# Système de déploiement d'un robot mobile autonome basé sur des balises

Lionel Génevé

UNIVERSITÉ DE STRASBOURG

ÉCOLE DOCTORALE MSII

**Laboratoire ICube** (UMR 7357)

# THÈSE présentée par :

**Lionel GÉNEVÉ**

soutenue le : **25 Septembre 2017**

pour obtenir le grade de : **Docteur de l'université de Strasbourg**
Discipline/Spécialité : **Robotique**

# Système de déploiement d'un robot mobile autonome basé sur des balises

**THÈSE dirigée par :**

| | |
|---|---|
| **M. LAROCHE Édouard** | Professeur des Universités, Université de Strasbourg |
| **M. KERMORGANT Olivier** | Maître de conférences, École Centrale de Nantes |

**RAPPORTEURS :**

| | |
|---|---|
| **M. JAULIN Luc** | Professeur des Universités, Université de Bretagne Occidentale |
| **M. BONNIFAIT Philippe** | Professeur des Universités, Université de Technologie de Compiègne |

**AUTRES MEMBRES DU JURY :**

| | |
|---|---|
| **M. GARCIA Gaetan** | Maître de conférences, École Centrale de Nantes |
| **M. BAYLE Bernard** | Professeur des Universités, Université de Strasbourg |

# Résumé

Cette thèse CIFRE s'inscrit dans le cadre d'un projet visant à développer un robot mobile autonome capable de réaliser des tâches spécifiques dans une zone préalablement définie. Pour ce faire, deux phases distinctes se succèdent. La première permet l'apprentissage des limites de la zone d'opération ainsi que des éventuels obstacles à l'intérieur. La seconde correspond à la réalisation des tâches par le robot en totale autonomie. Afin de faciliter la mise en œuvre du système, des balises radiofréquences fournissant des mesures de distance par rapport au robot sont disposées au préalable autour du terrain.

Au cours de ces travaux, différentes techniques sont développées pour permettre la réalisation des deux phases précitées. Ainsi, la phase d'apprentissage qui consiste à reconstruire la trajectoire des limites du terrain ainsi que la position des balises, correspond au problème de localisation et cartographie simultanées (SLAM). Afin de traiter ce problème, une solution basée sur un filtre de Kalman étendu (EKF) est proposée. Le filtre implémenté permet de fusionner les informations fournies par les codeurs incrémentaux du robot avec les mesures de distance provenant des balises afin d'estimer la pose du robot ainsi que la position des balises. Cependant, l'utilisation de mesures de distance uniquement, requiert une attention particulière au moment de l'insertion de la position des balises dans le vecteur d'état du filtre. Dans ce contexte-là, une nouvelle méthode d'initialisation des balises dans le filtre EKF est proposée.

L'intérêt principal de déterminer la position des balises durant la phase d'apprentissage est d'utiliser uniquement un algorithme de localisation pendant la phase d'opération du robot. Afin de résoudre ce problème de localisation, différentes méthodes sont proposées.

Dans un premier temps, des méthodes utilisant uniquement les mesures de distance sont présentées, dont plusieurs variantes de l'algorithme de trilatération. En remarquant la faible robustesse aux mesures aberrantes de la majorité des algorithmes, une nouvelle méthode robuste basée sur la théorie des polynômes en sommes de carrées (SoS) et les inégalités matricielles linéaires (LMI) est proposée. Cette méthode permet, pour certains types de bruit, de retourner un intervalle avec la garantie de contenir la vraie solution. Moins précise que les autres méthodes lorsque le bruit sur les mesures de distance est faible, l'approche proposée surpasse la majorité des autres méthodes lorsque le niveau de bruit et le nombre de mesures erronées augmentent.

Dans un second temps, en tenant compte qu'en pratique le robot est équipé de divers capteurs, un algorithme permettant de fusionner les données de différents capteurs est implémenté. Tout comme pour le SLAM, l'algorithme est basé sur un filtre EKF. Une première version 2D est présentée, puis une seconde en 3D, qui permet en plus, de fusionner les données d'une centrale inertielle et d'un GPS. La version 2D est alors testée en simulation et sur un jeu de données réelles, puis comparée à d'autres algorithmes de référence. Toujours dans un souci de robustesse aux mesures aberrantes, un test de Mahalanobis est mis en place avec succès au sein

du filtre EKF comme le démontre les expériences réalisées.

Enfin, le dernier axe de recherche de ce travail a porté sur le placement optimal des balises. En effet, le positionnement ainsi que le nombre des balises utilisées influencent la qualité de localisation du robot. Pour résoudre ce problème d'optimisation, un algorithme génétique est utilisé et différentes fonctions de coût sont proposées. Les critères utilisés pour implémenter les différentes fonctions de coût reposent sur la maximisation de la couverture et de la précision de localisation du terrain. En s'inspirant des fonctions de coût basées sur le déterminant de la matrice de Fisher présentes dans la littérature, une nouvelle formulation est proposée. Cette dernière consiste à maximiser le pourcentage de terrain dont la précision de localisation est inférieure à un seuil donné, et possède l'avantage de pouvoir spécifier la précision désirée sur l'ensemble ou en chaque position du terrain. Les différentes fonctions de coût ont ensuite été testées et comparées. S'il ressort que l'algorithme de placement permet bien d'optimiser le coût par rapport à des placements uniforme et aléatoire, cela ne se traduit pas toujours par une amélioration de la localisation du robot en utilisant un algorithme de trilatération. En effet, certaines fonctions de coût induisent des placements où la couverture et la précision de localisation ne sont pas homogènes sur tout le terrain, ce qui entraîne des difficultés de localisation dans certaines zones. De plus, les fonctions de coût étant basées sur l'algorithme de trilatération, celles-ci ne sont plus aussi bien adaptées quand le robot est localisé par un algorithme de fusion de données tel que le filtre EKF. Enfin, dans l'optique d'obtenir de façon automatique, le nombre de balises ainsi que leurs positionnements, une fonction de coût multicritère adaptée à l'algorithme génétique est utilisée. Cette nouvelle formulation permet d'obtenir un diagramme de Pareto où il est possible de visualiser le pourcentage de terrain couvert par une précision de localisation donnée en fonction du nombre de balises.

**Mots clés :** Robotique mobile, localisation, SLAM, placement optimal de capteurs, balises radiofréquence, mesures de distance.

# Abstract

This CIFRE thesis is part of a project which aims at developing an autonomous mobile robot able to perform specific tasks in a preset area. The system is deployed in two distinct stages. The first one consists of learning the limits of the operating area and the potential obstacles inside. The second step corresponds to the operation of the robot in complete autonomy. To ease the setup of the system, radio-frequency beacons providing range measurements with respect to the robot are set up beforehand on the borders of the robot's workspace.

In this work, different techniques are developed in order to allow the achievement of the two aforementioned steps. Thus, the learning step which consists in estimating the trajectory of the workspace's borders and the beacon's positions, corresponds to the simultaneous localization and mapping (SLAM) problem. In order to deal with it, a solution based on an extended Kalman Filter (EKF) is proposed. The filter fuses the information provided by the wheel's encoders and the range measurements from the beacon in order to estimate the robot's pose and the beacon's positions. However, using the range measurements only requires a particular attention while inserting the beacon's position in the state vector of the filter. In this context, a new method to initialize the beacons in the EKF is proposed.

The main advantage of estimating the beacon's positions during the learning phase is to rely only on a localization algorithm during the operating phase of the robot. In order to solve this localization problem, various methods are proposed.

Firstly, some range-only based methods are presented, and in particular different versions of the trilateration algorithm. By noticing that most of these algorithms are sensitive to outliers, a new robust method based on the theory of sums-of-squares (SoS) polynomials and linear matrix inequalities (LMI) is proposed. This method allows, for some types of noise, to return an interval where the true solution is guaranteed to belong inside. Less accurate than the other methods when the noise on the range measurements is low, the proposed approach outperforms the majority of them when the level of noise and the number of outliers increase.

Secondly, by taking into account that, in practice, various sensors are mounted on the robot, an algorithm fusing the data from different sensors is implemented. As for the SLAM, this algorithm is based on the EKF framework. A 2D, and then a 3D version of the algorithm are presented. In the 3D version, the possibility to fuse the data from an inertial measurement unit (IMU), and from a GPS is also added. The 2D version is first tested in simulation and with a real dataset, and then compared to other popular localization algorithms. Always with the robustness to outliers in mind, a Mahalanobis test is successfully integrated in the EKF, like it is shown during the experiments.

Finally, le last line of research was devoted to the sensor placement problem. Indeed, the positioning along with the number of beacons used leverage the localization performances of the robot. In order to solve this problem, a genetic algorithm (GA) is used and different cost functions are proposed. The criteria used to implement the cost functions are based on the maximization of the coverage and the localization accuracy over the field. By drawing inspiration from the cost functions based on the Fisher Information Matrix (FIM) determinant found in the literature, a new cost function is formulated. The proposed formulation consists of maximizing the percentage of the workspace where the localization accuracy is below a given threshold. The main advantage of the formulation, is that it is possible to control the desired accuracy on the whole workspace or at each position of it. Once implemented, the different cost functions are then tested and compared. If it shows up that the placement algorithm is indeed optimizing the cost with respect to a uniform and a random placement, it does not always imply that the localization of a robot using a trilateration algorithm is improved. Indeed, some cost functions return beacon placements where the coverage and the localization accuracy are not homogeneous over the whole workspace, which leads in some areas to difficulties to localize the robot. Moreover, as the cost functions are based on the trilateration algorithm, they are not as suitable when the robot is localized by a sensor fusion algorithm like the Kalman filter. At last, in order to get the number of beacons and their positions in an automatic way, a multi-criteria cost function adapted to the genetic algorithm is used. With this new formulation, it is possible to obtain a Pareto front, where one can visualize the percentage of the area covered with a given localization accuracy as a function of the number of beacons.

**Keywords :** Mobile robotics, localization, SLAM, optimal sensor placement, radiofrequency beacons, range-only measurements.

# REMERCIEMENTS

Je voudrais dans un premier temps remercier MM. Treger et Laroche qui ont été à l'origine de cette thèse CIFRE, et qui ont assuré son bon déroulement jusqu'au bout. Au sein de l'entreprise et plus particulièrement du centre de recherche, je souhaite remercier tout d'abord Laurent D. pour ses conseils avisés, sa patience à répondre à mes questions et pour m'avoir épaulé lors de la conception de mes premières cartes électroniques. Par la suite se sont Guillaume F. et Fabien Q. qui ont pris sa relève et qui m'ont permis d'avancer plus rapidement en me déchargeant de la réalisation et surtout de la réparation de certaines cartes électroniques, donc merci aussi à vous deux. Ayant très peu de connaissances en conception mécanique, j'ai pu compter sur l'aide d'Henry P. et de Damien W. notamment, pour la réalisation des supports des capteurs et le montage du prototype. Enfin, merci à MM. Kettering et Treger pour leur suivi et conseils lors de ma présence dans l'entreprise.

Du côté du laboratoire, où j'ai eu l'occasion et le plaisir de côtoyer un bon nombre de stagiaires et de doctorants de tous horizons, je voudrais saluer en particulier Sebastian F. qui partagea un temps mon bureau, Danda Pani P. pour sa bonne humeur et ses nombreuses anecdotes, Lijia G., Emmanuel R., Chinmay S., Xavier W., Maximilien L., Imane K. et Devesh A. Durant les premiers mois de la thèse, j'ai eu l'opportunité d'encadrer Hervé pour son stage d'école d'ingénieur de trois mois. Son aide pour rendre les simulations Gazebo un peu plus réalistes, en concevant certaines améliorations graphiques a été précieuse. Puis, en travaillant sur un algorithme de couverture de terrain, il a doté le robot d'une fonctionnalité essentielle à son déploiement final, qui n'a malheureusement pas encore pu être testée en pratique.

Je tiens aussi à saluer Adlane H. pour avoir partagé ses idées et ses travaux de recherche qui a abouti à la publication d'un article scientifique, ainsi que Loïc C. pour son aide sur la mise en œuvre des expérimentations et sur certaines aspect de programmation embarqué.

Avant de remercier mes deux encadrants du côté laboratoire, je voudrais remercier les membres du jury composé de MM. Jaulin L., Bonnifait P., Garcia G. et Bayle B. Tout d'abord, pour avoir accepté de faire partie du comité de thèse, mais aussi pour les relectures et les corrections ayant permis la finalisation d'un mémoire plus abouti.

Ainsi, j'en viens à remercier les deux personnes qui m'ont le plus apporté au cours de ces années de thèse : MM. Édouard Laroche et Olivier Kermorgant. Merci à Édouard pour avoir rempli son rôle de directeur de thèse à merveille, notamment par sa disponibilité, son expérience et sa gestion parfaite du début à la fin. Merci aussi à Olivier pour m'avoir fait découvrir les excellents outils que sont ROS et Gazebo au début de ma thèse et qui m'ont permis la réalisation de simulations de qualité. Merci à tous les deux pour votre apport essentiel dans les contributions, vos conseils au cours de nos échanges, vos relectures, et surtout de m'avoir fait prendre conscience de l'intérêt et du bénéfice d'aller jusqu'à la conclusion de ces trois années et demi de travaux.

Enfin, rien n'aurait été possible sans ma famille et en particulier ma mère et ma grand-mère. Merci pour votre contribution quotidienne qui m'a permis de bénéficier de conditions optimales tout au long de ce travail.

# TABLE DES MATIÈRES

# LISTE DES FIGURES

# LISTE DES TABLEAUX

# NOTATIONS

## Acronymes

— GPS : Global Positioning System
— RTK : Real Time Kinematic
— ECEF : Earth-Centered Earth-Fixed
— NED : North East Down
— ENU : East North Up
— MEMS : MicroElectroMechanical System
— IMU : Inertial Measurement Unit
— AHRS : Attitude and Heading Reference System
— INS : Inertial Navigation System
— SLAM : Simultaneous Localization And Mapping
— RO : Range-Only
— RF : Radio Frequency
— UWB : Ultra-Wide Band
— LOS/NLOS : Line-of-Sight/Non Line-of-sight
— RSSI : Received Signal Strength Indicator
— DOP : Dilution Of Precision
— PDOP : Positional Dilution Of Precision
— GDOP : Geometric Dilution Of Precision
— CRLB : Cramer Rao Lower Bound
— KF/EKF : Kalman Filter/Extended Kalman Filter
— GES : Globally Exponentially Stable
— LTI : Linear Time Invariant (sysem)
— PF : Particle Filter
— RBPF : Rao-Blackwellized Particle Filter
— LMI : Linear Matrix Inequality
— SoS : Sum Of Squares
— UGV : Unmanned Ground Vehicle
— AGV : Autonomous Guided Vehicle
— AUV : Autonomous Underwater Vehicle

— UAV : Unmanned Aerial Vehicle
— BLDC : BrushLess Direct Current
— PWM : Pulse Width Modulation
— ROS : Robot Operating System
— LIDAR : LIght Detection And Ranging
— ToF : Time Of Flight
— SPI : Serial Peripheral Interface
— $I^2C$ : Inter-Integrated Circuit
— GA : Genetic Algorithm
— FIM : Fisher Information Matrix
— RMSE : Root Mean Square Error
— SfM : Structure From Motion

# Notations

## Notations générales

— Les scalaires sont écrits en lettres minuscules : $x$,
— Les vecteurs sont écrits en gras et lettres minuscules : $\mathbf{x}$,
— Les matrices sont écrites en gras et lettres majuscules : $\mathbf{X}$.

## Algèbre

— $SO(3)$ : espace orthogonal des matrices de rotation,
— $SE(3)$ : espace euclidien des matrices homogènes,
— $\mathbf{M}_{ij}$ : élément de la $i^{\text{ème}}$ et $j^{\text{ème}}$ colonne de la matrice $\mathbf{M}$,
— $\mathbf{M}^T$ : transposée de la matrice $\mathbf{M}$,
— $\mathbf{M}^{-1}$ : inverse de la matrice $\mathbf{M}$,
— $\mathbf{M}^\dagger$ : pseudo-inverse de la matrice $\mathbf{M}$,
— $det(\mathbf{M})$ : déterminant de la matrice $\mathbf{M}$,
— $ker(\mathbf{M})$ : noyau de la matrice $\mathbf{M}$,
— $diag(\mathbf{M})$ : vecteur formé par la diagonale de la matrice $\mathbf{M}$,
— $diag(\mathbf{d})$ : matrice ayant pour diagonale le vecteur $\mathbf{d}$,
— $\mathbf{v}_\times$ : forme matricielle du produit vectoriel de $\mathbf{v}$,
— $\mathbf{0}_{n\times m}$ : matrice nulle de dimension $n \times m$,
— $\mathbf{I}_{n\times m}$ : matrice identité de dimension $n \times m$.

## Modélisation robot/capteurs

— $\mathbf{p} = (x, y)^T$ : position en 2D,
— $\mathbf{p} = (x, y, z)^T$ : position en 3D,
— $\theta$ : angle d'orientation du robot dans le plan (en 2D),
— $(\phi, \theta, \psi)$ : angles de roulis, tangage et lacet de l'orientation en 3D,

— $\mathbf{q} = (q_w, q_x, q_y, q_z)^T$ : quaternion (convention d'Hamilton),
— $\mathbf{R}$ : matrice de rotation en 2D ou 3D,
— $\mathbf{p} = (x, y, \theta)^T$ : pose du robot en 2D,
— $\mathbf{p} = (x, y, z, q_w, q_x, q_y, q_z)^T$ : pose en 3D.

# CHAPITRE 1

# INTRODUCTION GÉNÉRALE

L'ensemble des travaux réalisés au cours de cette thèse s'inscrit dans le domaine de la robotique, et plus précisément de la robotique mobile. La robotique est un domaine de recherche pluridisciplinaire faisant appel notamment à la mécanique, l'électronique, l'automatique, l'informatique, la vision, etc. Afin de bien cerner le cadre de ce travail et les motivations qui ont mené au démarrage de ce projet, nous allons au cours de ce chapitre introductif donner tout d'abord un bref aperçu sur ce qu'est la robotique mobile ainsi que ses applications. Dans un second temps, le contexte ainsi que les motivations du projet seront détaillés. Puis, les différentes contributions apportées au cours de ces travaux sont rapidement présentées, avant de conclure par l'organisation de la suite de ce mémoire.

## 1.1 Robotique mobile

Avec l'expansion de l'automatisation, de plus en plus de tâches sont aujourd'hui effectuées par des robots. Ce phénomène, qui a dans un premier temps principalement concerné l'industrie, s'étend maintenant progressivement à d'autres domaines. Ainsi, en parallèle des robots automates et robots manipulateurs généralement fixés à un endroit précis dans une usine, on rencontre de plus en plus de robots mobiles capables d'aller effectuer une mission spécifique sur un lieu ou dans un endroit donné.

Depuis les années 1970, et l'émergence des premiers robots mobiles avec notamment le robot Shakey développé par l'institut de recherche de Stanford [76], voir Figure 1.1, on trouve aujourd'hui des robots mobiles pouvant se déplacer sur Terre (UGV), dans les airs (UAV), sur et dans l'eau (AUV). Ces robots présentent aussi des formes diverses et variées en fonction notamment de leur mode de locomotion qui s'inspire parfois de la nature. Pour les robots terrestres, outre les robots à roues, on trouve aussi des robots s'inspirant de l'être humain (robot humanoïde [54]), de l'araignée [122], du serpent [137], et même du guépard avec le robot Cheetah développé au MIT [1] [14], voir Figure 1.2. Pour les robot aériens, en plus des robot reprenant les formes d'avion ou d'hélicoptère, on trouve les multi-rotors (en général 4, 6 ou 8) [91], et ceux s'inspirant des oiseaux comme le colibri [132], ou de la libellule [60], Figure 1.3.

Au vu de la multitude de formes que peuvent prendre les robots mobiles, on imagine facilement que les applications possibles se retrouvent dans de nombreux domaines dont les principaux sont : le militaire, l'industrie, l'agriculture, l'assistance, les services et les jouets. Ainsi,

1. Massachusetts Institute of Technology

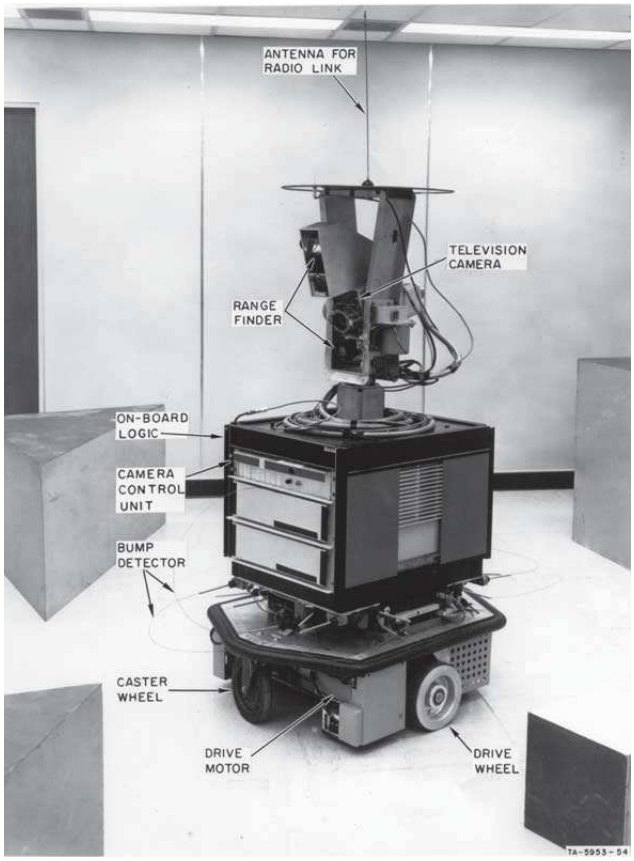



FIGURE 1.1 – Robot Shakey et ses capteurs (source : https://en.wikipedia.org/wiki/Shakey_the_robot)

FIGURE 1.2 – Robot Super Mini Cheetah inspiré du guépard (source : http://biomimetics.mit.edu/research/mit-super-mini-cheetah)

pour chaque domaine voici une liste non-exhaustive des applications où les robots mobiles sont présents, dont certaines sont visibles sur la Figure 1.4 :

— Militaire : déminage, reconnaissance, renseignement, transport de matériel,

— Industrie : collecte dans des entrepôts, inspection dans des usines, dans des zones de catastrophe (sites contaminés ou radioactifs),

— Agriculture : plantation et récolte, entretion du sol et désherbage, cartographie d'un champ,

— Assistance : aide à la visite dans un musée, dans un aéroport, dans les hôpitaux, aide aux personnes agées,

— Services : nettoyage et entretien : aspirateur, piscine, gouttières, vitre, pelouse,

FIGURE 1.3 – Robot Bionicopter de Festo inspiré de la libellule (source : https://www.festo.com/group/fr/cms/10224.htm)

— Jouets : voitures et drones téléguidés, photographie...

(a) Robot de téléprésence d'awabot

(b) Robot aspirateur Roomba 966 d'iRobot


(c) Robot de déminage Kobra 710 d'iRobot

(d) Robot de désherbage de Naio-Technologies

FIGURE 1.4 – Exemples d'applications actuelles des robots mobiles

Comme on peut le voir, les robots mobiles peuvent revêtir de multiples formes et servir dans de nombreuses applications. Dans le cadre de cette thèse, nous utiliserons un robot mobile terrestre à deux roues motrices, de type unicycle, destiné à accomplir des tâches en autonomie dans un environnement préalablement établi. Parmi les applications des robots mobiles citées, certaines nécessitent encore qu'un opérateur humain soit présent afin de téléopérer le robot dans l'exécution de sa tâche. Cependant, dans beaucoup de cas, l'objectif ultime serait que le robot puisse opérer de manière totalement autonome sans aucune intervention extérieure, ni aucun aménagement ou installation préalable. Le déroulement idéal du déploiement d'un robot mobile étant de venir avec son robot, de le placer dans sa zone d'opération, de le mettre en marche et de le laisser effectuer sa mission. Pour certaines applications bénéficiant de conditions favo-

rables, on arrive à se rapprocher de ce cas idéal comme pour les robots aspirateurs ou les robots utilisés pour la logistique dans les entrepôts par exemple, qui évoluent dans des environnements intérieurs fermés et structurés. Malheureusement, dans beaucoup d'autres cas, les contraintes liées à l'environnement et le manque de maturité technologique impliquent l'intervention d'un opérateur humain, l'adaptation et/ou l'apprentissage de l'environnement au préalable. Ainsi, au contraire des robots aspirateurs qui sont maintenant capables d'opérer en quasi-autonomie au sein d'une maison, les robots de tonte nécessitent encore l'installation fastidieuse d'un fil périmétrique pour délimiter le terrain et permettre le retour à la station de recharge en autonomie. De plus, la planification automatique de trajectoire et la prise en compte de l'évolution de l'environnement au cours du temps sont encore très peu présentes dans les solutions existantes. Dans ce travail, une méthode permettant le déploiement d'un robot mobile est présentée. La méthode proposée nécessite seulement l'installation de quelques balises, ainsi que la définition des limites de la zone d'opération du robot par l'utilisateur. Le robot est alors en capacité de réaliser les tâches assignées en autonomie tout en restant à l'intérieur de son périmètre.

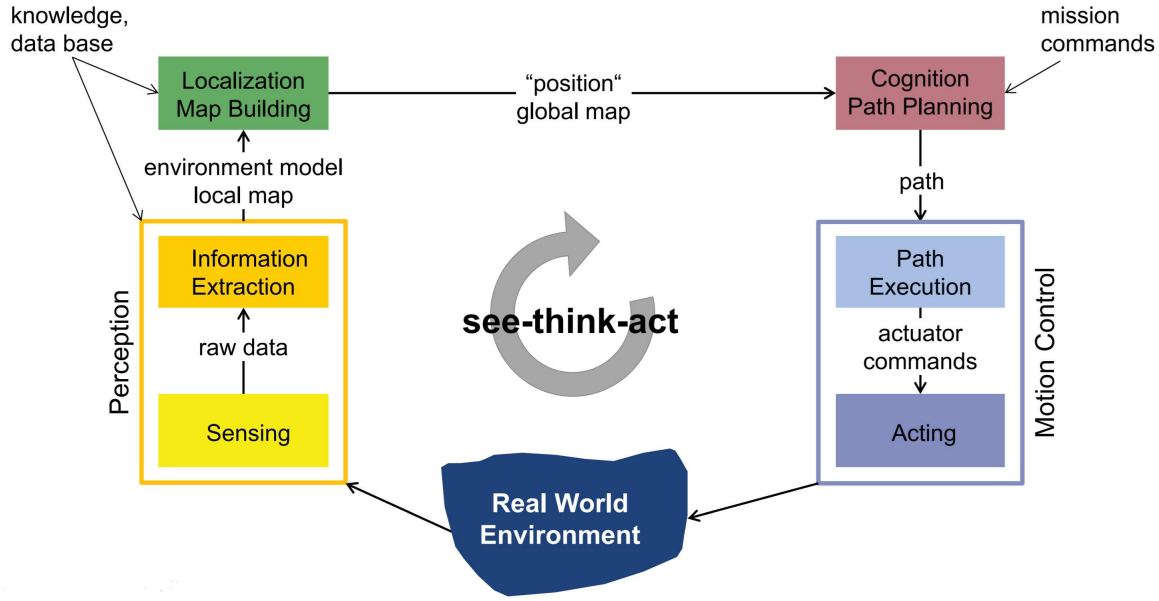



FIGURE 1.5 – Schéma général du fonctionnement d'un robot mobile représenté sous forme d'un cycle : perception-réflexion-action. Reproduit du livre : *An introduction to autonomous mobile robot* [118].

Dans l'objectif de la construction d'un robot mobile autonome, on peut identifier différentes composantes principales :

- Contrôle bas-niveau (moteurs), et haut-niveau (position, trajectoire, ...),
- Perception de l'environnement et de l'état du robot grâce aux capteurs,
- Localisation du robot et cartographie de son environnement,
- Évitement d'obstacles,
- Traitement des ordres de mission, planification de trajectoires.

Toutes ces fonctions sont liées et interagissent entre elles, comme le montre le schéma de la Figure 1.5. Ce schéma représente le cycle de perception-raisonnement-action et résume les différents éléments clés d'un robot mobile lui permettant de devenir autonome. Ainsi, lâché au milieu du monde réel, le robot doit être capable d'extraire les informations pertinentes (perception) lui permettant de se localiser et de construire une représentation de son environnement (localisation et cartographie). En tenant compte de sa position et de son entourage, le robot doit

alors être capable de planifier les différentes actions à accomplir afin de mener à bien sa mission, en fonction des objectifs qui lui sont données (raisonnement). Ces actions de haut-niveau sont alors traduites en actions de bas-niveau qui vont commander les actionneurs du robot et permettre au robot d'aller à une position donnée avec une vitesse donnée (contrôle du mouvement), le rapprochant ainsi de l'objectif à atteindre. Comme on peut le voir, la réalisation d'un robot mobile nécessite la mise en place d'un certain nombre de modules. Dans ce travail, nous nous intéressons plus particulièrement aux fonctions de perception et de localisation et cartographie car l'objectif principal poursuivit ici, est la réalisation d'un système permettant de localiser un robot dans un environnement inconnu. Toutes les parties sur le raisonnement et le contrôle du robot ne seront donc pas abordées même si elles sont nécessaires à la réalisation d'un prototype.

La localisation est donc la fonction permettant à un robot mobile de se localiser par rapport à un environnement connu au préalable. L'emploi du terme localisation suppose donc que l'environnement est connu à l'avance et disponible sous forme d'une carte. Cependant, dans la majorité des cas, la carte de l'environnement n'est pas disponible à priori, et il est donc nécessaire de la construire. Le robot doit alors être capable de se localiser par rapport à un environnement qu'il est en train de construire. Ce problème, bien connu en robotique mobile, est fait référence sous le terme de localisation et cartographie simultanées, ou SLAM en abrégé. Le challenge de ce problème étant qu'en partant de rien, le robot doit être capable de construire une représentation de son environnement et de connaître sa position au sein de celui-ci. La principale difficulté se trouvant dans les premiers instants ou très peu d'information est encore disponible, et ou la moindre erreur peut fausser complétement la procédure. Si tout se passe bien, au fur et à mesure de la collecte des informations, la carte de l'environnement devient plus complète et le robot peut se localiser de manière plus précise. En effet, la corrélation des différents éléments de la carte et le passage du robot dans des endroits déjà explorés permet de réduire les incertitudes. Une bonne introduction et description du problème du SLAM est disponible dans les tutoriels : [37], [3]. Il est aussi possible de trouver des chapitres consacrés au problème du SLAM dans des livres de référence en robotique mobile : *An introduction to autonomous mobile robot* [118], *Probabilistic Robotics* [126] et *Springer Handbook of Robotics* [117]. Plus récemment, Cadena et al. [18], présentent un bilan complet de l'état d'avancement du SLAM, ainsi que les principales difficultés qui restent encore à résoudre. Le problème du SLAM étant bien connu et traité depuis les années 1980, il existe maintenant des méthodes développées bénéficiant d'assez de maturité pour être utilisées dans des applications en dehors des laboratoires. Cependant, la plupart de ces méthodes reposent sur l'utilisation de capteurs, soit très coûteux, soit restreints à certains type d'environnement. Ainsi, on retrouve souvent des méthodes utilisant des lasers, des GPS, ou des centrales inertielles très précises et donc très chères. Le problème du SLAM est donc plutôt bien résolu avec ces capteurs, mais l'utilisation d'autres types de capteurs peut rendre le problème encore plus complexe. Justement, dans le cadre de ce travail, le choix a été fait d'utiliser une technologie à faible coût (de l'ordre d'une cinquantaine d'euros) moins commune, mais présentant des caractéristiques plus adaptées à l'application finale, comme cela est expliqué dans la suite.

## 1.2 Contexte et motivation

Ce travail s'inscrit dans le cadre d'un projet industriel ayant pour objectif le développement d'un robot mobile terrestre autonome. Le robot doit ainsi être capable de réaliser, en totale autonomie, des tâches spécifiques dans une zone préalablement définie par l'utilisateur. Les applications possibles étant par exemple : le nettoyage avec les robots aspirateurs, l'entretien avec les robots de tonte, ou la sécurité avec les robots de surveillance. Comme énoncé précédemment, de plus en plus de robots commercialisés sont capables d'évoluer avec une autonomie croissante.

