Les points sont projetés sur le modèle du RANSAC et les plus extrêmes sont sélectionnés (en rose).

(c) L'observation Z_l^v est générée au milieu des deux points extrêmes et la normale est calculée afin qu'elle soit dans le même sens que la carte.

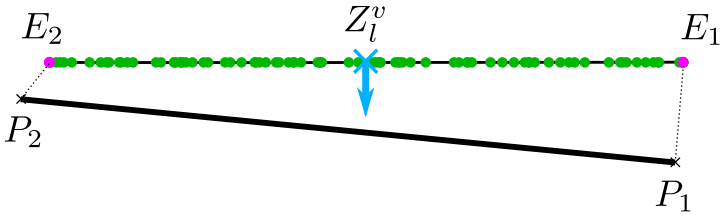


FIGURE 4.17. – Construction d'une observation via l'observation des points extrêmes d'une façade.

représentation de la façade sous forme de pose (section 4.5.2) :

$$x_l^v = \frac{x_i + x_{i+1}}{2}, \tag{4.35}$$

$$y_l^v = \frac{y_i + y_{i+1}}{2}, \tag{4.36}$$

$$\theta_l^v = \operatorname{atan2}\left(y_{i+1} - y_i, x_{i+1} - x_i\right) + \frac{\pi}{2}. \tag{4.37}$$

Afin d'orienter la normale dans le même sens que la carte, il faut que l'ordre des indices des points soient le même que lors de la construction de la carte. Les points extrêmes sont donc associés aux extrémités de la façade dans la carte, au plus proche. Ainsi, on conserve le même ordre d'indexation (figure 4.17c).

L'observation est retenue si la distance entre les deux points extrêmes est comprise entre 95 % et 105 % de la longueur de la façade dans la carte. Cela permet de rejeter les façades observées seulement en partie ou les observations de façades trop longues qui contiennent des points aberrants.

4.7. Lien avec le filtre

Les observations construites sont transmises au SKF, sous forme de poses relatives. À chaque observation est associé l'identifiant de la façade (et du véhicule pour collaborer avec les autres). On obtient ainsi l'observation $Z_{l_j}^{v_i}$ de la façade l_j par le véhicule v_i.

Pour rappel, le SKF utilisé pour la localisation et la cartographie collaborative, estime un état joint contenant la pose des véhicules et la pose des amers dans la carte $X =$

$\begin{pmatrix} X_V \\ X_L \end{pmatrix}$, où V est l'ensemble des véhicules et L l'ensemble des amers. Chaque pose d'un amer $X_{l_i} \in X_L$ correspond à une façade identifiée l_i dans la carte.

Lors de l'initialisation du filtre, l'état est construit en utilisant la représentation sous forme de poses de la carte, avec une covariance associée correspondant à l'incertitude de la carte initiale. Nous utilisons OSM comme source de la carte, et la plupart des bâtiments proviennent du cadastre. La covariance est donc initialisée en fonction de la précision du cadastre. L'ensemble de la carte de la zone d'évolution est chargé dans le filtre dès le départ pour des raisons de simplification de mise en œuvre expérimentale.

La transformation initiale de la NDT provient de l'étape de prédiction du filtre, au moment de la capture du nuage de points par le lidar. Il faut donc conserver un buffer contenant les dernières estimations avec la date associée pour gérer les délais dus aux calculs. La prédiction de la pose du véhicule provenant du SKF intervient seulement pour l'association point-à-façade qui suit.

4.8. Conclusion

Dans ce chapitre, la méthode utilisée pour extraire des observations utilisables pour la localisation collaborative et la mise à jour de carte a été présentée. Cette méthode, utilisant des nuages de points lidar 3D, est composée de cinq étapes. Tout d'abord, le nuage de points est segmenté sémantiquement et seuls les bâtiments sont conservés. Ensuite, le nuage de points global est aligné avec la carte en utilisant la NDT et des grilles de ND définies par façade. Les points sont ainsi groupés par façade, en sélectionnant la façade la plus proche. Avant d'extraire l'observation finale, les groupes de points par façade sont affinés avec un RANSAC pour garder un minimum d'*outliers*. Et enfin, les observations sont générées comme étant les milieux des points les plus éloignés dans chaque groupe pour avoir la même représentation que la carte. Ce choix du milieu du segment vient de l'objectif initial de pouvoir observer une façade de manière partielle, et d'avoir une même observation quelle que soit la proportion de la façade observée.

La présence d'observations dépend de la bonne association des points avec les façades, et donc du bon alignement entre le nuage de points et la carte, grâce à la NDT. Cependant, cela entraine une dépendance du nombre d'observations avec la qualité de l'estimation précédente du filtre, car cette estimation sert d'initialisation à la NDT après une phase de prédiction. L'erreur d'estimation n'aura pas une influence sur l'observation en elle-même, mais seulement sur son existence. Une divergence au niveau de l'estimation (par exemple, à la suite d'un long moment à l'odométrie) réduit considérablement le nombre d'observations qui sont pourtant les informations pouvant corriger cette divergence. Lorsque le filtre a dérivé de façon importante et qu'aucune observation ne peut être générée avec la méthode présentée dans ce chapitre, il faudrait utiliser une méthode permettant de détecter et d'identifier les façades observées sans avoir besoin d'une étape d'association avec la carte.

5. Étude expérimentale de la méthode proposée

Sommaire

5.1. Introduction

Ce chapitre présente les résultats de l'ensemble de la méthode mettant en œuvre le SKF, la KLA et la génération des observations à partir du lidar sur des données réelles. Les données ont été acquises avec des véhicules du laboratoire. Le SKF ayant déjà été évalué en simulation dans le chapitre 3, ce chapitre se focalise donc sur son application à des données réelles.

Un nombre important de développements informatiques supplémentaires ont été nécessaires afin de permettre de faire fonctionner toute la chaine de génération d'observations. Ils sont résumés dans ce chapitre, cela comprend l'adaptation de Cylinder3D aux données ROS et l'adaptation de la NDT développée par la PCL à notre problème.

5.2. Acquisition des données

Les données utilisées pour évaluer la méthode proposée dans cette thèse ont été capturées avec les véhicules du laboratoire, en juillet 2022. Les véhicules étaient trois Renault Zoé, chacune équipée des capteurs suivants (figure 5.1) :

— Un lidar rotatif Velodyne VLP32-C, fonctionnant à 10 Hz, avec 32 nappes et positionné sur le toit de la voiture.
— Un capteur d'odométrie fournissant la vitesse longitudinale et la vitesse de rotation à partir d'un gyromètre de lacet et des codeurs des roues à une fréquence de 100 Hz.

FIGURE 5.1. – Véhicules utilisés avec leurs capteurs, dans un environnement difficile constitués de bâtiments métalliques avec des passerelles. Une grille de ND verticale est représentée sur une façade.

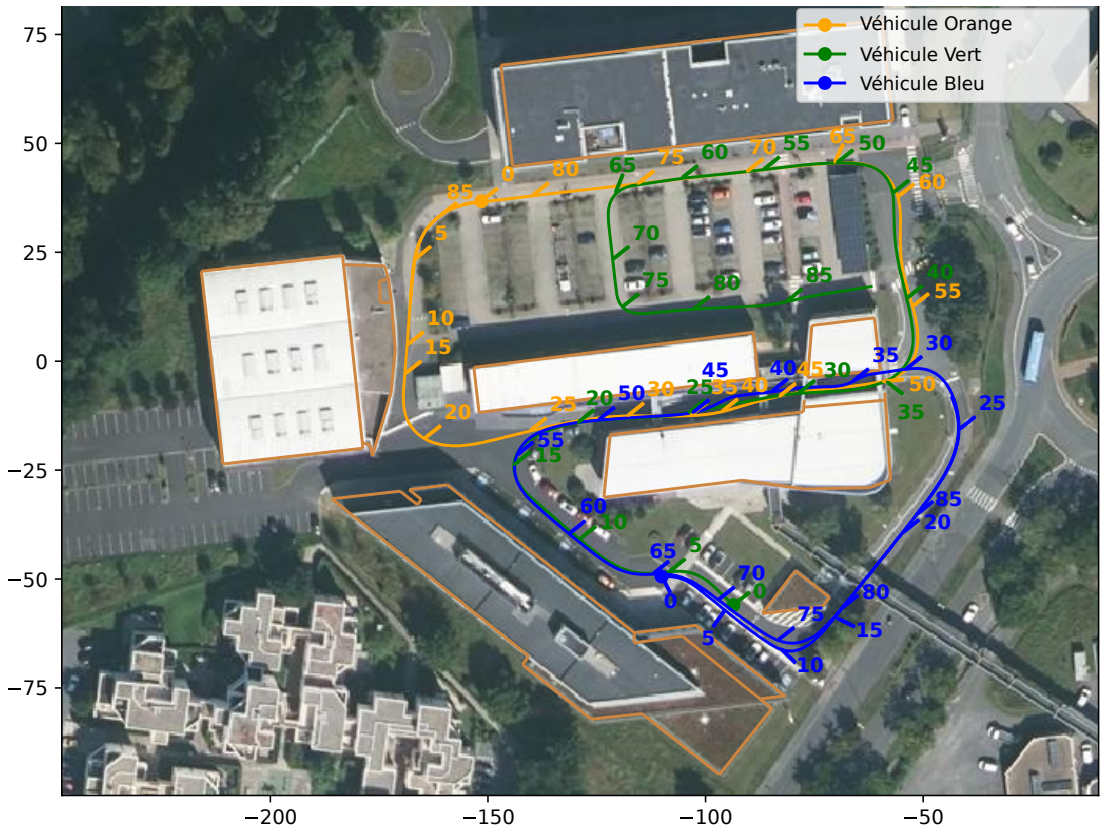


FIGURE 5.2. – Trajectoire des trois véhicules pour le scénario en données réelles avec les instants.

— Un système de positionnement RTK, utilisant une centrale inertielle SPAN-CPT, qui est post-traité, afin de fournir une vérité terrain de localisation.

L'enregistrement a été réalisé sur le campus de l'université, en suivant les trajectoires présentées à la figure 5.2. Ce scénario met en avant des zones de collaboration assez importantes, représentées avec des poses reliées par des traits (figure 5.3). L'expérimentation contient des zones assez compliquées pour le lidar comme par exemple la zone étroite entourée de bâtiments métalliques avec plusieurs passerelles et escaliers métalliques montrée à la figure 5.1.

Concernant la carte, les bâtiments sont extraits d'OpenStreetMap (OSM), puis ont été retravaillés pour n'avoir qu'un seul segment par façade. Une vérité terrain de la carte est disponible. Elle a été réalisée par des géomètres experts. Cependant, les représentations de la longueur des façades ne sont pas toujours les mêmes entre la vérité terrain et OSM. À cause de cela, les erreurs seront calculées suivant la normale de la façade.

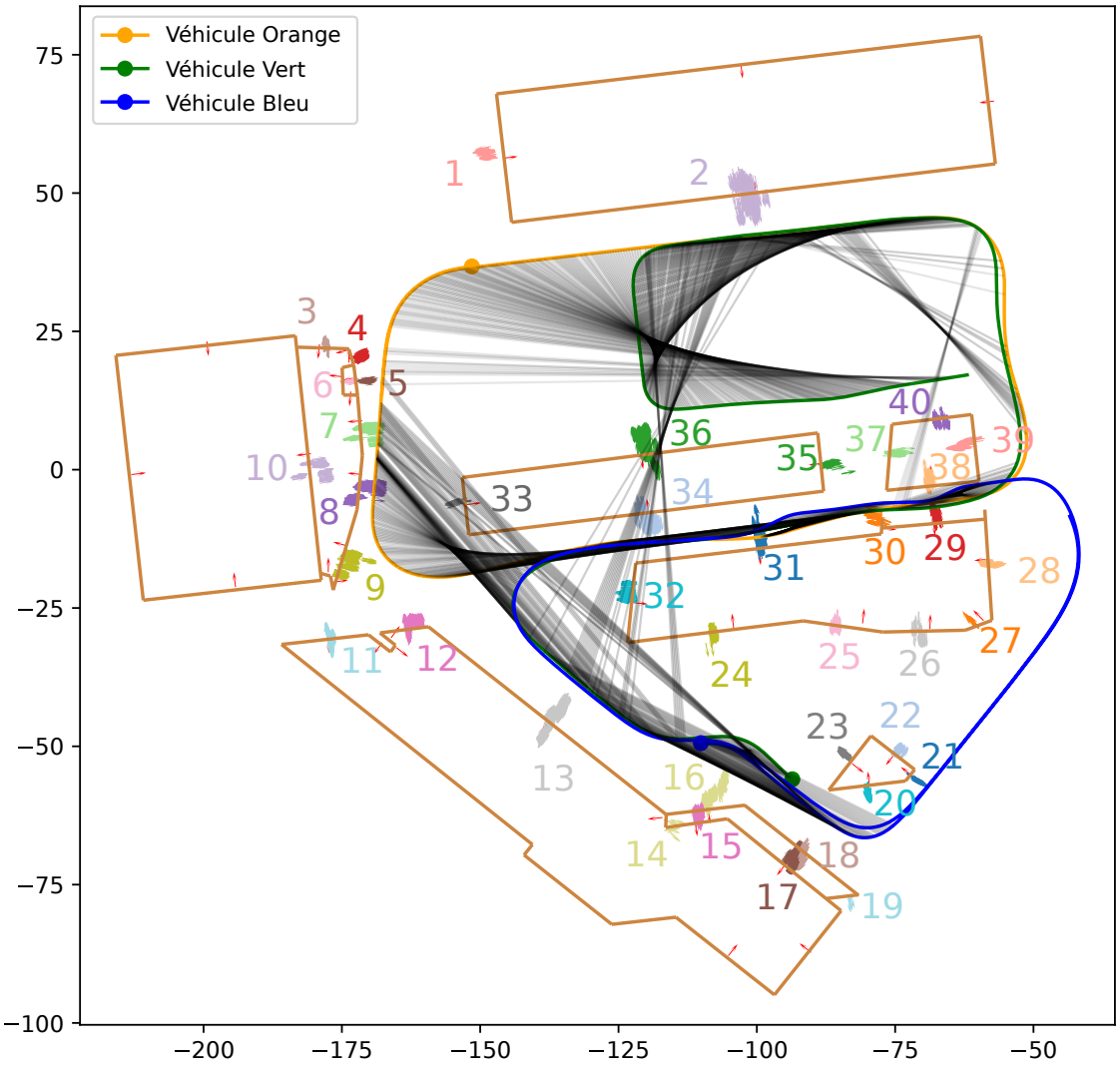


FIGURE 5.3. – Collaborations entre les véhicules. Les poses des véhicules permettant la collaboration sont reliées entre elles. Les observations des véhicules sont superposées à la carte sous forme de flèches avec les ID associés, ils sont convertis dans le repère de travail via la vérité terrain des véhicules.

5.3. Scénario

Les noms des véhicules correspondent à la couleur de leurs trajectoires (figure 5.2). Au début de l'expérimentation, les véhicules bleu et vert collaborent jusqu'à $t = 12$ s en se croisant. Le véhicule orange se déplace en mode isolé jusqu'à $t = 8$ s où il rejoint les deux autres véhicules, les trois véhicules se retrouvent alors en collaboration (via l'amer 13 à la figure 5.3). Ensuite, le véhicule bleu se retrouve tout seul (à partir de $t = 12$ s) pendant que le véhicule orange suit le véhicule vert dans le canyon, où ils collaborent. Le véhicule bleu retrouve le véhicule vert, vers $t = 30$ s. Cela permet au véhicule bleu de collaborer avec le vert puis l'orange pendant une vingtaine de secondes. Pour terminer, le véhicule bleu continue tout seul à partir de $t = 47$ s, et les véhicules orange et vert se suivent jusqu'à $t = 65$ s où leurs routes se séparent. Ils collaborent à travers les observations des amers 2 et 36 (entre autres). Ces deux véhicules ont quelques occasions pour collaborer avec le véhicule bleu via l'amer 9 autour de $t = 60$ s, ce qui permet une collaboration longue distance sans ligne de vue directe.

5.4. Adaptations expérimentales

Le code source de Cylinder3D permet une utilisation avec les formats des jeux de données SemanticKITTI et NuScenes. Dans le cadre de cette thèse, Cylinder3D (X. Zhu et al. 2021) a été adapté pour fonctionner sous ROS où un nœud a été créé pour recevoir les nuages de points sous forme de messages ROS, puis les transformer dans le format accepté par Cylinder3D.

Cylinder3D a été entrainé sur le jeu de données SemanticKITTI avec plusieurs tailles de grille. Pour les résultats présentés dans ce manuscrit, la grille la plus fine a été sélectionnée (360 segments angulaires, divisés en 32 couches en hauteur et 960 cellules le long du rayon). La segmentation avec Cylinder3D a été faite une fois, puis le nuage de points segmenté a été enregistré pour être rejoué par la suite. La figure 5.4 montre l'extraction des points classifiés comme étant des bâtiments ou des barrières.

Avec cette segmentation, des points n'appartenant pas à des bâtiments peuvent persister dans le nuage de points, ce qui peut poser des problèmes pour la suite, et rend la détection de défauts nécessaire. En effet, l'entrainement du réseau de neurones a été réalisé uniquement sur le jeu de données KITTI qui présente des différences par rapport à nos données réelles, sur le type de bâtiments présents.

Pour tous les traitements des nuages de points, nous utilisons la PCL (*Point Cloud Library*[1]). La NDT implémentée dans cette bibliothèque n'est pas très optimisée, notamment en termes de parallélisation. Nous utilisons donc une version multicœur[2] afin d'accélérer l'alignement.

D'autre part, la version originale de la NDT est uniquement adaptée pour le calcul des transformations entre deux nuages de points. Des adaptations ont été réalisées pour

1. `https://pointclouds.org/`
2. `https://github.com/koide3/ndt_omp`

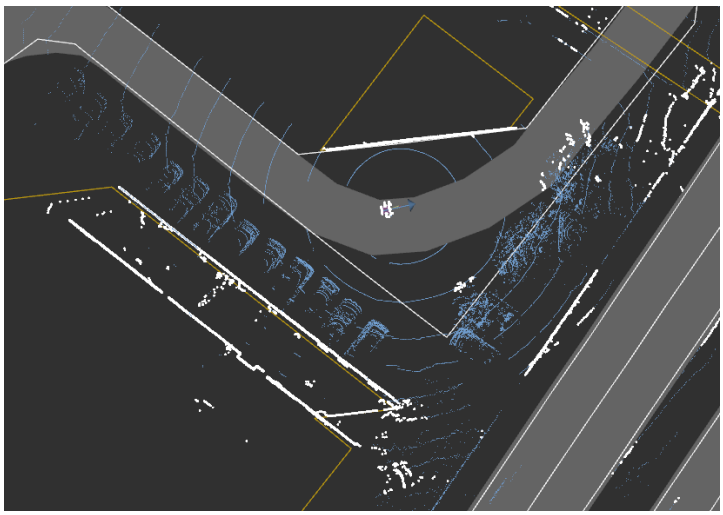


FIGURE 5.4. – Segmentation sémantique du nuage de points via Cylinder3D. Le nuage de points brut est représenté en bleu, et le nuage de points segmenté en blanc. Le fond de carte représente la route en gris et les bâtiments provenant d'OSM en marron.

pouvoir utiliser la NDT avec notre format (carte sous forme de distributions normales) et avec le kd-tree.

La NDT utilise des grilles avec des cellules de 1 m de côté, et une variance 0,1 m^2 suivant la normale de la façade. La distance maximale d'association entre un point et sa ND la plus proche est de 3 m, et le nombre maximal d'itérations a été limité à 50.

Le seuil utilisé pour le RANSAC est de 20 cm.

Un exemple d'observations générées par le véhicule bleu est montré au niveau de la figure 5.5 sous forme de poses représentées par des flèches.

5.5. Résultats de la localisation et de la mise à jour de carte

Cette section présente une comparaison entre les méthodes collaborative et isolée pour la localisation et la mise à jour de la carte sur le scénario présenté ci-dessus. Aucun défaut n'est injecté pour le moment. Cependant, le module FDIR est utilisé afin d'exclure les observations de mauvaises qualités, ce qui peut arriver fréquemment.

Au niveau des variances des observations, une variance de 0,15 rad^2 a été fixée sur l'orientation comme cette composante est assez bruitée. Cette valeur est importante, mais il a été observé de moins bons résultats avec une variance sur l'orientation plus faible. En effet, une erreur mal estimée sur l'orientation peut avoir de grands impacts au niveau de l'estimation de pose du véhicule, et cela d'autant plus lorsque la façade observée est loin. Au niveau de Q_p, une valeur conservatrice a été choisie, puisqu'il a été montré en simulation que l'impact de Q_p était faible lorsque la KLA est utilisée.

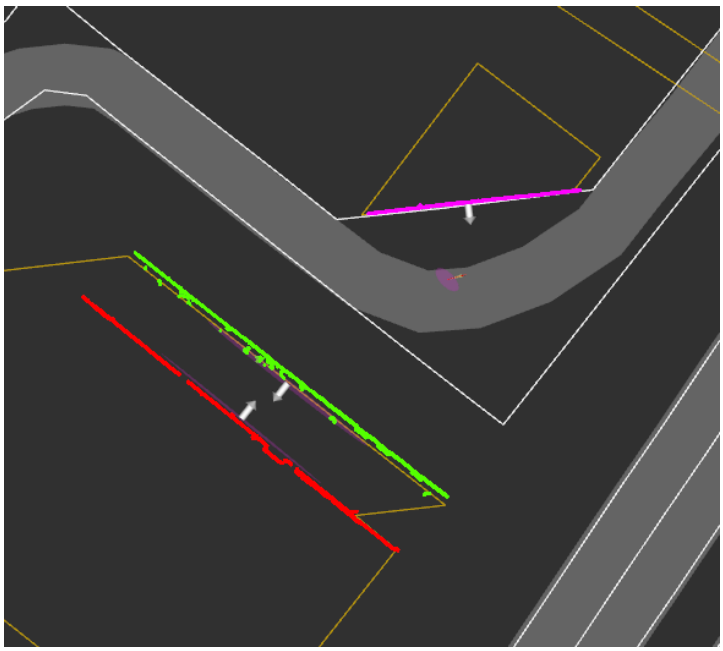


FIGURE 5.5. – Génération des observations par façade, par le véhicule bleu sous forme d'une pose représentée par une flèche.

Résultats

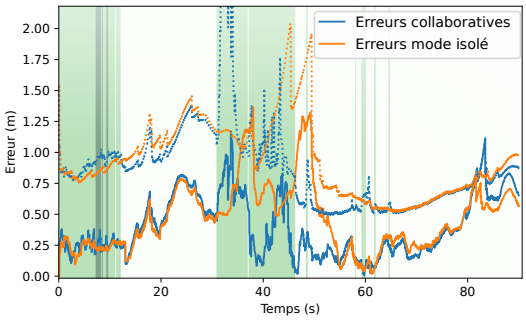
La figure 5.6 montre les erreurs avec les régions d'incertitude associées à 99,97 % pour les trois véhicules. Cette probabilité correspond à 4σ d'une distribution de Rayleigh, utilisée ici, car les erreurs correspondent à des distances euclidiennes. Les zones vertes en arrière-plan correspondent aux moments de collaboration : en vert clair lorsque le véhicule représenté collabore avec un autre et en vert foncé lorsqu'il collabore avec les deux autres véhicules.

Pour le véhicule vert, une augmentation de l'erreur en mode collaboratif peut être observée entre 25 et 30 secondes, ce qui se situe pendant la traversée de la zone difficile où les façades sont difficilement détectables. Puis l'erreur diminue pendant la collaboration avec le véhicule bleu vers $t = 30$ s et conserve des valeurs assez faibles après la fin de la collaboration à $t = 38$ s. Ceci s'explique par une augmentation du nombre d'observations, allant jusqu'à 9 sans collaboration. Cependant, en mode isolé, le véhicule subit une plus grande augmentation d'erreur dans la zone difficile comme moins d'observations sont disponibles. En effet, lors de la traversée de la zone difficile, seules une ou deux observations ont été réalisées par le véhicule en mode isolé. La collaboration apporte une ou deux autres observations, cela représente une hausse importante d'information disponible.

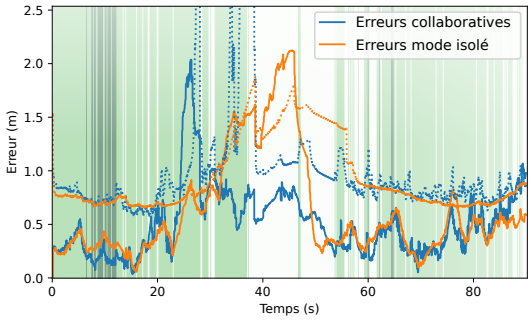
Au niveau du véhicule bleu, la méthode isolée est mise en difficulté aux alentours de 40 secondes, au moment du passage dans la zone difficile. On note cependant une grande similarité entre les deux méthodes au début et à la fin.

Les deux véhicules, vert et bleu, s'entraident via la collaboration et l'échange de carte.

(a) Véhicule Bleu.



(b) Véhicule Vert.



(c) Véhicule Orange.

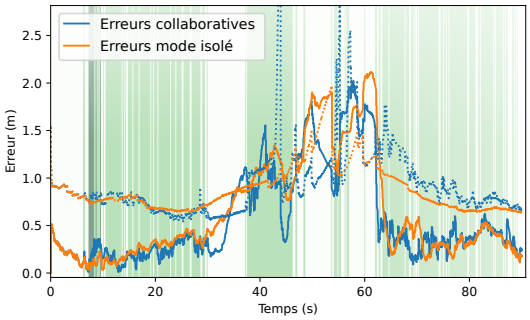


FIGURE 5.6. – Erreurs d'estimation des véhicules (en traits pleins), avec leur région d'incertitude à 99,97 % (en pointillés).

En effet, ces deux véhicules viennent de directions opposées. Lorsqu'ils se croisent, chacun récupère via la collaboration une meilleure version de la carte de la zone à venir, ce qui explique l'amélioration de leurs résultats après la collaboration.

Le véhicule orange ne collabore pas jusqu'à $t = 8$ s et par conséquent, les erreurs en mode isolé et collaboratif pour ce véhicule coïncident. À partir de $t = 8$ s, le véhicule orange collabore avec le véhicule vert puis le suit à partir de $t = 20$ s. L'erreur subit une augmentation entre 40 et 60 secondes, ce qui correspond à la zone difficile. La méthode isolée semble avoir plus de difficultés à se corriger, ce qui s'explique par le nombre plus important d'observations en mode collaboratif.

Les trois véhicules ont eu des difficultés à traverser la zone difficile. Cependant, la méthode collaborative se comporte mieux, car elle utilise plus d'observations et elle permet l'échange de cartes corrigées.

Les valeurs moyennes des erreurs des véhicules sont montrées à la table 5.1 et à la figure 5.7. Le mode collaboratif apporte une amélioration moyenne de l'ordre de 15 % par rapport au mode isolé. Ce faible écart s'explique par le grand nombre d'observations en mode isolé (jusqu'à 9 observations), par rapport aux observations collaboratives (qui en ajoutent de 1 à 5). Par conséquent, l'apport d'information en mode collaboratif est assez faible par rapport au mode isolé. Cependant, ces observations auront un impact majeur pour la détection des défauts cartes.