Mais dans la majorité des cas, l'environnement présente des caractéristiques favorables au déploiement du robot. Ainsi, les applications en extérieur ou dans des zones difficiles (terrain, visibilité, couverture, ...) sont quant à elles beaucoup plus rares. Il apparaît donc intéressant de proposer un système capable d'opérer efficacement dans des scénarios difficiles et déployable dans la majorité des environnements. Comme dans tout projet industriel, la solution développée doit être commercialement viable et respecter certaines contraintes définies dans le cahier des charges. Parmi ces contraintes, les plus importantes sont liées au coût du produit, et à la facilité d'utilisation du système par le client.

### 1.2.1 Objectifs

Dans ces conditions, le principal objectif est d'obtenir un robot mobile capable d'effectuer une ou plusieurs tâches définies par l'utilisateur de façon autonome, tout en nécessitant un minimum d'aménagements et d'interventions extérieurs. Pour répondre à ce besoin, différentes étapes sont identifiées :

— Définition et réalisation d'un système permettant le déploiement d'un robot mobile autonome,

— Choix des capteurs et autres modules équipant le robot et nécessaires à la réalisation du système,

— Développement d'un prototype du robot mobile,

— Développement des méthodes et algorithmes permettant au robot d'évoluer en autonomie,

— Réalisation d'un démonstrateur complet du système permettant de valider la solution développée.

### 1.2.2 Cadre et limitations

Afin de pouvoir utiliser le système dans une majorité de scénarios, aucune hypothèse n'est faite sur le type, la structure, et la taille de l'environnement. Ainsi, même si le développement sera plus axé sur une utilisation en extérieur, la solution proposée devra pouvoir fonctionner tout aussi bien en intérieur. Cependant, pour faciliter le développement, l'essentiel du travail a d'abord été effectué en supposant un environnement 2D, où tous les éléments sont supposés évoluer dans le même plan. Bien entendu, l'application finale devra prendre en compte la réalité 3D de notre monde, et il sera nécessaire d'effectuer les modifications plus ou moins triviales et complexes sur les solutions développées. En plus de l'adaptabilité en milieu extérieur, une autre contrainte doit être prise en compte. Cette contrainte impose que le robot doit être capable d'opérer de jour comme de nuit. Ainsi, en tenant compte des différentes contraintes environnementales, le choix des capteurs à utiliser pour l'application devient un point important. Par exemple, l'utilisation de tous types de caméras (mono, stéréo, omnidirectionnelle, RGBD, ...), qui sont de plus en plus utilisées actuellement, s'avère assez difficile dans de telles conditions. De même pour le GPS, où il n'est pas toujours évident d'avoir une vue parfaitement dégagée vers le ciel afin de garantir et d'optimiser la réception des signaux en provenance des satellites.

Par rapport au cas idéal du déploiement d'un robot mobile, où la seule intervention nécessaire consiste à déposer le robot à l'intérieur de sa zone d'opération, on suppose ici que c'est l'utilisateur qui est en charge de la définition des limites de l'environnement du robot. Durant cette phase, l'utilisateur "apprend" donc au robot les limites de son espace de travail et lui fournit ensuite le résultat. À partir de là, le robot doit alors être capable de se débrouiller en totale autonomie. Bien entendu, la solution permettant de définir les limites du terrain doit se

faire en minimisant les aménagements et être la plus simple possible pour l'utilisateur.

Afin de mener à bien les différents objectifs du projet, tout en tenant compte des différentes contraintes précitées, nous allons maintenant présenter les principales solutions et contributions proposées.

## 1.3 Contributions

Au cours de ce projet, deux types de contribution ont été faites. Le premier concerne le développement de la solution permettant de déployer le robot dans son environnement ainsi que la réalisation d'un prototype du système et du robot associés. Cette partie est brièvement présentée dans le paragraphe suivant. Le second type de contribution, d'ordre scientifique et qui fait l'objet de ce mémoire, concerne les méthodes développées afin de répondre aux différents problèmes de la localisation, du SLAM, et du placement de capteurs. Avant d'aborder plus en détails, les contributions scientifiques, nous allons rapidement présenter la solution retenue pour le déploiement du robot mobile autonome.

### 1.3.1 Déploiement du robot

La procédure de déploiement du robot se fait en deux étapes bien distinctes : l'apprentissage du terrain et de ses limites par l'utilisateur, puis la réalisation en autonomie des tâches assignées au robot. Le principe de la méthode proposée repose sur l'utilisation de balises radiofréquences (RF) permettant d'obtenir une mesure de distance entre deux balises. Ainsi, en préalable à la définition des limites du terrain par l'utilisateur, celui-ci doit dans un premier temps placer un certain nombre de balises sur le pourtour du terrain qu'il va délimiter. Une fois les balises en place, il peut parcourir ou faire parcourir au robot les limites du terrain à apprendre. Pendant cette phase, l'utilisation des balises ainsi que d'autres capteurs précisés par la suite, permet d'estimer la trajectoire effectuée par l'utilisateur ou le robot ainsi que la position des balises. On remarque que cette phase correspond au problème du SLAM, où l'on doit reconstruire une carte de l'environnement (position des balises) en même temps qu'estimer la position (trajectoire) du système mobile. Une fois la position des balises et les limites du terrain connues, l'utilisateur peut transférer les données au robot. Dans le même temps, l'utilisateur définit alors les tâches que le robot doit effectuer dans la zone préalablement délimitée. Il est par exemple possible d'indiquer au robot d'explorer et de couvrir la zone en totalité, ou d'effectuer des trajectoires spécifiques à l'intérieur. Le robot se charge alors de planifier ses missions en générant si besoin les trajectoires adéquates. À partir de ce moment, il est possible d'installer le robot dans sa zone d'opération et de le laisser opérer de façon autonome. Tout au long de la réalisation de sa mission, le robot pourra alors se localiser grâce aux balises dont la position est maintenant connue, et rester à l'intérieur des limites du terrain apprises durant la première phase.

Afin de valider la méthode de déploiement du robot, il a été nécessaire de réaliser un prototype du robot ainsi que des balises. Concernant le robot, en partant d'un châssis et des moteurs, il a été nécessaire de réaliser différentes cartes électroniques permettant notamment de contrôler les moteurs des roues et d'interfacer les différents capteurs. Pour la réalisation des balises, après la sélection de la technologie adéquate permettant d'obtenir les performances désirées, il a été nécessaire de prendre en main le composant RF et de développer une carte électronique permettant de récupérer les mesures de distance grâce à un microcontrôleur. Enfin, une fois l'ensemble des principaux composants fonctionnels et montés sur le robot, des premiers tests en conditions réelles ont été effectués. Toute la réalisation du prototype du robot et des balises a été effectuée en parallèle du travail de recherche dont les principales contributions sont

maintenant présentées.

### 1.3.2 Contributions scientifiques

La présentation des contributions scientifiques faisant l'objet de ce mémoire, seule une rapide description de chacune est donnée ici. Les principaux développements seront détaillés par la suite dans les différents chapitres. La principale contribution de ce travail concerne la réalisation d'un vaste état de l'art accompagné de comparaisons des performances de différentes méthodes de localisation et de SLAM utilisant des mesures de distance. La seconde contribution propose une nouvelle technique d'initialisation de la position des balises dans un filtre de Kalman étendu dans le cadre du problème de la localisation et de la cartographie simultanées. La troisième contribution concerne le développement d'une méthode de placement optimal des balises sur le périmètre de la zone d'opération d'un robot. La méthode d'optimisation est basée sur un algorithme génétique, et différentes fonctions de coût sont proposées.

Ces contributions ont fait l'objet de publications dans les conférences internationales suivantes :

— L. Génevé, O. Kermorgant et É. Laroche, A composite beacon initialization for EKF range-only SLAM, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hambourg, Allemagne, 2015.

— L. Génevé, O. Kermorgant et É. Laroche, Limits of trilateration-based sensor placement algorithms, IEEE Sensors Applications Symposium (SAS), Glassboro, USA, 2017.

— L. Génevé, A. Habed, É. Laroche et O. Kermorgant, Robust Range-Only Mapping via Sum-of-Squares Polynomials, IFAC World Congress, Toulouse, France, 2017.

## 1.4 Organisation du mémoire

La suite de ce manuscrit est décomposée en quatre chapitres.

Le **Chapitre 2** est consacré à la description du robot et des capteurs utilisés dans le cadre de ce travail. Ainsi, après une description sommaire de l'architecture du robot et de ses principaux composants, les équations du modèle du robot et de l'odométrie sont présentées. Dans un second temps, les différents capteurs employés pour localiser le robot (balises, accéléromètre, gyroscope, magnétomètre et GPS) sont décrits et leur modélisation présentée.

Le **Chapitre 3** traite du problème de la localisation d'un robot mobile, utilisant comme principales mesures extéroceptives, des mesures de distance entre le robot et des balises fixes disposées autour de la zone de travail. Ce chapitre est divisé en deux parties. Dans la première, différentes méthodes de localisation utilisant uniquement les mesures de distance sont présentées et comparées. L'approche basée sur les sommes aux carrés de polynômes et les inégalités linéaires matricielles ayant fait l'objet d'une publication y est notamment détaillée. Dans la seconde partie consacrée à la localisation par fusion de données, la réalisation d'un filtre de Kalman étendu (EKF) en 2D permettant de fusionner l'odométrie du robot avec les mesures de distance est détaillée. Les performances du filtre sont alors testées et comparées avec d'autres techniques classiques présentées dans l'état de l'art. Le chapitre se clôture sur la présentation de l'extension du filtre EKF au cas de la 3D.

Le **Chapitre 4** se situe dans la continuité du précédent, où l'on supprime l'hypothèse que la position des balises est connue au préalable. Ce chapitre traite ainsi du problème de la localisation et de la cartographie simultanées (SLAM). Dans le cas de l'utilisation de mesures de distance, ce problème comporte une difficulté supplémentaire avec l'initialisation de la position des balises. Après un état de l'art des différentes méthodes existantes, une nouvelle méthode

d'initialisation permettant un compromis entre rapidité d'initialisation et complexité est proposée. L'approche présentée est ensuite testée et comparée à d'autres techniques en termes notamment de précision et de rapidité d'exécution.

Le **Chapitre 5** traite quant à lui du placement optimal des balises sur le terrain. Il sera notamment vu que le placement a une influence sur les performances de localisation. Afin de résoudre le problème d'optimisation, un algorithme génétique est développé. Celui-ci permet de trouver la position et éventuellement le nombre des balises en optimisant une fonction de coût. La principale difficulté de la méthode étant alors de définir la bonne fonction de coût. Pour cela, différentes solutions reprises de la littérature, ainsi que de nouvelles formulations proposées sont testées et comparées. L'apport du placement optimisé des balises sera notamment testé sur les performances de localisation d'un robot dans son environnement. En fin de ce chapitre, les travaux sur les limitations des fonctions de coût basées sur la trilatération, ayant fait l'objet d'une publication, sont présentés.

Enfin, une conclusion générale sur l'ensemble du travail de recherche et sur le prototype est donnée. Les principaux résultats y sont notamment récapitulés avec des précisions sur les limitations et les perspectives d'améliorations.

# CHAPITRE 2

# MODÉLISATION ET ARCHITECTURE

L'une des premières étapes dans la réalisation d'un robot mobile consiste à définir le mode de locomotion à mettre en œuvre, ainsi que les différents éléments et capteurs qui le composent. Une fois les actionneurs et les capteurs définis et modélisés, il est alors possible de développer les algorithmes permettant au robot d'obtenir le degré d'autonomie souhaité pour la réalisation des tâches qui lui seront assignées. Dans ce premier chapitre, nous présentons l'architecture générale du robot et identifions les différents blocs fonctionnels qui le composent. Ensuite, nous rappelons les équations du modèle du robot, de type unicycle, qui seront utilisées tout au long de ce mémoire. À partir de ce modèle, nous dérivons les équations d'odométrie permettant d'estimer la position du robot en intégrant les vitesses des roues mesurées par des capteurs incrémentaux de position. Nous présentons ensuite les différents capteurs équipant le robot, dont les équations modélisant le processus de mesure seront utilisées dans les différents algorithmes développés dans la suite de ce travail. Pour finir, le choix de l'architecture de contrôle du robot utilisée pour la réalisation du prototype est présenté.

## 2.1 Architecture générale

Il est possible de décomposer le robot mobile en trois blocs fonctionnels différents. Tout d'abord le robot mobile doit pouvoir se déplacer afin d'effectuer ses tâches. Il a donc besoin d'actionneurs, qui dans le cas présent, sont des moteurs sans charbon (BLDC). Ensuite, le robot doit s'adapter à son environnement. Il a donc besoin de capteurs lui permettant d'obtenir des informations sur le monde qui l'entoure, mais aussi sur son état interne. Enfin, pour exploiter les données des capteurs et décider des actions à exécuter, le robot a besoin d'un calculateur qui va lui permettre de mener à bien sa mission.

La description "haut niveau" de l'organisation du robot avec le détail des différents blocs est schématisée par la Figure 2.1. Les principaux composants équipant le prototype développé au cours du projet y sont détaillés, ainsi que les relations et communications entre les différents éléments. Le système de locomotion du robot repose sur l'utilisation de roues. Plus précisément, le robot est basé sur le principe de la conduite différentielle, qui consiste à utiliser deux roues motrices actionnées de manière indépendante. Généralement, une ou deux roues additionnelles non motorisées sont ajoutées pour assurer l'équilibre et la stabilité de l'ensemble. En contrôlant la vitesse et le sens de rotation des deux moteurs associés aux roues, il est alors possible de déplacer le robot. Ce mode de locomotion a pour avantage d'être simple à mettre en œuvre et donc à contrôler. De plus, sur des surfaces planes, ce mode de locomotion possède un bon rendement énergétique, mais qui se dégrade assez vite dès lors que le terrain devient plus chaotique

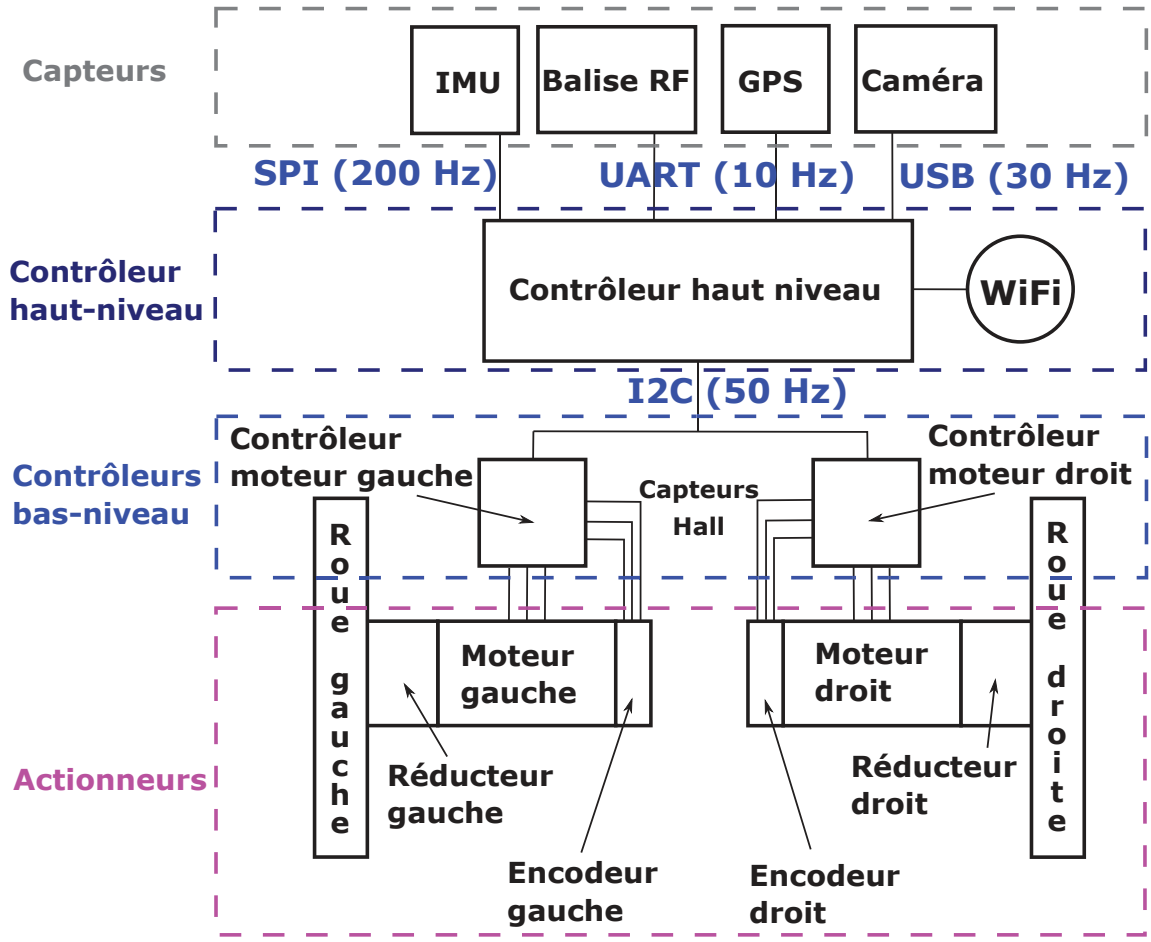



FIGURE 2.1 – Architecture du robot avec le détail des différents composants et communications

comme dans certains environnements extérieurs. En effet, sur un terrain boueux ou générant de nombreux glissements, le robot va beaucoup plus solliciter les moteurs afin de contrecarrer les glissements et de maintenir sa vitesse.

Sur la Figure 2.1, les actionneurs sont les moteurs des roues droite et gauche. Deux contrôleurs moteurs sont utilisés pour réaliser l'asservissement de la vitesse de rotation des roues. Ces contrôleurs moteurs sont construits à partir d'un onduleur de tension composé de trois bras de ponts (un par phase), ainsi que d'un microcontrôleur. À partir des consignes de vitesse, le microcontrôleur réalise l'asservissement et commande les demi-ponts en envoyant des signaux à largeur d'impulsion modulée (PWM). L'intérêt d'avoir des contrôleurs bas-niveau pour commander directement les moteurs est de mieux isoler la partie signal (capteurs, contrôleur haut niveau) de la partie puissance (batterie, moteurs) et ainsi de réduire les interférences. Ces contrôleurs moteurs communiquent via le protocole I2C avec un contrôleur haut niveau à une fréquence de 50 Hz. Les principales informations échangées sont les consignes de vitesse qui sont envoyées par le contrôleur haut niveau vers les deux contrôleurs moteur et les vitesses mesurées par les contrôleurs qui sont retournées au contrôleur haut niveau.

Le contrôleur haut-niveau est basé sur un ordinateur mono-carte miniature de type Raspberry Pi 3 [1], visible à la Figure 2.2, tournant avec un système d'exploitation Linux. L'utilisation d'un petit ordinateur comme contrôleur haut-niveau permet d'obtenir une grande flexibilité tant au niveau logiciel que matériel. Concernant le matériel, le Raspberry Pi 3 permet l'utilisation de différents moyens de communication filaires (SPI, I2C, UART, USB, ...) et sans fil (Wi-Fi,

1. https://www.raspberrypi.org

FIGURE 2.2 – Raspberry Pi 3 (https://www.raspberrypi.org)

Bluetooth). Ainsi, en configurant le Raspberry Pi 3 comme un point d'accès Wi-Fi, il est possible de se connecter via SSH et de contrôler l'ordinateur à distance. Ce mode de connexion facilite le contrôle du robot et permet d'obtenir des informations sur son état plus facilement. Le Raspberry Pi permet également la possibilité d'installer un noyau temps-réel si cela s'avère nécessaire pour l'application. Concernant la partie logicielle, l'utilisation d'un système Linux permet de faciliter le développement grâce un grand choix de langages de programmation et de bibliothèques disponibles. Parmi les nombreux outils accessibles avec Linux, il existe notamment un système d'opération dédié aux robots, appelé ROS [2] [23], et qui a été utilisé dans le cadre de ce projet. Cet ensemble d'outils au code source ouvert, devenu très populaire dans la communauté robotique, permet de faciliter le développement de programmes et leurs interactions. ROS intègre toute une série de fonctionnalités comme une infrastructure de communication inter-processus et inter-machine avec des messages standardisés, la possibilité d'enregistrer et de rejouer les messages, des outils de visualisation, des algorithmes de localisation, SLAM, navigation, ... Ainsi, afin de faciliter le développement du prototype et de bénéficier de ses nombreux avantages, le système ROS a donc été installé sur le Raspberry Pi 3, et l'ensemble des programmes correspondant à la gestion des capteurs et des actionneurs ont été réalisé à partir de cet environnement ROS.

Autour du contrôleur haut niveau, on retrouve différents capteurs. Tout d'abord, le système dispose d'une centrale inertielle (IMU) qui est composée d'un accéléromètre, d'un gyroscope et d'un magnétomètre et qui retourne l'accélération linéaire, la vitesse angulaire et le champ magnétique au contrôleur haut niveau à une fréquence de 200 Hz via une communication SPI. On retrouve aussi le capteur sur lequel est axé ce travail, qui est une balise radiofréquence (RF) permettant d'obtenir des mesures de distance par rapport à d'autres balises disposées dans la zone d'opération du robot. La balise du robot est reliée au contrôleur haut niveau par une communication série (UART) qui lui permet de faire remonter les informations de distance avec les identifiants des balises correspondantes à une fréquence de 10 Hz. Enfin, le robot est équipé d'une caméra, qui n'est utilisée ici que pour obtenir un retour visuel et qui ne sera pas utilisée pour la commande. De plus, un GPS est aussi monté sur le robot afin d'obtenir la position géographique du robot. Le GPS est connecté par une liaison UART au contrôleur haut-niveau et envoie les informations de position à une fréquence comprise entre 1 et 10 Hz.

Après ce bref aperçu de la composition matérielle du robot, nous allons maintenant détailler les différents modèles des composants, en commençant par les équations de modélisation du robot mobile utilisé.

2. http://www.ros.org

## 2.2 Modélisation du robot

### 2.2.1 Modèle cinématique

On considère dans le cadre de ce travail un robot mobile à roues basé sur une conduite différentielle, où les deux roues motrices du robot sont actionnées séparément. Le comportement de ce moyen de locomotion est modélisé par les équations cinématiques du modèle unicycle, [118]. Il s'agit de l'un des modèles les plus simples de robot à roues. En contrôlant les vitesses de rotation des deux roues et notamment en jouant sur la différence de vitesse entre la roue droite et la roue gauche, il est possible de déplacer le robot dans différentes directions : vers l'avant, vers l'arrière, à droite, et à gauche. Il est même possible de réaliser un demi-tour sur place. Un schéma illustratif des mouvements du robot et des vitesses associées est donné à la Figure 2.3. Il faut noter que le robot à conduite différentielle est un système sous-actionné. Ceci implique que le robot ne peut effectuer certains mouvements instantanément. En effet, le système est commandé par deux entrées (les vitesses de rotation des roues), alors que l'espace des configurations (degrés de liberté) du robot est de dimension trois en 2D (la position suivant les axes $x$ et $y$ du plan et la rotation $\theta$ dans ce plan). De manière plus intuitive, on s'aperçoit facilement que le robot n'est pas capable de se déplacer instantanément sur le côté. Comme dans le cas d'une voiture faisant un créneau, il est nécessaire d'opérer une certaine manœuvre (trajectoire) pour atteindre la position souhaitée. La contrainte sur les vitesses que le robot ne peut se translater dans la direction parallèle à l'axe des roues étant non-intégrable, le système unicycle est dit non-holonome [19], [105].

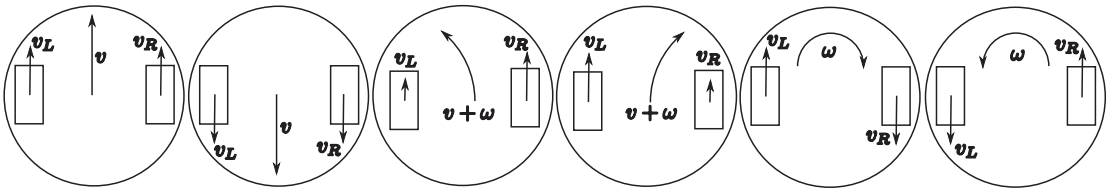



FIGURE 2.3 – Schémas de différents déplacements possibles avec un robot mobile à roues de type unicycle

Nous allons maintenant détailler comment la position, l'orientation et les vitesses (translation et rotation) du robot sont reliées aux vitesses de rotation des roues. Définissons tout d'abord les variables permettant de modéliser le comportement du robot. La Figure 2.4 illustre le modèle unicycle du robot ainsi que les différents paramètres utilisés. L'ensemble des paramètres sont aussi regroupés dans le Tableau 2.1. On considère le cas d'un robot évoluant dans un plan (environnement 2D) muni d'un repère global $\mathcal{G} = (0, \vec{g}_x, \vec{g}_y)$. Dans ce repère Cartésien, le robot est représenté par sa position, donnée par les coordonnées $(x, y)$, et par son orientation dans le plan, $\theta$. L'orientation est définie par rapport à l'axe $\vec{g}_x$ du repère global, comme le montre la Figure 2.4. L'ensemble des paramètres de position et d'orientation définit la pose du robot, notée :

$$\mathbf{p} = (x, y, \theta)^T \tag{2.1}$$

En supposant que les roues roulent sans glissement, il est maintenant possible de relier la vitesse de rotation des roues droite et gauche définies respectivement par $\dot{\varphi}_R$ et $\dot{\varphi}_L$ aux vitesses linéaire et angulaire du robot. Pour cela, on introduit les vitesses linéaires des roues droite et gauche, dont les expressions sont données par :

$$\begin{cases} v_R = r\dot{\varphi}_R \\ v_L = r\dot{\varphi}_L \end{cases}$$

| Symboles | Unités | Paramètres |
|---|---|---|
| $\dot{\varphi}_R$, $\dot{\varphi}_L$ | rad/s | Vitesses angulaires des roues droite et gauche |
| $v_R$, $v_L$ | m/s | Vitesses linéaires des roues droite et gauche |
| $(x, y)$ | m | Position du robot dans le plan |
| $\theta$ | rad | Orientation du robot dans le plan |
| $(\dot{x}, \dot{y})$ | m | Vitesses du robot dans le plan |
| $v$ | m/s | Vitesse linéaire du robot |
| $\omega$ | rad/s | Vitesse angulaire du robot |
| $r$ | m | Rayon des roues droite et gauche |
| $b$ | m | Distance entre les deux points de contact au sol des roues droite et gauche |
| $\rho$ | m | Rayon de courbure de la trajectoire circulaire du robot |

Tableau 2.1 – Paramètres du modèle du robot unicycle

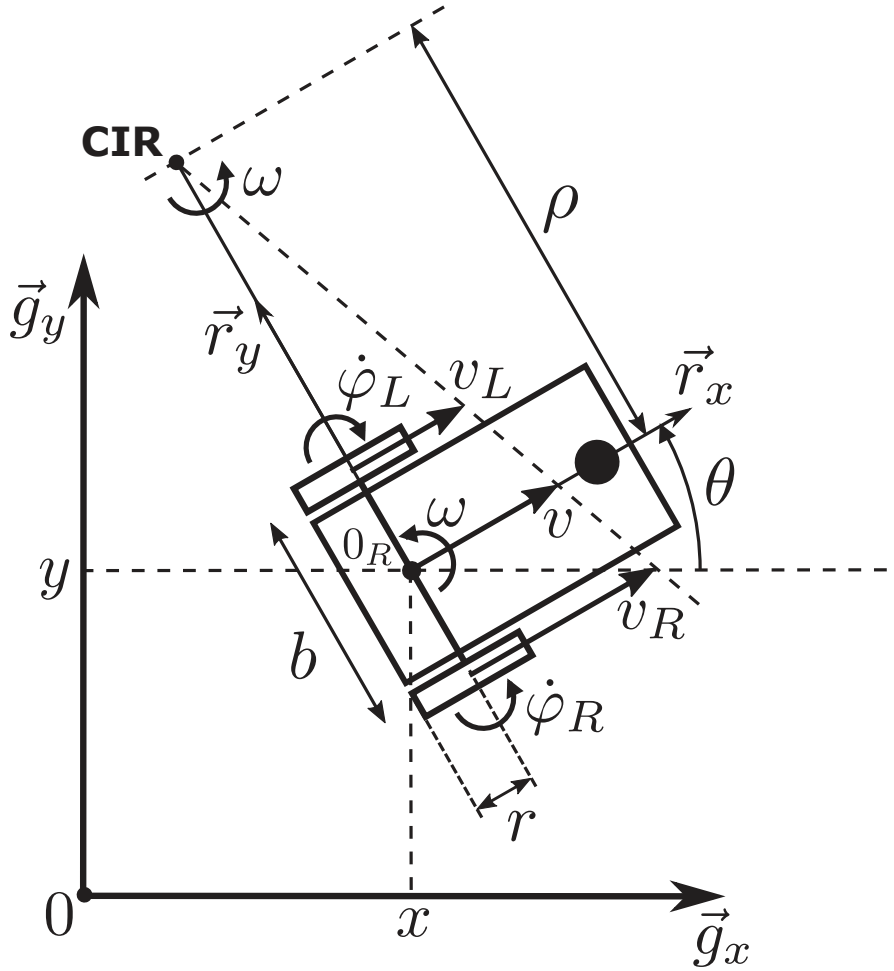



Figure 2.4 – Modélisation du robot unicycle (inspiré de la figure 2.6 dans http://eavr.u-strasbg.fr/~bernard/education/3a_robmob/3a_robmob_poly.pdf)

où $r$ est le rayon des roues (supposé le même pour les deux roues, mais la modélisation s'étend facilement à des rayons différents). Les vitesses de rotation $\dot{\varphi}_R$, $\dot{\varphi}_L$ sont exprimées en rad/s, alors que les vitesses linéaires $v_R$ et $v_L$ sont exprimées en m/s.

Les deux roues motrices ayant le même axe de rotation, le centre instantané de rotation (CIR) du robot est un point de cet axe, voir Figure 2.4. Si l'on définit $\rho$ comme le rayon de

courbure de la trajectoire du robot, c'est-à-dire la distance entre le CIR et le centre du robot $0_R$, $\omega$ comme la vitesse de rotation du robot autour du CIR, et $b$ comme la distance entre les deux points de contact des roues avec le sol. Alors les vitesses des roues droite et gauche s'écrivent aussi :

$$\begin{cases} v_R = r\dot{\varphi}_R = (\rho + \frac{b}{2})\omega & (1) \\ v_L = r\dot{\varphi}_L = (\rho - \frac{b}{2})\omega & (2) \end{cases}$$

En faisant $(1) + (2)$ et $(1) - (2)$, on obtient :

$$\begin{cases} (1) + (2) \Leftrightarrow r(\dot{\varphi}_R + \dot{\varphi}_L) = 2\rho\omega \\ (1) - (2) \Leftrightarrow r(\dot{\varphi}_R - \dot{\varphi}_L) = b\omega \end{cases}$$

De plus, comme $v = \rho\omega$, la première équation du système précédent peut se réécrire :

$$r(\dot{\varphi}_R + \dot{\varphi}_L) = 2v$$

On obtient alors les vitesses linéaire (m/s) et angulaire (rad/s) du robot grâce aux équations :

$$\begin{cases} v = r\frac{\dot{\varphi}_R + \dot{\varphi}_L}{2} = \frac{v_R + v_L}{2} \\ \omega = r\frac{\dot{\varphi}_R - \dot{\varphi}_L}{b} = \frac{v_R - v_L}{b} \end{cases} \tag{2.2}$$

Ces deux dernières équations représentent ce qu'on appelle le modèle cinématique inverse du robot.