La figure 5.8 et la table 5.1 montrent que les erreurs des amers restent dans le même

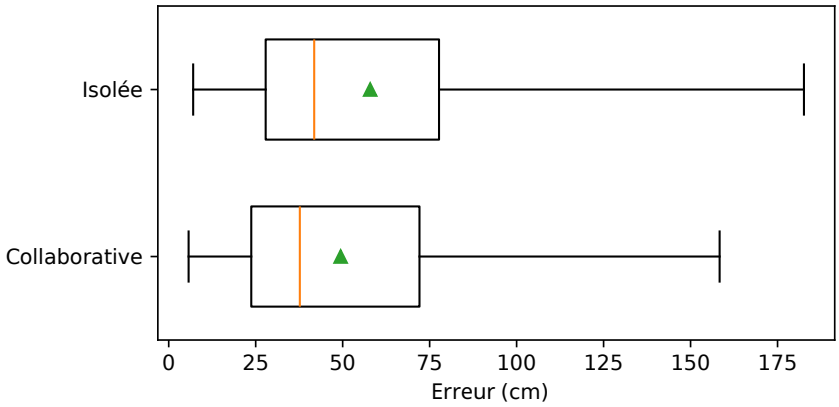
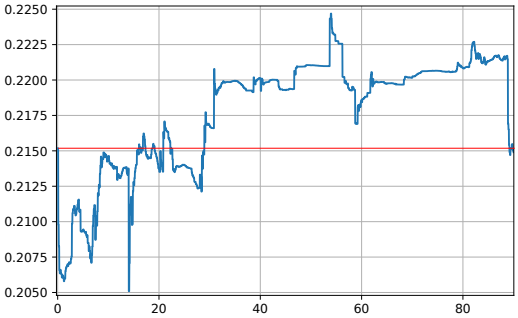


FIGURE 5.7. – Boîte à moustache représentant les moyennes (triangles verts), médianes (trait orange), les premier et troisième quartiles (rectangle) ainsi que les premier et quatre-vingt-dix-neuvième percentiles (extrémités) de la localisation des véhicules.

(a) Véhicule Vert collaboratif.



(b) Véhicule Vert isolé.

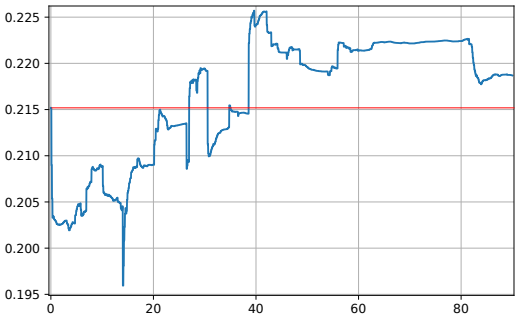


FIGURE 5.8. – Biais global de la carte en collaboratif et en mode isolé, exprimé en mètres. La ligne rouge correspond au biais initial.

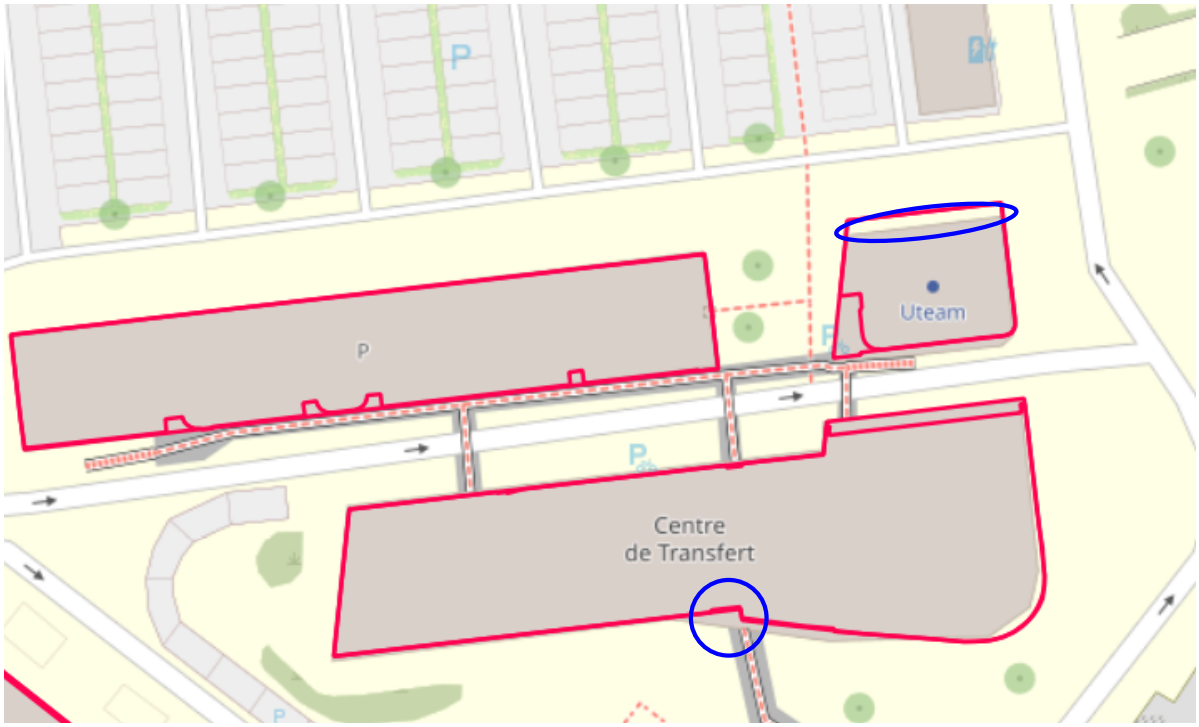


FIGURE 5.9. – Différence de représentation des façades entre la vérité terrain (rouge) et OSM (en fond).

Méthodes \ Véhicules	Bleu	Vert	Orange	Tous	Moyenne carte
Collaborative	39,4	50,9	57,9	49,4	36,0
Isolée	46,5	62,6	64,6	57,9	36,7

TABLE 5.1. – Erreurs moyennes en centimètres des véhicules et de la carte en fonction de la méthode.

Méthodes \ Véhicules	Bleu	Vert	Orange	Tous
Collaborative	99,1	95,4	82,0	92,2
Isolée	97,7	85,3	77,7	86,9

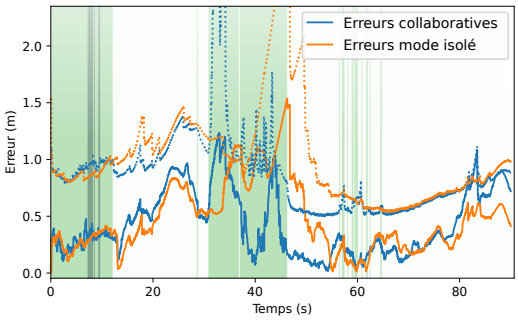
TABLE 5.2. – Proportions (%) des erreurs inférieures à la borne d'incertitude à 99,7 %.

ordre de grandeur pour les méthodes collaboratives et isolées. Ceci s'explique par une carte initiale avec de bons positionnements relatifs des amers. La vérité terrain n'utilise pas la même représentation des longueurs de façades qu'OSM, certaines façades ont donc un centre décalé (figure 5.9). Cette différence n'est pas observable par notre méthode. C'est pourquoi nous avons choisi de calculer les erreurs de carte suivant les normales des façades, afin de mesurer les performances du filtre là où cela est pertinent. Le biais final de la carte se situe entre 20 et 22 cm avec une légère amélioration pour la méthode collaborative. Ce biais n'est pas observable sans mesures absolues.

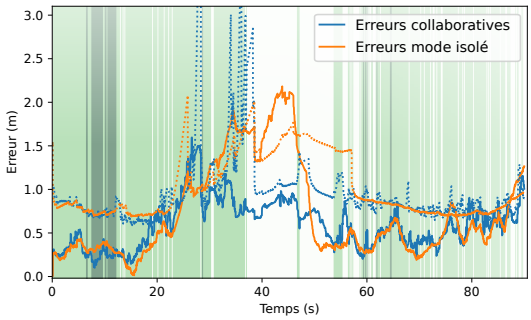
La table 5.2 montre une étude de consistance avec les proportions d'erreurs qui dépasse la région à 99,7 %, ce qui correspond à 3σ pour une distribution normale. Afin de calculer ces chiffres, les proportions de dépassement des erreurs suivant x et y par rapport à leur composante de covariance sont déterminées puis moyennées. Bien que seul le véhicule bleu atteigne les 99,1 % correspondant à la proportion attendue avec une région d'incertitude à 3σ, on observe que la collaboration permet d'améliorer la consistance des estimations. En effet, 92,2 % des erreurs sont à l'intérieur de la région d'incertitude dans le cas collaboratif alors que cette valeur n'est que de 86,9 % en mode isolé. Cela est dû surtout au plus grand nombre d'observations utilisées grâce à la collaboration, permettant une meilleure estimation de la localisation des véhicules et des amers.

Pour conclure, le pourcentage d'amélioration sur la localisation et sur les cartes dépend largement du scénario. Dans cet exemple, les véhicules observent quasiment les mêmes amers, et par conséquent, aucun véhicule ne profite des informations sur un amer non observé provenant d'un autre véhicule.

(a) Véhicule Bleu.



(b) Véhicule Vert.



(c) Véhicule Orange.

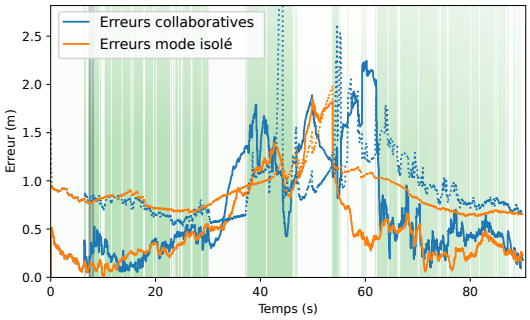


FIGURE 5.10. – Erreurs d'estimation des véhicules (en traits pleins), avec leurs incertitudes à 99,97 % (en pointillés), après injection de défauts.

Méthodes \ Véhicules	Bleu	Vert	Orange	Tous	Moyenne carte
Collaborative	**43,10**	**59,67**	69,71	**57,50**	**38,6**
Isolée	47,56	72,56	**53,05**	57,72	49,3

TABLE 5.3. – Erreurs moyennes en centimètres des véhicules et de la carte en fonction de la méthode, avec des défauts d'injectés.

5.6. Résultats sur la détection et l'exclusion des défauts

Afin d'évaluer la détection de défauts, des biais ont été simulés au niveau des observations entre les instants 2 et 15 s pour tous les véhicules, en vérifiant les hypothèses décrites au chapitre 2, à savoir pas plus de deux défauts à la fois. De plus, les amers 13 et 31 (de la figure 5.3) dans la carte ont été modifiés pour simuler un biais de positionnement de 5 m et 2,6 m respectivement.

La figure 5.10 montre les erreurs de localisation des véhicules. La différence entre le mode collaboratif et le mode isolé est très faible au départ, dans la période où les défauts d'observation ont été injectés, car les deux méthodes rejettent les observations erronées.

Du point de vue de la correction de la carte, l'amer erroné 13, observé dès le départ

(a) Estimation de l'amer 13 par le véhicule vert.

(b) Estimation de l'amer 31 par le véhicule orange.

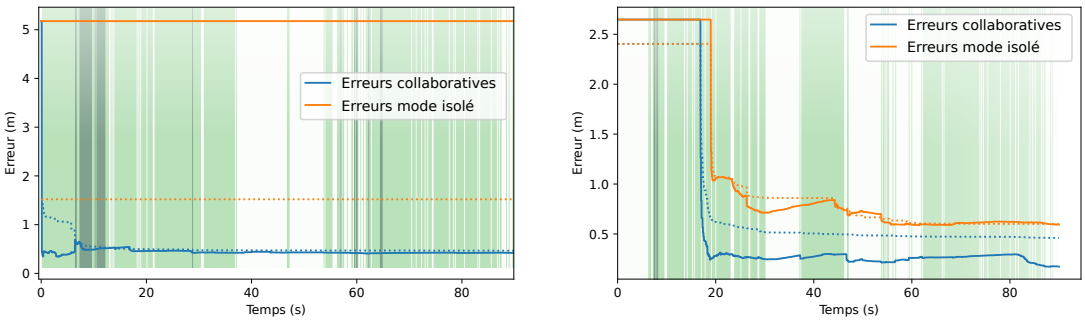
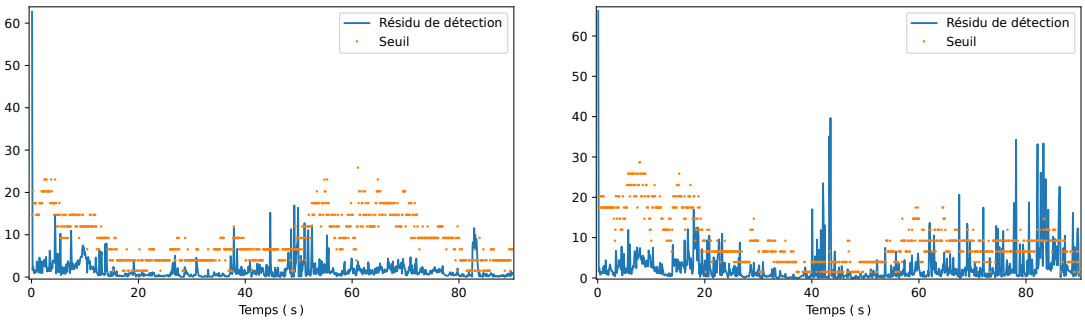


FIGURE 5.11. – Estimations des amers erronés (en traits pleins), ainsi que leurs incertitudes (en pointillés).

(a) Véhicule Bleu.

(b) Véhicule Vert.



(c) Véhicule Orange.