En inversant ces relations, on obtient alors les équations du modèle cinématique direct. Les équations permettant de passer des vitesses linéaire et angulaire du robot aux vitesses linéaires des roues sont donc données par :

$$\begin{cases} v_R = v + \frac{b\omega}{2} \\ v_L = v - \frac{b\omega}{2} \end{cases} \tag{2.3}$$

Pour arriver à la vitesse de rotation des roues (rad/s), on divise simplement par $r$ :

$$\begin{cases} \dot{\varphi}_R = \frac{v}{r} + \frac{b\omega}{2r} \\ \dot{\varphi}_L = \frac{v}{r} - \frac{b\omega}{2r} \end{cases} \tag{2.4}$$

Il reste maintenant à voir les équations cinématiques du robot, c'est-à-dire le lien entre les vitesses linéaire et angulaire du robot et la dérivée de la pose du robot. Ceci permet de comprendre comment les vitesses linéaire et angulaire induisent le changement de pose du robot. On définit pour cela un deuxième repère que l'on attache au robot : $\mathcal{R} = (0_R, \vec{r}_x, \vec{r}_y)$. Plus précisément on considère que l'origine du repère se trouve au centre de l'axe des roues motrices du robot. Si l'on considère que la vitesse linéaire du robot dans son repère est donnée par $\mathbf{v} = ||\mathbf{v}||\vec{r}_x$ et que la vitesse angulaire du robot dans son repère est donnée par $\omega = ||\omega||\vec{r}_z$, avec $\vec{r}_z$ l'axe perpendiculaire au plan, alors les dérivées temporelles de la pose, $\dot{\mathbf{p}} = (\dot{x}, \dot{y}, \dot{\theta})^T$, sont données par :

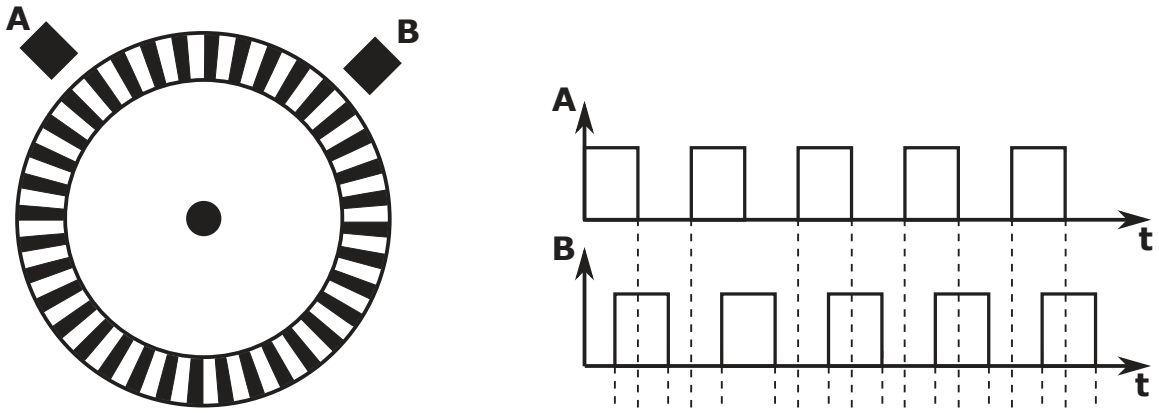
$$\begin{cases} \dot{x} = v\cos\theta \\ \dot{y} = v\sin\theta \\ \dot{\theta} = \omega \end{cases} \tag{2.5}$$

Ces équations cinématiques du robot, en plus d'être essentielles pour modéliser le comportement du robot, seront utilisées par les différents algorithmes de localisation développés par la suite. Elles permettront notamment de prédire l'évolution de la pose du robot à partir des vitesses de rotation des roues.

Les moteurs de roue du robot étant équipés de codeurs incrémentaux afin de réaliser l'asservissement en vitesse, il est aussi possible d'utiliser ces mesures pour estimer le déplacement du robot. Nous allons ainsi maintenant voir comment à partir de la mesure des vitesses de rotation des roues du robot et des équations données dans (2.5), il est possible d'estimer la pose du robot mobile.

### 2.2.2 Odométrie des roues

Dans la majorité des cas en robotique mobile, les moteurs de roue sont équipés de codeurs incrémentaux. Ces capteurs ont pour rôle de mesurer la vitesse de rotation de l'axe du moteur et éventuellement la position du rotor, [13], [118]. Les codeurs incrémentaux sont des capteurs proprioceptifs, c'est-à-dire qu'ils donnent une information sur l'état interne du robot. Parmi les codeurs incrémentaux, on distingue les capteurs qui donnent une mesure relative du déplacement, des capteurs qui donnent une mesure absolue. Dans notre cas, pour estimer la vitesse, une mesure relative est suffisante. Différentes technologies de codeurs incrémentaux existent, comme les capteurs magnétiques ou les capteurs optiques, mais le principe de fonctionnement de base reste le même. Ce dernier étant de compter un nombre d'impulsions qui est proportionnel à la vitesse de rotation du moteur.




(a) Disque et capteurs optiques A et B (b) Diagramme des signaux A et B

FIGURE 2.5 – Exemple d'un disque et des capteurs optiques A et B pour un encodeur rotatif (a) avec les signaux correspondants (b)

Par exemple, dans le cas d'un encodeur optique, un disque composé d'une alternance de franges blanches et noires est monté sur l'axe du moteur. Deux capteurs optiques, appelés canaux A et B sont placés face à ce disque, voir Figure 2.5. Les signaux des deux canaux sont déphasés de 90 degrés et correspondent aux transitions entre les franges blanches et noires détectées par les deux capteurs A et B. La Figure 2.5a montre les signaux typiquement obtenus en sortie des deux canaux. En comptant le nombre de transitions et en analysant les transitions des deux canaux, il est possible de compter le déplacement angulaire avec un précision de $\frac{\pi}{(4N}$ où $N$ est le nombre de points par tour, et d'en déduire la vitesse ainsi que le sens de rotation. Le nombre de points par tour, $N$, correspond à la résolution du capteur. Plus cette dernière sera élevée, meilleure sera l'estimation de la position et de la vitesse.

Pour déterminer la vitesse de rotation de la roue attachée au moteur à partir d'un codeur incrémental, il existe différentes techniques. La solution la plus simple consiste à compter le nombre d'impulsions pendant une période d'échantillonnage donnée. En multipliant le nombre d'impulsions par un facteur $f$ (rad), on obtient alors l'incrément de position de la roue. Ce

facteur $f$ est donné par, [13] :

$$f = \frac{\pi}{gN}$$

et dépend de :

— $g$, le rapport de réduction du réducteur entre la roue et le moteur,

— $N$, le nombre d'impulsions par tour du moteur.

Pour obtenir la vitesse de rotation, il suffit alors de diviser l'incrément de position de la roue par la période d'échantillonnage. Si l'on note $n$ le nombre d'impulsions comptées durant la période d'échantillonnage $\Delta T$ (en seconde), alors la vitesse de rotation de la roue (en rad/s) est donnée par :

$$\dot{\varphi} = \frac{nf}{\Delta T}$$

Les mesures étant réalisées de manière discrète, on utilisera plus souvent l'incrément de distance parcourue par les roues droite et gauche pendant l'intervalle $\Delta T$. Ainsi, si l'on suppose les vitesses des roues constantes sur l'intervalle $\Delta T$, on obtient :

$$\begin{cases} \delta U_R = r_R f n_R \\ \delta U_L = r_L f n_L \end{cases}$$

avec $\delta U_{R/L}$ l'incrément de distance, $r_{R/L}$ le rayon et $n_{R/L}$ le nombre d'impulsions des roues droite (R) et gauche (L) respectivement.

En connaissant les incréments de distance sur les deux roues, il est possible de retrouver les incréments de distance et d'angle du robot. En utilisant les équations données par (2.2) dans leurs formes discrètes, on obtient :

$$\begin{cases} \delta U = \frac{(\delta U_R + \delta U_L)}{2} \\ \delta \Theta = \frac{(\Delta U_R - \delta U_L)}{b} \end{cases} \tag{2.6}$$

avec $\delta U$ l'incrément de distance linéaire, et $\delta\Theta$ l'incrément d'angle du robot.

Le principe de l'odométrie pour un robot de type unicycle consiste donc à intégrer les mesures des déplacements incrémentaux des roues pour estimer la pose du robot. En appliquant ce principe à chaque nouvelle mesure, il est possible de reconstruire les poses successives du robot. Pour dériver les équations d'odométrie du robot et donc pouvoir estimer la pose actuelle du robot à partir de la pose précédente et des mesures des codeurs incrémentaux, il faut reprendre les équations (2.5). Un développement de Taylor à l'ordre 1 permet d'obtenir :

$$\begin{cases} x_{k+1} = x_k + \delta U \cos\theta_k \\ y_{k+1} = y_k + \delta U \sin\theta_k \\ \theta_{k+1} = \theta_k + \delta\Theta \end{cases}$$

avec $\mathbf{p}_{k+1} = (x_{k+1}, y_{k+1}, \theta_{k+1})^T$ et $\mathbf{p}_k = (x_k, y_k, \theta_k)^T$ les poses du robot aux instants $t_{k+1} = (k+1)\Delta T$ et $t_k = k\Delta T$.

Pour une meilleure précision, il est possible de réaliser un développement de Taylor à l'ordre 2, [105], [118] :

$$\begin{cases} x_{k+1} = x_k + \delta U \cos(\theta_k + \frac{\delta\Theta}{2}) \\ y_{k+1} = y_k + \delta U \sin(\theta_k + \frac{\delta\Theta}{2}) \\ \theta_{k+1} = \theta_k + \delta\Theta \end{cases} \tag{2.7}$$

Ces dernières équations peuvent aussi se retrouver géométriquement, comme le montre la Figure 2.6, en supposant que pendant la période d'intégration, on peut assimiler le déplacement du robot à une trajectoire circulaire. Ainsi, à partir des équations (2.7), on dispose déjà d'un pre-

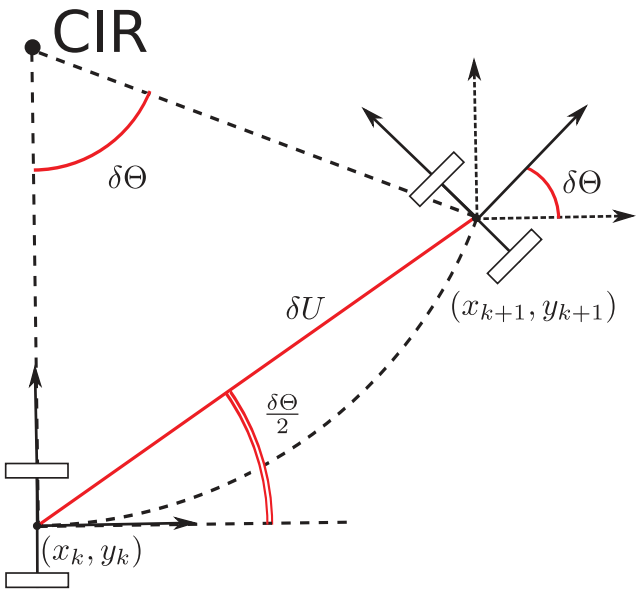



FIGURE 2.6 – Modélisation de l'odométrie du robot permettant d'estimer la position du robot à partir des mesures des codeurs incrémentaux des roues (CIR = centre instantané de rotation)

mier algorithme basique permettant de localiser le robot, si l'on suppose que la pose de départ est connue. Il faut toutefois noter que cette méthode ne donne une information de positionnement fiable que sur une très courte période. En effet, les mesures des codeurs incrémentaux sont entachées d'erreurs, qui sont directement intégrées par les équations (2.7). En intégrant au fur et à mesure ces erreurs, même infimes, la pose estimée du robot va très rapidement s'éloigner de la pose réelle. L'augmentation de la résolution des codeurs incrémentaux peut améliorer l'estimation, mais ne pourra pas éviter les écarts liés à des perturbations extérieures comme le glissement des roues sur le sol. En effet, un sol plat et sans aspérités favorisant le roulement du robot permet d'obtenir de très bons résultats. Au contraire, un terrain bosselé ou boueux, va entraîner des glissements et des pertes de contact avec le sol. Les roues du robot tournant alors dans le vide, les mesures des codeurs incrémentaux vont engendrer une surestimation de la vitesse des roues et donc du déplacement du robot.

Enfin, la précision de l'odométrie dépend aussi de la précision des paramètres géométriques $r$ et $b$ du modèle cinématique. Pendant tout le développement précédent, on a supposé que ces paramètres étaient parfaitement connus. Bien souvent ces paramètres sont disponibles avec une assez bonne précision si l'on dispose du modèle 3D du robot dessiné pendant sa conception. Cependant, dans certains cas, il est possible que ces paramètres ne soient pas connus ou difficilement mesurables. Ainsi, il se peut qu'après l'étape de production, les deux roues ne se retrouvent pas exactement avec le même rayon. En outre, l'usure inévitable peut être différente d'une roue à l'autre. En fonction du niveau de précision voulu, il peut donc être intéressant, voir nécessaire de prendre en compte ces imperfections. Dans ce cas-là, en plus d'estimer la pose du robot, il faut aussi estimer les valeurs des paramètres que l'on souhaite retrouver avec précision. Ainsi, dans les différents algorithmes proposés par la suite, ces paramètres ne devront plus être considérés comme fixes, mais bien comme des variables à estimer.

Les modèles géométrique et cinématique ayant été présentés, nous allons maintenant détailler les autres capteurs embarqués sur le robot, qui lui permettront de se localiser pleinement en corrigeant l'odométrie.

## 2.3 Modélisation des capteurs

L'odométrie permettant d'estimer la pose du robot à partir des seules mesures de la vitesse des roues, que nous avons vu précédemment, est entachée d'une erreur croissante au cours du temps. Afin de remédier à ce problème, deux approches complémentaires permettent d'améliorer la précision de localisation du robot. La première consiste à ajouter des capteurs proprioceptifs comme une centrale inertielle, de façon à améliorer les performances de l'odométrie en fusionnant de multiples sources d'information complémentaires. La seconde approche consiste à prendre en compte des informations extérieures au robot, comme la mesure de position absolue via un récepteur GPS, ou des mesures de distance par rapport aux objets environnant le robot. Communément en robotique mobile, les capteurs extéroceptifs sont présents sous la forme de caméras ou de Lidar (LIght Detection And Ranging). Pour des raisons de coût, l'utilisation d'un télémètre laser n'a pu être possible. De plus, le système développé devant pouvoir opérer de nuit, l'utilisation de caméras devenait fortement compromise. Pour ces raisons, dans ce mémoire nous considérons que les seules mesures extéroceptives disponibles proviennent d'un capteur permettant de mesurer des distances par rapport à un certain nombre de balises présentes dans l'environnement du robot. Dans ces travaux, un GPS est monté sur le robot, mais celui-ci n'est utilisé que comme un capteur secondaire pouvant servir éventuellement si les conditions le permettent. De plus, ce dernier était envisagé pour fournir une vérité terrain et comparer les différents algorithmes développés dans la suite. Cependant, la précision obtenue durant les expériences n'a pas été suffisante pour que le GPS puisse jouer son rôle de vérité terrain.

Dans le cas d'un robot évoluant sur une surface 3D non plane, c'est la fusion des données des différents capteurs précédents (codeurs incrémentaux, centrale inertielle, balises), qui permettra une bonne estimation de la pose et de la vitesse du robot.

Un capteur étant un dispositif mesurant une quantité physique, une étape essentielle dans la réalisation des différents algorithmes consiste à modéliser le principe de fonctionnement de la mesure. La précision du modèle du capteur peut influencer la précision des algorithmes, et la plupart du temps il y a un compromis à faire entre la précision et la complexité du modèle utilisé. Nous détaillons maintenant les modèles des différents capteurs utilisés au cours de ces travaux. Nous présentons dans un premier temps les balises RF, puis les composants de la centrale inertielle et enfin le GPS.

### 2.3.1 Balises RF

Les capteurs permettant de mesurer des distances sont pour la plupart basés sur le principe du calcul du temps de vol d'une onde [50]. Dans le cas des capteurs ultrasons, il s'agit d'une onde sonore d'environ 40 kHz qui est émise. Le principe repose sur un émetteur qui génère un train d'ondes, alors qu'un récepteur permet de détecter un éventuel écho si un obstacle est présent pour renvoyer l'onde vers le capteur. Dans le cas des balises radiofréquences (RF), plusieurs méthodes existent. La première consiste à calculer la distance à partir de la puissance reçue qui est d'après la formule de Friis [47] inversement proportionnelle au carré de la distance entre les deux antennes des balises. Ce principe est aussi connu sous le nom de RSSI (Received Signal Strength Indicator), qui est souvent utilisé par les réseaux Wi-Fi pour avoir une estimation de la qualité de la connexion ou pour localiser les personnes se connectant à un réseau. L'inconvénient majeur de cette méthode est la faible précision obtenue, qui est généralement de l'ordre de quelques mètres.

Une autre technique consiste à mesurer le temps de vol de l'onde radio entre un récepteur et un émetteur. La principale difficulté, réside ici, dans l'obtention d'une mesure précise du temps de vol. En effet, l'onde radio se propage à la vitesse de la lumière, qui est d'environ $300 \times 10^6$

m/s, et parcourt donc environ 30 cm en une nanoseconde. Autrement dit, une erreur d'une nanoseconde dans la mesure du temps de vol engendre donc une erreur d'environ 30 cm sur la distance retournée. Pour le bon fonctionnement de la méthode, il est donc fondamental de posséder des composants électroniques capables de mesurer avec une telle précision. De plus, le choix de la fréquence joue aussi un rôle important sur les performances du système. Depuis quelques années, les recherches sur l'utilisation des fréquences UWB, [116], [56], [107], pour réaliser des mesures de distance ont permis l'arrivée sur le marché de composants dotés de performances intéressantes pour la robotique mobile, en terme de précision notamment. Ainsi, la technologie de mesure de distance basée sur les ondes radios UWB est considérée comme la plus prometteuse dans le domaine des systèmes de positionnement et notamment en intérieur. Parmi les principales entreprises développant cette technologie, on peut citer : Nanotron [3], Time Domain [4], BeSpoon [5], et Decawave [6]. Un signal radio UWB est défini par une bande passante fractionnelle plus grande que 0.2 ou par une bande passante absolue d'au moins 500 MHz [115]. La bande passante absolue $B$ étant calculée comme la différence entre la fréquence haute $f_H$ du point d'émission à -10 dB et la fréquence basse $f_L$ du point d'émission à -10 dB :

$$B = f_H - f_L$$

Alors que la bande passante fractionnelle est définie par :

$$B_{\text{frac}} = \frac{B}{f_c} = \frac{2(f_H - f_L)}{f_H + f_L}$$

avec $f_c$ la fréquence centrale donnée par :

$$f_c = \frac{f_H + f_L}{2}$$

Les principaux avantages des dispositifs de mesure de distance par ondes radios UWB sont :

— Une bonne précision de mesure grâce à une résolution temporelle très fine,

— Pas de fausses détections de mesure : une balise ne fournira jamais une mesure qui n'a pas été envoyée,

— Le problème d'association d'une mesure à une balise est facilement résolu grâce à l'envoi des identifiants des balises dans les messages échangés lors du processus de mesure,

— Une haute pénétration des ondes UWB dans les matériaux, ce qui permet d'obtenir des mesures même à travers la plupart des obstacles,

— Un coût relativement bas de la technologie,

— Une faible consommation d'énergie.

Par contre, parmi les principaux inconvénients on peut citer :

— Les phénomènes de chemins multiples (signal radio atteignant l'antenne par différents chemins) et les obstacles traversés par l'onde radio (NLOS) peuvent grandement détériorer la précision des mesures de distance,

— Capteur actif nécessitant une source d'alimentation (batterie),

— L'installation à priori des balises sur la zone à couvrir.

Pour la réalisation des balises utilisées par la suite, le module DWM1000 de Decawave, visible sur la Figure 2.7, a été choisi. Le principe de la méthode est illustré à la Figure 2.8 qui a été reprise de [25]. On considère deux balises, l'une appelée ancre (balise fixe) et l'autre

3. http://nanotron.com
4. http://www.timedomain.com
5. http://bespoon.com
6. http://www.decawave.com


FIGURE 2.7 – Module DWM1000 de Decawave (http://www.decawave.com) utilisé pour obtenir les mesures de distance

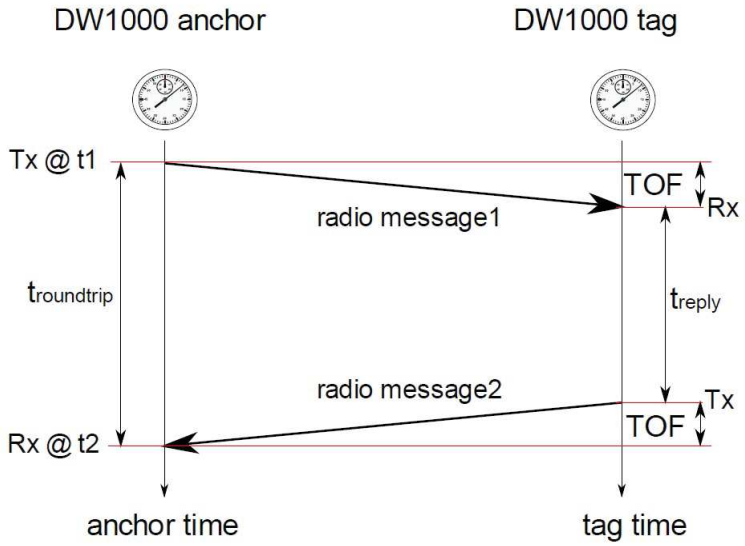



FIGURE 2.8 – Principe de la mesure de distance par mesure du temps de vol aller-retour de l'onde radio (schéma repris de [25])

appelée tag (balise mobile). Dans un premier temps, l'ancre transmet un message radio au tag et enregistre le temps correspondant au début de la transmission, noté $t_1$. Le tag reçoit le message et après un temps donné, noté $t_r$, répond à l'ancre. Finalement, l'ancre réceptionne la réponse du tag et enregistre le temps d'arrivée du message, noté $t_2$. À partir des temps précédents, il est maintenant possible de connaître le temps de vol (ToF) de l'onde, noté $t$, et qui est donné par :

$$t = \frac{(t_2 - t_1) - t_r}{2}$$

La distance $d$ entre l'ancre et le tag est alors simplement obtenue en multipliant par la vitesse de l'onde radio $c$ :

$$d = ct = c\frac{(t_2 - t_1) - t_r}{2} \tag{2.8}$$

Comme cela est expliqué dans [25], le processus de mesure par aller-retour est entaché d'erreurs

causées par les dérives de l'horloge et de la fréquence. Afin de réduire ces erreurs, la solution implémentée par l'entreprise DecaWave consiste à calculer la distance en utilisant les mesures aller-retour de l'ancre vers le tag et les mesures aller-retour du tag vers l'ancre. Cette solution rajoute donc un autre échange entre le tag et l'ancre pour la réalisation du calcul. La mesure de distance prend donc deux fois plus de temps, mais la précision s'en trouve améliorée.

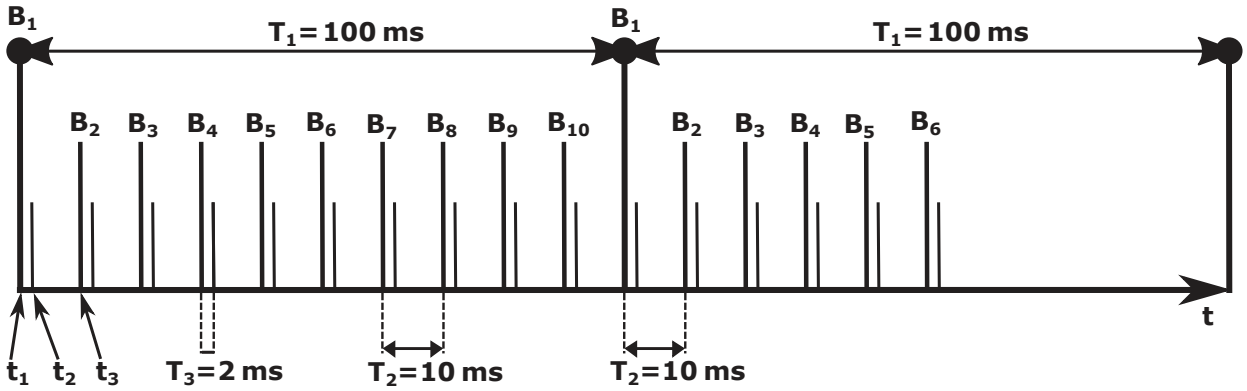



FIGURE 2.9 – Principe de synchronisation et chronologie des mesures de distance par rapport aux balises

Afin d'obtenir une mesure de distance par rapport à chaque balise dans un temps donné et d'éviter les interférences, il est nécessaire d'organiser le déroulement de la prise de mesures. Pour cela, un mécanisme de synchronisation permettant de récupérer les mesures des balises les unes après les autres, tout en respectant une période donnée, est mis en place. La chronologie montrée à la Figure 2.9 présente la méthode appliquée en pratique. On notera tout d'abord que le temps pour effectuer une mesure de distance est de l'ordre de $T_3 \sim 2$ ms. Compte tenu de la vitesse du robot, du nombre de balises et de la précision des balises notamment, la période pendant laquelle on souhaite avoir une mesure de distance par rapport à chaque balise a été fixée à $T_1 = 100$ ms. Sachant qu'une mesure de distance prend environ 2 ms, et en ajoutant une marge d'erreur de 8 ms, la période pour réaliser une mesure de distance par rapport à une balise est fixée à $T_2 = 10$ ms. Avec ce découpage, on peut donc obtenir 10 mesures de distance provenant de balises différentes 10 fois par secondes. Il est bien sûr possible de réduire ou d'augmenter la marge d'erreur afin d'obtenir plus ou moins de mesures par rapport à différentes balises. Cependant, en pratique, une dizaine de balises est généralement suffisant, et la diminution de la marge d'erreur entraîne inévitablement plus d'erreurs de synchronisation et donc plus de mesures manquantes.

Ainsi, à l'instant $t_1$ de la Figure 2.9 qui représente le début d'une nouvelle période, la balise montée sur le robot attend un message de la balise $B_1$ afin de réaliser la mesure. À $t_2 \sim 2$ ms, le processus de mesure de distance est fini et la balise du robot envoie directement la mesure au contrôleur haut niveau. À l'instant $t_3 = 10$ ms, la balise du robot attend maintenant un message de la balise $B_2$ afin d'initier un nouveau processus de mesure. Ceci est alors répété pour chaque balise du terrain $B_i$, $i \in [1, N]$ ($N \leq 10$). Après $t_1 + T_1$, une nouvelle période recommence avec l'attente d'un message de la balise $B_1$.

Concernant, l'utilisation des mesures de distance par les algorithmes développés par la suite, deux options sont envisageables. La première consiste à utiliser les mesures dès qu'elles sont disponibles, c'est-à-dire tous les $T_2 = 10$ ms. On traite donc chaque mesure de distance séparément. Cette solution est, par exemple, très bien adaptée à un filtre de Kalman où les mesures peuvent être fusionnées dès leur réception. La deuxième option consiste à utiliser les mesures tous les $T_1 = 100$ ms, afin de traiter toutes les mesures de distance par rapport à chaque balise en même temps. Cette solution introduit une erreur en cas de mouvement du robot pendant

la période $T_1$. Cependant, en pratique, au vu de la vitesse du robot qui est inférieure à 1 m/s, le déplacement du robot peut être négligé entre l'instant de mesure de la première balise et l'instant de mesure de la dernière balise. De plus, pour certains algorithmes comme la trilatération, il est nécessaire de posséder un certain nombre de mesures de distance par rapport à des balises différentes à un instant donné pour obtenir une estimation de position. Ainsi, dans la majorité des algorithmes développés par la suite, les mesures de distances seront utilisées à une fréquence de 10 Hz, et les décalages temporels entre la réalisation des mesures au sein d'une période seront négligées.

Maintenant que le principe de la mesure de distance a été exposé, nous allons voir comment il peut être modélisé par une équation de mesure.

### Modèle des mesures de distance des balises

Pour modéliser le processus de mesure de distance, on considère deux balises $i$ et $j$ dont les positions dans un environnement 3D sont données par : $\mathbf{p}_i = (x_i, y_i, z_i)^T$ et $\mathbf{p}_j = (x_j, y_j, z_j)^T$. Le modèle de mesure de distance entre la balise $i$ et la balise $j$ est alors donné par [42] :

$$r_{i,j} = s_{i,j} d_{i,j} + b_{i,j} + \epsilon_{i,j} \tag{2.9}$$

avec :

- $r_{i,j}$ : la distance mesurée par les balises,
- $s_{i,j}$ : le facteur d'échelle entre la distance mesurée et la distance réelle, peut éventuellement dépendre de la distance ($s_{i,j}(d_{i,j})$),
- $d_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}$ : la distance réelle entre les deux balises,
- $b_{i,j}$ : le biais de la mesure, peut éventuellement dépendre de la distance ($b_{i,j}(d_{i,j})$),
- $\epsilon_{i,j}$ : le bruit de mesure, modélisé par un bruit blanc Gaussien de moyenne nulle et de variance $\sigma^2_{i,j}$. La variance peut éventuellement dépendre de la distance : $\sigma^2_{i,j}(d_{i,j}) = (1 + \eta_{i,j} d_{i,j})^2 \sigma^2_{i,j,0}$ avec $\eta_{i,j}$ le paramètre de dépendance à la distance et $\sigma^2_{i,j,0}$ la variance initiale de la mesure [98].

Afin de modéliser la dépendance à la distance de la variance du bruit de mesure, la formulation présentée dans [98] a été utilisée. Il existe cependant d'autres variantes, dont notamment celles proposées dans [67] et [109]. Enfin, on notera que le processus de mesure de distance est possible dans les deux sens. Ainsi, si la balise $i$ peut mesurer sa distance par rapport à la balise $j$, il en est de même pour la balise $j$ vers la balise $i$. La seule différence entre les deux étant l'initiateur de la mesure. Par convention, on notera $r_{ij}$ une mesure de distance initiée par la balise $i$ par rapport à la balise $j$. Dans le cas des algorithmes de localisation, on s'intéressera principalement aux mesures de distance entre la balise équipant le robot mobile et les balises fixes de position connue disposées sur le terrain. Cependant, dans le cas où la position des balises n'est pas connue à l'avance, il peut aussi être bénéfique d'exploiter les mesures inter-balises (entre les balises fixes du terrain) afin d'améliorer et d'accélérer l'estimation de la position des balises et par la suite celle du robot.

### Modèle des mesures de distance des balises avec décalage

Dans le modèle précédent, il est implicitement supposé que la position de l'antenne de la balise mobile corresponde exactement à la position que l'on cherche à estimer. Dans le cas d'un robot mobile équipé d'une balise, il se peut, pour diverses raisons, que l'antenne de la balise du robot se retrouve assez éloignée de l'origine du repère du robot, qui dans le cas présent correspond au centre de l'axe des roues arrière du robot. On peut souhaiter prendre en compte ce décalage dans la modélisation afin d'améliorer la précision.

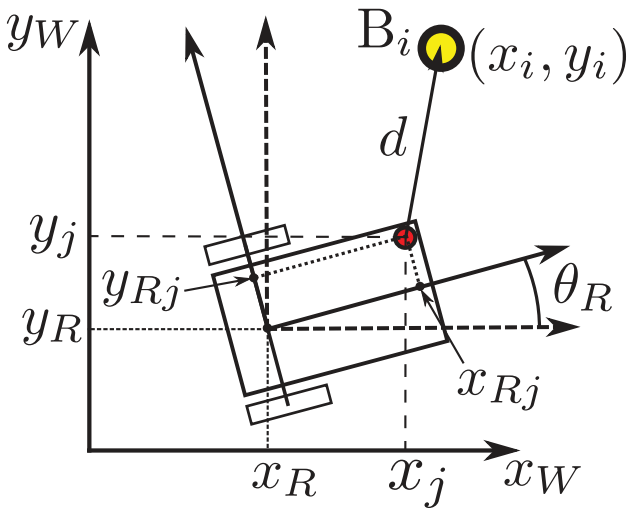



FIGURE 2.10 – Modélisation d'une mesure de distance robot/balise dans le cas où l'antenne de la balise du robot est décalée par rapport au centre du robot

Le modèle des mesures de distance avec décalage dans le cas 2D est représenté par le schéma de la Figure 2.10. L'équation de base donnée par l'équation (2.9) reste valable, seul le terme $d_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}$ change. Si l'on considère que l'indice $j$ correspond à la balise équipant le robot, les coordonnées $(x_j, y_j)$ de la balise doivent être remplacées par celle du robot. On obtient donc :

$$\begin{cases} x_j = x_R + x_{Rj}\cos\theta_R - y_{Rj}\sin\theta_R \\ y_j = y_R + x_{Rj}\sin\theta_R + y_{Rj}\cos\theta_R \end{cases}$$

avec $(x_R, y_R, \theta_R)$ la pose du robot à l'instant de la mesure, et $(x_{Rj}, y_{Rj})$ la position de l'antenne de la balise dans le repère du robot.

De plus, dans le cas 2D ou le robot évolue dans un environnement plan, il se peut que l'antenne de la balise du robot et les antennes des balises disposées sur le terrain ne soient pas à la même hauteur ($z_j \neq z_i$). Dans ce cas, on peut prendre en compte la différence de hauteur $\delta_z = z_i - z_j$ afin de supprimer le décalage.

Dans le cas 3D où l'on cherche par exemple à localiser un système composé d'une centrale inertielle et d'une balise, il se peut que les centres des deux capteurs ne soient pas exactement alignés. Si le centre de la centrale inertielle est utilisé comme origine du repère pour l'estimation, alors la position de l'antenne de la balise dans le repère du monde est donnée par :

$$\mathbf{p}_j = \mathbf{p}_I + \mathbf{R}_I \mathbf{p}_{Ij}$$

avec $\mathbf{p}_j = (x_j, y_j, z_j)^T$ la position de l'antenne dans le repère du monde, $\mathbf{p}_I = (x_I, y_I, z_I)^T$ la position de la centrale inertielle dans le repère du monde, $\mathbf{R}_I$ la matrice d'orientation de la centrale inertielle dans le repère du monde, et $\mathbf{p}_{Ij} = (x_{Ij}, y_{Ij}, z_{Ij})^T$ la position de l'antenne de la balise dans le repère de la centrale inertielle. La formule précédente suppose que la translation $\mathbf{p}_{Ij}$ est connue. Dans le cas où il n'est pas possible de mesurer directement cette translation sur le système physique, il est possible d'inclure ce paramètre dans l'algorithme de fusion de données et d'estimer sa valeur en temps réel.

Les modèles de mesure de distance qui viennent d'être présentés sont assez généraux. En fonction de la précision, de la complexité et du nombre de paramètres du modèle souhaités, il est possible de simplifier plus ou moins les équations du modèle. Dans le cas d'un faible décalage

entre l'antenne de la balise et l'origine du repère que l'on souhaite localiser, il est possible de négliger ce dernier dans le modèle. De plus, en fonction de la technologie utilisée pour obtenir la mesure de distance, certains paramètres peuvent s'avérer inutiles (facteur d'échelle $s$ égal à 1, ou biais $b$ égal à 0 par exemple). Dans tous les cas, il est recommandé de procéder à des tests en conditions réelles avec le dispositif de mesure afin de valider la pertinence du modèle. Une fois que l'on dispose d'un modèle des mesures de distance, il reste finalement à déterminer les différents paramètres qui composent ce modèle. Pour cela, il est nécessaire d'effectuer des expériences avec les capteurs et d'en extraire les paramètres à l'aide d'algorithmes de calibration. Cette procédure de calibration pour les balises est présentée en annexe A. Pour les balises développées et utilisées dans le cadre de ce travail, les paramètres obtenus après la procédure de calibration sont :

— Facteur d'échelle : $s = 1$,

— Biais : $b = 0.4$ m,

— Déviation standard : $\sigma = 0.1$ m,

— Paramètre de dépendance à la distance pour la variance : $\eta = 0$.

Nous allons maintenant voir les modèles de mesure de l'accéléromètre, du gyroscope, et du magnétomètre qui composent généralement une centrale inertielle.

### 2.3.2 Accéléromètre

En complément des codeurs incrémentaux équipant les moteurs de roue, et qui permettent uniquement d'estimer la pose du robot dans un plan, nous allons maintenant présenter brièvement une série de trois capteurs qui sont souvent regroupés ensemble dans ce que l'on appelle une centrale inertielle (ou IMU). La centrale inertielle permet d'obtenir une estimation de l'orientation et de la position en 3D [5], à partir des mesures d'un accéléromètre, d'un gyroscope et éventuellement d'un magnétomètre. La Figure 2.11 montre le modèle MPU9250 de Invensense ayant été sélectionné pour équiper le robot, alors que la Figure 2.12a montre la direction des différentes mesures suivant les axes du capteur ainsi que les angles d'orientation (roulis $\phi$, tangage $\theta$, lacet $\psi$) correspondants.


Figure 2.11 – Centrale inertielle MPU9250 de Invensense montée sur le robot. À noter les petites dimensions du module de l'ordre de $2 \times 2$ cm et son faible coût d'une dizaine d'euros.

Le premier capteur présenté, l'accéléromètre, permet de mesurer l'accélération subie par l'objet auquel il est attaché, dans le repère du capteur. Un accéléromètre peut être composé d'un, deux, ou trois axes $(x, y, z)$, suivant lesquels l'accélération est mesurée. En s'appuyant sur la loi fondamentale de la dynamique ($\sum \mathbf{f} = m\mathbf{a}$), un accéléromètre mesure aussi toutes les forces qui s'appliquent à un objet de masse $m$ suivant les axes du capteur [36]. La force de gravitation

étant inclue, il est nécessaire de la soustraire pour retrouver uniquement les composantes de l'accélération du capteur. Il faut bien noter que les mesures de l'accéléromètre sont faites dans le repère lié au capteur. Le principe de fonctionnement de l'accéléromètre suivant un axe $x$ peut être assimilé à un système masse ressort attaché au repère du capteur. Quand une force s'applique sur le capteur suivant l'axe $x$, le système masse ressort se déplace. En mesurant le déplacement $\delta x$, et en connaissant la raideur du ressort $k$ ainsi que la masse $m$, il est possible de déterminer l'accélération subie par le capteur suivant l'axe $x$ :

$$f = ma \leftrightarrow f = -k\delta x \leftrightarrow a = -\frac{k\delta x}{m}$$

Suivant le type de technologie utilisée par le capteur, la technique permettant de mesurer le déplacement $\delta x$ diffère. Cependant, dans la plupart des cas, ce déplacement est transformé en une tension proportionnelle à la valeur de l'accélération. Le capteur fournit alors en sortie une tension analogique ou bien une version numérique directement convertie en interne par le capteur. Ainsi, on peut modéliser le processus de mesure d'un accéléromètre trois axes par l'équation suivante :

$$\mathbf{z}_a = \mathbf{K}_a\mathbf{R}^T(\mathbf{a} - \mathbf{g}) + \mathbf{b}_a + \boldsymbol{\epsilon}_a \tag{2.10}$$

avec :

- $\mathbf{z}_a$ : la mesure d'accélération du capteur exprimée dans le repère du capteur en $\mathrm{m.s}^{-2}$,
- $\mathbf{K}_a$ : la matrice d'échelle,
- $\mathbf{R}$ : la matrice de rotation du repère lié à l'accéléromètre par rapport au repère d'inertie,
- $\mathbf{a}$ : l'accélération subie par le capteur exprimée dans le repère d'inertie en $\mathrm{m.s}^{-2}$,
- $\mathbf{g} = (0, 0, \pm g)^T$ : le vecteur gravité dans le repère d'inertie ($g \sim 9.81\ \mathrm{m.s}^{-2}$),
- $\mathbf{b}_a$ : le biais de l'accéléromètre,
- $\boldsymbol{\epsilon}_a$ : le bruit de mesure, modélisé par un bruit blanc Gaussien de moyenne nulle et de matrice de covariance $\boldsymbol{\Sigma}_a$.

Si l'on prend en compte la vitesse de rotation de la Terre $\boldsymbol{\omega}_G$, alors l'équation de mesure devient [100] :

$$\mathbf{z}_a = \mathbf{K}_a\mathbf{R}^T\Big(\mathbf{a} - \mathbf{g} + 2[\boldsymbol{\omega}_G]_\times\mathbf{v}_R + [\boldsymbol{\omega}_G]_\times^2\mathbf{p}_R\Big) + \mathbf{b}_a + \boldsymbol{\epsilon}_a \tag{2.11}$$

avec $\mathbf{v}_R$ et $\mathbf{p}_R$, la vitesse et la position respectivement du capteur, exprimées dans le repère d'inertie. Cette deuxième équation bien qu'apportant une modélisation plus précise, n'est utile que dans le cas d'un capteur d'accélération très précis. Dans le cas du capteur utilisé, il n'est pas nécessaire d'avoir recours à ce modèle. Pour cela, dans la suite, on se basera uniquement sur le premier modèle donné par (2.10).

Quand l'accéléromètre n'est pas placé à l'origine du repère de localisation, il est sujet à des accélérations centripètes causées par les vitesses angulaires. Ces accélérations doivent être soustraites aux mesures afin de ne considérer que les accélérations liées au centre du repère. La contribution centripète, $\mathbf{c}_i$, pour chaque axe $i$ de l'accéléromètre est donnée par [79] :

$$\mathbf{c}_i = \boldsymbol{\omega} \times (\boldsymbol{\omega} \times \mathbf{r}_i) + \dot{\boldsymbol{\omega}} \times \mathbf{r}_i, \quad i \in \{x, y, z\} \tag{2.12}$$

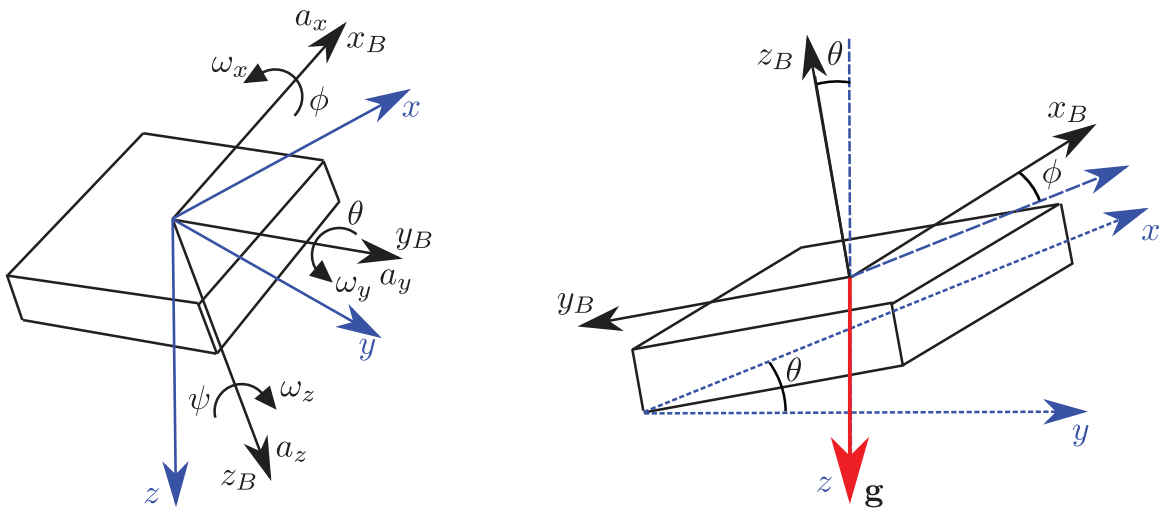
avec : $\boldsymbol{\omega} = (\omega_x, \omega_y, \omega_z)^T$ la vitesse angulaire, et $\mathbf{r}_i$ la position de l'accéléromètre lié à l'axe $i$ par rapport au centre du robot.

Un paramètre important dans la modélisation de la mesure d'accélération est le biais. Ce dernier correspond à la différence (offset) entre la valeur mesurée par le capteur et la vraie valeur. Lorsque le capteur est au repos, le biais correspond à la valeur moyenne du signal de

sortie. Quand on veut estimer la vitesse ou la position à partir de la mesure d'accélération, le biais peut vite devenir problématique. En effet, un biais supposé constant, qui est intégré deux fois donne une erreur en position qui croit de façon quadratique avec le temps :

$$\iint (a + b_a) \rightarrow x + b_a \frac{t^2}{2}$$

Le biais du capteur ainsi que d'autres paramètres comme la matrice d'échelle, $\mathbf{K}_a$, peuvent être estimés grâce à une procédure de calibration. Une telle procédure de calibration est présentée en annexe B. Pour le biais, il est aussi possible d'estimer sa valeur quand le capteur est au repos (pas de mouvement) et que l'on connaît l'orientation et la norme du vecteur gravité. En effet, au repos, la seule force mesurée par l'accéléromètre doit être la force de gravitation. Ainsi si l'on connaît sa direction par rapport au repère du capteur, il est possible d'obtenir une estimation du biais en prenant la valeur moyenne de l'accéléromètre pendant une certaine période. Par la suite il suffira de soustraire les valeurs de biais trouvées afin d'obtenir une mesure plus fiable de l'accélération. Une autre méthode pour compenser le biais consiste, comme on le verra par la suite, à ajouter ce paramètre aux variables à estimer dans les différents algorithmes. Ceci permettra ainsi d'avoir une estimation en temps réel de la valeur du biais et de pouvoir compenser son éventuelle dérive avec le temps.




(a) Orientation IMU

(b) Accéléromètre au repos

FIGURE 2.12 – Schémas des axes de mesure de l'IMU (a), et des angles de roulis, $\phi$, et tangage, $\theta$, de l'accéléromètre au repos mesurant uniquement le vecteur gravité.

En plus d'estimer la vitesse et la position du capteur quand celui-ci est en mouvement, un accéléromètre permet aussi, quand il est au repos, d'obtenir une estimation de deux angles (roulis $\phi$ et tangage $\theta$) de l'orientation du capteur, Figure 2.12b. En effet, en reprenant l'équation (2.10), et en supprimant l'accélération linéaire du capteur $\mathbf{a}$, on obtient :

$$\mathbf{z}_a = -\mathbf{K}_a \mathbf{R}^T \mathbf{g} + \mathbf{b}_a + \boldsymbol{\epsilon}_a$$

Ainsi, au repos, l'accéléromètre mesure uniquement la force de gravitation dans son repère (en supposant $\mathbf{K}_a$ et $\mathbf{b}_a$ connus). Comme généralement la direction du vecteur gravité dans le repère d'inertie est connue, il est possible d'obtenir une estimation des angles de roulis, $\phi$, et de tangage, $\theta$, du capteur par rapport au repère d'inertie. Comme l'estimation de l'orientation avec l'accéléromètre n'est valable que quand les accélérations linéaires subies par le capteur sont faibles ou négligeables devant la force de gravité, on utilise presque toujours en complément un gyroscope qui permet d'obtenir une estimation de l'orientation en intégrant les vitesses angulaires mesurées par le capteur.

### 2.3.3 Gyroscope

Le gyroscope est un capteur permettant de mesurer la vitesse de rotation angulaire suivant un ou plusieurs axes dans le repère qui lui est associé. On distingue principalement deux types de technologie : les gyroscopes optiques qui utilisent l'effet Sagnac et les gyroscopes mécaniques qui utilisent l'effet Coriolis [36]. Ces derniers, de par leur faible coût et la miniaturisation des composants sont devenus prépondérants dans la fabrication des centrales inertielles. Le principe d'un gyroscope mécanique repose sur un corps vibrant produisant un signal sinusoïdal d'amplitude constante lorsque le capteur est au repos. Quand le capteur est soumis à une rotation, les forces de Coriolis induisent alors des vibrations dont l'amplitude est proportionnelle à la vitesse angulaire, [92]. L'amplitude est alors généralement convertie en un signal électrique dont la tension ou sa version numérisée est proportionnelle à la vitesse angulaire. L'équation de mesure d'un gyroscope à trois axes est donc modélisée par :

$$\mathbf{z}_g = \mathbf{K}_g\boldsymbol{\omega} + \mathbf{b}_g + \boldsymbol{\epsilon}_g \tag{2.13}$$

avec :

- $\mathbf{z}_g$ : la mesure de vitesse angulaire du capteur exprimée dans le repère du capteur, en rad/s,
- $\mathbf{K}_g$ : la matrice d'échelle,
- $\boldsymbol{\omega}$ : la vitesse angulaire du capteur par rapport au repère d'inertie exprimée dans le repère attaché au capteur, en rad/s,
- $\mathbf{b}_g$ : le biais du capteur qui varie avec le temps et la température, en rad/s,
- $\boldsymbol{\epsilon}_g$ : le bruit de la mesure qui est modélisé par un bruit blanc Gaussien de moyenne nulle et de matrice de covariance $\boldsymbol{\Sigma}_g$.

Comme pour l'accéléromètre, si l'on prend en compte la vitesse de rotation de la Terre $\boldsymbol{\omega}_G$, on obtient d'après [100] :

$$\mathbf{z}_g = \mathbf{K}_g\boldsymbol{\omega} + \mathbf{R}^T\boldsymbol{\omega}_G + \mathbf{b}_g + \boldsymbol{\epsilon}_g$$

Cependant pour la même raison que pour l'accéléromètre, la première formulation donnée par (2.13) sera privilégiée dans la suite par rapport à la formulation prenant en compte la vitesse de rotation de la Terre.

De même que pour l'accéléromètre, le biais du gyroscope est une source importante de l'erreur de mesure. Ainsi, pour obtenir l'orientation du capteur, il est nécessaire d'intégrer une fois les vitesses angulaires mesurées. Et un biais qui est intégré une fois, donne alors une erreur angulaire qui croit linéairement avec le temps :

$$\int(\boldsymbol{\omega} + \mathbf{b}_g) \rightarrow \mathbf{q} + \mathbf{b}_g t$$

En revanche, l'estimation de la valeur de ce biais est plus facile que pour un accéléromètre. Lorsque le capteur est au repos (immobile), on sait que normalement la mesure du gyroscope doit être nulle. Ainsi, en maintenant le capteur au repos sur une certaine durée, tout en enregistrant les mesures, il suffit alors de faire la moyenne des mesures enregistrées pour obtenir une estimation du biais sur les différents axes du capteur. Le biais est alors compensé en le soustrayant aux mesures futures du gyroscope.

Par contre, il est connu que le biais du gyroscope est soumis à une dérive au cours du temps et qu'il est fonction de la température [5]. Ceci a pour conséquence que la valeur initiale du biais peut très rapidement devenir caduque. Cette dérive dépend beaucoup de la qualité du capteur. Afin de mieux prendre en compte son influence, il est souvent préférable comme

pour le biais de l'accéléromètre, d'estimer en temps réel sa valeur dans les algorithmes de fusion.

Le gyroscope fournit une mesure relative de l'orientation du capteur, c'est-à-dire soumise à l'intégration de la vitesse angulaire. L'accéléromètre, lui, fournit au repos, une mesure absolue de l'orientation du capteur en mesurant le vecteur gravité, mais seulement pour deux angles (roulis et tangage). Pour éviter la dérive suivant le dernier angle (lacet), un troisième capteur, le magnétomètre, est souvent ajouté à la centrale inertielle afin d'obtenir une mesure absolue de l'orientation par rapport au Nord magnétique ou au Nord géographique.

### 2.3.4 Magnétomètre

Le magnétomètre est un capteur mesurant le module et la direction du champ magnétique dans son repère. Le magnétomètre permet donc de mesurer la projection du champ magnétique terrestre local dans le repère du capteur :

$$\mathbf{z}_m = \mathbf{K}_m \mathbf{R}^T \mathbf{m} + \mathbf{b}_m + \boldsymbol{\epsilon}_m \tag{2.14}$$

avec :

- $\mathbf{z}_m$ : la mesure du champ magnétique exprimée dans le repère du capteur (en Tesla, T),
- $\mathbf{K}_m$ : la matrice d'échelle,
- $\mathbf{R}$ : la matrice de rotation du repère lié au magnétomètre par rapport au repère d'inertie,
- $\mathbf{m}$ : le vecteur du champ magnétique terrestre dans le repère d'inertie,
- $\mathbf{b}_m$ : le biais du magnétomètre,
- $\boldsymbol{\epsilon}_m$ : le bruit de la mesure qui est modélisé par un bruit blanc Gaussien de moyenne nulle et de matrice de covariance $\boldsymbol{\Sigma}_m$.

Le magnétomètre, en mesurant le champ magnétique terrestre, permet d'avoir une information sur la direction du Nord magnétique et par conséquent du Nord géographique. Le magnétomètre donne donc une mesure absolue de l'angle de lacet dans le repère d'inertie [20].

Cependant, pour retrouver l'orientation du magnétomètre, il est nécessaire de connaître l'intensité et la direction du champ magnétique terrestre $\mathbf{m}$. Or, ces valeurs dépendent de la position sur Terre et du temps (évolution temporelle). Heureusement, il existe des modèles permettant de calculer ces valeurs, que l'on peut par exemple obtenir sur le site de la National Oceanic and Atmospheric Administration (NOAA) [7]. Il est possible de caractériser le champ magnétique terrestre à l'aide d'un modèle simplifié. Ce modèle, qui est illustré à la Figure 2.13a, considère que le champ possède deux composantes, l'une horizontale, $m_h$, et l'autre verticale, $m_v$, [89]. La composante verticale est due à l'inclinaison du champ magnétique vers le centre de la Terre. La composante horizontale, qui est nulle aux pôles magnétiques, permet d'obtenir l'information sur la direction du Nord magnétique. En effet, si l'on place le magnétomètre dans un plan horizontal (perpendiculaire à la surface de la Terre) et que l'on effectue une rotation du capteur dans ce plan, la composante verticale reste constante, alors que seule la composante horizontale varie, voir Figure 2.13b. C'est donc cette dernière qui permet de retrouver le Nord magnétique. Le champ magnétique terrestre, $\mathbf{m}$, a donc la forme suivante :

$$\mathbf{m} = (m_x, 0, m_z)^T = (m_h, 0, m_v)^T = \Big(||\mathbf{m}|| \cos\delta, 0, ||\mathbf{m}|| \sin\delta\Big)^T \tag{2.15}$$

avec $\delta$ l'angle d'inclinaison du champ magnétique terrestre par rapport à l'axe $x$. Ainsi d'après le site de la NOAA, les valeurs à Strasbourg de l'angle d'inclinaison et de l'intensité du champ sont d'environ : $\delta = 64° \ 21'$, et $||\mathbf{m}|| = 48.25 \ \mu$T respectivement. Finalement, afin de retrouver

7. https://www.ngdc.noaa.gov/geomag-web

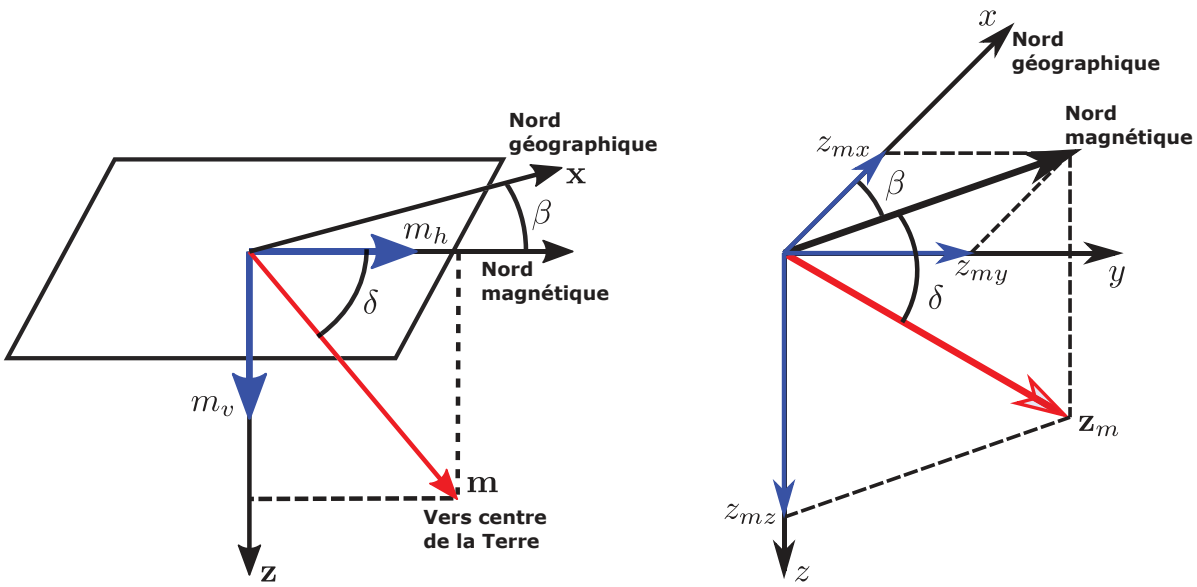
la direction du Nord géographique, il est nécessaire de connaître l'angle de déclinaison, $\beta$, qui représente l'angle entre la direction du Nord magnétique et la direction du Nord géographique. La valeur de cet angle dépend aussi de la position sur Terre et du temps. À Strasbourg, sa valeur est de $\beta = 2° \ 5'$ en direction de l'est.

L'angle de lacet, $\psi$, ou azimut, peut donc s'obtenir à partir de la mesure du magnétomètre en procédant comme suit. On ramène dans un premier temps les mesures du magnétomètre, $\mathbf{z}_m$, dans le plan horizontal de la Terre grâce aux équations suivantes [20] :

$$\begin{cases} m_{xh} = z_{mx}\cos\theta + z_{my}\sin\theta\sin\phi - z_{mz}\sin\theta\cos\phi \\ m_{yh} = z_{my}\cos\phi + z_{mz}\sin\phi \end{cases}$$

Puis, à partir des composantes du champ dans le plan horizontal, l'angle d'azimut par rapport au Nord géographique est obtenu grâce à l'équation suivante :

$$\psi = \text{atan2}(m_{yh}, m_{xh}) + \beta \tag{2.16}$$




(a) Modèle du champ magnétique terrestre (b) Modèle de mesure du magnétomètre

FIGURE 2.13 – Modèle du champ magnétique terrestre (a), et de la mesure du magnétomètre dans le plan de la Terre (b), qui permet de retrouver l'angle de lacet $\psi$.

L'inconvénient majeur du magnétomètre est que le champ magnétique est modifié par les objets magnétiques environnants (moteurs, batteries, structures métalliques, ...). En effet, en plus du champ magnétique terrestre, le magnétomètre mesure aussi l'ensemble des champs magnétiques générés localement. Cet inconvénient rend souvent difficile l'utilisation du magnétomètre en environnement intérieur. Pour cela, le magnétomètre est plus communément utilisé pour des applications extérieures où les probabilités de perturbations sont plus faibles. On distingue ainsi deux sources principales d'interférences magnétiques :

— Les sources d'interférences fixes dans le repère du capteur (Hard iron biases), incluses dans le biais $\mathbf{b}_m$, et qui peuvent être compensées par calibration,

— Les sources d'interférences fixes dans le repère de la Terre (Soft iron biases), difficilement corrigeables, et qui sont la cause d'erreurs dans la mesure de la direction du champ magnétique terrestre.

Une méthode permettant d'obtenir la direction, $\mathbf{m}$, du champ magnétique terrestre dans le repère de la Terre à partir de la mesure du magnétomètre, $\mathbf{z}_m$, tout en éliminant certaines interférences est proposée par Madgwick dans [89]. Cette méthode consiste dans un premier temps à normaliser la mesure du magnétomètre. Puis, une rotation par l'orientation courante du capteur, $\mathbf{q}$, est effectuée sur la mesure normalisée :

$$\mathbf{m} = (m_x, m_y, m_z)^T = \mathbf{q} \otimes \frac{\mathbf{z}_m}{||\mathbf{z}_m||} \otimes \mathbf{q}^*$$

avec $\mathbf{q}$ et $\mathbf{q}^*$ le quaternion et son conjugué représentant l'orientation du capteur, et $\otimes$ l'opérateur de multiplication sur les quaternions. La direction du champ magnétique terrestre est alors obtenue d'après :

$$\mathbf{m} = (\sqrt{m_x^2 + m_y^2}, 0, m_z)^T$$

Nous venons de présenter consécutivement trois capteurs (accéléromètre, gyroscope et magnétomètre) qui permettent indépendamment ou de façon combinée de retrouver dans certaines conditions un ou plusieurs angles de l'orientation globale par rapport à un repère d'inertie. Nous allons maintenant présenter un autre capteur, le GPS, qui permet lui, d'obtenir une estimation globale de la position et éventuellement de la vitesse dans un repère d'inertie.

### 2.3.5 GPS, DGPS-RTK

Le GPS, acronyme de Global Positioning System, est un système de positionnement global initialement déployé par l'armée américaine, et qui permet de se localiser presque partout sur la Terre grâce à des satellites en orbite. Le principe de ce système de localisation repose sur la trilatération. En mesurant le temps de vol des signaux envoyés par un minimum de quatre satellites distincts, il est possible d'obtenir une estimation de la position en 3D, [36], [39]. Le GPS est un moyen très populaire et efficace de localiser un robot mobile en environnement extérieur.

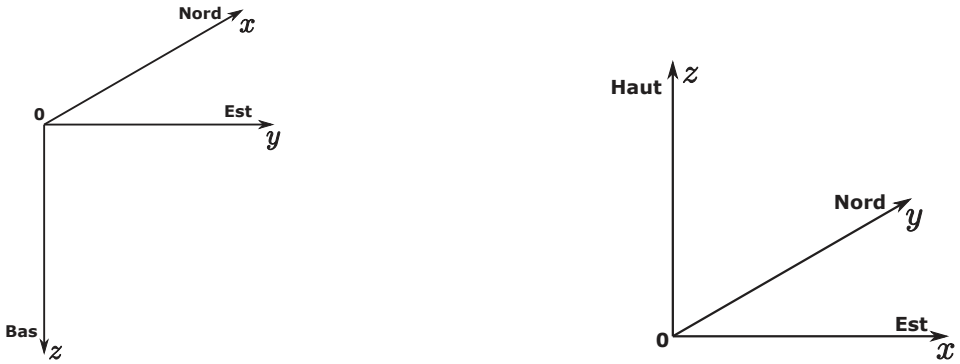
Cependant, ce système présente aussi un certain nombre de désavantages. Le principal étant la nécessité d'avoir une bonne visibilité entre le robot et les satellites afin d'avoir un minimum de quatre satellites et d'éviter les réflexions des ondes qui peuvent grandement influencer la qualité de la position estimée. Ainsi, l'utilisation du GPS est compromise dans les zones couvertes et denses, comme en milieu urbain ou dans des environnements boisés. De plus, pour certaines applications, la précision du GPS n'est pas suffisante. En effet, le GPS permet en général d'obtenir une précision comprise de l'ordre de $\pm 1$ à $\pm 10$ mètres suivant les conditions et technologies utilisées (SBAS, GBAS ...). Pour l'application considérée ici, la précision souhaitée est de l'ordre de $\pm 10$ cm.

En utilisant un système DGPS-RTK (Differential GPS with Real Time Kinematic), il est possible d'atteindre cette précision mais au prix de l'installation d'une station de base et d'un coût plus important. Le principe du DGPS-RTK consiste à utiliser deux GPS. L'un est fixe (base) et de position parfaitement connue, l'autre, mobile, est embarqué sur le système à localiser. Grâce au GPS fixe de la base, dont la position est connue, il est possible de corriger certaines erreurs sur les paramètres liés aux perturbations atmosphériques dans l'estimation de la position du GPS mobile. De plus, un filtre de Kalman étendu (EKF) est généralement employé, afin d'obtenir une meilleure estimation en temps réel de la position.

Dans le cadre de ce projet, le GPS n'est utilisé qu'en tant que capteur additionnel. Ainsi, quand celle-ci est disponible, l'information de position fournie par le GPS est simplement fusionnée dans le filtre de Kalman de localisation développé et présenté par la suite. Dans l'optique d'obtenir une vérité terrain durant les tests du prototype, une tentative de réaliser un système DGPS-RTK à bas coût, basé sur le projet *RTKLIB*[8] en utilisant deux GPS UBlox NEO-M8T

8. http://www.rtklib.com

a été faite, mais sans succès. En effet, la précision obtenue lors des essais était loin des ±10 cm attendus.




(a) Repère NED (b) Repère ENU

FIGURE 2.14 – Deux exemples de repère d'inertie lié à la Terre : (a) repère Nord-Est-Bas, (b) repère Est-Nord-Haut.