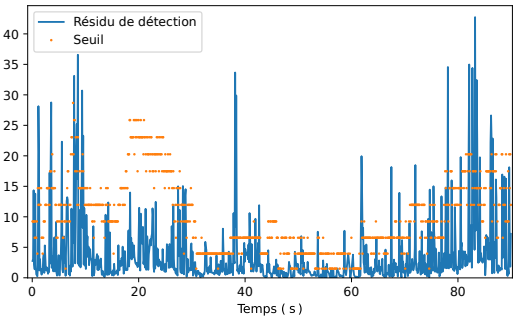


FIGURE 5.12. – Résidus de détection des trois véhicules lors de l'injection de défauts.

Méthodes \ Véhicule	Bleu	Vert	Orange	Tous
Collaborative	**98,7**	**94,3**	75,3	**89,4**
Isolée	97,9	77,3	**88,7**	88,0

TABLE 5.4. – Proportions (%) des erreurs inférieures à la borne d'incertitude à 99,7 %, avec des défauts d'injectés.

par les véhicules bleu et vert, est instantanément corrigé en collaboratif grâce au FDIR. La figure 5.11a montre l'estimation de cet amer par le véhicule vert avec les deux modes. En mode isolé, l'observation correspondante à l'amer 13 est simplement rejetée par les deux véhicules. Ceci n'affecte pas la précision de localisation des véhicules grâce aux nombreuses observations présentes dans le système. Cependant, le positionnement de l'amer ne peut pas être corrigé.

Le comportement du deuxième amer erroné, 31, est montré à la figure 5.11b comme estimé par le véhicule orange. Cet amer a été corrigé par le véhicule sans passer par l'étape de détection de défaut. Le véhicule orange en mode isolé arrive aussi à le corriger. La correction de l'amer n'est pas due à la méthode de FDIR, mais simplement au fait que le biais injecté n'était pas assez important vis-à-vis de la covariance pour déclencher la correction de la carte via le FDIR. L'amer est donc corrigé naturellement. Cependant, la correction en mode collaboratif est plus importante et permet d'obtenir une meilleure localisation de l'amer, car plus d'observations sont utilisées pour cette correction. De plus, le véhicule orange est le seul à le corriger en mode isolé. Le véhicule vert n'y arrive pas car cela dépend des observations. Grâce à la collaboration, le véhicule vert profite de la correction par le véhicule orange.

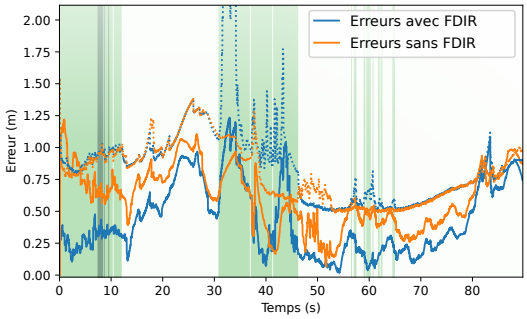
Ces deux corrections permettent aux véhicules en collaboration de mieux corriger leurs cartes. Ceci est montré au niveau de la table 5.3 où l'amélioration en termes d'erreurs moyennes peut être constatée. Pour le véhicule orange, on peut constater une réduction de l'erreur en mode isolé par rapport à la table 5.1. Ceci est dû à une observation de mauvaise qualité non détectée en mode isolé après injection des défauts vers $t = 55$ s. En mode collaboratif, après injection de défauts, cet amer est détecté et entraine une petite dérive.

La figure 5.12 montre les résidus utilisés pour la détection des défauts pour les trois véhicules. Des défauts sont détectés tout au long de l'exécution et ne se limitent pas aux défauts injectés. En effet, un nombre important d'observations présente des problèmes, à cause du processus d'extraction d'observation qui n'est pas suffisamment robuste aux bruits présents dans les nuages de points. On observe ainsi que certains des défauts d'observation injectés au début (entre 2 et 15 s) ne sont pas particulièrement importants (en termes d'intensité) par rapport aux défauts naturels présents tout le long de la trajectoire.

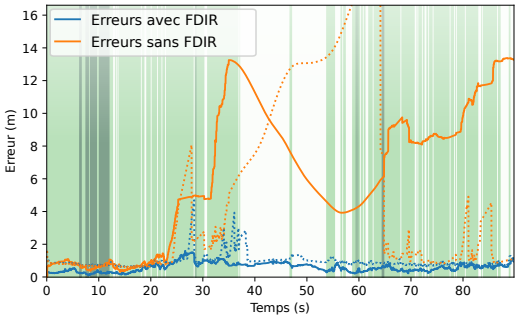
En termes de consistance, la table 5.4 montre que les erreurs sont mieux bornées en collaboratif, mais on observe tout de même une baisse de la consistance par rapport à l'absence d'injection de défauts (table 5.2). Le véhicule orange est ici moins consistant en mode collaboratif qu'en mode isolé à cause des observations aux alentours de $t = 55$ s comme expliqué ci-dessus. Cependant, on observe une amélioration pour les autres véhicules, notamment avec le véhicule vert qui profite largement de la collaboration pour améliorer sa consistance.

La figure 5.13 montre une comparaison des erreurs des véhicules avec et sans détection de défauts, en mode collaboratif. Au début de la trajectoire, on peut constater l'amélioration apportée par la détection de défaut. Cet effet ne se limite pas aux défauts injectés mais

(a) Véhicule Bleu.



(b) Véhicule Vert.



(c) Véhicule Orange.

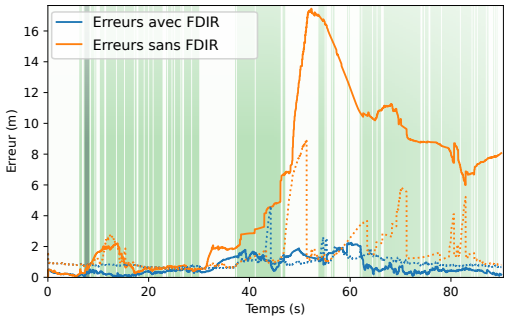


FIGURE 5.13. – Erreurs d'estimation des véhicules, avec leurs incertitudes à 99,97 % (en pointillés), lors de l'injection de défauts, en collaboration, mais avec ou sans méthode de tolérance aux défauts.

Méthodes \ Véhicule	Bleu	Vert	Orange	Tous
Avec FDIR	98,7	94,3	75,3	89,4
Sans FDIR	89,0	38,8	23,0	50,3

TABLE 5.5. – Proportions (%) des erreurs inférieures à la borne d'incertitude à 99,7 %, avec des défauts d'injectés, et avec ou sans la détection de défauts.

aussi à la zone problématique vers $t = 40$ s où une dérive limitée dans le temps se produit. Le système de FDIR est donc très utile dans les zones difficiles, où il y a peu d'observations à utiliser. Dès qu'un défaut est intégré au système de localisation, il se propage et provoque une dégradation de l'estimation de tous les véhicules en collaboration. Du point de vue des régions d'incertitude, on constate que l'absence d'une étape de FDIR rend l'estimation inconsistante. En effet, la proportion des erreurs inférieure à leurs bornes passe de 50,3 % sans module de FDIR, à 89 % avec (table 5.5). Seul le véhicule bleu conserve une consistance acceptable, bien que largement moins bonne qu'avec l'utilisation de FDIR. Le véhicule orange est celui qui présente le plus de problèmes avec une valeur de 23 % sans FDIR.

5.7. Conclusion

Ce chapitre a présenté une évaluation de la méthode complète sur des données réelles.

Les résultats expérimentaux (que ce soit en mode collaboratif ou en mode isolé) montrent que la localisation en milieu urbain à l'aide des bâtiments perçus par un lidar et sans aucune autre source de localisation absolue qu'une carte OpenStreetMap incertaine (hormis l'utilisation du GNSS pour l'initialisation) peut atteindre une précision moyenne de l'ordre de 50 cm. C'est un premier résultat intéressant.

Ces résultats montrent aussi que la collaboration améliore la localisation et la consistance des estimations des véhicules et de la carte, comme ceci avait été observé en simulation. L'avantage de la collaboration pour la correction de la carte a été mis en avant grâce au FDIR proposé. Le FDIR est donc un composant essentiel pour gérer les mesures erronées qui peuvent apparaitre régulièrement dans les environnements difficiles.

L'amélioration des résultats en mode collaboratif par rapport au mode isolé est moins spectaculaire que sur les résultats en simulation. En effet, avec les données réelles, le nombre d'observations en mode isolé est élevé (aux alentours de dix observations par étape de correction) par rapport au mode collaboratif (qui en ajoute aux alentours de deux ou trois en moyenne et cinq au maximum). Cela réduit donc l'apport des observations collaboratives comparées aux observations locales. De même, dans nos conditions expérimentales, les trois véhicules ont observé quasiment les mêmes amers, ce qui réduit l'effet de la collaboration sur l'estimation de la carte.

Sur la base de cette étude, on peut dire que les améliorations apportées par la collaboration dépendent du scénario, du nombre d'amers observés par chaque véhicule et du nombre d'observations à chaque instant.

Conclusion générale et perspectives

Conclusion

Les travaux présentés dans ce manuscrit ont porté sur la proposition et l'étude d'une méthode permettant à un ensemble de robots de se localiser avec une carte imprécise, de manière collaborative et décentralisée.

Nous avons présenté dans un premier temps une revue de la littérature sur ces méthodes, ce qui nous a permis de développer une méthode adaptée à notre problème. Celle-ci utilise le filtre de Schmidt-Kalman pour estimer la localisation des robots, tout en corrigeant les amers dans la carte. Ce filtre est particulièrement bien adapté à l'utilisation d'une architecture décentralisée.

Des contraintes de communication ont été prises en compte pour utiliser un minimum d'échanges entre les robots. Cela a conduit à une adaptation de la méthode en introduisant la fusion des états de chaque robot avec la moyenne de Kullback-Leibler, permettant de n'échanger qu'un seul message par étape de correction lorsque les robots sont en collaboration grâce à des observations communes via l'envoi de l'état complet du système collaboratif.

Pour assurer la précision et l'intégrité de l'estimation d'état, une étape de détection et d'isolation de défauts est nécessaire. La méthode que nous avons proposée a montré de bonnes performances en simulation et sur des données réelles.

Le système collaboratif permet ainsi de mettre à jour la carte et d'estimer la localisation des robots sans dépendre d'une communication permanente entre les robots et sans serveur central. Si des messages sont perdus, cela n'empêche pas les robots de continuer à pouvoir se localiser. Chaque robot peut, en effet, estimer seul sa localisation et la collaboration permet de l'améliorer quand elle est disponible. La collaboration permet aussi de rendre l'isolation de défauts plus fiable, et elle permet de détecter des défauts de carte qui sont impossibles à déterminer sans elle. Cela conduit à une meilleure estimation globale, à la fois de la carte qui s'en trouve corrigée, mais aussi des robots qui peuvent utiliser des observations supplémentaires pour se localiser.

L'utilisation de la KLA pour réaliser la fusion des estimations provenant de chaque robot fait que certains « perdent » par moments de l'information, mais ceci se fait alors au profit des autres robots. Cela est quand même bénéfique quand on regarde le groupe de robots dans son ensemble, même si le robot ayant partagé cette information peut être moins bien localisé que s'il n'avait pas collaboré du tout. En effet, la KLA se comporte comme une moyenne pondérée entre les états à fusionner et lorsqu'un élément est mal localisé (un amer ayant une mauvaise position initiale par exemple), un robot qui aurait été en mesure de corriger cet élément peut voir son estimation être dégradée par cet

élément mal localisé en provenance d'un autre.

La méthode a été évaluée sur des données réelles, collectées grâce aux véhicules expérimentaux du laboratoire, équipés de lidars multi-couches. Pour permettre l'utilisation de ces capteurs, une méthode permettant d'extraire des observations de poses à partir de nuages de points a été proposée. Les observations correspondent aux milieux des façades des bâtiments, avec l'orientation choisie comme étant la normale de la façade. Cette méthode combine l'utilisation de segmentation sémantique et géométrique, avec des techniques d'alignement de nuages de points avec la carte, ainsi qu'une méthode pour obtenir le milieu de la façade par traitement analytique. La méthode proposée tâche de ne pas créer de dépendance entre les observations générées et l'estimation du filtre. Cependant, certaines limitations ont été relevées, notamment sur l'impossibilité d'obtenir des observations si le filtre diverge ou si la façade n'est pas présente dans la carte.

Nos résultats montrent que la méthode présentée améliore la localisation des robots et des amers grâce à la collaboration, que ce soit en simulation ou sur des données réelles. L'amélioration apparait surtout dans le cas où le nombre d'observations en mode isolé est petit par rapport au nombre d'observations collaboratives. Dans le cas contraire, l'impact des observations collaboratives sur la solution globale est réduit. Cette collaboration reste cependant nécessaire pour pouvoir détecter des défauts cartes. L'avantage de la méthode collaborative apparait encore lorsqu'un robot se retrouve dans une situation moins avantageuse par rapport aux autres. Par exemple, dans le cas d'une carte contenant un amer parfait, la collaboration permet au robot qui n'observe pas cet amer de corriger sa carte.

Du point de vue de la consistance, la collaboration montre ses avantages tout en étant le moins pessimiste possible. C'est un résultat intéressant qui va s'étendre à l'étude de l'intégrité et de la disponibilité des systèmes. De ce point de vue, l'absence de l'étape FDIR ne dégrade pas seulement la précision mais aussi la consistance.

Pour finir, sur les données réelles, nous atteignons une précision de localisation de l'ordre de 50 cm, ce qui montre que la localisation avec des bâtiments est une approche assez prometteuse.

Perspectives

La méthode proposée dans ce manuscrit est à l'état de preuve de concept. Le but était de montrer que la méthode fonctionne et qu'elle répond au problème. Cependant, des optimisations peuvent être faites facilement, afin d'accélérer le processus, encore loin du fonctionnement temps réel utilisable en conditions réelles ou encore pour robustifier certains aspects.

Modification de l'état du SKF pour n'utiliser que les amers utiles

Un des avantages du SKF est qu'il est plus rapide qu'un filtre de Kalman travaillant sur le même état. En effet, le SKF n'estime que la partie s, et donc tous les calculs réalisés dans la partie p sont évités. Seule l'inter-covariance P_{sp} est estimée. Ainsi, le SKF

est plus rapide lorsque la partie p est suffisamment grande. Cependant, dans notre cas, la partie p contient uniquement les robots ne collaborant pas. Avec peu de robots comme dans les travaux réalisés, elle reste petite par rapport à la partie s contenant l'intégralité de la carte. Cela fait qu'avec les manipulations nécessaires sur les vecteurs et les matrices, le SKF est au final plus long qu'un EKF.

Une première optimisation serait donc de ne garder que les amers observés dans la partie s et ne pas estimer les autres. On conserve ainsi les propriétés recherchées, avec $H_p = \mathbb{0}$, tout en limitant la taille de la matrice P_{ss}. Le reste de la méthode reste strictement identique.

Fusion des états avec la KLA

La KLA dans notre cas est réalisée à chaque étape de correction du filtre, ce qui nécessite l'échange d'états assez volumineux. Cependant, cette étape n'est pas nécessaire à chaque instant et on peut la limiter à des intervalles de temps bien déterminés. Une optimisation possible serait que, lors d'étapes de correction successives avec les mêmes robots en collaboration, on n'échange les états et on ne fait la KLA que la première fois. Lors des étapes suivantes, on pourrait envoyer uniquement la contribution informationnelle de la prédiction et de la correction précédente concernant la localisation du robot. La KLA serait de nouveau réalisée au bout d'un certain temps, afin de remettre les pendules à l'heure. Cela permettrait de limiter les échanges de données lors des phases de collaboration, qui sont souvent étendues dans le temps.

Étude d'autres types d'observation

Dans cette thèse, nous avons choisi d'utiliser des observations sous la forme d'une pose qui représente le milieu d'une façade. Ce choix reposait sur l'objectif initial de pouvoir observer des façades partiellement avec toujours le même milieu, sans avoir besoin d'observer les coins la délimitant, ce qui peut arriver souvent dans les couloirs urbains. Cependant, nous en sommes venus à utiliser des observations de façades complètes car la méthode d'observation partielle n'a pas été concluante. Dans ce cas, on peut se poser la question de n'observer que les coins. En effet, l'observation complète d'une façade signifie qu'on observe les deux coins la délimitant. C'est une piste à explorer, mais qui ne résout pas le problème des canyons urbains.

Une autre possibilité serait de mesurer une distance entre le robot et le plan de la façade, permettant d'avoir une information de localisation dans le sens de la normale de la façade, mais aucune sur la direction perpendiculaire à la normale.

Ces changements d'observations nécessitent une nouvelle modélisation du système et une nouvelle approche pour extraire les observations des nuages de points.

Observation indépendante de la carte

La méthode proposée dans cette thèse nécessite l'utilisation de la carte pour générer les observations à travers une étape d'association. Cependant, cette étape limite la généralisation simple de la méthode pour ajouter des façades manquantes à la carte.

Pour se passer de la carte, il faut que l'observation se décrive elle-même afin de pouvoir lui associer son identifiant sans utiliser de proximité avec la carte. Les méthodes utilisant les descripteurs, telles que (Dube et al. 2017), seraient un bon point de départ. Il faut que les descripteurs soient capables de différencier des bâtiments ayant une forte ressemblance comme ça peut être le cas dans un milieu urbain. Les descripteurs doivent aussi être robustes aux changements d'angle de vue du robot.

Filtrage et optimisation

Nous avons choisi d'utiliser une méthode basée filtrage afin d'inscrire nos travaux dans une recherche d'intégrité externe de localisation, nécessitant une estimation de l'erreur de localisation. La détection et l'isolation de défauts a aussi beaucoup bénéficié du filtrage. Cependant, une approche utilisant l'optimisation est aussi possible. Les méthodes de *bundle adjustment* et de *pose graph* ont montré leurs preuves et sont très utilisées dans le SLAM.

Pour bénéficier des avantages des deux méthodes, l'optimisation peut concerner qu'une partie du problème, la carte. Ainsi, le rôle du filtre peut se limiter à une estimation de la localisation des robots. La suppression de la carte du vecteur d'état peut résoudre plusieurs problèmes liés aux calculs des inter-covariances. C'est une partie exploratoire qui nécessiterait une étude détaillée.

Centralisation de la carte

L'utilisation d'une approche centralisée au niveau de la carte peut être envisagée. Le besoin d'avoir une carte globale optimisée à chaque instant n'est pas nécessaire. Par conséquent, une unité centrale pourrait se charger de cette optimisation. Chaque robot pourrait ainsi se localiser en estimant une carte locale et en coopérant avec les autres robots. La carte globale serait mise à jour par le serveur central périodiquement, ou bien lorsque celui-ci serait accessible. Les robots pourraient récupérer de nouvelles informations cartographiques plus à jour lorsqu'ils reviennent dans la zone d'évolution par exemple.

Retrait et ajout d'éléments dans la carte

Dans cette thèse, nous n'avons pas eu le temps de gérer le retrait et l'ajout d'éléments de la carte. Cette étape était un objectif au début de la thèse.

Concernant le premier point, les travaux de (Berrio et al. 2020) sont prometteurs dans cette direction. Pour ce faire, il s'agit de quantifier la probabilité d'existence de chaque amer, en fonction d'un modèle d'observabilité et d'un modèle de capteur. Cela signifie qu'en fonction des obstructions dans l'environnement et en fonction des points morts du capteur, il est possible de déterminer si un amer doit être observé. Sa probabilité d'existence est ensuite mise à jour. Si cette probabilité tombe en dessous d'un certain seuil, l'amer est retiré de la carte.

Concernant l'ajout de nouveaux amers, cela nécessite une analyse beaucoup plus fine du nuage de points. Pour cela, les réseaux de neurones pourraient être utiles. Cependant, la complexité de cette tâche est liée au type de *features* utilisées.

Affinage de la carte

Un des objectifs initiaux de la thèse était la mise à jour des grilles de distributions normales (ND) représentant les façades, de façon à raffiner leurs descriptions et ainsi d'améliorer la précision de la localisation. Nous n'avons pas eu le temps de nous y intéresser. L'idée serait d'utiliser les nuages de points pour mettre à jour les ND dans la carte afin d'avoir une représentation plus fine de la carte. Cet objectif nous a guidé dans certains choix comme la NDT. L'utilisation de grilles ND couplées à des grilles d'occupation, comme dans (Saarinen et al. 2013) serait un bon point de départ pour faire une étude dans cette direction.

Approches basées apprentissage pour la définition des seuils

Les seuils utilisés lors des étapes de détection et d'isolation des défauts ont été fixés d'une façon empirique. Cependant, les seuils affectent directement l'intégrité de localisation et doivent être fixés d'une façon optimale. Pour cela, des méthodes basées sur l'apprentissage pourraient être étudiées pour définir des seuils adaptatifs qui conduisent à la meilleure prise de décision.

A. Décomposition des inter-covariances pour les maintenir à jour

A.1. Introduction

Le SKF permet de n'estimer que la partie s de l'état. Cependant, même si la partie p n'est effectivement pas utilisée, il n'en est pas de même pour les inter-covariances entre la partie s et la partie p qui sont mise à jour lors de l'étape de correction à l'équation (2.84). Nous allons voir dans cette annexe comment répartir ces inter-covariances dans une architecture décentralisée où les robots ne peuvent pas communiquer en permanence.

A.2. Types d'inter-covariances

Tout d'abord, les différents types d'inter-covariances peuvent être les suivants. Les inter-covariances *robot-robot* sont prédites par l'étape du même nom via l'équation (2.43) et les deux robots sont impliqués via leurs jacobiennes $F_{i,k}$ et $F_{j,k}$. Il faudrait donc qu'ils s'échangent leurs F à chaque étape de prédiction, ce qui engendrerait des communications supplémentaires obligatoires que l'on souhaite éviter. Pour l'étape de correction, soit les deux robots sont dans s et alors il n'y a pas de problème puisqu'ils peuvent s'échanger des informations, soit un des deux est dans p et la correction est effectuée de chaque côté de manière différente (avec des observations différentes). Il faudrait donc que les deux robots communiquent, mais on veut l'éviter.

Au niveau des inter-covariances *robot-amer*, elles sont prédites par le robot dans l'étape de prédiction, et ne nécessitent rien venant de l'amer (la jacobienne F d'un amer est toujours l'identité I). Lors de l'étape de correction, on a choisi de garder tous les amers dans s donc la correction de l'inter-covariance est aussi possible tant que le robot est dans s. À noter que si nous n'avions pas choisi de garder toute la carte dans s, cela aurait complexifié le problème ici. Si le robot est dans p, alors on ne peut pas communiquer avec lui et la correction est effectuée de son côté.

Enfin, pour les inter-covariances amer-amer, il n'y a pas de problème sur les étapes de prédiction puisqu'ils sont statiques. Au niveau des étapes de correction, la carte étant entièrement dans s, il n'y a pas de problème non plus.

Il faut donc un moyen de maintenir l'estimation des inter-covariances robot-robot et robot-amer en dehors des phases de collaboration. Pour les inter-covariances robot-amer, nous avions choisi de laisser le robot s'en occuper seul, puis de les communiquer aux autres

robots lors de la collaboration. Pour les inter-covariances robot-robot, nous utilisions la décomposition présentée dans (Luft et al. 2016).

A.3. Décomposition des inter-covariances robot-robot

Afin d'estimer les inter-covariances entre les robots entre deux moments de collaboration, v_i et v_j, les auteurs dans (Luft et al. 2016) utilisent une décomposition en un produit de deux matrices. On écrira i pour le robot v_i et j pour le robot v_j.

$$P_{ij} = \sigma_{ij} \left(\sigma_{ji}\right)^T. \tag{A.1}$$

Ainsi, le robot v_i gardera la partie σ_{ij} et le robot v_j, σ_{ji}. Notons que σ_{ij} et σ_{ji} peuvent prendre n'importe quelle valeur, tant qu'elles satisfont l'équation (A.1). En particulier, après une étape de correction, un des deux robots gardera la covariance complète $\sigma_{ij,k|k} = P_{ij,k|k}$ tandis que l'autre verra sa part de covariance devenir l'identité $\sigma_{ji,k|k} = I$.

L'étape de prédiction des robots i et j peut donc être écrite ainsi :

$$\sigma_{ij,k+1|k} = F_{i,k}\sigma_{ij,k|k}, \tag{A.2}$$
$$\sigma_{ji,k+1|k} = F_{j,k}\sigma_{ji,k|k}, \tag{A.3}$$

avec $F_{i,k}$ (respectivement $F_{j,k}$) la matrice Jacobienne du modèle d'évolution du robot i (respectivement j) tel qu'il a été défini à l'équation (2.27).

Ensuite, lorsque les deux robots communiquent, ils peuvent échanger leurs pièces et reconstituer la covariance croisée complète avant d'effectuer la mise à jour :

$$P_{ij,k+1|k} = \sigma_{ij,k+1|k} \left(\sigma_{ji,k+1|k}\right)^T, \tag{A.4}$$
$$= F_{i,k}\sigma_{ij,k|k} \left(\sigma_{ji,k|k}\right)^T F_{j,k}^T, \tag{A.5}$$
$$= F_{i,k}P_{ij,k|k}F_{j,k}^T. \tag{A.6}$$

Cette reconstruction est possible grâce à l'utilisation de produits à gauche uniquement (équations (A.2) et (A.3)) et est équivalente à l'étape de prédiction de la covariance croisée donnée dans l'équation (2.43). Les robots sont maintenant prêts pour l'étape de mise à jour.

Dans l'étape de correction, lorsque des robots collaborent, l'inter-covariance complète peut-être reconstituée avec l'équation (A.4). Cependant, ce n'est pas le cas pour les inter-covariances entre des robots ne collaborant pas : un robot dans s (noté s_i) et un robot dans p (noté p_j). En utilisant l'équation (2.84), les termes de l'inter-covariance entre ces deux robots peuvent être écrits ainsi :

$$\begin{aligned} P_{s_ip_j,k|k} = &(I - K_{s_i,k}H_{s_i,k})P_{s_ip_j,k|k-1} \\ &- \sum_{r=1,r\neq i}^{n} K_{s_i,k}H_{s_r,k}P_{s_rp_j,k|k-1}. \end{aligned} \tag{A.7}$$

$P_{s_i p_j,k|k-1} = \sigma_{s_i p_j,k|k-1} \left(\sigma_{p_j s_i,k|k-1}\right)^T$ et $P_{s_r p_j,k|k-1} = \sigma_{s_r p_j,k|k-1} \left(\sigma_{p_j s_r,k|k-1}\right)^T$ ne peuvent pas être reconstitués parce que les robots dans s ne communiquent pas avec p_j. De même, l'équation (A.7) ne peut pas être écrite sous la forme d'une matrice multipliée par by $P_{s_i p_j,k|k-1}$, une forme autorisant la décomposition de $P_{s_i p_j,k|k}$.

Ainsi, pour garder trace des mises à jour des inter-covariances, et être capable de les reconstituer à la prochaine collaboration, une approximation est nécessaire, telle que proposée dans Luft et al. 2016 :

$$P_{s_i p_j,k|k} \approx P_{s_i s_i,k|k}(P_{s_i s_i,k|k-1})^{-1} P_{s_i p_j,k|k-1}, \tag{A.8}$$

où la partie conservée par s_i est :

$$\sigma_{s_i p_j,k|k} \approx P_{s_i s_i,k|k}(P_{s_i s_i,k|k-1})^{-1} \sigma_{s_i p_j,k|k-1}. \tag{A.9}$$

Notons que cette approximation est exacte si les éléments de s sont totalement décorrélés, ou fortement corrélés, comme démontré dans Luft et al. 2016.