La position retournée par un GPS est généralement exprimée en coordonnées géographiques (latitude, longitude, altitude), mais il est possible, en utilisant des modèles prédéfinis, de la convertir dans d'autres représentations (NED, ENU, ECEF ...), voir Figure 2.14 pour un exemple des représentations NED et ENU. Ainsi, indépendamment des coordonnées utilisées, l'équation modélisant la mesure de position d'un GPS est donnée par :

$$\mathbf{z}_{\text{GPS}} = \mathbf{p} + \boldsymbol{\epsilon}_{\text{GPS}} \tag{2.17}$$

avec :

- $\mathbf{z}_{\text{GPS}} = (x_{\text{GPS}}, y_{\text{GPS}}, z_{\text{GPS}})^T$ : la mesure de position donnée par le GPS,
- $\mathbf{p} = (x, y, z)^T$ : le vecteur correspondant à la vraie position,
- $\boldsymbol{\epsilon}_{\text{GPS}}$ : le bruit affectant la mesure du GPS, généralement considéré comme un bruit blanc Gaussien de moyenne nulle et de matrice de covariance $\boldsymbol{\Sigma}_{\text{GPS}}$.

Il existe d'autres modèles plus complexes permettant de modéliser le processus de mesure de position du GPS, notamment en prenant en compte directement les mesures du temps de vol des ondes radiofréquences par rapport aux satellites [39]. Cependant, ces modèles sont généralement déjà utilisés en interne par le GPS pour calculer l'estimation de position. La modélisation présentée précédemment peut sembler simpliste, mais elle correspond bien aux données de sortie d'un GPS classique.

Dans cette section, nous avons présenté différents capteurs équipant le prototype du robot et servant principalement à sa localisation. Parmi eux, on distingue ceux permettant de se localiser dans un repère local, comme les codeurs incrémentaux, l'accéléromètre et le gyroscope, qui permettent d'estimer la pose par intégration des mesures. Alors qu'avec les balises, le GPS, le magnétomètre et la mesure du vecteur de gravité par l'accéléromètre, il est possible de se localiser globalement dans un repère d'inertie.

Cependant, afin d'être totalement autonome, le robot a aussi besoin d'autres capteurs lui permettant notamment d'éviter les obstacles. Ainsi, sur le prototype développé, des capteurs ultrasons, des capteurs infrarouges et des capteurs de contact, ont aussi été installés. Ces capteurs bien qu'étant essentiels, ne sont pas présentés ici, car ils ne sont pas directement liés à la problématique de localisation.

Après avoir vu le principe de fonctionnement du robot et en supposant que les capteurs présentés dans cette section permettent d'obtenir une estimation de toutes les variables liées au robot (pose, vitesse, ...), nous allons maintenant présenter l'architecture de commande du robot.

## 2.4 Architecture de commande

En s'appuyant sur les mesures fournies par les différents capteurs, ainsi que les modèles du robot et des moteurs, il est possible de construire l'architecture de commande du robot. La commande du robot consiste à déterminer les tensions à appliquer aux deux moteurs des roues en fonction des mesures afin d'atteindre une consigne donnée. L'architecture de commande, adoptée ici, repose sur plusieurs fonctions imbriquées.

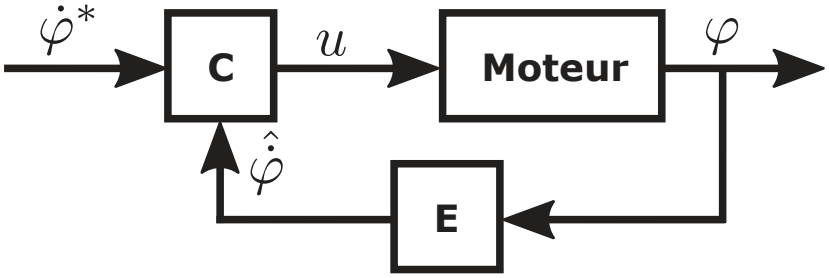



FIGURE 2.15 – Schéma-bloc de commande en vitesse d'un moteur de roue

La première fonction correspond à la commande bas-niveau des deux moteurs de roue. Chaque moteur est asservi par un contrôleur en vitesse. Comme on peut le voir sur la Figure 2.15, le correcteur **C** calcule la commande $u$, qui est une tension, à partir de la vitesse de rotation de référence, $\dot{\varphi}^*$, et de la vitesse de rotation, $\hat{\dot{\varphi}}$, estimée par le bloc **E**. Le bloc d'estimation **E**, permet d'estimer la vitesse de rotation du moteur en dérivant la mesure de position, $\varphi$, du moteur. En pratique, le correcteur **C** est réalisé avec un contrôleur proportionnel intégral (PI). L'asservissement PI en vitesse de chaque roue est alors implémenté sur les deux microcontrôleurs des cartes contrôleurs de moteur.

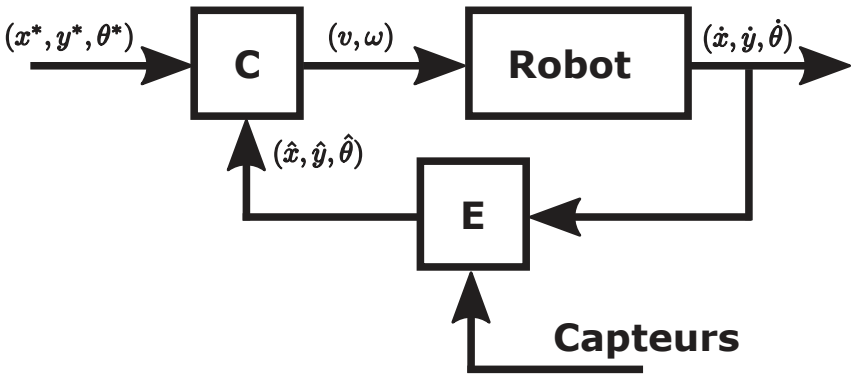



FIGURE 2.16 – Schéma-bloc de commande en position du robot

La deuxième fonction correspond à l'asservissement en position ou pose du robot. Le principe de la commande, visible à la Figure 2.16, est similaire au cas précédent. Le correcteur **C** calcule les commandes, qui sont les vitesses linéaire $v$ et angulaire $\omega$ du robot, à partir de l'estimation de la pose courante du robot, $(\hat{x}, \hat{y}, \hat{\theta})$, et de la consigne $(x^*, y^*, \theta^*)$. L'estimation de la pose courante du robot, $(\hat{x}, \hat{y}, \hat{\theta})$, est réalisée par le bloc **E**, qui utilise pour cela, le modèle du robot représenté par $(\dot{x}, \dot{y}, \dot{\theta})$ et donné par les équations (2.5), ainsi que des mesures provenant de

différents capteurs. Le bloc **E** peut être réalisé, par exemple, avec un filtre de Kalman, comme nous le verrons par la suite. Concernant le correcteur **C**, aucun choix précis n'a été fait, mais plusieurs solutions ont été implémentées, dont notamment :

— Un asservissement en position décomposé en deux phases et qui découple la commande en vitesse linéaire et angulaire. Dans la première phase, on tourne le robot vers la position à atteindre, et dans la seconde on fait avancer le robot vers l'objectif. Chacune des phases utilise un correcteur proportionnel : $v = k_v e_v$, $\omega = k_\omega e_\omega$,

— Un asservissement par suivi de chemin [99],

— Un asservissement par suivi de trajectoire [99],

— Un asservissement par suivi d'un véhicule de référence fictif ayant les mêmes caractéristiques que le robot à commander [99].

Enfin, la troisième fonction, qui est optionnelle, correspond à l'asservissement des vitesses linéaire $v$, et angulaire $\omega$ du robot. Les commandes en vitesse et en position du robot sont toutes les deux implémentées avec ROS sur le Raspberry Pi 3.

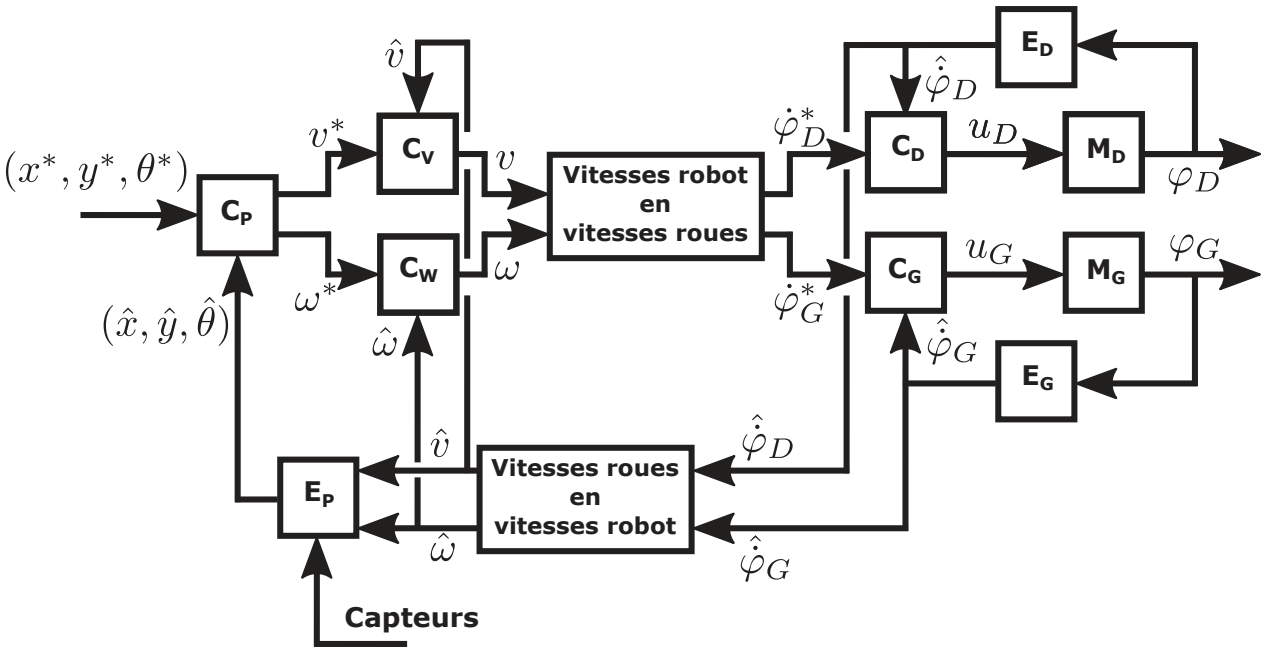



FIGURE 2.17 – Schéma-bloc de commande complet du robot

L'architecture globale de commande du robot est visible à la Figure 2.17. Comme détaillé précédemment, la commande repose sur trois boucles imbriquées. La boucle extérieure correspond à l'asservissement de la pose du robot avec le correcteur $\mathbf{C_p}$ et l'estimateur $\mathbf{E_p}$. La boucle du milieu correspond à l'asservissement en vitesse du robot avec les correcteurs $\mathbf{C_v}$ et $\mathbf{C_\omega}$. Enfin, la boucle intérieure comprend les asservissements des moteurs de roue droit et gauche, $\mathbf{M_D}$ et $\mathbf{M_G}$, par les correcteurs $\mathbf{C_D}$ et $\mathbf{C_G}$. Les blocs qui transforment les vitesses du robot en vitesses des roues et vice versa correspondent aux équations (2.4) et (2.2) respectivement.

Comme on peut le voir sur la Figure 2.17, les différentes boucles d'asservissement nécessitent une mesure des variables à asservir que sont les vitesses des roues $(\hat{\dot{\varphi}}_D, \hat{\dot{\varphi}}_G)$, les vitesses du robot $(\hat{v}, \hat{\omega})$, et la pose du robot $(\hat{x}, \hat{y}, \hat{\theta})$. Dans les deux chapitres qui suivent nous allons voir différents algorithmes de fusion de données permettant d'obtenir une estimation de ces variables et notamment de la pose du robot.

## 2.5 Conclusion

Au cours de ce chapitre axé sur la modélisation, le robot mobile ainsi que ses principaux composants ont été introduits. Dans un premier temps, l'architecture retenue pour réaliser le prototype, avec le détail des différents composants (actionneurs, capteurs et unités de calculs) et des communications entre les différents blocs a été présentée. Ensuite, le modèle du robot, de type unicycle, ainsi les équations d'odométrie permettant d'estimer la pose du robot à partir des mesures des codeurs incrémentaux ont été détaillés. Puis, les équations de modélisation des principaux capteurs équipant le robot (balise RF, centrale inertielle et GPS) ont été présentées. Enfin, pour terminer ce chapitre, le choix de l'architecture de commande des moteurs et du robot est rapidement expliqué.

L'étape de modélisation qui vient d'être présentée, va maintenant servir à mettre en œuvre les différents algorithmes de localisation et de cartographie présentés dans les chapitres qui suivent. Pour cela, les modèles et les équations de ce chapitre seront réutilisés dans la réalisation des algorithmes. Ainsi, par exemple, dans le chapitre suivant, deux approches de localisation sont présentées. La première utilise uniquement les mesures de distance, alors que la seconde permet de combiner l'odométrie du robot avec les mesures de distance provenant des balises.

# CHAPITRE 3

# LOCALISATION AVEC MESURES DE DISTANCE

## 3.1 Introduction

Après un premier chapitre centré sur la présentation du robot, des capteurs et de leurs modèles, nous allons maintenant voir dans les deux prochains chapitres comment utiliser et combiner ces informations afin d'estimer les différents états du robot et notamment sa pose (position et orientation). Dans ce chapitre nous allons nous intéresser au problème de localisation du robot dans son environnement. La localisation peut se définir comme l'estimation de la pose du robot relativement à l'environnement, considéré connu, dans lequel il évolue. Dans le chapitre précédent, le modèle du robot unicycle a été présenté ainsi que le moyen d'obtenir une première estimation de sa pose en intégrant les données des codeurs incrémentaux des roues à partir d'une pose initiale connue. Cette technique peut déjà être considérée comme une méthode de localisation du robot, bien que l'intégration des erreurs sur les vitesses des roues va rapidement faire dévier la pose estimée de la vraie pose du robot, et ce d'autant plus que le terrain sera accidenté ou la résolution des codeurs incrémentaux mauvaise. Pour améliorer l'estimation de la pose du robot, il est donc nécessaire d'intégrer des informations provenant d'autres capteurs. Ainsi, parmi les capteurs équipant communément les robots mobiles on retrouve les télémètres laser, les caméras (mono, stéréo...), et les GPS dans le cas d'une utilisation extérieure. Cependant, comme déjà expliqué au chapitre précédent, en raison du cahier des charges du robot à développer, aucun des capteurs précités ne peut être utilisé. Pour cela, les seuls capteurs considérés seront les codeurs incrémentaux des roues et/ou une centrale inertielle combiné(s) avec des balises fournissant des mesures de distance dont le principe a été présenté au chapitre précédent. On distingue les algorithmes de localisation n'utilisant qu'une seule source d'information (les mesures de distance par exemple), et les algorithmes permettant de fusionner plusieurs capteurs différents. Dans le dernier cas, en connaissant son environnement le robot va déterminer sa pose par rapport à celui-ci en fusionnant les données de ses capteurs internes fournissant une information sur son mouvement, et les données de capteurs extéroceptifs qui donnent des informations relatives à son environnement. Ce principe assez général illustré à la Figure 3.1 peut se résumer en deux phases :

— La **prédiction**, où le robot prédit sa future pose en utilisant un modèle du système ou en intégrant les informations de ses capteurs proprioceptifs,

— La **correction** qui consiste à recaler sa position relativement à son environnement grâce aux capteurs extéroceptifs.

Pour la localisation, l'environnement est supposé connu à l'avance par le robot. Il est donc nécessaire de définir une représentation de celui-ci. Différentes représentations peuvent être uti-

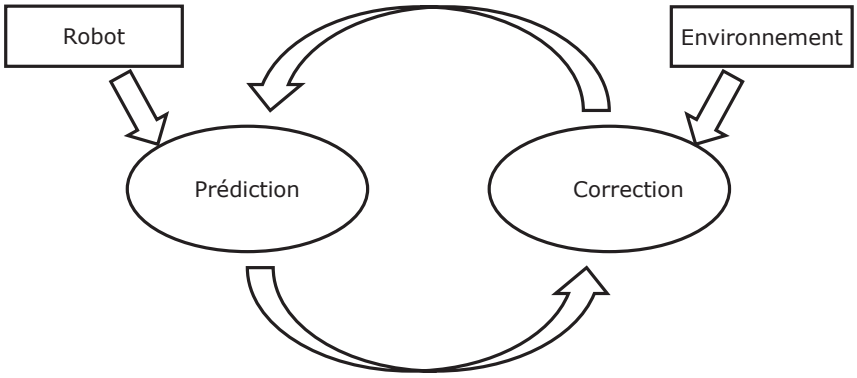



FIGURE 3.1 – Cycle de prédictions et de corrections sur lequel repose un grand nombre d'algorithmes de fusion de données.

lisées parmi lesquelles on trouve les grilles d'occupation [38]. Le terrain est ainsi discrétisé en cellules comportant chacune la probabilité d'occupation par un obstacle. Une autre représentation possible repose sur l'utilisation de repères caractéristiques artificiels ou naturels facilement identifiable par les capteurs [84]. Dans le cadre de ce travail, la deuxième représentation sera privilégiée car les balises RF fournissant les mesures de distance peuvent être considérées comme des repères artificiels caractéristiques de l'environnement. Ainsi, dans la majorité des cas, quand il sera fait référence à l'environnement, il s'agira en réalité des balises RF et plus précisément de leurs positions. Il sera ainsi considéré dans la suite de ce chapitre que la position des balises est connue avec précision dans un repère donné, lié à l'environnement. Le but sera alors de déterminer la pose ou position du robot par rapport à ce repère et donc par rapport aux balises. Notons que cette représentation, bien que suffisante pour localiser le robot, peut-être incomplète pour l'exécution d'autres fonctionnalités comme la navigation et l'évitement d'obstacles. Si l'environnement dans lequel évolue le robot est composé d'obstacles statiques ou dynamiques, il est donc nécessaire de disposer d'une autre représentation plus complète afin que le robot puisse prendre en compte tous les éléments significatifs lors de la réalisation de ses tâches. Enfin, un avantage d'utiliser des repères artificiels comme les balises RF, est une plus grande robustesse au caractère dynamique de l'environnement. Bien que les obstacles dynamiques évoluant dans l'environnement puissent perturber les mesures de distance de façon imprédictible, le fait d'obtenir des mesures par rapport à des repères fixes et identifiables permet d'éviter les problèmes d'association et de mauvaises correspondances que l'on peut par exemple rencontrer dans le cas de la localisation avec une caméra.

Le terme localisation englobe différents problèmes de difficultés variées [126]. On peut ainsi différencier la localisation locale et globale. Concernant la localisation locale, on fait l'hypothèse que la pose initiale du robot est connue et l'on cherche alors à suivre l'évolution de la pose du robot, d'où l'autre nom qu'on lui donne souvent qui est le suivi de pose. En revanche si le robot démarre en n'ayant aucune idée de l'endroit ou il se trouve, on parle alors du problème plus complexe de localisation globale. Enfin, il existe une dernière variante, connue sous le nom du problème de l'enlèvement, ou à un moment donné le robot est déplacé à une autre position. Le robot estime ainsi qu'il est quelque part alors qu'en fait il est ailleurs. Toute la difficulté consiste alors à détecter ce changement à partir des données des capteurs qui ne vont plus être cohérentes avec l'estimation courante de la pose du robot et à retrouver la bonne pose.

La suite de ce chapitre consacré a la localisation est organisée comme suit. Tout d'abord un état de l'art sur les différents algorithmes de localisation est présenté en mettant l'accent sur ceux basés sur des mesures de distance. Dans un second temps, la difficulté de la localisation par mesures de distance sera montrée en étudiant l'observabilité du problème de localisation par mesures de distance seulement. Ensuite, différents algorithmes de localisation, utilisant comme seule source d'information les mesures de distance, seront présentés et comparés en termes de robustesse au bruit et aux mesures aberrantes. Parmi eux, un nouvel algorithme de localisation robuste utilisant la théorie des inégalités linéaires matricielles (LMI) et des sommes de polynômes au carré (SoS) est proposé et comparé aux méthodes précédentes. Enfin, différentes implémentations d'algorithmes de fusion de données pour la localisation en 2D et en 3D seront présentées et comparées en termes de précision, de temps de calcul et de robustesse. Le chapitre se conclut sur un résumé des performances des différentes techniques de localisation développées et les améliorations encore envisageables.

### 3.1.1 Contributions

Au cours de ce chapitre trois contributions sont apportées en plus d'une comparaison extensive de différents algorithmes de localisation par mesures de distance. La première concerne la dérivation d'une formulation alternative de l'algorithme de trilatération sous forme d'un système linéaire homogène. La seconde présente une nouvelle technique de trilatération robuste utilisant la théorie des LMIs et des polynômes SoS. Enfin, la troisième et dernière contribution de ce chapitre est la réalisation d'un algorithme de fusion de données en 3D, basé sur un filtre de Kalman étendu, qui fusionne les données des codeurs incrémentaux des roues du robot, de la centrale inertielle (accéléromètre, gyroscope et magnétomètre) et les mesures de distance robot/balises.

Nous allons maintenant présenter un état de l'art des différentes techniques de localisation d'un robot mobile et plus particulièrement celles utilisant des mesures de distance comme principale source d'information extéroceptive.

## 3.2 État de l'art

Le problème de localisation d'un robot mobile a déjà fait l'objet de nombreuses recherches et l'on trouve depuis quelques années déjà des algorithmes de localisation matures présentant des caractéristiques de précision, de robustesse et de temps d'exécution compatibles avec un bon nombre d'applications. Des explications détaillées de la théorie, ainsi que des exemples d'implémentations en pseudo-code des algorithmes les plus populaires, ont déjà fait l'objet de livres de référence comme [126] et [118]. Parmi les algorithmes de fusion de données les plus populaires pour la localisation on citera le filtre de Kalman [71] et le filtre particulaire [125]. Suivant les capteurs équipant le robot et le modèle utilisé pour représenter l'environnement connu a priori, les algorithmes peuvent varier en complexité. Dans le cas le plus simple de la localisation, on considère souvent que l'environnement est représenté par des repères artificiels ou naturels identifiables par le robot et dont la position est connue. On considère alors que le robot est capable de mesurer la distance et l'angle relatifs entre lui et un repère de référence. Ainsi à partir de ces deux mesures le robot peut connaître sa position relative par rapport au repère. Ce sont ces deux mesures qui sont alors utilisées dans l'étape de correction du filtre de fusion de données. Cependant, il existe des cas où le robot ne peut mesurer que l'une de ces deux informations : soit l'angle relatif comme c'est le cas avec une caméra monoculaire, soit la distance relative par rapport au repère comme c'est le cas avec les balises RF. On appelle cela alors de la localisation par mesure d'angle uniquement ou par mesures de distance uniquement. Ici, comme nous disposons de balises RF capables de mesurer des distances, nous

nous trouvons donc dans le second cas. Ainsi nous allons maintenant présenter les différentes méthodes développées dans le cadre de la localisation par mesures de distance uniquement. Tout d'abord nous allons voir les techniques de localisation utilisant uniquement les mesures de distance, puis dans un deuxième temps nous allons présenter des algorithmes permettant de fusionner les informations de capteurs additionnels.

### 3.2.1 Mesures de distance uniquement

La méthode de localisation par mesures de distance uniquement la plus basique est l'algorithme de trilatération. Cette méthode repose sur le fait que pour un espace à $N$ dimensions, $N = 2, 3$, si l'on a $N + 1$ mesures de distance par rapport à des balises de positions connues, alors la position à estimer est l'intersection de $N + 1$ cercles/sphères centrés sur la position des balises et de rayons correspondant à la mesure de distance. En l'absence de bruit cette intersection existera toujours et sera unique, sauf dans des cas particuliers de configuration des balises. En revanche, en présence de bruit sur les mesures de distance, il y a de fortes chances qu'une intersection unique et commune à toutes les mesures n'existe pas. Les différentes méthodes de trilatération vont donc chercher à estimer l'intersection qui correspond le mieux aux mesures de distance. Plusieurs versions de l'algorithme de trilatération seront présentées dans une section ultérieure, mais on peut déjà citer les méthodes présentées dans [116], ainsi que par Zhou [138] où ce dernier utilise une formulation basée sur les moindres-carrés. Parmi les alternatives à la trilatération, on trouve la méthode proposée par Doherty et al. [35], qui transforme le problème de localisation en un problème d'optimisation convexe en dérivant des contraintes d'inégalités linéaires matricielles convexes à partir des mesures de distance. La solution est alors trouvée en résolvant un problème d'optimisation semi-définie positive (SDP). Enfin, dans [75], une technique de régression de noyau est proposée pour résoudre le problème de la localisation par mesures de distance. Une approche générique permettant de réaliser la fonction noyau est présentée. La localisation se fait en deux étapes. Dans la première, qui se fait hors-ligne, un algorithme de Levenberg-Marquardt est utilisé pour résoudre le problème d'optimisation de la régression de noyau. Enfin, un filtre de Kalman étendu est utilisé afin de localiser en temps réel le robot mobile. Nous allons maintenant passer en revue les méthodes qui fusionnent différentes informations en plus des mesures de distance.

### 3.2.2 Mesures de distance fusionnées avec d'autres capteurs

Quand il s'agit de fusionner les informations de différents capteurs pour obtenir une estimation en temps réel, le problème est généralement résolu en utilisant une technique de filtrage. Plus récemment, des techniques d'optimisation fonctionnant en temps réel et apportant souvent de meilleurs résultats sont aussi apparues. Ainsi nous allons d'abord nous intéresser aux deux filtres les plus populaires en robotique mobile : le filtre de Kalman et le filtre particulaire. Puis nous présenterons quelques méthodes alternatives aux techniques de filtrage.

#### Filtres de Kalman

Kantor et al. [72] appliquent deux algorithmes classiques : le filtre de Kalman étendu et le filtre particulaire dans le cadre de la localisation utilisant comme seules informations l'odométrie du robot et des mesures de distance par rapport à des balises fixes de positions connues. Les résultats expérimentaux montrent que même avec des erreurs sur les mesures de distance de l'ordre de 1.8 m, il est possible d'estimer la position du robot avec une erreur de moins de 0.3 m. Ceci montre ainsi que la fusion de données permet à partir de différents capteurs d'obtenir une meilleure précision en position que la précision des capteurs. Dans des travaux ultérieurs [77], les résultats expérimentaux de la fusion dans un filtre de Kalman étendu de l'odométrie des

roues et des mesures de distance par rapport à des balises RF sont présentés. Une étude des caractéristiques du bruit des mesures de distance est faite, et à nouveau malgré une erreur de mesure de l'ordre de 1.5 m, l'erreur moyenne de localisation du robot sur des jeux de données de plusieurs dizaines de minutes n'est que d'environ 0.3 m. Graefenstein et al. [55] ont développé une méthode de localisation robuste bien adaptée aux environnements extérieurs, qui utilise l'indicateur de l'intensité du signal reçu (RSSI) de balises RF à faible puissance. Un nouveau modèle de mesure est présenté afin d'obtenir une estimation de la distance à partir du RSSI. Un filtre particulaire est alors utilisé afin de fusionner ces informations avec les données d'odométrie des roues, des capteurs ultrasons et une connaissance à priori de certains obstacles du terrain. Les résultats expérimentaux démontrent qu'une précision inférieure au mètre peut être obtenue. Dans [101] Mueller et al. fusionnent les mesures de distance de balises RF UWB avec les données d'un accéléromètre et d'un gyroscope dans un filtre de Kalman étendu pour estimer la position d'un quadricoptère. Les données inertielles sont utilisées dans l'étape de prédiction du filtre, alors que les mesures de distance sont utilisées pour la correction. Grâce à la haute fréquence des mesures de distance (200 Hz), les expériences menées démontrent la faisabilité d'utiliser la position estimée comme retour d'état dans une boucle de contrôle en position du drone. Dans Busanelli [16], deux manières de fusionner des mesures de distance avec l'odométrie des roues dans un filtre de Kalman étendu sont présentées. La première méthode, TDOA (Time Differences of Arrivals), fusionne uniquement une seule mesure distance provenant d'une balise donnée. Alors que la deuxième, TTDOA (Twin-receiver TDOA), utilise deux mesures de distance provenant de deux balises différentes pour obtenir en plus une estimation de l'angle du robot. Les deux méthodes sont évaluées en simulations, et il est montré que la deuxième méthode permet de réduire significativement les erreurs en position et en orientation du robot comparée à la première méthode. Ainsi, pour un écart-type des mesures de distance de 0.1 m, l'erreur en position du robot obtenue par la première méthode est de l'ordre de 0.1 m, alors qu'elle n'est plus que de 0.01 m pour la deuxième méthode.

### Filtres particulaires

Dans González et al. [53], un filtre particulaire est employé pour localiser un robot mobile avec des mesures de distance provenant de balises RF UWB. Une étude approfondie des caractéristiques des mesures de distance sans et en présence d'obstacle est d'abord réalisée afin de dériver un modèle de mesure utilisé dans le filtre particulaire. En plus d'estimer la pose du robot, le filtre inclut aussi le biais des mesures de distance par rapport à chaque balise. Des expériences sont conduites en intérieur dans deux conditions : avec des lignes de vue parfaites (pas d'obstacles entre le robot et les balises) et avec des mesures de distance faites à travers des obstacles. Dans les conditions de ligne de vue parfaites, l'erreur en position obtenue est de l'ordre de 5 cm. De plus, dans les conditions ou des obstacles se trouvent sur le chemin des mesures de distance, il est montré que l'estimation du biais des mesures de distance améliore grandement la robustesse de l'algorithme et permet une meilleure localisation.

Dans Jourdan et al. [68] un algorithme de localisation en intérieur basé sur un réseau de balises RF UWB est présenté. Un filtre particulaire est utilisé pour la fusion des données. Comme précédemment, le biais des mesures de distance est inclus dans l'état à estimer et permet d'améliorer les résultats obtenus en présence d'obstacles.

### Autres méthodes

Parmi les autres techniques d'estimation que le filtrage, on retrouve notamment les méthodes d'optimisation. Une approche particulièrement performante est présentée par Kaess et al. dans [70] et [26]. Bien que cette méthode n'a pas été présentée dans le cas des mesures de distance uniquement, elle peut facilement s'y adapter. Ainsi, des exemples pour la localisation

par mesures de distance sont présentés au sein de la bibliothèque de fonctions, appelée GT-SAM, et mis à disposition sur le site des auteurs [1]. Une autre technique d'estimation reposant sur l'analyse par intervalles a été appliquée dans le cadre de la localisation avec des mesures de distance par Jaulin et al. [65] et [119]. Cette technique permet d'avoir une garantie sur la localisation du robot dans le cas où les erreurs de mesure sont inconnues mais bornées avec les bornes connues. Il est alors possible d'obtenir une estimation de la position du robot sous forme d'un intervalle dont on garantit qu'il contient la position du robot.

Après ce passage en revue des différentes méthodes de localisation par mesures de distance, nous allons dans un premier temps nous intéresser au problème de l'estimation d'une position en utilisant uniquement des mesures de distance avant de passer au problème de la fusion de données pour localiser un robot mobile.