Quand deux robots sont dans s, les inter-covariances peuvent être reconstituées et l'étape de correction peut être réalisée. Ensuite, un des σ_{ij} prend la valeur de $P_{s_i p_j}$ corrigé, et l'autre est défini comme étant l'identité.

Les inter-covariances entre les robots et la carte sont gérés d'une autre façon. Chaque robot garde sa carte locale et ses inter-covariances avec chaque amer, puis les corrige quand nécessaire. Ces inter-covariances ne sont pas décomposées et sont totalement transmises lors des collaborations. Ainsi, les robots peuvent reconstituer P_{ss}.

Afin de travailler sur les mêmes cartes, une fusion des m cartes provenant des m robots en collaboration est faite en utilisant la KLA Giorgio Battistelli et Luigi Chisci 2016 :

$$\bar{P}^{-1}_{L,k|k-1} = \frac{1}{m} \sum_{i=1}^{m} (P^i_{L,k|k-1})^{-1}, \tag{A.10}$$

$$\bar{X}_{L,k|k-1} = \bar{P}_{L,k|k-1} \frac{1}{m} \sum_{i=1}^{m} (P^i_{L,k|k-1})^{-1} X^i_{L,k|k-1}, \tag{A.11}$$

avec P^i_L et X^i_L la covariance et l'état de la partie correspondant à la carte estimée par le robot v_i.

A.4. Limites de la méthode dans notre problème

Cette méthode de décomposition fonctionne parfaitement tant qu'il n'y a que deux acteurs. Pour les inter-covariances entre deux robots, elle est donc adaptée. Le problème vient de la gestion des inter-covariances robot-amer.

Chaque robot ayant sa propre carte, et cette carte étant mise à jour, il faudrait pouvoir répartir les inter-covariances entre les robots et les amers entre tous les robots, donc plus que deux acteurs. Nous avions choisi de transférer les inter-covariances robot-amer du robot concerné vers les autres, mais en dehors du fait qu'il nous manque une partie de

l'estimation de l'inter-covariance, cette opération n'est pas mathématiquement acceptable pour conserver les propriétés de la covariance (notamment sa définition positive).

Un autre problème est les multiples versions des estimations de carte qui sont fusionnées avec une KLA uniquement sur cette partie. Cela détériore la cohérence de la matrice de covariance complète, et en particulier, la cohérence des inter-covariances robot-amer.

En conclusion, cette décomposition serait possible uniquement si les amers de la carte possédaient leurs propres capacités de communication et de calcul (on peut imaginer une carte centralisée). Il serait alors possible de décomposer chaque inter-covariance entre deux acteurs, et l'amer se chargerait de mettre à jour son σ. Or, quitte à avoir une carte centralisée, autant centraliser plus d'éléments. Nous perdrions aussi notre indépendance au réseau, car il faudrait absolument pouvoir communiquer avec l'amer pour reconstituer les inter-covariances robot-amer, et procéder à l'étape correction.

B. Observations partielles de façade avec la NDT

L'approche initiale pour la génération d'observations est d'utiliser la NDT pour aligner le groupe de points correspondant à la façade avec la carte, en utilisant s.

Dans l'objectif de pouvoir utiliser des observations partielles de façades, c'est-à-dire sans observer les deux coins simultanément, il faut pouvoir trouver la transformation à appliquer au groupe de points pour s'aligner parfaitement sur la façade. Cela est fait avec la NDT en utilisant la représentation sous forme de grille de ND de la façade (section 4.5.3). Cette NDT propre à chaque façade se fait en plus de la NDT globale, cette dernière permettant de faire un premier alignement.

B.1. Repères utilisés

Étant donné le grand nombre de repères utilisés, nous faisons un point dans la suite (figure B.1).

1. Le nuage de point lidar est exprimé dans le repère du lidar L_i, qui peut varier en fonction des jeux de données utilisés. Ce repère est exprimé par rapport à la pose du véhicule.
2. La pose du véhicule définit un repère propre au véhicule, V_i, à partir duquel sont définis les repères des capteurs. Les transformations entre les capteurs et la base du véhicule sont fixes.

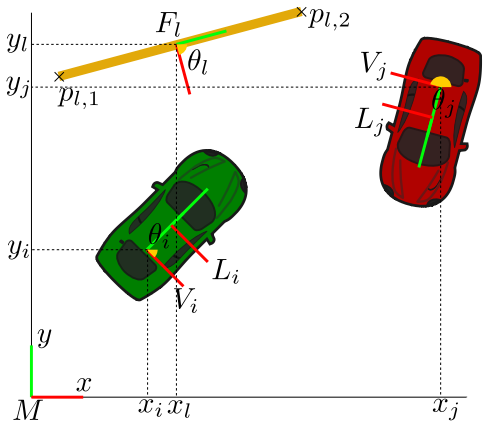


FIGURE B.1. – Repères.

3. Le repère qui sera considéré comme absolu est le repère de la carte M (qui est considéré confondu avec le repère ENU local). Les poses des véhicules ainsi que celles des façades sont exprimées dans ce repère.
4. Chaque façade possède une grille de distributions normales, localisées par rapport au centre de la façade, définissant un repère F_k. La pose de la façade est l'origine du repère F_k dans M.
5. Dans le groupe de points k associé à la façade F_k, on peut définir un repère O_k , qui, lorsque le nuage de point est correctement aligné avec la façade, se superpose au repère de la façade F_k. L'origine de O_k définit une pose dans F_k, qui sera l'observation utilisée par le filtre de Schmidt-Kalman.

Pour résumer, la figure B.2 représente un graphe des dépendances entre les repères.

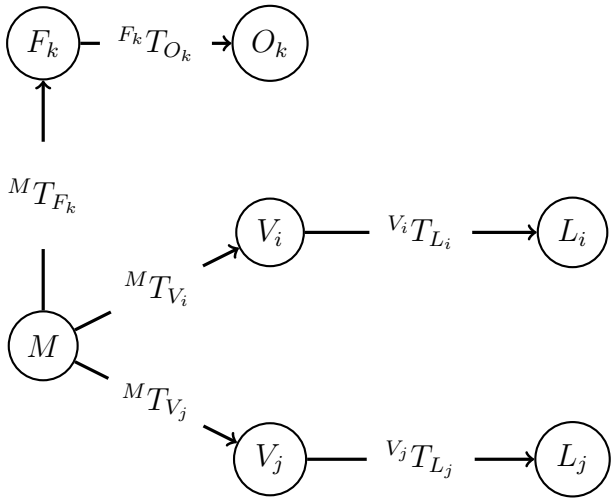


FIGURE B.2. – Dépendances entre les repères

B.2. Alignement via NDT

Le nuage de points initial est dans le repère du lidar L_i, et pour pouvoir faire l'alignement avec la NDT, il faut se placer dans le repère de la façade F_k. La transformation du cluster k dans le repère de la façade nécessite d'utiliser les poses estimées du véhicule et de la façade.

La transformation de la base du véhicule V_i vers la carte M est donnée par :

$$ {}^{M}T_{V_i} = \begin{bmatrix} \cos\theta & -\sin\theta & x \\ \sin\theta & \cos\theta & y \\ 0 & 0 & 1 \end{bmatrix} \tag{B.1} $$

où x, y et θ représentent la pose du véhicule dans le repère de la carte.

La transformation de la façade F_k vers la carte M est :

$$ {}^{M}T_{F_k} = \begin{bmatrix} \cos\theta_{F_k} & -\sin\theta_{F_k} & x_{F_k} \\ \sin\theta_{F_k} & \cos\theta_{F_k} & y_{F_k} \\ 0 & 0 & 1 \end{bmatrix} \tag{B.2} $$

où x_{F_k}, y_{F_k} et θ_{F_k} représentent la pose du point milieu de la façade dans le repère de la carte R_M.

Les transformations chaînées sont donc :

$$ {}^{M}T_{L_i} = {}^{M}T_{V_i}.{}^{V_i}T_{L_i} \tag{B.3} $$

$$ {}^{F_k}T_{L_i} = {}^{M}T^{-1}_{F_k}.{}^{M}T_{V_i}.{}^{V_i}T_{L_i} \tag{B.4} $$

La position du point p, exprimée en premier lieu dans le repère du lidar L_i, dans le repère de la façade F_k est donc :

$$ {}^{F_k}p = {}^{F_k}T_{L_i}.{}^{L_i}p \tag{B.5} $$

B.3. Traitement du résultat de la NDT

Lorsque la NDT a fini l'alignement, la transformation appliquée au nuage de points est accessible, notée ${}^{F_k}T_{A_k}$. Cependant, la pose issue de cette transformation n'est pas l'observation que nous cherchons. Pour obtenir l'observation utilisée dans le SKF, il faut construire la pose la façade comme elle serait dans le groupe de points avant alignement. Pour ceci, il suffit d'appliquer la transformation inverse à celle issue de l'alignement, comme illustré à la figure B.3.

Pour trouver la pose de l'observation dans le repère du véhicule, il faut donc faire :

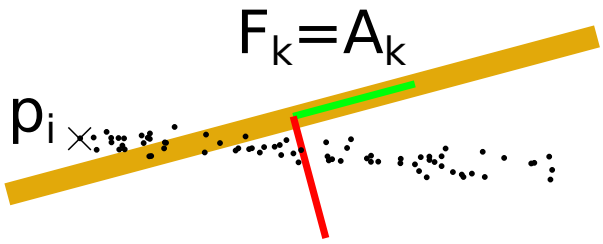
$$ {}^{V_i}T_{O_k} = {}^{M}T^{-1}_{V_i}.{}^{M}T_{F_k}.{}^{F_k}T_{O_k} \tag{B.6} $$

$$ = {}^{M}T^{-1}_{V_i}.{}^{M}T_{F_k}.{}^{F_k}T^{-1}_{A_k} \tag{B.7} $$

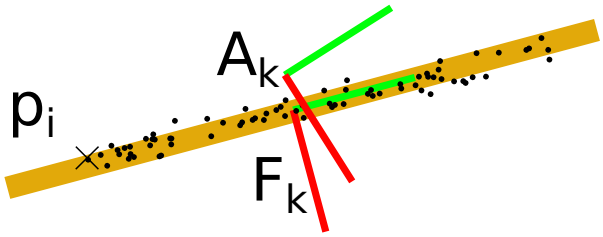
et la pose sera donc définie par l'origine du repère O_k dans le repère du véhicule V_i, c'est-à-dire la pose issue de la transformation ${}^{V_i}T_{O_k}$.

B.4. Conclusion sur cette méthode

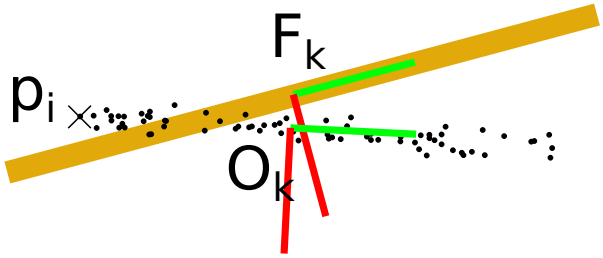
Les résultats des premiers essais ont montré une faible robustesse de l'alignement, avec de nombreux mauvais alignements. Nous avons pu limiter cela en ne prenant que les groupes de points observants plus de la moitié de la façade. Cette approche ne peut pas être appliquée à cause de la dépendance entre l'observation et l'estimation de l'état, que ce soit par la pose du véhicule ou la pose de la façade. Cette dépendance rejette une hypothèse de base du SKF utilisé, qui est l'indépendance conditionnelle des observations avec l'estimation de l'état.



(a) Initialisation de la NDT après la transformation du groupe de points dans le repère de la façade F_k.



(b) Alignement du groupe de points avec la façade. Le repère du groupe de points A_k a été déplacé.



(c) Pour obtenir l'observation, il faut donc replacer le centre de la façade F_k à sa place dans le groupe de point, mais avant l'alignement. Cela correspond à appliquer la transformation inverse à l'alignement sur F_k, ce qui donne le repère O_k.

FIGURE B.3. – Étapes pour la NDT locale à chaque façade.

Table des figures

Liste des tableaux

Bibliographie

Agamennoni, Gabriel, Simone Fontana, Roland Y. Siegwart et Domenico G. Sorrenti (oct. 2016). « Point Clouds Registration with Probabilistic Data Association ». In : *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, p. 4092-4098.

Akai, Naoki, Luis Yoichi Morales, Takuma Yamaguchi, Eijiro Takeuchi, Yuki Yoshihara, Hiroyuki Okuda, Tatsuya Suzuki et Yoshiki Ninomiya (oct. 2017). « Autonomous Driving Based on Accurate Localization Using Multilayer LiDAR and Dead Reckoning ». In : *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. Yokohama : IEEE, p. 1-6.

Al Hage, Joelle (2016). « Fusion de données tolérante aux défaillances : Application à la surveillance de l'intégrité d'un système de localisation ». Thèse de doct. Université de Lille. 202 p.

Al Hage, Joelle, Maan E. El Najjar et Denis Pomorski (1er sept. 2017). « Multi-Sensor Fusion Approach with Fault Detection and Exclusion Based on the Kullback–Leibler Divergence : Application on Collaborative Multi-Robot System ». In : *Information Fusion* 37, p. 61-76.

Allig, Christoph et Gerd Wanielik (juin 2019). « Alignment of Perception Information for Cooperative Perception ». In : *2019 IEEE Intelligent Vehicles Symposium (IV)*, p. 1849-1854.

Amemiya, Takeshi (1985). *Advanced Econometrics*. Harvard University Press. 540 p.