## 3.3 Algorithmes de trilatération

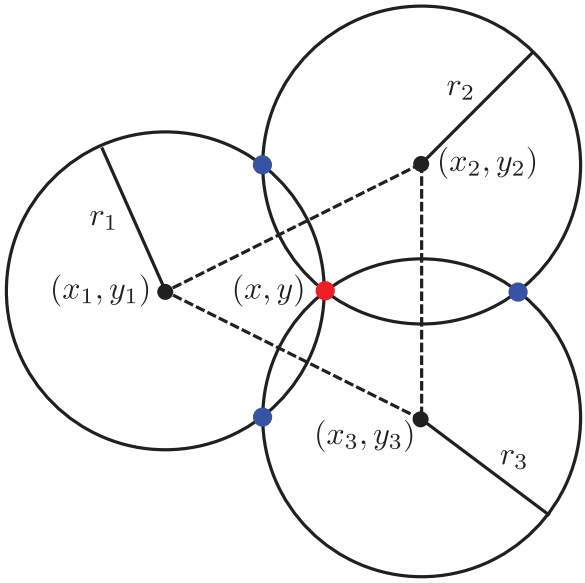
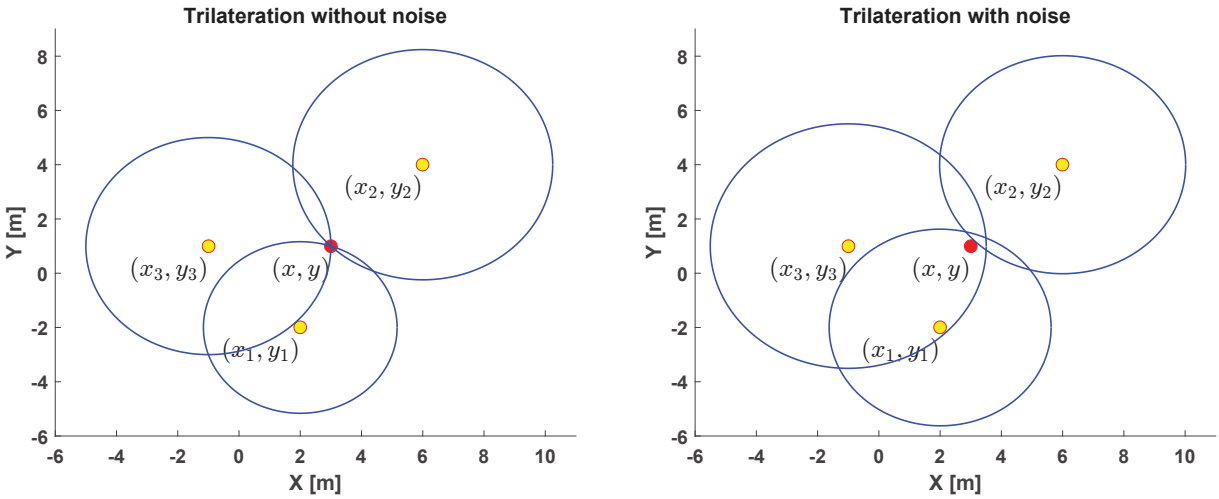



FIGURE 3.2 – Principe de la trilatération : les mesures de distance par rapport aux balises peuvent être vues comme les rayons de cercles centrés sur les positions des balises, et dont l'intersection donne le point que l'on cherche à estimer.

La trilatération est l'un des algorithmes de localisation par mesures de distance le plus simple. Il ne requiert que des mesures de distance par rapport à des balises fixes de positions connues pour obtenir une estimation de la position. Cependant, un inconvénient de la méthode est la nécessité d'avoir un minimum de $n+1$ mesures de distance pour un espace à $n$ dimensions (3 en 2D, 4 en 3D) à un instant donné pour calculer la position. L'idée de la méthode repose sur de la géométrie simple : l'intersection de trois cercles dans le plan est un point, voir Figure 3.2. En effet, en considérant des mesures de distance parfaites (non bruitées), les positions supposées connues des trois balises correspondent au centre des cercles dont le rayon est la mesure de distance, voir Figure 3.3a. Ainsi, en collectant un minimum de trois mesures de distance par rapport à des balises de position connue à un instant donné, il est possible d'estimer la position courante du robot. Dans le cas où les mesures sont bruitées, il est probable que les trois cercles ne s'intersectent plus en un même point, voir Figure 3.3b. L'algorithme de trilatération va

1. https://research.cc.gatech.edu/borg/projects/gtsam

alors retourner le point qui satisfait au mieux les équations. Il est aussi important de noter que certaines configurations des balises sont plus ou moins sensibles au bruit de mesure. Par exemple, on évitera autant que possible de calculer une position quand les balises utilisées sont sur la même ligne. Au contraire, la configuration où la position à estimer se trouve au milieu d'un triangle inscrit par les trois balises est optimale.




(a) Trilatération sans bruit

(b) Trilatération avec bruit

FIGURE 3.3 – Exemple de deux configurations de trilatération, l'une sans (a), l'autre avec (b) bruit sur les mesures de distance.

Il existe différents algorithmes de trilatération, dont les principaux sont maintenant détaillés. Tous les algorithmes sont présentés dans le cas où la position à estimer se trouve dans le plan, cependant l'extension au cas de la 3D est le plus souvent trivial et consiste uniquement à ajouter les équations correspondantes à la troisième coordonnée z. L'une des principales différences des algorithmes de trilatération réside dans la construction du système à résoudre à partir des mesures de distance. Certaines approches s'attachent à résoudre un système linéaire et d'autres consistent à résoudre un système non-linéaire.

Pour toutes les méthodes, on cherche à estimer la position du point $\mathbf{p} = (x, y)^T$ à partir des mesures de distance par rapport à $N$ balises dont les positions sont connues. On peut donc regrouper les différentes équations de mesures de distance dans un même système :

$$\begin{cases} r_1 = \sqrt{(x - x_1)^2 + (y - y_1)^2} \\ r_2 = \sqrt{(x - x_2)^2 + (y - y_2)^2} \\ r_3 = \sqrt{(x - x_3)^2 + (y - y_3)^2} \\ \vdots \\ r_N = \sqrt{(x - x_N)^2 + (y - y_N)^2} \end{cases} \tag{3.1}$$

avec $r_i$ la distance par rapport à la i$^{\text{ème}}$ balise et $\mathbf{p}_i = (x_i, y_i)^T$ sa position.

### 3.3.1 Méthodes linéaires

#### Méthode 1

La principe de la méthode présentée dans [116], consiste à mettre au carré les équations de mesures de distance, puis à soustraire l'équation de l'une des mesures de distance au reste

des équations. Sans perte de généralité, on peut utiliser la première équation. En mettant les équations du système (3.1) au carré et en développant, on obtient :

$$\begin{cases} r_1^2 = x^2 + x_1^2 + y^2 + y_1^2 - 2xx_1 - 2yy_1 \ (1) \\ r_2^2 = x^2 + x_2^2 + y^2 + y_2^2 - 2xx_2 - 2yy_2 \ (2) \\ r_3^2 = x^2 + x_3^2 + y^2 + y_3^2 - 2xx_3 - 2yy_3 \ (3) \\ \vdots \\ r_N^2 = x^2 + x_N^2 + y^2 + y_N^2 - 2xx_N - 2yy_N \ (3) \end{cases}$$

En faisant (2)-(1), (3)-(1), ..., (N)-(1), on obtient :

$$\begin{cases} r_2^2 - r_1^2 = x_2^2 - x_1^2 + y_2^2 - y_1^2 - 2x(x_2 - x_1) - 2y(y_2 - y_1) \\ r_3^2 - r_1^2 = x_3^2 - x_1^2 + y_3^2 - y_1^2 - 2x(x_3 - x_1) - 2y(y_3 - y_1) \\ \vdots \\ r_N^2 - r_1^2 = x_N^2 - x_1^2 + y_N^2 - y_1^2 - 2x(x_N - x_1) - 2y(y_N - y_1) \end{cases}$$

Ce qui donne en isolant les inconnues d'un seul côté :

$$\begin{cases} x(x_2 - x_1) + y(y_2 - y_1) = \frac{1}{2}(x_2^2 - x_1^2 + y_2^2 - y_1^2 - (r_2^2 - r_1^2)) \\ x(x_3 - x_1) + y(y_3 - y_1) = \frac{1}{2}(x_3^2 - x_1^2 + y_3^2 - y_1^2 - (r_3^2 - r_1^2)) \\ \vdots \\ x(x_N - x_1) + y(y_N - y_1) = \frac{1}{2}(x_N^2 - x_1^2 + y_N^2 - y_1^2 - (r_N^2 - r_1^2)) \end{cases}$$

En posant $b_i = \frac{1}{2}(x_i^2 - x_1^2 + y_i^2 - y_1^2 - (r_i^2 - r_1^2))$, on obtient :

$$\begin{cases} x(x_2 - x_1) + y(y_2 - y_1) = b_2 \\ x(x_3 - x_1) + y(y_3 - y_1) = b_3 \\ \vdots \\ x(x_N - x_1) + y(y_N - y_1) = b_N \end{cases} \tag{3.2}$$

Ces équations peuvent alors se mettre sous la forme du système matriciel suivant :

$$\mathbf{Ax} = \mathbf{b} \tag{3.3}$$

avec :

$$\mathbf{A} = \begin{pmatrix} (x_2 - x_1) & (y_2 - y_1) \\ (x_3 - x_1) & (y_3 - y_1) \\ \vdots & \vdots \\ (x_N - x_2) & (y_N - y_2) \end{pmatrix}$$

$\mathbf{b} = (b_2, b_3, \ldots, b_N)^T$ et $\mathbf{x} = (x, y)^T$.
La solution de ce système linéaire est donnée par l'équation normale :

$$\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b} \tag{3.4}$$

En effet, en pratique le système $\mathbf{Ax} = \mathbf{b}$ ne sera jamais parfaitement satisfait à cause du bruit de mesure. On aura alors $\mathbf{Ax} \sim \mathbf{b}$. On peut alors définir un vecteur d'erreur $\mathbf{r}$ (appelé aussi résidu) tel que :

$$\mathbf{r} = \mathbf{Ax} - \mathbf{b} \tag{3.5}$$

On va alors cherche à minimiser ce résidu, ou plus exactement sa norme au carré $||\mathbf{r}||^2 = \mathbf{r}^T\mathbf{r}$ :

$$\min_{\mathbf{x}} \mathbf{r}^T\mathbf{r} = \min_{\mathbf{x}} (\mathbf{Ax} - \mathbf{b})^T(\mathbf{Ax} - \mathbf{b}) \tag{3.6}$$

En développant on obtient :

$$\begin{aligned}\mathbf{r}^T\mathbf{r} &= \mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x} - \mathbf{x}^T\mathbf{A}^T\mathbf{b} - \mathbf{b}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{b} \\ &= \mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x} - 2\mathbf{x}^T\mathbf{A}^T\mathbf{b} - \mathbf{b}^T\mathbf{b}\end{aligned}$$

Comme on cherche à minimiser, il faut trouver où s'annule la dérivée de l'équation $\mathbf{r}^T\mathbf{r}$ donnée par :

$$\frac{\partial(\mathbf{r}^T\mathbf{r})}{\partial\mathbf{x}} = 2\mathbf{A}^T\mathbf{A}\mathbf{x} - 2\mathbf{A}^T\mathbf{b}$$

Cette dérivée s'annule quand :

$$\mathbf{A}^T\mathbf{A}\mathbf{x} - \mathbf{A}^T\mathbf{b} = \mathbf{0}$$

On retrouve alors l'équation (3.4) en isolant $\mathbf{x}$ et en divisant par $\mathbf{A}^T\mathbf{A}$ :

$$\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$$

Il existe également d'autres méthodes permettant de résoudre le système linéaire (3.3), la factorisation de Cholesky, la factorisation QR ou la décomposition en valeurs singulières (SVD).

### Méthode 2

Le développement de la méthode est repris de [15]. Le principe de la méthode consiste tout d'abord, comme précédemment, à mettre les équations au carré. Ensuite, on crée une nouvelle variable à estimer $t = x^2 + y^2$, qui va permettre de réécrire le système d'équations sous forme d'un système matriciel à trois inconnues.

En mettant au carré les équations du système (3.1), on obtient :

$$\begin{cases} r_1^2 = x_1^2 + x^2 + y_1^2 + y^2 - 2xx_1 - 2yy_1 \\ r_2^2 = x_2^2 + x^2 + y_2^2 + y^2 - 2xx_2 - 2yy_2 \\ \vdots \\ r_N^2 = x_N^2 + x^2 + y_N^2 + y^2 - 2xx_N - 2yy_N \end{cases}$$

Si l'on définit une troisième inconnue $t$ telle que $t = x^2 + y^2$ et que l'on remplace, on obtient :

$$\begin{cases} r_1^2 = x_1^2 + y_1^2 + t - 2xx_1 - 2yy_1 \\ r_2^2 = x_2^2 + y_2^2 + t - 2xx_2 - 2yy_2 \\ \vdots \\ r_N^2 = x_N^2 + y_N^2 + t - 2xx_N - 2yy_N \end{cases}$$

En regroupant les inconnues du même côté on obtient :

$$\begin{cases} 2xx_1 + 2yy_1 - t = x_1^2 + y_1^2 - r_1^2 \\ 2xx_2 + 2yy_2 - t = x_2^2 + y_2^2 - r_2^2 \\ \vdots \\ 2xx_N + 2yy_N - t = x_N^2 + y_N^2 - r_N^2 \end{cases}$$

Si l'on définit le vecteur des inconnues comme $\mathbf{x} = (x, y, t)^T$, on peut réécrire le système d'équations sous la forme du système matriciel donné par (3.3) :

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

avec

$$\mathbf{A} = \begin{pmatrix} 2x_1 & 2y_1 & -1 \\ 2x_2 & 2y_2 & -1 \\ \vdots & \vdots & \vdots \\ 2x_N & 2y_N & -1 \end{pmatrix}$$

et

$$\mathbf{b} = \begin{pmatrix} x_1^2 + y_1^2 - r_1^2 \\ x_2^2 + y_2^2 - r_2^2 \\ \vdots \\ x_N^2 + y_N^2 - r_N^2 \end{pmatrix}$$

La solution au sens des moindres carrés est comme pour la méthode précédente donnée par l'équation suivante :

$$\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$$

Avec cette méthode, si l'on connaît l'incertitude (la variance) associée au bruit de chaque mesure de distance, il est possible d'obtenir et de résoudre un système des moindres carrés pondérés (**W**eighted **L**east **S**quares). La pondération $\omega_i$ associée à la $i^{\text{ème}}$ mesure est donnée par :

$$\omega_i = \sigma_i^{-2}$$

avec $\sigma_i^2$ la variance associée à la $i^{\text{ème}}$ mesure. La pondération indique le niveau de confiance que l'on a dans la mesure de distance. Les mesures avec une pondération plus importante influenceront plus le résultat que celles ayant un poids plus faible. Cette fois-ci, pour le système à résoudre on ne minimise plus l'erreur au carré comme dans (3.6), mais on minimise la distance de Mahalanobis définie par :

$$d_\Sigma = ||\mathbf{r}||_\Sigma^2 = \mathbf{r}^T\mathbf{\Sigma}^{-1}\mathbf{r}$$

avec $\mathbf{\Sigma}^{-1} = \text{diag}(\frac{1}{\sigma_1^2}, \ldots, \frac{1}{\sigma_i^2}, \ldots, \frac{1}{\sigma_N^2})$, et $\mathbf{r}$ le vecteur des résidus donné par l'équation (3.5).

Si l'on cherche de nouveau à minimiser cette distance $d_\Sigma$, on obtient en suivant le même principe que précédemment :

$$\mathbf{A}^T\mathbf{\Sigma}^{-1}\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{\Sigma}^{-1}\mathbf{b}$$

En posant $\mathbf{C} = \mathbf{A}^T\mathbf{\Sigma}^{-1}\mathbf{A}$ et $\mathbf{d} = \mathbf{A}^T\mathbf{\Sigma}^{-1}\mathbf{b}$, on se ramène à un système du type $\mathbf{C}\mathbf{x} = \mathbf{d}$, que l'on sait déjà résoudre. La solution est donc donnée par :

$$\mathbf{x} = (\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T\mathbf{d} \tag{3.7}$$

En propageant l'incertitude des mesures de distance à travers les équations du système, il est possible d'obtenir la matrice de covariance $\mathbf{P}$ correspondant à la position estimée $\mathbf{x}$. Celle-ci s'obtient à partir de l'équation suivante :

$$\mathbf{P} = \mathbf{T}\mathbf{\Sigma}\mathbf{T}^T$$

avec

$$\mathbf{T} = \mathbf{A}^\dagger \text{diag}(-2r_1, \ldots, -2r_i, \ldots, -2r_N)$$

et $\mathbf{A}^\dagger = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T = \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}$ la pseudo-inverse de Moore Penrose. $\mathbf{T}$ étant le Jacobien du système d'équation par rapport aux mesures de distance. On peut noter qu'ici, la position des balises est considérée comme parfaitement connue, et que seules les mesures de distance sont entachées d'erreurs.

### Méthode 3

Cette méthode suit le même développement que celle précédente avec en plus l'ajout d'un 1 dans le vecteur $\mathbf{x}$ des inconnues. Cet ajout permet d'obtenir un nouveau système linéaire homogène de la forme $\mathbf{Ax} = \mathbf{0}$.

En mettant au carré les équations du système (3.1), on obtient :

$$\begin{cases} r_1^2 = x_1^2 + x^2 + y_1^2 + y^2 - 2xx_1 - 2yy_1 \\ r_2^2 = x_2^2 + x^2 + y_2^2 + y^2 - 2xx_2 - 2yy_2 \\ \vdots \\ r_N^2 = x_N^2 + x^2 + y_N^2 + y^2 - 2xx_N - 2yy_N \end{cases}$$

En posant $t = x^2 + y^2$ et $t_i = x_i^2 + y_i^2$, on obtient :

$$\begin{cases} r_1^2 = t_1 + t - 2xx_1 - 2yy_1 \\ r_2^2 = t_2 + t - 2xx_2 - 2yy_2 \\ \vdots \\ r_N^2 = t_N + t - 2xx_N - 2yy_N \end{cases}$$

En passant toutes les variables du même côté, on obtient :

$$\begin{cases} 2xx_1 + 2yy_1 - t - t_1 + r_1^2 = 0 \\ 2xx_2 + 2yy_2 - t - t_2 + r_2^2 = 0 \\ \vdots \\ 2xx_N + 2yy_N - t - t_N + r_N^2 = 0 \end{cases}$$

Si l'on définit le vecteur des inconnues comme $\mathbf{x} = (x, y, t, 1)^T$, on peut remplacer le système d'équations précédent par le système matriciel suivant :

$$\mathbf{Ax} = \mathbf{0}$$

avec

$$\mathbf{A} = \begin{pmatrix} 2x_1 & 2y_1 & -1 & r_1^2 - t_1 \\ 2x_2 & 2y_2 & -1 & r_2^2 - t_2 \\ \vdots & \vdots & \vdots & \vdots \\ 2x_N & 2y_N & -1 & r_N^2 - t_N \end{pmatrix}$$

Ce système linéaire homogène peut se résoudre en effectuant une décomposition en valeurs singulières (SVD) de la matrice $\mathbf{A}$ [52] :

$$\mathbf{A} = \mathbf{USV}^T$$

La solution $\mathbf{x}$ est alors donnée par la dernière colonne de la matrice $\mathbf{V}^T$ correspondant à la plus petite valeur singulière.

## 3.3.2 Méthode non-linéaire

Le système d'équations donné par (3.1) peut se réécrire sous la forme d'une fonction $\mathbf{f}$ à valeurs dans $\mathcal{R}^N$ qui dépend des paramètres que l'on cherche à estimer ($\mathbf{x} = (x, y)^T$) :

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} \sqrt{(x_1 - x)^2 + (y_1 - y)^2} \\ \sqrt{(x_2 - x)^2 + (y_2 - y)^2} \\ \vdots \\ \sqrt{(x_N - x)^2 + (y_N - y)^2} \end{pmatrix}$$

Ce qui donne le système suivant :

$$\mathbf{f}(\mathbf{x}) = \mathbf{b} \tag{3.8}$$

avec $\mathbf{b} = (r_1, r_2, \ldots, r_N)^T$. En pratique, avec le bruit sur les mesures, l'égalité ne sera jamais vérifiée. On cherche alors à minimiser l'erreur $\mathbf{e}(\mathbf{x}) = \mathbf{f}(\mathbf{x}) - \mathbf{b}$ entre les mesures $\mathbf{b}$ et les prédictions $\mathbf{f}(\mathbf{x})$ faites à partir des paramètres. En réalité, comme précédemment, on va minimiser la norme de l'erreur au carré ou la norme pondérée de l'erreur (distance de Mahalanobis). On cherche donc à résoudre le problème de minimisation non-linéaire suivant :

$$\min_{\mathbf{x}} ||\mathbf{f}(\mathbf{x}) - \mathbf{b}||_2^2 \tag{3.9}$$

$$\Leftrightarrow \min_{\mathbf{x}} ||\mathbf{e}(\mathbf{x})||_2^2 \tag{3.10}$$

Si l'on connaît une première estimation des paramètres $\mathbf{x}_0$, la solution au problème de minimisation précédent (3.9) peut être obtenue de manière itérative en utilisant un algorithme d'optimisation non-linéaire. La méthode de Gauss-Newton comme la majorité des optimiseurs permet, à partir de la solution initiale $\mathbf{x}_0$, de calculer l'incrément $\delta\mathbf{x}$ sur les paramètres qui va permettre de réduire l'erreur $\mathbf{e}(\mathbf{x})$. Pour cela on approxime la fonction d'erreur par un développement de Taylor à l'ordre un :

$$\mathbf{e}(\mathbf{x}_k + \delta\mathbf{x}) \simeq \mathbf{e}(\mathbf{x}_k) + \mathbf{J}\delta\mathbf{x} \tag{3.11}$$

avec $\mathbf{J}$ le Jacobien de $\mathbf{e}(\mathbf{x})$ calculé en $\mathbf{x}_k$ et dont l'expression est donnée par :

$$\mathbf{J} = \frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} = \begin{pmatrix} \frac{-dx_1}{d_1} & \frac{-dy_1}{d_1} \\ \frac{-dx_2}{d_2} & \frac{-dy_2}{d_2} \\ \vdots & \vdots \\ \frac{-dx_N}{d_N} & \frac{-dy_N}{d_N} \end{pmatrix} \tag{3.12}$$

et $dx_i = (x_i - x)$, $dy_i = (y_i - y)$, $d_i = \sqrt{dx_i^2 + dy_i^2}$.

On cherche alors à minimiser :

$$||\mathbf{e}(\mathbf{x}_k + \delta\mathbf{x})||_2^2 = \mathbf{e}(\mathbf{x}_k + \delta\mathbf{x})^T \mathbf{e}(\mathbf{x}_k + \delta\mathbf{x}) \tag{3.13}$$

$$= (\mathbf{e}(\mathbf{x}_k) + \mathbf{J}\delta\mathbf{x})^T (\mathbf{e}(\mathbf{x}_k) + \mathbf{J}\delta\mathbf{x}) \tag{3.14}$$

$$= \delta\mathbf{x}^T \mathbf{J}^T \mathbf{J}\delta\mathbf{x} + 2\mathbf{e}(\mathbf{x}_k)^T \mathbf{J}\delta\mathbf{x} + \mathbf{e}(\mathbf{x}_k)^T \mathbf{e}(\mathbf{x}_k) \tag{3.15}$$

Cela revient alors à résoudre l'équation normale suivante :

$$\mathbf{J}^T \mathbf{J}\delta\mathbf{x} = -\mathbf{J}^T \mathbf{e}(\mathbf{x}) \tag{3.16}$$

qui se résout avec :

$$\delta\mathbf{x} = -(\mathbf{J}^T \mathbf{J})^{-1} \mathbf{J}^T \mathbf{e}(\mathbf{x}) \tag{3.17}$$

La solution est alors cherchée de façon itérative : $\mathbf{x}_{k+1} = \mathbf{x}_k + \delta\mathbf{x}$ jusqu'à atteindre un des critères d'arrêt suivant : convergence de la solution ou atteinte d'un nombre d'itérations donné. Un résumé des étapes de l'algorithme est donné dans l'Algorithme 1.

Un élément essentiel de ce type d'algorithme concerne les valeurs données aux paramètres à estimer au moment de l'initialisation. En effet, les valeurs initiales ne doivent pas être trop éloignée de la vraie solution, sinon l'algorithme risque de diverger ou de rester bloqué dans un minimum local au lieu de converger vers le minimum global. La première estimation de la position peut par exemple être obtenue en utilisant l'un des algorithmes de trilatération présentés précédemment.

**Input :** $\mathbf{x}_0$, $(r_1, r_2, \ldots, r_N)$, nIters, seuil
**Output :** $\mathbf{x}$

$\mathbf{x} = \mathbf{x}_0$ ;
**for** $i = 1 \ldots nIters$ **do**
  $\mathbf{e} = \mathbf{f}(\mathbf{x}) - \mathbf{b}$ ;
  $\mathbf{J} = \ldots$ ;
  $\delta\mathbf{x} = -(\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T\mathbf{e}$ ;
  **if** $\delta\mathbf{x} <$*seuil* **then**
    return $\mathbf{x}$ ;
  **end**
  $\mathbf{x} += \delta\mathbf{x}$
**end**
return $\mathbf{x}$ ;

**Algorithme 1 :** Algorithme de trilatération non-linéaire basé sur la méthode de Gauss-Newton

On peut aussi ajouter des pondérations à chaque mesure en utilisant la distance de Mahalanobis ($||.||^2_{\mathbf{\Sigma}}$) à la place de la norme au carré ($||.||^2_2$). Si on sait que le bruit de chaque mesure de distance $r_i$ a une variance de $\sigma_i^2$, alors on construit la matrice $\mathbf{\Sigma} = \text{diag}(\sigma_1^2, \ldots, \sigma_N^2)$, et on minimise :

$$\min_{\mathbf{x}} ||\mathbf{f}(\mathbf{x}) - \mathbf{b}||^2_{\mathbf{\Sigma}} \tag{3.18}$$

L'équation normale à résoudre devient :

$$(\mathbf{J}^T\mathbf{\Sigma}^{-1}\mathbf{J})\mathbf{\Delta x} = \mathbf{J}^T\mathbf{\Sigma}^{-1}(\mathbf{b} - \mathbf{f}(\mathbf{x})) \tag{3.19}$$

### 3.3.3 Méthode robuste avec RANSAC

Les méthodes de trilatération présentées lors des sections précédentes fonctionnent plutôt bien tant que le bruit sur les mesures de distance reste faible. Par contre, il suffit d'une seule mesure aberrante pour que la solution retournée soit totalement éloignée de la vraie solution. Il est donc essentiel de disposer d'un mécanisme de détection des mesures aberrantes afin de les éliminer et d'employer les algorithmes de trilatération précédents uniquement sur des "bonnes" mesures. Une technique communément employée en vision pour sélectionner les bonnes correspondances de points entre deux images est l'algorithme RANSAC (pour RANdom SAmple Consensus) [46]. La stratégie de l'algorithme est de trouver le modèle, ici la position à estimer, qui s'accorde avec un maximum de données, ici les mesures de distance. Cet algorithme suppose que la majorité des bonnes mesures s'accorde avec le modèle, alors qu'inversement si le modèle est établi avec une mesure aberrante, la plupart des mesures ne s'accorderont pas avec le modèle.

Le fonctionnement de l'algorithme est maintenant présenté directement dans le contexte de la trilatération. Tout d'abord on sélectionne aléatoirement un nombre minimal de mesures de distance (ici trois) afin de calculer la position via l'un des algorithmes de trilatération linéaire précédent. Ensuite, on calcule le support de cette position, c'est-à-dire le nombre de mesures de distance qui s'accordent avec la position. Pour cela on définit une erreur, $e_i$, et un seuil, $t$. Toutes les mesures dont l'erreur correspondante sera inférieure au seuil fixé seront considérées comme des bonnes mesures, alors que les autres seront des mesures aberrantes. Dans le cas présent, l'erreur utilisée est la valeur absolue entre la mesure de distance $r_i$, et la distance $d_i$, prédite à partir de la position estimée par le modèle $(x, y)$ et la position de la balise correspondant $(x_i, y_i)$ :

$$e_i = |r_i - d_i| \tag{3.20}$$

On répète alors $N$ fois les opérations précédentes, et la position ayant obtenue le plus grand support est considérée comme la solution robuste de l'algorithme. Il est enfin possible et recommandé pour une meilleure estimation, de recalculer la position par trilatération en utilisant cette fois-ci toutes les bonnes mesures. Pour affiner encore le résultat, on peut finir par utiliser l'algorithme de trilatération non-linéaire proposé dans la section précédente.

Les paramètres de l'algorithme sont le seuil $t$ et le nombre $N$ d'itérations. Il existe différentes approches permettant de régler au mieux ces paramètres, ici nous utiliserons celles décrites dans [57]. Le seuil $t$ est défini proportionnellement à la déviation standard des mesures de distance. Pour le nombre d'itérations, on adopte une version adaptative de l'algorithme RANSAC, décrite dans [57], et qui permet de déterminer de façon automatique le nombre d'itérations au cours de son exécution. Le détail de l'algorithme en pseudo-code est proposée dans l'Algorithme 2.

**Input :** Mesures de distance $(r_1, r_2, \ldots, r_N)$, seuil $t$
**Output :** Position estimée $(x, y)$, les bonnes mesures correspondantes $\mathcal{S}_{\text{inliers}}$
nIters $= \infty$ ;
ct $= 0$ ;
// Ensemble contenant les bonnes mesures de distance
$\mathcal{S}_{\text{inliers}} = \emptyset$ ;
nInliers $= 0$ ;
**while** *nIters* $>$ *ct* **do**
  Choisir aléatoirement 3 mesures de distance : $r_j, r_k, r_l$ ;
  /* Calculer la position correspondante avec un algorithme de trilatération linéaire */
  $(x, y) =$ trilatLineaire$(r_j, r_k, r_l)$ ;
  // Compter le nombre de bonnes mesures
  $n = 0$ ;
  $\mathcal{S} = \emptyset$ ;
  **for** $i = 1 \ldots N$ **do**
    $d_i = \sqrt{(x_i - x)^2 + (y_i - y)^2}$ ;
    $e_i = |r_i - d_i|$ ;
    **if** $e_i < t$ **then**
      $n = n + 1$ ;
      $\mathcal{S} \leftarrow \mathcal{S} + r_i$ ;
    **end**
  **end**
  // Garder le meilleur ensemble des bonnes mesures
  **if** $n >$ *nInliers* **then**
    nInliers $= n$ ;
    $\mathcal{S}_{\text{inliers}} = \mathcal{S}$ ;
  **end**
  // Calcul adaptatif du nombre d'itérations
  $\epsilon = 1 - (\text{nInliers}/N)$ ;
  nIters $= \log(1 - p)/\log\left(1 - (1 - \epsilon)^s\right)$ ;
  ; // $p = 0.99$, $s = 3$
  ct $=$ ct $+ 1$ ;
**end**
// Recalculer la position avec toutes les bonnes mesures
$(x, y) =$ trilatLineaire$(\mathcal{S}_{\text{inliers}})$ ;
// Optionnel : utiliser un algorithme de trilatération non-linéaire
$(x, y) =$ trilatNonLineaire$\left((x, y), \mathcal{S}_{\text{inliers}}\right)$ ;
return $(x, y)$, $\mathcal{S}_{\text{inliers}}$ ;

**Algorithme 2 :** Algorithme de trilatération robuste avec RANSAC

## 3.4 Localisation robuste par mesures de distance via LMI et SoS

L'un des problèmes majeurs rencontré par la plupart des algorithmes de trilatération, qu'ils soient linéaires ou non-linéaires, provient de leur manque de robustesse aux mesures aberrantes. Il suffit qu'une seule des mesures utilisées dans l'algorithme soit complètement fausse pour que la solution estimée se retrouvent totalement éloignée de la vraie solution. Une méthode permettant

de pallier ce problème a déjà été présentée, il s'agit de la méthode de trilatération avec RANSAC. Cette dernière permet de sélectionner le plus grand nombre de mesures qui supportent une solution du problème. Cette méthode, bien que robuste, ne présente aucune garantie que la solution retournée soit réellement la bonne. Le mécanisme aléatoire de la méthode ne permet pas de dire avec certitude que la solution retournée correspond vraiment à la vérité terrain. Dans cette section, une nouvelle approche permettant d'avoir des garanties sur la solution est présentée. L'algorithme proposé retourne en effet un intervalle, une boîte en 2D, dont on a la garantie que la solution se trouve à l'intérieur. Cette nouvelle approche, qui se rapproche du travail fait dans [35], combine la théorie des inégalités linéaires matricielles (LMIs) [102] et des sommes de polynômes au carré (SoS) [80, 95]. Les LMI [48] sont depuis longtemps utilisées en automatique et sont devenues un outil puissant dans beaucoup de domaines s'appuyant sur l'optimisation. De plus, grâce aux travaux de Nesterov [102] sur la méthode des points intérieurs, les LMI peuvent être résolues de manière rapide et efficace. La méthode présentée ici, est inspirée du travail réalisé dans [108], dans le contexte de la correspondance point/plan en vision par ordinateur.