Avižienis, Algirdas (14 nov. 1967). « Design of Fault-Tolerant Computers ». In : *Proceedings of the November 14-16, 1967, Fall Joint Computer Conference*. AFIPS '67 (Fall). New York, NY, USA : Association for Computing Machinery, p. 733-743.

Bailey, T. et H. Durrant-Whyte (sept. 2006). « Simultaneous Localization and Mapping (SLAM) : Part II ». In : *IEEE Robotics Automation Magazine* 13.3, p. 108-117.

Bailey, Tim, Juan Nieto, Jose Guivant, Michael Stevens et Eduardo Nebot (oct. 2006). « Consistency of the EKF-SLAM Algorithm ». In : *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Beijing : IEEE, p. 3562-3568.

Bar-Shalom, Y. (juill. 2002). « Update with Out-of-Sequence Measurements in Tracking : Exact Solution ». In : *IEEE Transactions on Aerospace and Electronic Systems* 38.3, p. 769-777.

Battistelli, Giorgio et Luigi Chisci (mars 2014). « Kullback–Leibler Average, Consensus on Probability Densities, and Distributed State Estimation with Guaranteed Stability ». In : *Automatica* 50.3, p. 707-718.

Battistelli, Giorgio et Luigi Chisci (juin 2016). « Stability of Consensus Extended Kalman Filter for Distributed State Estimation ». In : *Automatica* 68, p. 169-178.

Behley, Jens, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss et Juergen Gall (16 août 2019). *SemanticKITTI : A Dataset for Semantic Scene Understanding of LiDAR Sequences*. preprint.

Berrio, Julie Stephany, Stewart Worrall, Mao Shan et Eduardo Nebot (1er août 2020). « Long-Term Map Maintenance Pipeline for Autonomous Vehicles ». In : *arXiv e-prints* 2008, arXiv :2008.12449.

Besl, Paul J. et Neil D. McKay (30 avr. 1992). « Method for Registration of 3-D Shapes ». In : *Sensor Fusion IV : Control Paradigms and Data Structures*. T. 1611. International Society for Optics and Photonics, p. 586-606.

Biber, P. et W. Strasser (oct. 2003). « The Normal Distributions Transform : A New Approach to Laser Scan Matching ». In : *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*. T. 3, 2743-2748 vol.3.

Bourgault, Frederic et Hugh F Durrant-Whyte (juill. 2004). « Communication in General Decentralized Filters and the Coordinated Search Strategy ». In : *Proc. of The 7th Int. Conf. on Information Fusion*, p. 8.

Bresson, G., Z. Alsayed, L. Yu et S. Glaser (sept. 2017). « Simultaneous Localization and Mapping : A Survey of Current Trends in Autonomous Driving ». In : *IEEE Transactions on Intelligent Vehicles* 2.3, p. 194-220.

Bresson, Guillaume, Romuald Aufrère et Roland Chapuis (1er déc. 2015). « A General Consistent Decentralized Simultaneous Localization And Mapping Solution ». In : *Robotics and Autonomous Systems* 74, p. 128-147.

Brown, R. Grover (1992). « A Baseline GPS RAIM Scheme and a Note on the Equivalence of Three RAIM Methods ». In : *NAVIGATION* 39.3, p. 301-316.

Cadena, Cesar, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid et John J. Leonard (déc. 2016). « Past, Present, and Future of Simultaneous Localization and Mapping : Toward the Robust-Perception Age ». In : *IEEE Transactions on Robotics* 32.6, p. 1309-1332.

Calamai, Paul H. et Jorge J. Moré (1er sept. 1987). « Projected Gradient Methods for Linearly Constrained Problems ». In : *Mathematical Programming* 39.1, p. 93-116.

Campos, Carlos, Richard Elvira, Juan J. Gómez Rodríguez, José M. M. Montiel et Juan D. Tardós (déc. 2021). « ORB-SLAM3 : An Accurate Open-Source Library for

Visual, Visual–Inertial, and Multimap SLAM ». In : *IEEE Transactions on Robotics* 37.6, p. 1874-1890.

Carrillo-Arce, Luis C., Esha D. Nerurkar, José L. Gordillo et Stergios I. Roumeliotis (nov. 2013). « Decentralized Multi-Robot Cooperative Localization Using Covariance Intersection ». In : *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, p. 1412-1417.

Chen, Chang, Hua Zhu, Menggang Li et Shaoze You (sept. 2018). « A Review of Visual-Inertial Simultaneous Localization and Mapping from Filtering-Based and Optimization-Based Perspectives ». In : *Robotics* 7.3 (3), p. 45.

Chen, Yang et Gérard Medioni (1er avr. 1992). « Object Modelling by Registration of Multiple Range Images ». In : *Image and Vision Computing*. Range Image Understanding 10.3, p. 145-155.

Cho, Younghun, Giseop Kim, Sangmin Lee et Jee-Hwan Ryu (avr. 2022). « OpenStreetMap-Based LiDAR Global Localization in Urban Environment Without a Prior LiDAR Map ». In : *IEEE Robotics and Automation Letters* 7.2, p. 4999-5006.

Cieslewski, Titus, Siddharth Choudhary et Davide Scaramuzza (16 oct. 2017). « Data-Efficient Decentralized Visual SLAM ». In : *ICRA 2018*.

Das, Arun et Steven L. Waslander (oct. 2012). « Scan Registration with Multi-Scale k-Means Normal Distributions Transform ». In : *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, p. 2705-2710.

Dellenbach, Pierre, Jean-Emmanuel Deschaud, Bastien Jacquet et François Goulette (mai 2022). « CT-ICP : Real-time Elastic LiDAR Odometry with Loop Closure ». In : *2022 International Conference on Robotics and Automation (ICRA)*, p. 5580-5586.

Delobel, Laurent (8 mai 2018). « Agrégation d'information pour la localisation d'un robot mobile sur une carte imparfaite ». Thèse de doct. 253 p.

Ding, Steven X. (2013). *Model-Based Fault Diagnosis Techniques : Design Schemes, Algorithms and Tools*. Advances in Industrial Control. London : Springer.

Dong, Jing, Erik Nelson, Vadim Indelman, Nathan Michael et Frank Dellaert (mai 2015). « Distributed Real-Time Cooperative Localization and Mapping Using an Uncertainty-Aware Expectation Maximization Approach ». In : *2015 IEEE International Conference on Robotics and Automation (ICRA)*. Seattle, WA, USA : IEEE, p. 5807-5814.

Dube, Renaud, Daniel Dugas, Elena Stumm, Juan Nieto, Roland Siegwart et Cesar Cadena (mai 2017). « SegMatch : Segment Based Place Recognition in 3D Point Clouds ». In : *2017 IEEE International Conference on Robotics and Automation (ICRA)*. Singapore, Singapore : IEEE, p. 5266-5272.

Dubé, Renaud, Andrei Cramariuc, Daniel Dugas, Juan Nieto, Roland Siegwart et Cesar Cadena (26 juin 2018). « SegMap : 3D Segment Mapping Using Data-Driven Descriptors ». In : *Robotics : Science and Systems XIV*.

Dubois, Rodolphe, Alexandre Eudes, Julien Moras et Vincent Fremont (24 oct. 2020). « Dense Decentralized Multi-robot SLAM Based on Locally Consistent TSDF Submaps ». In : *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Las Vegas, NV, USA : IEEE, p. 4862-4869.

Dudek, G., M. Jenkin, E. Milios et D. Wilkes (juill. 1993). « A Taxonomy for Swarm Robots ». In : *Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '93)*. T. 1, 441-447 vol.1.

Durrant-Whyte, Hugh (27 sept. 2002). « Introduction to Decentralised Data Fusion ». In : p. 194.

Escourrou, Maxime, Joelle Al Hage et Philippe Bonnifait (août 2021). « NDT Localization with 2D Vector Maps and Filtered LiDAR Scans ». In : *European Conference on Mobile Robots (ECMR)*, p. 1-6.

Escourrou, Maxime, Joelle Al Hage et Philippe Bonnifait (juill. 2022). « Decentralized Collaborative Localization with Map Update Using Schmidt-Kalman Filter ». In : *25th International Conference on Information Fusion (FUSION)*, p. 1-8.

finances, Ministère de l'économie et des (2013). *Standard d'échange des objets du plan cadastral informatisé fondé sur la norme EDIGéO*.

Fischler, Martin A. et Robert C. Bolles (1er juin 1981). « Random Sample Consensus : A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography ». In : *Communications of the ACM* 24.6, p. 381-395.

Fong, Whye Kit, Rohit Mohan, Juana Valeria Hurtado, Lubing Zhou, Holger Caesar, Oscar Beijbom et Abhinav Valada (23 déc. 2021). *Panoptic nuScenes : A Large-Scale Benchmark for LiDAR Panoptic Segmentation and Tracking*. preprint.

Gallego, Guillermo, Tobi Delbrück, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew J. Davison, Jörg Conradt, Kostas Daniilidis et Davide Scaramuzza (jan. 2022). « Event-Based Vision : A Survey ». In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.1, p. 154-180.

Gao, L., G. Battistelli et L. Chisci (déc. 2020). « Random-Finite-Set-Based Distributed Multirobot SLAM ». In : *IEEE Transactions on Robotics* 36.6, p. 1758-1777.

Goelles, Thomas, Birgit Schlager et Stefan Muckenhuber (jan. 2020). « Fault Detection, Isolation, Identification and Recovery (FDIIR) Methods for Automotive Perception Sensors Including a Detailed Literature Survey for Lidar ». In : *Sensors* 20.13 (13), p. 3662.

Graham Miller, Olivier et Vaibhav Gandhi (1er jan. 2021). « A Survey of Modern Exogenous Fault Detection and Diagnosis Methods for Swarm Robotics ». In : *Journal of King Saud University - Engineering Sciences* 33.1, p. 43-53.

Guo, Hongliang, Jiankang Zhu et Yunping Chen (2022). « E-LOAM : LiDAR Odometry and Mapping with Expanded Local Structural Information ». In : *IEEE Transactions on Intelligent Vehicles*, p. 1-1.

Hamadi, Hussein, Benjamin Lussier, Isabelle Fantoni et Clovis Francis (1er oct. 2022). « Data Fusion Fault Tolerant Strategy for a Quadrotor UAV under Sensors and Software Faults ». In : *ISA Transactions* 129, p. 520-539.

Hou, Yuenan, Xinge Zhu, Yuexin Ma, Chen Change Loy et Yikang Li (juin 2022). « Point-to-Voxel Knowledge Distillation for LiDAR Semantic Segmentation ». In : *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. New Orleans, LA, USA : IEEE, p. 8469-8478.

Hough, Paul V. C. (18 déc. 1962). « Method and Means for Recognizing Complex Patterns ». Brev. amér. 3069654A. Paul V C Hough.

Howard, A., M.J. Matark et G.S. Sukhatme (sept. 2002). « Localization for Mobile Robot Teams Using Maximum Likelihood Estimation ». In : *IEEE/RSJ International Conference on Intelligent Robots and Systems*. T. 1, 434-439 vol.1.

Huang, Guoquan P., Anastasios I. Mourikis et Stergios I. Roumeliotis (mai 2009). « On the Complexity and Consistency of UKF-based SLAM ». In : *2009 IEEE International Conference on Robotics and Automation*, p. 4401-4408.

Javanmardi, Ehsan, Yanlei Gu, Mahdi Javanmardi et Shunsuke Kamijo (1er avr. 2019). « Autonomous Vehicle Self-Localization Based on Abstract Map and Multi-Channel LiDAR in Urban Area ». In : *IATSS Research* 43.1, p. 1-13.

Julier, S.J. et J.K. Uhlmann (mai 2001). « A Counter Example to the Theory of Simultaneous Localization and Map Building ». In : *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation*. T. 4, 4238-4243 vol.4.

Julier, Simon J et Jeffrey K Uhlmann (28 juill. 1997a). « A New Extension of the Kalman Filter to Nonlinear ». In : *Signal Processing, Sensor Fusion, and Target Recognition VI* 3068.

Julier, Simon J et Jeffrey K Uhlmann (juin 1997b). « A Non-Divergent Estimation Algorithm in the Presence of Unknown Correlations ». In : *Proceedings of the 1997 American Control Conference*. T. 4, 2369-2373 vol.4.

Kalman, R. E. (1er mars 1960). « A New Approach to Linear Filtering and Prediction Problems ». In : *Journal of Basic Engineering* 82.1, p. 35-45.

Ke, Tong, Kejian J Wu et Stergios I Roumeliotis (nov. 2019). « RISE-SLAM : A Resource-aware Inverse Schmidt Estimator for SLAM ». In : *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, p. 354-361.

Kellalib, Billel, Nouara Achour, Vincent Coelen et Abdelkrim Nemra (fév. 2021). « Towards Simultaneous Localization and Mapping Tolerant to Sensors and Software Faults : Application to Omnidirectional Mobile Robot ». In : *Proceedings of the Institution of Mechanical Engineers, Part I : Journal of Systems and Control Engineering* 235.2, p. 269-288.

Kurazume, R., S. Nagata et S. Hirose (mai 1994). « Cooperative Positioning with Multiple Robots ». In : *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, 1250-1257 vol.2.

Lajoie, Pierre-Yves, Benjamin Ramtoula, Fang Wu et Giovanni Beltrame (10 mars 2022). « Towards Collaborative Simultaneous Localization and Mapping : A Survey of the Current Research Landscape ». In : *Field Robotics* 2.1, p. 971-1000.

Lázaro, María T., Lina M. Paz, Pedro Piniés et José A. Castellanos (juin 2013). « Distributed Localization and Submapping for Robot Formations Using a Prior Map* ». In : *IFAC Proceedings Volumes* 46.10, p. 138-145.

Li, Hao, Fawzi Nashashibi et Ming Yang (déc. 2013). « Split Covariance Intersection Filter : Theory and Its Application to Vehicle Localization ». In : *IEEE Transactions on Intelligent Transportation Systems* 14.4, p. 1860-1871.

Li, Nanxi, Chong Pei Ho, Jin Xue, Leh Woon Lim, Guanyu Chen, Yuan Hsing Fu et Lennon Yao Ting Lee (2022). « A Progress Review on Solid-State LiDAR and Nanophotonics-Based LiDAR Sensors ». In : *Laser & Photonics Reviews* 16.11, p. 2100511.

Lima, Antoine (2023). « Cooperative Perception Integrity for Intelligent Vehicles ». Thèse de doct. Université de Technologie de Compiègne.

Lima, Antoine, Philippe Bonnifait, Veronique Cherfaoui et Joelle Al Hage (19 sept. 2021). « Data Fusion with Split Covariance Intersection for Cooperative Perception ». In : *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, p. 7.

Lin, Yangbin, Cheng Wang, Bili Chen, Dawei Zai et Jonathan Li (sept. 2017). « Facet Segmentation-Based Line Segment Extraction for Large-Scale Point Clouds ». In : *IEEE Transactions on Geoscience and Remote Sensing* 55.9, p. 4839-4854.

Luft, L., T. Schubert, S. Roumeliotis et W. Burgard (2016). « Recursive Decentralized Collaborative Localization for Sparsely Communicating Robots ». In : *Robotics : science and systems*.

Magnusson, Martin (2009). « The Three-Dimensional Normal-Distributions Transform : An Efficient Representation for Registration, Surface Analysis, and Loop Detection ». Thèse de doct. Örebro University. 220 p.

Magnusson, Martin, Narunas Vaskevicius, Todor Stoyanov, Kaustubh Pathak et Andreas Birk (mai 2015). « Beyond Points : Evaluating Recent 3D Scan-Matching Algorithms ». In : *2015 IEEE International Conference on Robotics and Automation (ICRA)*, p. 3631-3637.

Marchand, Olivier Le, Philippe Bonnifait, Javier Bañez-Guzmán, François Peyret et David Betaille (avr. 2008). « Performance Evaluation of Fault Detection Algorithms as Applied to Automotive Localisation ». In : Toulouse, France.

Martins, Allan De Medeiros, Adrião Duarte Dória Neto et Jorge Dantas De Melo (2004). « Comparison between Mahalanobis Distance and Kullback-Leibler Divergence in Clustering Analysis ». In : *Wseas Transactions on Systems* 3.2.

Mendes, Ellon, Pierrick Koch et Simon Lacroix (oct. 2016). « ICP-based Pose-Graph SLAM ». In : *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. Lausanne, Switzerland : IEEE, p. 195-200.

Moisan, Lionel, Pierre Moulon et Pascal Monasse (19 mai 2012). « Automatic Homographic Registration of a Pair of Images, with A Contrario Elimination of Outliers ». In : *Image Processing On Line* 2, p. 56-73.

Mu, H., T. Bailey, P. Thompson et H. Durrant-Whyte (avr. 2011). « Decentralised Solutions to the Cooperative Multi-Platform Navigation Problem ». In : *IEEE Transactions on Aerospace and Electronic Systems* 47.2, p. 1433-1449.

Muntzinger, Marc M., Michael Aeberhard, Sebastian Zuther, Mirko Mählisch, Matthias Schmid, Jürgen Dickmann et Klaus Dietmayer (juin 2010). « Reliable Automotive Pre-Crash System with out-of-Sequence Measurement Processing ». In : *2010 IEEE Intelligent Vehicles Symposium*, p. 1022-1027.

Murphy, Kevin Patrick (automne 2002). « Dynamic Bayesian Networks : Representation, Inference and Learning ». Thèse de doct. Berkeley : University of California. 255 p.

Nettleton, Eric W. et Hugh F. Durrant-Whyte (4 oct. 2001). « Delayed and Asequent Data in Decentralized Sensing Networks ». In : sous la dir. de Gerard T. McKee et Paul S. Schenker. Boston, MA, p. 1-9.

Ouyang, Ming, Xuesong Shi, Yujie Wang, Yuxin Tian, Yingzhe Shen, Dawei Wang, Peng Wang et Zhiqiang Cao (22 août 2021). *A Collaborative Visual SLAM Framework for Service Robots*. preprint.

Panzieri, Stefano, Federica Pascucci et Roberto Setola (oct. 2006). « Multirobot Localisation Using Interlaced Extended Kalman Filter ». In : *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, p. 2816-2821.

Paz, L.M., P. Jensfelt, J.D. Tardos et J. Neira (avr. 2007). « EKF SLAM Updates in O(n) with Divide and Conquer SLAM ». In : *Proceedings 2007 IEEE International Conference on Robotics and Automation*, p. 1657-1663.

Ragot, José (1er fév. 2021). « Aberrant Measurements : Detection, Localization, Suppression, Acceptance and Robustness ». In : *Measurement* 172, p. 108872.

Renzler, Tobias, Michael Stolz, Markus Schratter et Daniel Watzenig (mai 2020). « Increased Accuracy For Fast Moving LiDARS : Correction of Distorted Point Clouds ». In : *2020 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, p. 1-6.

Rheaume, Francois et Abder Rezak Benaskeur (déc. 2008). « Forward Prediction-Based Approach to Target-Tracking with Out-of-Sequence Measurements ». In : *2008 47th IEEE Conference on Decision and Control*, p. 1326-1333.

Roumeliotis, S.I. et G.A. Bekey (oct. 2002). « Distributed Multirobot Localization ». In : *IEEE Transactions on Robotics and Automation* 18.5, p. 781-795.

Roumeliotis, S.I., G.S. Sukhatme et G.A. Bekey (oct. 1998). « Sensor Fault Detection and Identification in a Mobile Robot ». In : *Proceedings. 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications*. T. 3, 1383-1388 vol.3.

Saarinen, Jari, Henrik Andreasson, Todor Stoyanov, Juha Ala-Luhtala et Achim J. Lilienthal (mai 2013). « Normal Distributions Transform Occupancy Maps : Application to Large-Scale Online 3D Mapping ». In : *2013 IEEE International Conference on Robotics and Automation*, p. 2233-2238.

Schmidt, STANLEY F. (1er jan. 1966). « Application of State-Space Methods to Navigation Problems ». In : *Advances in Control Systems*. Sous la dir. de C. T. Leondes. T. 3. Elsevier, p. 293-340.

Schnabel, R., R. Wahl et R. Klein (2007). « Efficient RANSAC for Point-Cloud Shape Detection ». In : *Computer Graphics Forum* 26.2, p. 214-226.

Schulz, Cornelia et Andreas Zell (mai 2020). « Real-Time Graph-Based SLAM with Occupancy Normal Distributions Transforms ». In : *2020 IEEE International Conference on Robotics and Automation (ICRA)*, p. 3106-3111.

Segal, A., D Haehnel et S Thrun (2009). « Generalized ICP ». In : *Robotics : science and systems* 2.4, p. 435.

Shi, Xiuying, Jianjun Peng, Jiping Li, Pitao Yan et Hangyu Gong (2019). « The Iterative Closest Point Registration Algorithm Based on the Normal Distribution Transformation ». In : *Procedia Computer Science* 147, p. 181-190.

Smith, Randall, Matthew Self et Peter Cheeseman (1990). « Estimating Uncertain Spatial Relationships in Robotics ». In : *Autonomous Robot Vehicles*. Sous la dir. d'Ingemar J. Cox et Gordon T. Wilfong. New York, NY : Springer, p. 167-193.

Stoyanov, Todor, Martin Magnusson, Henrik Andreasson et Achim J Lilienthal (oct. 2012). « Fast and Accurate Scan Registration through Minimization of the Distance between Compact 3D NDT Representations ». In : *The International Journal of Robotics Research* 31.12, p. 1377-1393.

Strasdat, H., J. M. M. Montiel et A. J. Davison (mai 2010). « Real-Time Monocular SLAM : Why Filter ? » In : *2010 IEEE International Conference on Robotics and Automation*, p. 2657-2664.

Takeuchi, Eijiro et Takashi Tsubouchi (oct. 2006). « A 3-D Scan Matching Using Improved 3-D Normal Distributions Transform for Mobile Robotic Mapping ». In : *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, p. 3068-3073.

Tessier, C., C. Cariou, C. Debain, F. Chausse, R. Chapuis et C. Rousset (sept. 2006). « A Real-Time, Multi-Sensor Architecture for Fusion of Delayed Observations : Application to Vehicle Localization ». In : *2006 IEEE Intelligent Transportation Systems Conference*, p. 1316-1321.

Thrun, Sebastian, Daphne Koller, Zoubin Ghahramani, Hugh Durrant-Whyte et Andrew Y. Ng (2004). « Simultaneous Mapping and Localization with Sparse Extended Information Filters : Theory and Initial Results ». In : *Algorithmic Foundations of Robotics V*. Sous la dir. de Jean-Daniel Boissonnat, Joel Burdick, Ken Goldberg et Seth Hutchinson. Springer Tracts in Advanced Robotics. Berlin, Heidelberg : Springer, p. 363-380.

Thrun, Sebastian et Yufeng Liu (2005). « Multi-Robot SLAM with Sparse Extended Information Filers ». In : *Robotics Research. The Eleventh International Symposium*. Sous la dir. de Paolo Dario et Raja Chatila. Springer Tracts in Advanced Robotics. Berlin, Heidelberg : Springer, p. 254-266.

Tipaldi, Gian Diego et Kai O. Arras (mai 2010). « FLIRT - Interest Regions for 2D Range Data ». In : *2010 IEEE International Conference on Robotics and Automation*, p. 3616-3622.

Triggs, Bill, Philip F. McLauchlan, Richard I. Hartley et Andrew W. Fitzgibbon (2000). « Bundle Adjustment — A Modern Synthesis ». In : *Vision Algorithms : Theory and Practice*. Sous la dir. de Bill Triggs, Andrew Zisserman et Richard Szeliski. Réd. par Gerhard Goos, Juris Hartmanis et Jan Van Leeuwen. T. 1883. Berlin, Heidelberg : Springer Berlin Heidelberg, p. 298-372.

Uhlmann Jeffrey K., Simon Julier (2009). « General Decentralized Data Fusion with Covariance Intersection ». In : *Handbook of Multisensor Data Fusion*. 2e éd. CRC Press.

Ulas, Cihan et Hakan Temeltas (1er jan. 2011). « A 3D Scan Matching Method Based On Multi-Layered Normal Distribution Transform ». In : *IFAC Proceedings Volumes*. 18th IFAC World Congress 44.1, p. 11602-11607.

Vemprala, Sai H. et Srikanth Saripalli (2021). « Collaborative Localization for Micro Aerial Vehicles ». In : *IEEE Access* 9, p. 63043-63058.

Vicentini, Federico (1er juin 2020). « Terminology in Safety of Collaborative Robotics ». In : *Robotics and Computer-Integrated Manufacturing* 63, p. 101921.

Von Neumann, John (1956). « Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components ». In : *Automata studies* 34, p. 43-98.

Welch, Greg et Gary Bishop (1997). *An Introduction to the Kalman Filter*. Chapel Hill : University of North Carolina, p. 16.

WooYeon Jeong et Kyoung Mu Lee (2005). « CV-SLAM : A New Ceiling Vision-Based SLAM Technique ». In : *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Edmonton, Alta., Canada : IEEE, p. 3195-3200.

Wu, Kejian J et Stergios I Roumeliotis (sept. 2016). *Inverse Schmidt Estimators*. 2016-003. University of Minnesota : Multiple Autonomous Robotic Systems Labroratory, p. 21.

Xia, Shaobo, Dong Chen, Ruisheng Wang, Jonathan Li et Xinchang Zhang (2020). « Geometric Primitives in LiDAR Point Clouds : A Review ». In : *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 13, p. 685-707.

Yan, Xu, Jiantao Gao, Chaoda Zheng, Chao Zheng, Ruimao Zhang, Shenghui Cui et Zhen Li (14 oct. 2022). *2DPASS : 2D Priors Assisted Semantic Segmentation on LiDAR Point Clouds*. preprint.

Zanetti, Renato et Christopher D'Souza (déc. 2013). « Recursive Implementations of the Schmidt-Kalman 'Consider' Filter ». In : *The Journal of the Astronautical Sciences* 60.3-4, p. 672-685.

Zhang, Ji et Sanjiv Singh (12 juill. 2014). « LOAM : Lidar Odometry and Mapping in Real-time ». In : *Robotics : Science and Systems X*. Robotics : Science and Systems Foundation.

Zhang, Tianjun, Lin Zhang, Yang Chen et Yicong Zhou (2022). « CVIDS : A Collaborative Localization and Dense Mapping Framework for Multi-Agent Based Visual-Inertial SLAM ». In : *IEEE Transactions on Image Processing* 31, p. 6562-6576.

Zhang, Yanhao, Teng Zhang et Shoudong Huang (mai 2018). « Comparison of EKF Based SLAM and Optimization Based SLAM Algorithms ». In : *2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA)*. Wuhan : IEEE, p. 1308-1313.

Zhu, N., J. Marais, D. Bétaille et M. Berbineau (sept. 2018). « GNSS Position Integrity in Urban Environments : A Review of Literature ». In : *IEEE Transactions on Intelligent Transportation Systems* 19.9, p. 2762-2778.

Zhu, Xinge, Hui Zhou, Tai Wang, Fangzhou Hong, Yuexin Ma, Wei Li, Hongsheng Li et Dahua Lin (juin 2021). « Cylindrical and Asymmetrical 3D Convolution Networks for LiDAR Segmentation ». In : *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, p. 9934-9943.

Zhu, Ziyu, Kongtao Zhu, Zhentan Zheng, Shitao Chen et Nanning Zheng (oct. 2022). « Multi-L : A Novel Multi-Robot Cooperative Localization Method in Indoor Environment ». In : *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, p. 2436-2443.