Le fonctionnement de la méthode a été repris et adapté au contexte de la localisation d'un robot à partir de balises ou le problème inverse qui consiste à déterminer la position des balises à partir de la trajectoire connue du robot. Ainsi, de manière générique, en utilisant uniquement des mesures de distance $r_i$ entre des positions connues $(x_i, y_i)$ et une position $(x, y)$, le but est de déterminer la position $(x, y)$. La méthode consiste à vérifier que toutes les mesures de distance sont compatibles avec une boîte censée contenir la vraie position $(x, y)$. Cette vérification est faite en transformant l'équation des mesures de distance en un polynôme. Si ce polynôme est une somme de carrés (SoS), alors la mesure correspondante est inconsistante avec la boîte. La procédure permettant de trouver la solution, consiste à découper l'espace de recherche en boîtes. Celles qui sont inconsistantes sont écartées, alors que les autres sont subdivisées jusqu'à atteindre la boîte solution. Comme il sera expliqué par la suite, le problème SoS peut être transformé en un problème de faisabilité de LMI, qu'on peut alors facilement résoudre. Un avantage de cette méthode est qu'il n'est pas nécessaire de déterminer un modèle du bruit des mesures ou de faire des hypothèses sur le processus de générations des mesures de distance. Il est aussi possible de réduire la taille des boîtes retournées, en tolérant un certain pourcentage des mesures aberrantes qui ne seraient pas cohérentes avec la solution estimée. Ce mécanisme permet ainsi de détecter les mesures aberrantes et d'affiner la taille de la boîte contenant la vraie solution.

Dans la suite de cette section, la formulation du problème est d'abord présentée avec quelques rappels sur la théorie des SoS et des LMIs. Puis, l'algorithme proposé est expliqué en détails avec notamment la procédure permettant de vérifier si une boîte donnée est inconsistante avec une mesure de distance. Enfin, les résultats de la méthode sur des simulations et de jeux de données sont présentés, ainsi qu'une comparaison avec d'autres techniques de trilatération présentant des degrés de robustesse différents.

### 3.4.1 Présentation des notions utilisées par l'algorithme

Avant de présenter la formulation de la méthode, voici d'abord quelques informations sur les concepts utilisées.

#### Fonctions d'appartenance

La fonction d'appartenance permet d'avoir une expression mathématique de l'appartenance d'une variable à un intervalle. Soit l'intervalle $[\underline{x}, \overline{x}]$ dans lequel se trouve $x$, la fonction d'ap-

partenance, appelée $g(x)$ est donnée par :

$$g(x) = (x - \underline{x})(\overline{x} - x) \tag{3.21}$$

Cette fonction a pour propriété d'être positive quand $x$ est à l'intérieur de l'intervalle et négative quand il est à l'extérieur, voir Figure 3.4. On a donc l'équivalence suivante :

$$x \in [\underline{x}, \overline{x}] \iff g(x) \geq 0$$

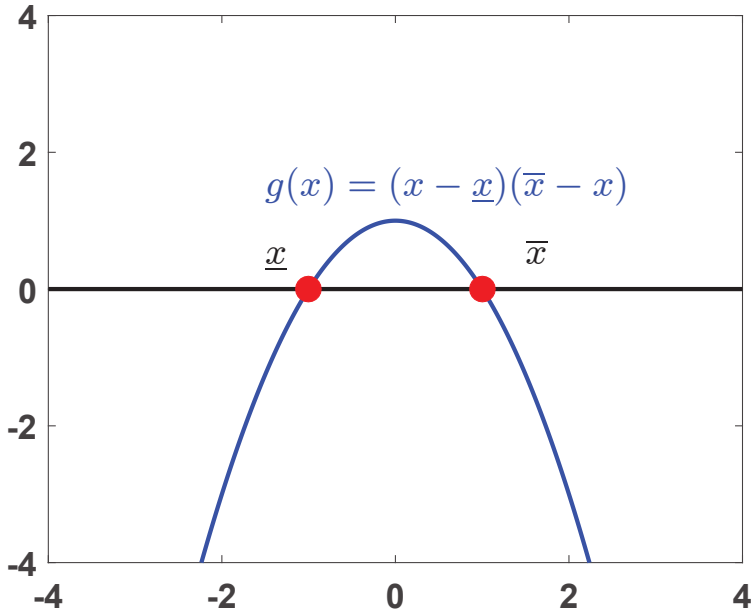



FIGURE 3.4 – Fonction d'appartenance $g(x)$

La notion d'intervalle à une dimension peut être étendue à des dimensions plus élevées. En 2D, on appelle cela une boîte et on la définit comme le produit de deux intervalles (un pour chaque dimension) :

$$\mathbb{B} = \left\{ \mathbf{x} = (x, y) \in \mathbb{R}^2 \mid (x, y) \in [\underline{x}, \overline{x}] \times [\underline{y}, \overline{y}] \right\} \tag{3.22}$$

En utilisant une formulation avec les fonctions d'appartenances, une boîte $\mathbb{B}$ se définit de façon équivalente par :

$$\mathbb{B} := \left\{ \mathbf{x} \in \mathbb{R}^2 \,\middle|\, g_x(\mathbf{x}) \geq 0,\ g_y(\mathbf{x}) \geq 0 \right\} \tag{3.23}$$

avec :

$$\begin{cases} g_x(\mathbf{x}) = (x - \underline{x})(\overline{x} - x) \\ g_y(\mathbf{x}) = (y - \underline{y})(\overline{y} - y) \end{cases}$$

### Théorie des SoS et LMIs

On note $\mathcal{S}^{n \times n}$ l'ensemble des matrices symétriques de taille $n \times n$, et $\mathbb{R}_{2d}[\mathbf{x}]$ l'ensemble des polynômes de degré $2d$ à coefficients réels. Un polynôme $p(\mathbf{x}) = p(x_1, \ldots, x_n) \in \mathbb{R}_{2d}[\mathbf{x}]$ est positif si :

$$p(\mathbf{x}) \geq 0, \forall \mathbf{x} \in \mathcal{R}^n$$

Un polynôme $p(\mathbf{x})$ est une somme de carrés (SoS) s'il existe des polynôme $q_i(\mathbf{x})$ tels que :

$$p(\mathbf{x}) = \sum_i q_i^2(\mathbf{x})$$

Si un polynôme $p(\mathbf{x})$ est SoS alors il est positif. Un polynôme $p(\mathbf{x})$ de degré $2d$ est SoS si et seulement si il existe une matrice symétrique définie positive $\mathbf{G} \in \mathcal{S}$ telle que :

$$p(\mathbf{x}) = m(\mathbf{x})^T \mathbf{G} m(\mathbf{x})$$

avec $m(\mathbf{x})$ le vecteur de tous les monômes de degré inférieur à $d$. La matrice $\mathbf{G}$ est souvent appelée matrice de Gram [110] et a pour dimensions : $\binom{n+d}{d} \times \binom{n+d}{d}$, avec $\binom{n}{k}$ un coefficient binomial.

Si la matrice $\mathbf{G}(\mathbf{l})$ est une matrice symétrique affine par rapport au vecteur des paramètres $\mathbf{l}$, alors :

$$\mathbf{G}(\mathbf{l}) \geq \mathbf{0}$$

est appelée une inégalité linéaire matricielle. Pour une matrice symétrique réelle, $\mathbf{G} \geq 0$ signifie que toutes ses valeurs propres sont positives et réelles. Tester la positivité d'un polynôme $p(\mathbf{x}, \mathbf{l}) = m(\mathbf{x})^T \mathbf{G}(\mathbf{l}) m(\mathbf{x})$, peut donc se faire en vérifiant la faisabilité d'une LMI [95, 110] :

$$\mathbf{G}(\mathbf{l}) \geq \mathbf{0} \Rightarrow p(\mathbf{x}, \mathbf{l}) \text{ SoS}$$

Finalement, la positivité du polynôme $p(\mathbf{x})$ peut être restreinte à un domaine particulier. Par exemple en 2D, $\mathbf{p}(\mathbf{x})$ est positif sur $\mathbb{B}$, s'il existe des scalaires positifs $\sigma_x$ et $\sigma_y$ tels que :

$$\mathbf{p}(\mathbf{x}) - \sigma_x g_x(\mathbf{x}) - \sigma_y g_y(\mathbf{x})$$

soit SoS.
En effet, si $\mathbf{x} \in \mathbb{B}$, alors $g_x(\mathbf{x}) \geq 0$ et $g_y(\mathbf{x}) \geq 0$. De plus, si :

$$\begin{aligned} &\mathbf{p}(\mathbf{x}) - \sigma_x g_x(\mathbf{x}) - \sigma_y g_y(\mathbf{x}) \text{ SoS} \\ \Rightarrow &\mathbf{p}(\mathbf{x}) - \sigma_x g_x(\mathbf{x}) - \sigma_y g_y(\mathbf{x}) \geq 0 \\ \Leftrightarrow &\mathbf{p}(\mathbf{x}) \geq \sigma_x g_x(\mathbf{x}) + \sigma_y g_y(\mathbf{x}) \geq 0 \end{aligned}$$

donc $\mathbf{p}(\mathbf{x})$ est positif pour tout $\mathbf{x} \in \mathbb{B}$.

### 3.4.2 Description de l'algorithme

Avant de présenter l'algorithme en détails, un bref survol des différentes étapes est d'abord introduit. Ensuite, la partie centrale de l'algorithme est présentée avec les équations permettant d'arriver au test de consistance avec les LMI. Enfin, la procédure permettant d'explorer et de chercher la plus petite boîte compatible avec la majorité des mesures de distance est expliquée.

#### Description générale

Pour le problème de localisation considéré, on suppose que l'on possède $N$ mesures de distance bruitées $r_i$ entre des positions statiques connues $\mathbf{x}_i$ :

$$\mathbf{x}_i = (x_i, y_i)^T,\ i \in [1, N]$$

et une position $\mathbf{x}$ :

$$\mathbf{x} = (x, y)^T$$

On cherche alors à déterminer la position inconnue $\mathbf{x}$ en accord avec toutes les mesures ou une certaine fraction d'entre elles. Les mesures de distance sont les seules informations utilisées pour estimer la position $\mathbf{x}$.

Une mesure de distance, $r_i$, entre les positions $\mathbf{x}$ et $\mathbf{x}_i$ peut être modélisée par :

$$r_i = d_i$$

avec $d_i = \sqrt{(x_i - x)^2 + (y_i - y)^2}$.

La méthode proposée est basée sur la possibilité de tester si une mesure de distance $r_i$ est incompatible avec une boîte englobant la position $\mathbf{x}$ à estimer. Pour cela, à partir d'une mesure de distance $r_i$, on définit une contrainte $f_i(\mathbf{x})$ comme étant la différence entre la mesure au carrée et la distance prédite au carrée :

$$f_i(\mathbf{x}) = r_i^2 - d_i^2 \tag{3.24}$$

La forme de cette fonction de contrainte pour une mesure de distance est visible à la Figure 3.5. On remarquera que la fonction s'annule pour les positions situées sur un cercle de rayon $r_i$ et de centre $\mathbf{x}_i$.

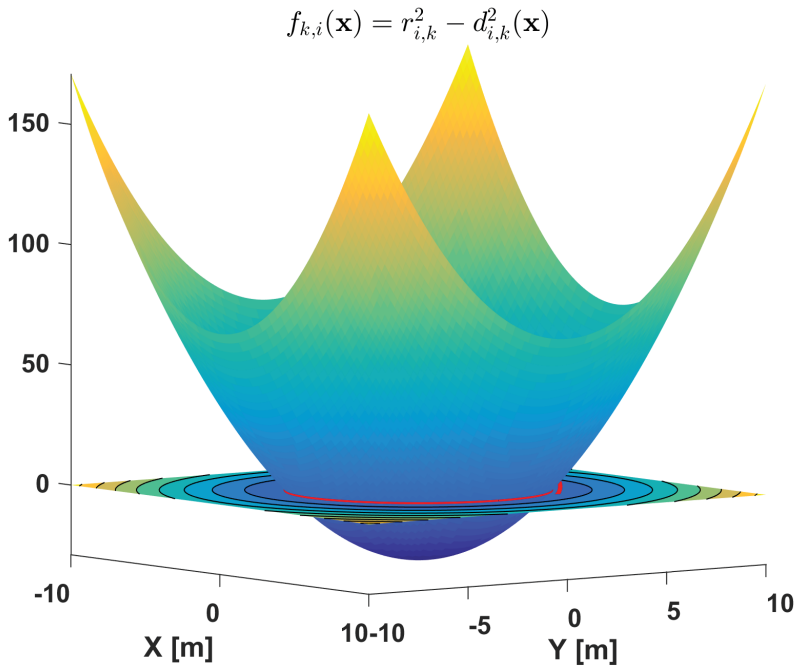



FIGURE 3.5 – Fonction de contrainte sur la mesure de distance $f_i(\mathbf{x})$

Une mesure de distance $r_i$ est compatible avec la boîte $\mathbb{B}$ si la fonction $f_i(\mathbf{x})$ passe par zéro en un point à l'intérieur de la boîte :

$$f_i(\mathbf{x}) = r_i^2 - d_i^2 = 0 \tag{3.25}$$

Si l'on arrive à prouver que la fonction $f_i(\mathbf{x})$ est strictement positive ou négative dans $\mathbb{B}$, alors on est sûr que $r_i$ est incompatible avec la boîte $\mathbb{B}$ considérée. On peut remarquer que la contrainte de l'équation (3.24) est quadratique par rapport à l'inconnue $\mathbf{x}$, ce qui permet alors d'appliquer directement la théorie des polynômes SoS. Dans la suite de cette section, la contrainte qui vient d'être présentée sera utilisée afin de développer une méthode permettant de retrouver la plus petite boîte associée à la position $\mathbf{x}$ qui aura le moins de mesures incompatibles voir aucunes, en testant la positivité de cette contrainte à l'intérieur de la boîte. Le test de consistance réalisé pour une boîte et une mesure de distance est maintenant détaillé.

### Test de consistance LMI

Le test de consistance LMI est le suivant : si la fonction $f_i(\mathbf{x})$ est strictement positive sur la boîte $\mathbb{B}$ :

$$f_i(\mathbf{x}) > 0,\ \forall \mathbf{x} \in \mathbb{B}$$

alors la mesure de distance $r_i$ correspondante est incompatible avec cette boîte $\mathbb{B}$. Pour cela, la condition précédente est testée en vérifiant la positivité de la fonction polynomiale $h_i(\mathbf{x})$, construite à partir de la contrainte donnée par (3.24) et des polynômes d'appartenance définis dans (3.21) :

$$h_i(\mathbf{x}) = \lambda f_i(\mathbf{x}) - \sum_{j=\{x,y\}} g_j(\mathbf{x})\sigma_j \tag{3.26}$$

ou $\lambda = \pm 1$ et $\sigma_j > 0$. La variable $\lambda$ est ajoutée afin de prendre en compte les cas $\pm f_i(\mathbf{x}) > 0$. L'équation (3.26) est réécrite sous forme quadratique :

$$h_i(\mathbf{x}) = \mathbf{m}^T(\mathbf{x})\mathbf{G}(\lambda, \sigma_x, \sigma_y)\mathbf{m}(\mathbf{x}) \tag{3.27}$$

où $\mathbf{G}(\lambda, \sigma_x, \sigma_y)$ est la matrice de Gram du polynôme $h_i(\mathbf{x})$ et $\mathbf{m}(\mathbf{x}) = (x, y, 1)^T$ le vecteur des monômes de degrés inférieurs à deux.

On obtient donc que, pour une boîte donnée $\mathbb{B}$, s'il existe un scalaire réel $\lambda = \pm 1$ et des scalaires positifs $\sigma_j > 0$, $j = \{x, y\}$, tels que le polynôme $h_i(\mathbf{x})$ soit SoS, alors $\lambda f_i(\mathbf{x}) > 0$ pour tout $\mathbf{x} \in \mathbb{B}$. En effet, si $h_i(\mathbf{x})$ est SoS, alors $h_i(\mathbf{x}) > 0$ et donc :

$$\begin{aligned} &\lambda f_i(\mathbf{x}) - \sum_{j=\{x,y\}} g_j(\mathbf{x})\sigma_j > 0 \\ \Leftrightarrow\ &\lambda f_i(\mathbf{x}) > \sum_{j=\{x,y\}} g_j(\mathbf{x})\sigma_j \end{aligned}$$

Or si, $\mathbf{x} \in \mathbb{B}$, alors $g_j(\mathbf{x}) > 0$ et comme $\sigma_j > 0$, on a donc $\sum_{j=\{x,y\}} g_j(\mathbf{x})\sigma_j > 0$, d'où :

$$\begin{aligned} &\lambda f_i(\mathbf{x}) > \sum_{j=\{x,y\}} g_j(\mathbf{x})\sigma_j > 0 \\ \Rightarrow\ &\lambda f_i(\mathbf{x}) > 0 \end{aligned}$$

$f_i(\mathbf{x})$ est donc soit strictement positive, soit strictement négative sur $\mathbb{B}$, et ne contient ainsi aucun point pour lequel $f_i(\mathbf{x}) = 0$. La mesure de distance $r_i$ est alors garantie d'être incompatible avec la boîte $\mathbb{B}$. Dans le cas contraire, $r_i$ est potentiellement une bonne mesure.

La vérification que le polynôme $h_i(\mathbf{x})$ est SoS se fait en vérifiant que sa matrice $\mathbf{G}(\lambda, \sigma_x, \sigma_y)$ est définie positive : $\mathbf{G}(\lambda, \sigma_x, \sigma_y) \geq 0$. Cette dernière condition se vérifie en résolvant le problème de faisabilité d'une LMI pour les variables $\lambda$ and $\sigma_j$, $j = \{x, y\}$. En effet, la matrice $\mathbf{G}(\lambda, \sigma_x, \sigma_y)$ peut être décomposée comme suit :

$$\mathbf{G}(\lambda, \sigma_x, \sigma_y) = \mathbf{G}_0 + \lambda\mathbf{G}_1 + \sigma_x\mathbf{G}_2 + \sigma_y\mathbf{G}_3 \tag{3.28}$$

avec

$$\mathbf{G}_0 = \mathbf{0}_{3\times 3} \tag{3.29}$$

$$\mathbf{G}_1 = \begin{pmatrix} 1 & 0 & -x_i \\ 0 & 1 & -y_i \\ -x_i & -y_i & x_i^2 + y_i^2 - r_i^2 \end{pmatrix} \tag{3.30}$$

$$\mathbf{G}_2 = \begin{pmatrix} 1 & 0 & -\frac{1}{2}(\underline{x} + \overline{x}) \\ 0 & 0 & 0 \\ -\frac{1}{2}(\underline{x} + \overline{x}) & 0 & \underline{x}\overline{x} \end{pmatrix} \tag{3.31}$$

$$\mathbf{G}_3 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & -\frac{1}{2}(\underline{y} + \overline{y}) \\ 0 & -\frac{1}{2}(\underline{y} + \overline{y}) & \underline{y}\overline{y} \end{pmatrix} \tag{3.32}$$

Lors de la construction du système LMI, il ne faut pas oublier d'imposer les contraintes de positivité sur les $\sigma_j$. De plus, le $\lambda = \pm 1$ est traité en testant séparément les deux cas.

En résumé, si le problème LMI est faisable, alors on a $\mathbf{G}(\lambda, \sigma_x, \sigma_y) \succeq 0$, et donc $h_i(\mathbf{x})$ est SoS, ce qui signifie que la mesure de distance est une mesure aberrante pour la boîte considérée. Dans le cas contraire, si le solveur LMI n'a pas trouvé de solution, la mesure de distance est potentiellement une bonne mesure. En réalisant le test venant d'être décrit pour l'ensemble des mesures de distance, il est possible de définir si une boîte est consistante ou non. Si une ou un pourcentage des mesures est inconsistant avec une boîte donnée, cette dernière est écartée. Il reste maintenant à établir une procédure permettant de rechercher la plus petite boîte compatible avec l'ensemble des mesures. Dans le paragraphe suivant, une telle procédure, basée sur un algorithme de parcours en largeur, est présentée.

### Algorithme complet

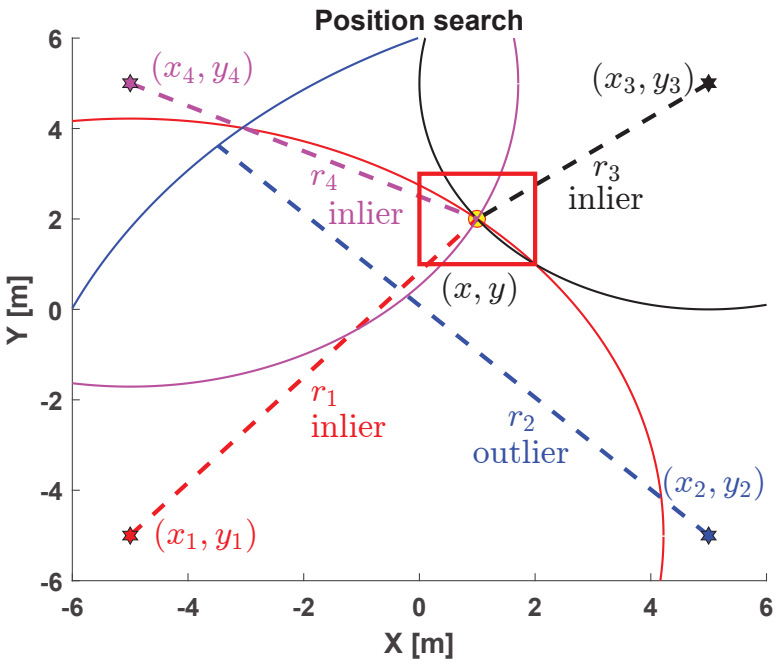



FIGURE 3.6 – Illustration du procédé de recherche de la plus petite boîte compatible avec le maximum de mesures de distance

Le test précédent permet uniquement de déterminer si une mesure de distance est incompatible avec une boîte donnée. Afin d'estimer la position à partir de mesures de distance, il est

nécessaire d'utiliser et de combiner au minimum trois mesures. De plus, pour obtenir l'estimation la plus précise possible, il faut définir un processus permettant de rechercher la plus petite boîte consistante avec un maximum des mesures de distance. Pour cela, la technique présentée est basée sur un algorithme de parcours en largeur (**B**readth-**F**irst **S**earch, [82]). L'algorithme prend comme paramètres d'entrée les mesures de distance avec la position des balises associées, une boîte de départ $\mathbb{B}_0$, et deux seuils. Le premier seuil permet de contrôler le pourcentage de mesures aberrantes $t_{outliers}$ que l'on souhaite rejeter, alors que le second sert à stopper l'algorithme quand une certaine précision est atteinte (l'aire de la boîte). En démarrant avec une boîte initiale assez grande, ou plus petite si l'on a déjà des connaissances à priori sur la zone à chercher, l'algorithme va, de manière récursive, tester les boîtes. Dans le cas où le test de consistance est positif pour une boîte, cette dernière est divisée en boîtes petites boîtes. Si le test de consistance échoue, la boîte est écartée de la recherche. La procédure est répétée jusqu'à ce que plus aucune boîte ne soit consistante avec les mesures de distance. Le test de consistance est réalisé pour chaque mesure de distance comme expliqué précédemment. Si une mesure aberrante est détectée, ou si le pourcentage de mesures aberrantes tolérées atteint le seuil prédéfini, alors la boîte est écartée. Le réglage du pourcentage de mesures aberrantes est très utile et efficace quand les mesures sont fortement bruitées comme cela est montré lors des expériences présentées par la suite. Pour savoir si une boîte doit être explorée ou non, l'algorithme maintient une liste de boîtes consistantes qui restent encore à explorer. Si une boîte passe le test de consistance, alors elle est ajoutée à la liste pour être explorer par la suite. Le processus est itéré jusqu'à ce qu'il n'y ait plus de boîtes à explorer dans la liste ou qu'un niveau de précision donné est atteint. Le niveau de précision est défini comme étant l'aire de la boîte considérée, le seuil est fixé à 1 $cm^2$ lors des expériences. Le fonctionnement détaillé du processus de recherche est présenté dans l'Algorithme 3. Une illustration du principe de la localisation d'une position à partir de quatre mesures de distance, dont l'une d'entre elles (balise 2), est aberrante est visible à la Figure 3.6. La boîte considérée (en rouge) est compatible avec seulement trois mesures de distance car les cercles correspondants intersectent la boîte. Ceci n'est pas le cas pour la balise 2, dont le cercle de la mesure de distance n'intersecte jamais la boîte. En choisissant un pourcentage des mesures aberrantes à 0.25, l'algorithme sera capable de retourner une boîte rouge de faibles dimensions autour du point jaune correspondant à la vraie solution. La subdivision des boîtes s'avère être une étape délicate. Dans l'implémentation actuelle, une boîte est divisée en quatre sous-boîte en coupant chaque dimension en deux. Cependant, si la vraie position à estimer se trouve sur la frontière ou tout proche de la séparation de deux boîtes, alors il est possible que l'algorithme rejette la boîte alors qu'en réalité elle est valide. Ce phénomène est d'autant plus amplifié que le bruit sur les mesures est grand. Il est possible de contourner ce problème en divisant les boîtes suivant deux schémas différents afin d'éviter d'avoir les mêmes frontières, mais au prix d'une augmentation du nombre de boîtes à explorer. Ce phénomène est expliqué plus en détail dans la suite ou les résultats des expériences sont présentés.

### 3.4.3 Simulations

Après avoir présenté en détails l'algorithme proposé, nous allons maintenant valider son bon fonctionnement en présence de différents niveaux et types de bruit. Ensuite, une comparaison avec d'autres algorithmes de trilatération plus ou moins robustes aux mesures erronées est réalisée. Les simulations ont été effectuées en utilisant trois des jeux de données : Gesling2, Plaza1 et Plaza2, présentés dans [32] [2]. Afin de maîtriser correctement les perturbations appliquées aux mesures de distance, les mesures provenant de la vérité terrain ont d'abord été générées, puis les différents bruits ont été ajoutés. Les différents types et niveaux de bruits utilisés pour les tests sont :

2. http://www.frc.ri.cmu.edu/projects/emergencyresponse/RangeData

**Input** : Positions : $(\mathbf{x}_1, \ldots, \mathbf{x}_i, \ldots, \mathbf{x}_n)$, Range measurements : $\mathbf{r} = (r_1, \ldots, r_i, \ldots, r_n)$, Inital box : $\mathbb{B}_0 := (\underline{\mathbf{x}_0}, \overline{\mathbf{x}_0})$, Outliers threshold : $t_{outliers}$
**Output** : Smallest box : $\mathbb{B}_s := (\underline{\mathbf{x}_s}, \overline{\mathbf{x}_s})$

```
// List of boxes to test
list.append(𝔹_0)
// Initialize first solution
𝔹_s ← 𝔹_0
while list not empty do
    // Extract first box in the list
    𝔹 ← list.pop()
    // Split box
    [𝔹_1, ..., 𝔹_k, ...] ← split(𝔹)
    foreach 𝔹_k do
        nRanges = 0
        nOutliers= 0
        foreach x_i do
            if r_i exists then
                // Test if r_i respects the condition
                isOutlier = LmiTest(x_i, r_i,𝔹_k)
                if isOutlier == True then
                    nOutliers++
                end
                nRanges++
            end
        end
        outlierRatio = nOutliers / nRanges
        if outlierRatio < t_outliers then
            list.append(𝔹_k)
            if area(𝔹_k) < area(𝔹_s) then
                𝔹_s ← 𝔹_k
            end
        end
    end
end
```

**Algorithme 3 :** Algorithme de recherche de la plus petite boîte cohérente avec les mesures de distance

— Sans bruit : les mesures de distance correspondent à la vérité terrain,

— Std1 : ajout d'un bruit blanc Gaussien de moyenne nulle et d'écart-type égal à 0.0333 m,

— Std2 : bruit blanc Gaussien de moyenne nulle et d'écart-type égal à 0.167 m,

— Std1+biais : bruit blanc Gaussien de moyenne nulle et de d'écart-type égal à 0.0333 m et ajout d'un biais de 0.2 m,

— Std1+bias+10% outliers : bruit blanc Gaussien de moyenne nulle et d'écart-type égal à 0.0333 m avec un biais de 0.2 m, et dont 10% des mesures sont erronées,

— Std1+$x$% outliers : bruit blanc Gaussien de moyenne nulle et d'écart-type égal à 0.0333 m, dont $x$% des mesures sont erronées ($x$=10%,20%,30%,40%,50%).

Ainsi, pour chacun des trois jeux de données 100 positions du robot sont utilisées pour calculer les vraies mesures de distance par rapport aux balises avant d'ajouter les différents bruits précités. Les jeux de données Plaza1 et Plaza2 sont utilisés dans le but d'estimer la position

des balises à partir des positions du robot, alors que le jeu de données Gesling2, lui, utilise la position des balises pour estimer les positions du robot.

Nous allons maintenant analyser les résultats de la méthode proposée pour les différents niveaux de bruit avant de passer à la comparaison avec d'autres méthodes, notamment par rapport à la robustesse aux mesures aberrantes.

| Bruits | Balise 1 | Balise 2 | Balise 3 | Balise 4 |
|---|---|---|---|---|
| Sans bruit | 0.044 | 0.042 | 0.068 | 0.017 |
| Std1 | 0.068 | 0.078 | 0.088 | 0.058 |
| Std2 | 0.641 | 0.078 | 0.848 | 0.365 |
| Std1+biais | 0.641 | 0.250 | 0.848 | 0.084 |
| Std1+biais+10% outliers | 0.190 | 0.250 | 0.848 | 0.084 |
| Std1+10% outliers | 0.096 | 0.086 | 0.088 | 0.058 |
| Std1+20% outliers | 0.096 | 0.086 | 0.088 | 0.058 |
| Std1+30% outliers | 0.060 | 0.086 | 0.120 | 0.157 |
| Std1+40% outliers | 0.096 | 0.086 | 0.120 | 0.017 |
| Std1+50% outliers | 0.096 | 0.078 | 0.224 | 0.017 |

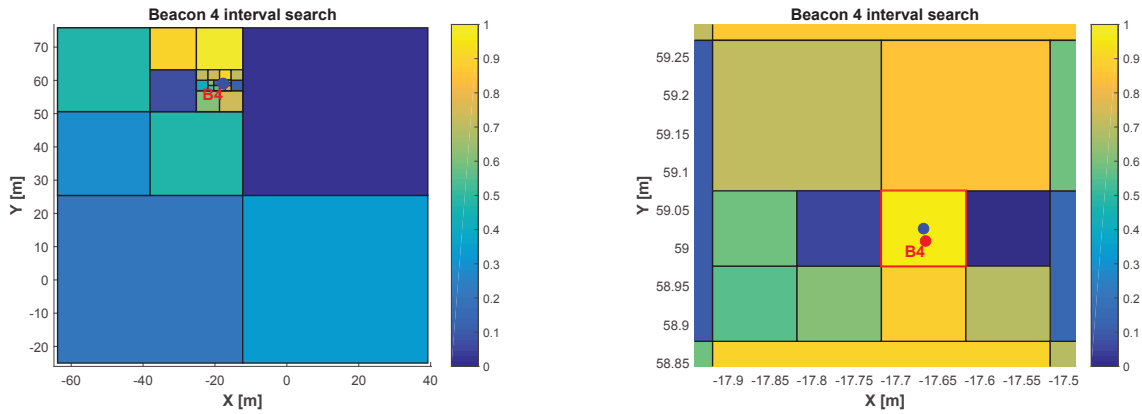
Tableau 3.1 – Erreurs d'estimation en mètres de la position des balises pour l'approche LMI+SoS sur le jeu de données Plaza1

| Bruits | Balise 1 | Balise 2 | Balise 3 | Balise 4 |
|---|---|---|---|---|
| Sans bruit | 0.009 | 0.009 | 0.018 | 0.010 |
| Std1 | 0.045 | 0.246 | 0.129 | 0.082 |
| Std2 | 0.310 | 0.294 | 0.264 | 0.633 |
| Std1+biais | 0.106 | 1.715 | 1.586 | 0.392 |
| Std1+biais+10% outliers | 0.106 | 0.199 | 0.264 | 0.197 |
| Std1+10% outliers | 0.045 | 0.136 | 0.068 | 0.082 |
| Std1+20% outliers | 0.045 | 0.136 | 0.264 | 0.082 |
| Std1+30% outliers | 0.045 | 0.085 | 0.068 | 0.082 |
| Std1+40% outliers | 0.031 | 0.085 | 0.068 | 0.082 |
| Std1+50% outliers | 0.045 | 0.031 | 0.068 | 0.079 |

Tableau 3.2 – Erreurs d'estimation en mètres de la position des balises pour l'approche LMI+SoS sur le jeu de données Plaza2

### Sans bruit

En regardant les erreurs de position des balises des jeux de données Plaza1 et Plaza2 données dans les tableaux 3.1 et 3.2 sur la première ligne (sans bruit), on constate que les erreurs sont très petites mais non nulles. Si les erreurs ne sont pas nulles cela vient de la façon de calculer la position estimée par l'approche. En effet, l'algorithme utilisant des boîtes pour englober la solution, la position estimée est approximée par le centre de la boîte finale retournée par l'algorithme. En regardant la Figure 3.8a, qui montre la position finale estimée des balises pour le jeu de données Plaza1 sans bruit, on ne peut distinguer les vraies positions des positions estimées par l'algorithme car celles-ci se confondent. Cependant, en regardant l'aire des boîtes finales retournées (indiquée par a=... sur les figures), on constate qu'elles sont toutes égales à 0.01 $m^2$, qui correspond au seuil imposé sur l'aire permettant de stopper l'algorithme. Le processus de recherche de la plus petite boîte par l'algorithme BFS est illustré à la Figure 3.7




(a) Balise 4 (b) Balise 4 (zoom)

FIGURE 3.7 – Illustration du processus de recherche des intervalles pour la balise 4 du jeu de données Plaza1 sans bruit

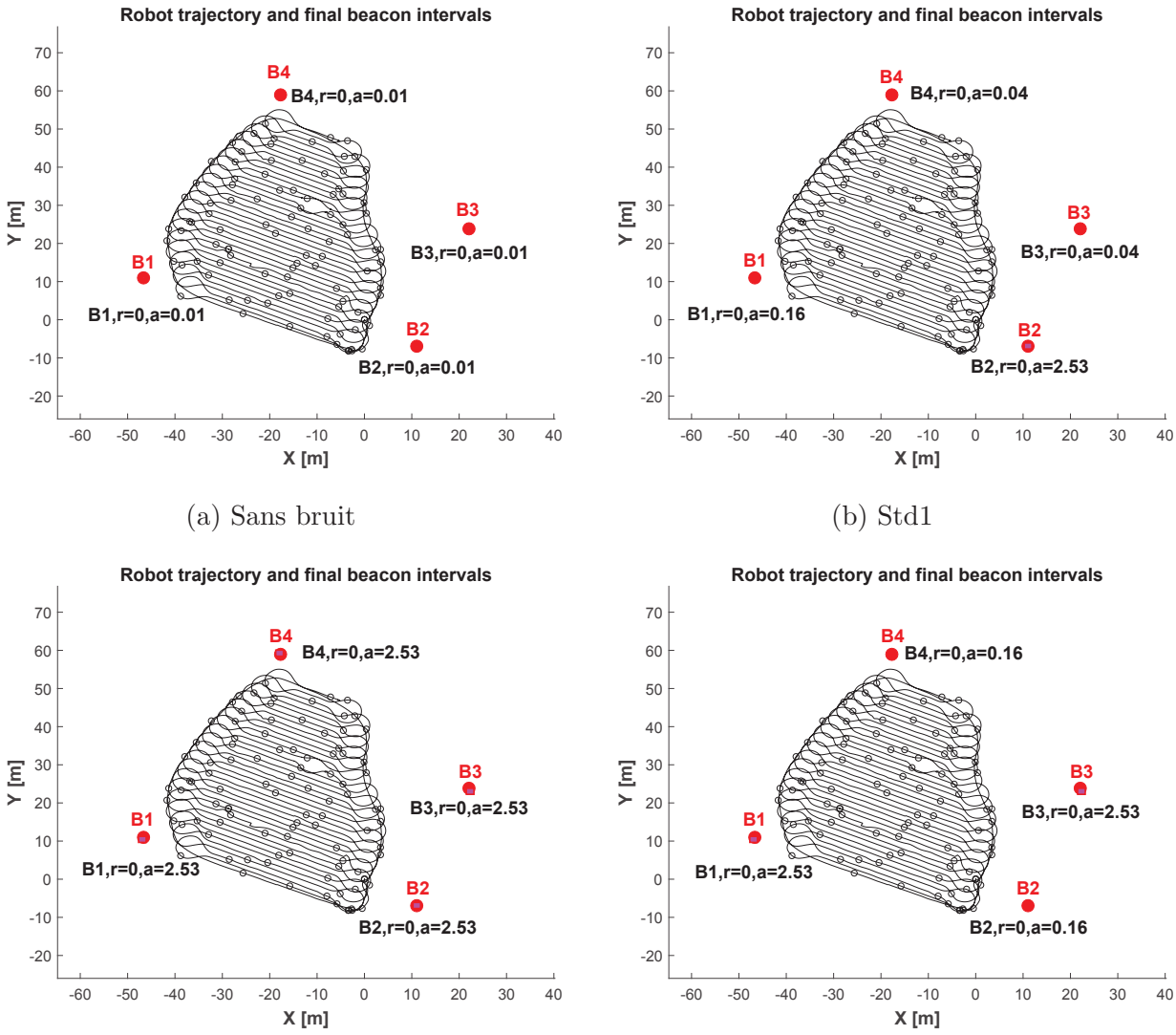
pour la balise 4 du jeu de donnée Plaza1 sans bruit. En regardant le zoom sur l'intervalle final retourné, on voit bien que le point bleu au milieu de l'intervalle jaune cerclé de rouge qui correspond à la position estimée est proche de la vraie position représentée par le point rouge. On constate donc que l'algorithme proposé est capable, sans bruit, de retourner à chaque fois un intervalle de taille minimale englobant la vraie solution.

### Avec bruit Gaussien et biais

En ajoutant un bruit Gaussien, on s'aperçoit que l'erreur d'estimation augmente légèrement par rapport au cas sans bruit et ce d'autant plus que la variance du bruit est importante. En effet, si l'on compare les trois premières lignes des tableaux 3.1 et 3.2, on constate une augmentation de l'erreur de position des balises d'une ligne à l'autre. De même, pour la taille de la boîte retournée, on peut voir en comparant les figures 3.8a, 3.8b et 3.8c que l'aire des boîtes retournées (a=...) augmentent en même temps que la déviation standard.

Quand on ajoute du biais (Std1+biais), on s'aperçoit en comparant les lignes 2 et 4 des tableaux 3.1 et 3.2, que l'erreur de position est plus importante en présence de biais. En comparant les figures 3.8b et 3.8d, on constate aussi que la taille (aire) des boîtes augmente avec l'ajout d'un biais. De plus, en regardant les agrandissements sur la position estimée des balises 2 et 3 avec le bruit Std1+biais, donnés aux figures 3.9a et 3.9b, on remarque pour la balise 2, que la boîte retournée par la méthode proposée ne contient plus la vraie position. Pour la balise, la vraie position se trouve juste sur la limite de la boîte retournée. Ceci montre que la méthode proposée échoue à contenir la vraie position à l'intérieur de la boîte retournée en présence de biais sur les mesures de distance. La méthode ne garantit donc pas que la solution se trouve à l'intérieur de la boîte avec du biais. Le cas de la balise 3, lui, illustre le problème du processus de subdivision d'une boîte en plus petites boîtes. En effet, quand la subdivision se fait très près de la vraie position à estimer, cela peut avoir pour conséquence de stopper l'algorithme plus tôt que prévu et ceci d'autant plus que le bruit sur les mesures est important. Quand la limite d'une boîte se trouve sur la vraie position, le test de consistance risque de désigner beaucoup plus de mesures comme erronées et ainsi d'écarter la boîte alors qu'elle était bonne.

On peut donc dire que l'algorithme proposé fonctionne encore très bien avec un bruit Gaussien, et cela même avec une déviation standard correspondant à un bruit sur les mesures de distance de l'ordre de $\pm 0.5$ m (Std2). Par contre, en ajoutant du biais, l'algorithme n'est plus capable de garantir que la vraie position se trouvera effectivement à l'intérieur de la boîte retournée, mais retournera quand même une position estimée assez proche de la vraie solution.




(a) Sans bruit

(b) Std1

(c) Std2

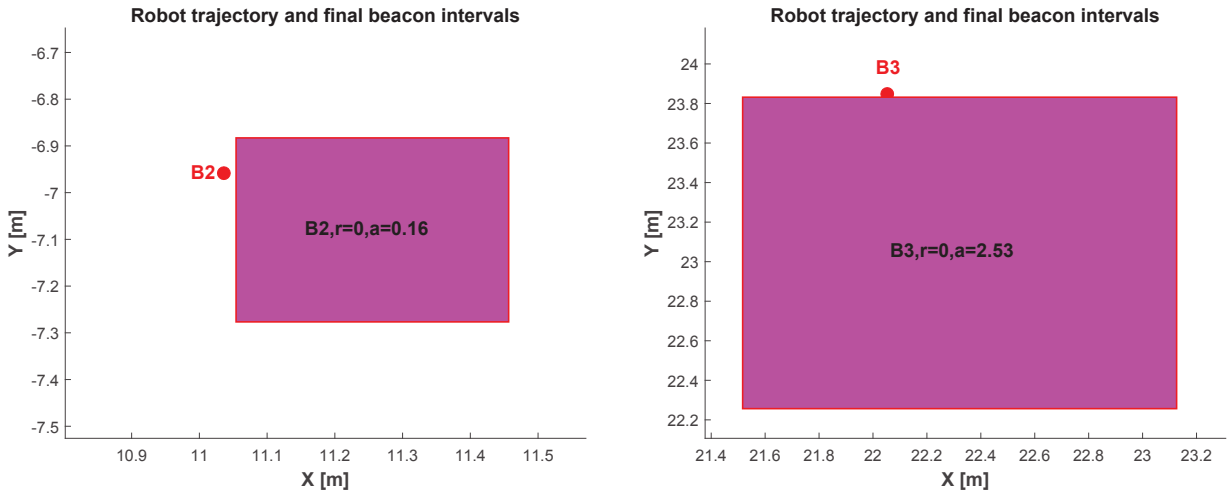
(d) Std1+biais

FIGURE 3.8 – Positions finales estimées des balises pour le jeu de données Plaza1 avec différents bruits (r = pourcentage de mesures erronées, a = aire de l'intervalle)

### En présence de mesures erronées

Pour le bruit Std1+bias+10% outliers, on constate en comparant les lignes 4 et 5 des tableaux 3.1 et 3.2 que l'erreur de position n'augmente pas et qu'elle a même tendance à diminuer par rapport au bruit Std1+bias seul. La raison à cela est qu'avec l'ajout des mesures erronées, on permet à l'algorithme de tolérer le même pourcentage de mesures erronées. Ce qui signifie que l'algorithme explore maintenant des boîtes pour lesquelles un pourcentage donné de mesures erronées est détecté, alors qu'avant aucune mesure erronée n'était tolérée. Ceci augmente donc le nombre de boîtes explorées et permet d'obtenir des boîtes de plus petites tailles en fonction de la situation. Le seuil sur le pourcentage de mesures erronées tolérées permet donc d'affiner la recherche. L'inconvénient étant qu'un mauvais réglage du seuil par rapport au pourcentage réel des mesures erronées peut entraîner l'algorithme à retourner une boîte ne contenant pas la solution.

En comparant les bruits Std1+$x$% outliers des dernières lignes des tableaux 3.1 et 3.2, on constate que l'erreur de position des balises reste approximativement constante quand le pourcentage de mesures erronées augmente. Ceci montre qu'en ayant une connaissance approxima-




(a) Balise 2 (b) Balise 3

FIGURE 3.9 – Zoom sur les positions finales estimées des balises 2 et 3 pour le jeu de données Plaza1 avec bruit Std1+Biais

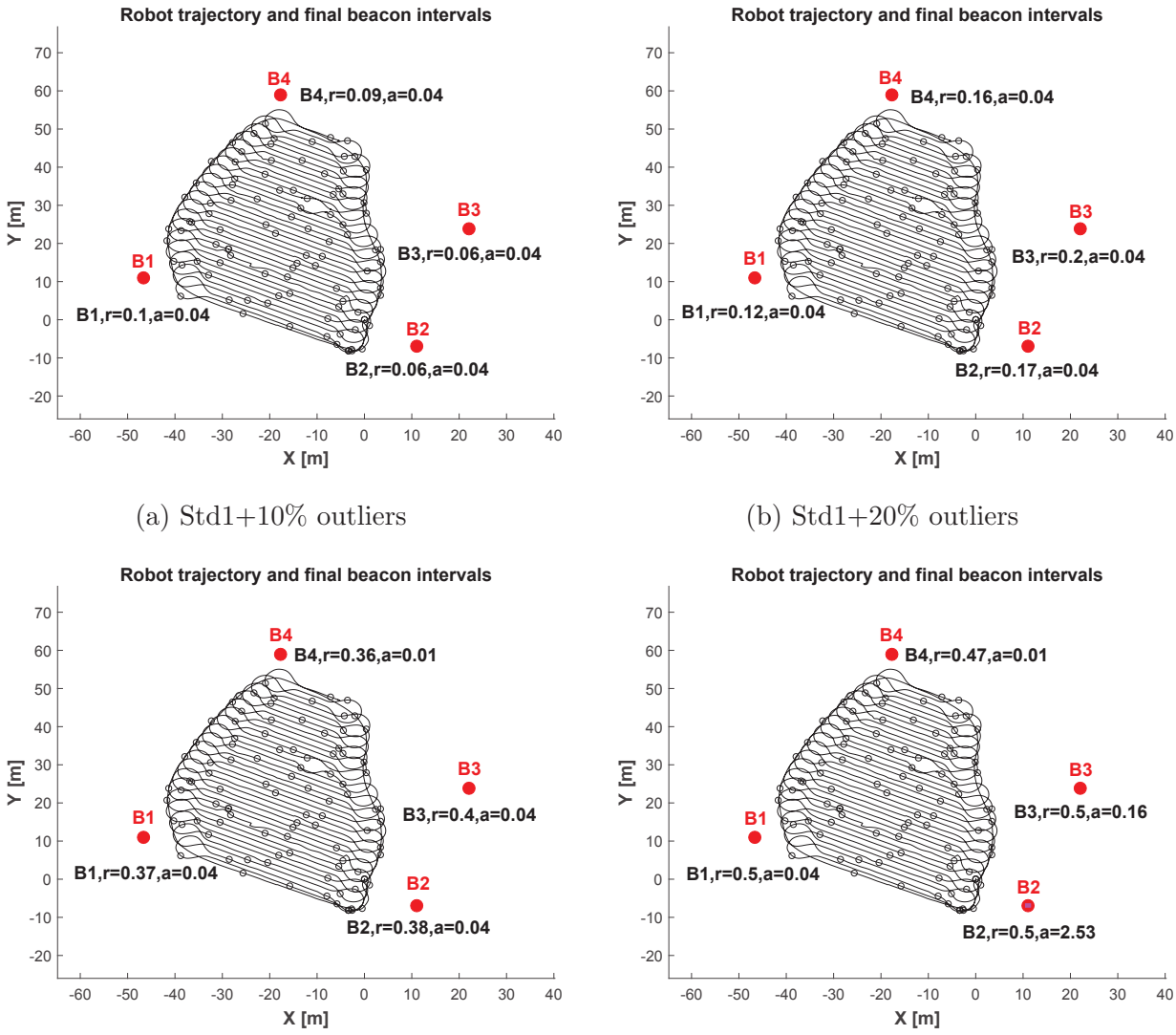
tive du pourcentage de mesures erronées pour régler le seuil de l'algorithme, celui-ci est capable d'estimer avec une bonne précision la position des balises et cela quelque soit le pourcentage de mesures erronées. De plus, en regardant la Figure 3.10 qui montre les positions finales estimées des balises pour le jeu de données Plaza1 pour des pourcentages de mesures erronées croissants, on peut voir que les aires des intervalles n'augmentent pas non plus significativement avec le pourcentage de mesures erronées. En regardant les pourcentages de mesures erronées tolérés pour les intervalles finaux (r=...), on constate que les valeurs sont généralement assez proches du vrai pourcentage de mesures erronées. Le processus de recherche de l'algorithme semble donc bien s'adapter en cas de mesures erronées en utilisant le bon seuil et permet de retourner des estimations de la position des balises proches de la vérité terrain. Le bon comportement de l'algorithme proposé en présence de mesures erronées est maintenant comparé à d'autres algorithmes de localisation par mesures de distance.

### Comparaison de la robustesse avec d'autres algorithmes

Ainsi d'après les simulations menées avec des niveaux de bruit de difficulté croissante, l'approche développée démontre bien sa capacité à localiser la position dans un intervalle donné et cela même en présence d'un fort pourcentage de mesures aberrantes. Nous allons maintenant comparer, la précision de localisation ainsi que la robustesse de la méthode proposée avec d'autres algorithmes de localisation par mesures de distance plus ou moins robustes. Les algorithmes utilisés pour la comparaison avec la méthode proposée sont :

— Algorithmes de trilatération linéaires (*Trilat1*, *Trilat2*, *Trilat3*),
— Algorithme de trilatération non-linéaire (*Trilat+LM*),
— Algorithme de trilatération linéaire avec RANSAC (*Trilat+RANSAC*),
— Algorithme de trilatération non-linéaire avec RANSAC (*Trilat+RANSAC+LM*),
— Algorithmes de grilles d'occupation (*occGrid1*, *occGrid2*, *occGrid3*).

Les algorithmes de trilatération linéaires, non-linéaire et avec RANSAC sont ceux présentés précédemment dans ce chapitre. Pour les algorithmes basés sur une grille d'occupation, le principe consiste à diviser l'espace de recherche en cellules carrées de taille fixe (10 cm lors des expériences). À chaque mesure de distance, toutes les cellules de la grille sont mises à jour avec




(a) Std1+10% outliers

(b) Std1+20% outliers

(c) Std1+40% outliers

(d) Std1+50% outliers

FIGURE 3.10 – Positions estimées des balises pour le jeu de données Plaza1 en présence de mesures erronées (r = pourcentage de mesures erronées, a = aire de l'intervalle)

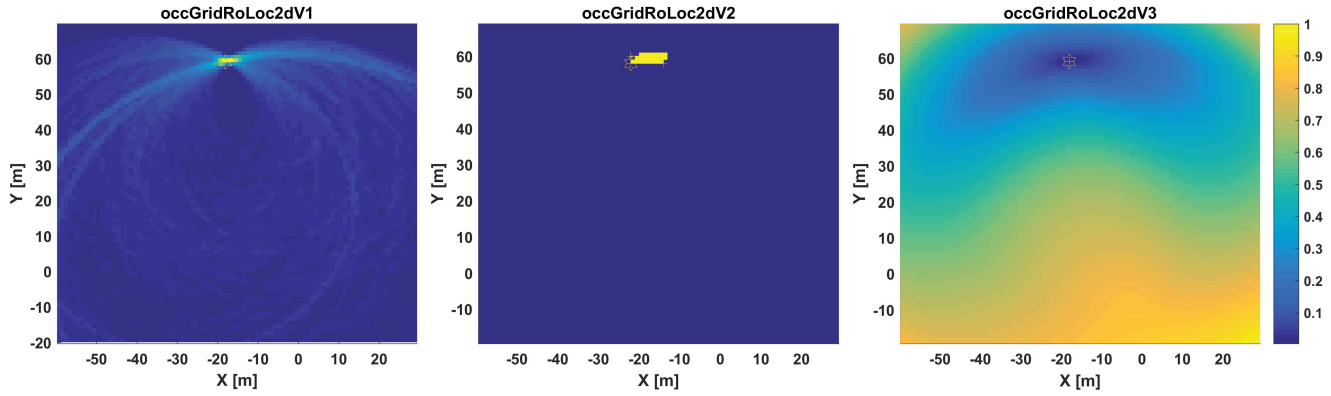
une valeur représentant la possibilité que la mesure soit compatible avec la cellule considérée. La valeur attribuée à la cellule dépend de la version de l'algorithme. Le principe des différentes versions est le suivant :

— *occGrid1* : la valeur attribuée à chaque cellule est égale à 1 si $r_i - e \leq |d_i| \leq r_i + e$ et 0 sinon. Avec cette version on compte pour chaque cellule le nombre de fois ou la cellule se trouve dans l'anneau centrée sur la position connue, de rayon égal à la mesure de distance et de largeur égale à $2e$. Voir Figure 3.11a.

— *occGrid2* : la valeur attribuée à chaque cellule est égale à la probabilité que la mesure de distance $r_i$ provienne de la cellule : $p = \frac{1}{\sqrt{2\pi\sigma_i}} \exp -\frac{(r_i - d_i)^2}{2\sigma_i^2}$, avec $\sigma_i^2$ la variance de la mesure de distance. Voir Figure 3.11b.

— *occGrid3* : la valeur attribuée à chaque cellule est égale à la valeur absolue de l'erreur entre la distance, $d_i$, prédite à partir de la position de la cellule et la mesure de distance $r_i$ : $|r_i - d_i|$. Voir Figure 3.11c.

La position finale estimée est alors obtenue en cherchant la cellule de la grille ayant la valeur maximale (*occGrid1*, *occGrid2*) ou minimale (*occGrid3*).




(a) *occGrid1* (b) *occGrid2* (c) *occGrid3*

FIGURE 3.11 – Exemples de grilles d'occupation obtenues avec les 3 algorithmes *occGrid* pour la balise 4 du jeu de données Plaza1

L'objectif principal de la comparaison étant de déterminer quels sont les algorithmes les plus robustes aux mesures aberrantes, les différents bruits utilisés ici sont : Std1 et Std1+$x$% outliers, avec $x = 10, 20, 30, 40, 50$. Ainsi, pour chaque algorithme, nous analysons l'évolution de la précision de localisation en fonction du pourcentage de mesures aberrantes.

| Outliers : | 0% | 10% | 20% | 30% | 40% | 50% |
|---|---|---|---|---|---|---|
| *Trilat1* | 0.057 | 1.804 | 9.152 | 11.830 | 15.269 | 31.138 |
| *Trilat2* | 0.009 | 2.406 | 8.744 | 10.004 | 14.092 | 22.607 |
| *Trilat3* | 0.010 | 12.146 | 65.798 | 152.572 | 206.608 | 416.687 |
| *Trilat+LM* | 0.006 | 1.625 | 4.151 | 5.630 | 8.097 | 10.850 |
| *Trilat+RANSAC* | 0.009 | 0.016 | 0.016 | 0.021 | 0.008 | 0.023 |
| *Trilat+RANSAC+LM* | 0.006 | 0.006 | 0.008 | 0.015 | 0.005 | 0.012 |
| *occGrid1* | 0.046 | 0.046 | 0.046 | 0.046 | 0.068 | 0.044 |
| *occGrid2* | 24.721 | 77.871 | 77.871 | 77.871 | 77.871 | 77.871 |
| *occGrid3* | 0.046 | 0.048 | 0.046 | 0.046 | 0.055 | 7.234 |
| *LMI+SoS* | 0.073 | 0.082 | 0.082 | 0.106 | 0.080 | 0.104 |

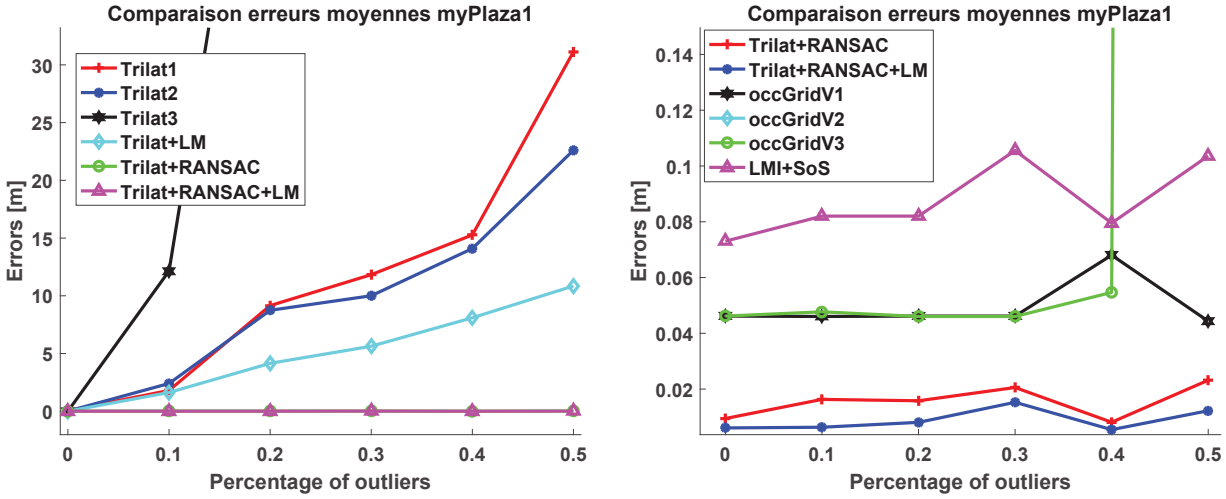
TABLEAU 3.3 – Erreurs moyennes d'estimation en mètres de la position des balises fournies par les algorithmes pour les différents pourcentages de mesures erronées avec le jeu de données Plaza1

| Outliers : | 0% | 10% | 20% | 30% | 40% | 50% |
|---|---|---|---|---|---|---|
| *Trilat1* | 0.007 | 3.958 | 9.746 | 12.581 | 16.458 | 27.773 |
| *Trilat2* | 0.009 | 4.976 | 9.172 | 13.465 | 17.261 | 20.087 |
| *Trilat3* | 0.012 | 330.675 | 91.627 | 160.440 | 161.828 | 144.522 |
| *Trilat+LM* | 0.007 | 2.089 | 3.806 | 6.467 | 8.152 | 10.111 |
| *Trilat+RANSAC* | 0.009 | 0.011 | 0.007 | 0.018 | 0.016 | 0.015 |
| *Trilat+RANSAC+LM* | 0.007 | 0.004 | 0.006 | 0.010 | 0.008 | 0.012 |
| *occGrid1* | 0.650 | 0.046 | 0.086 | 0.086 | 0.171 | 0.046 |
| *occGrid2* | 0.075 | 79.160 | 79.160 | 79.160 | 79.160 | 79.160 |
| *occGrid3* | 0.046 | 0.046 | 0.046 | 0.046 | 0.051 | 1.019 |
| *LMI+SoS* | 0.126 | 0.083 | 0.132 | 0.070 | 0.066 | 0.056 |

TABLEAU 3.4 – Erreurs moyennes d'estimation en mètres de la position des balises fournies par les algorithmes pour les différents pourcentages de mesures erronées avec le jeu de données Plaza2

L'ensemble des erreurs moyennes sur la position des balises obtenues avec les différents algorithmes et pour les différents pourcentages de mesures erronées sont regroupées dans les tableaux 3.3 et 3.4. De manière générale, on constate qu'avec l'augmentation du pourcentage de mesures erronées, l'erreur moyenne en position augmente aussi. Ceci se vérifie surtout pour les premières lignes des tableaux qui correspondent aux algorithmes de trilatération non robustes (*Trilat1*, *Trilat2*, *Trilat3*, *Trilat+LM*). Alors que pour les autres (*Trilat+RANSAC*, *Trilat+RANSAC+LM*, *occGrid1*, *occGrid3*, *LMI+SoS*) les erreurs moyennes n'augmentent quasiment pas. Pour des raisons non élucidées, l'algorithme *occGrid2* ne semble marcher que dans le cas sans bruit avec le jeu de donnée Plaza2.

Afin de mieux comparer les algorithmes entre eux, une séparation en deux groupes a été faite. D'un côté les algorithmes de trilatération : *Trilat1*, *Trilat2*, *Trilat3*, *Trilat+LM*, *Trilat+RANSAC*, *Trilat+RANSAC+LM*. De l'autre les algorithmes dits robustes : *Trilat+RANSAC*, *Trilat+RANSAC+LM*, *occGrid1*, *occGrid2*, *occGrid3*, *LMI+SoS*. On peut ainsi voir sur les figures 3.12 et 3.13 l'évolution des erreurs de positions moyennes pour les jeux de données Plaza1 et Plaza2 respectivement. Les sous-figures (a) comparent les algorithmes de trilatération, alors que les sous-figures (b) comparent les algorithmes dits robustes. Ainsi en comparant les algorithmes de trilatération, on observe comme l'on pourrait s'y attendre que seuls les deux algorithmes utilisant RANSAC sont robustes et conservent une erreur plus ou moins constante avec l'augmentation du nombre de mesures erronées. Pour les autres algorithmes : *Trilat1*, *Trilat2*, *Trilat3* et *Trilat+LM*, on constate une rapide augmentation de l'erreur moyenne dès l'apparition de quelques mesures erronées, ce qui montre la réelle sensibilité de ces méthodes. De plus, parmi les algorithmes basés sur la trilatération, on constate que les versions *Trilat1*, *Trilat2* présentent approximativement le même comportement avec un léger avantage pour la version *Trilat2*, alors que la méthode *Trilat3* n'apparaît efficace que sans mesures erronées. On constate aussi que l'utilisation d'un algorithme d'optimisation non-linéaire *Trilat+LM*, afin de raffiner l'estimation, remplit bien son rôle car l'erreur moyenne est presque toujours inférieure à celle de la méthode linéaire utilisée seule *Trilat2*.

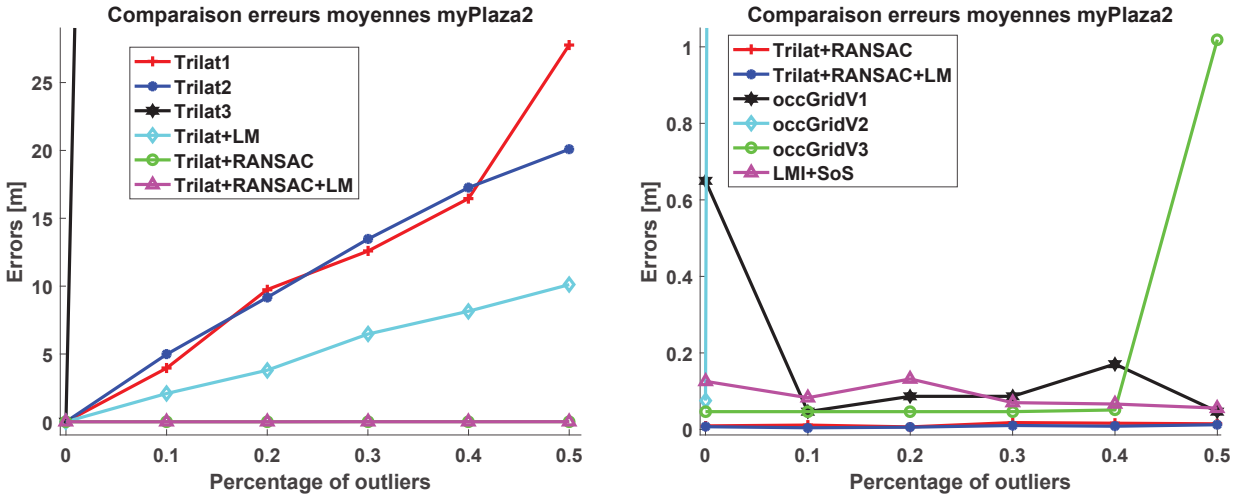



(a) Algorithmes de trilatération

(b) Algorithmes robustes

FIGURE 3.12 – Évolution de l'erreur moyenne en fonction du pourcentage de mesures erronées pour le jeu de données Plaza1

En comparant les algorithmes robustes aux figures 3.12b et 3.13b, on constate pour l'ensemble des algorithmes que les erreurs moyennes restent quasiment constantes au fur et à mesure de l'augmentation du nombre de mesures aberrantes et inférieures à un mètre. La méthode fonctionnant le mieux est celle basée sur la trilatération avec RANSAC *Trilat+RANSAC*



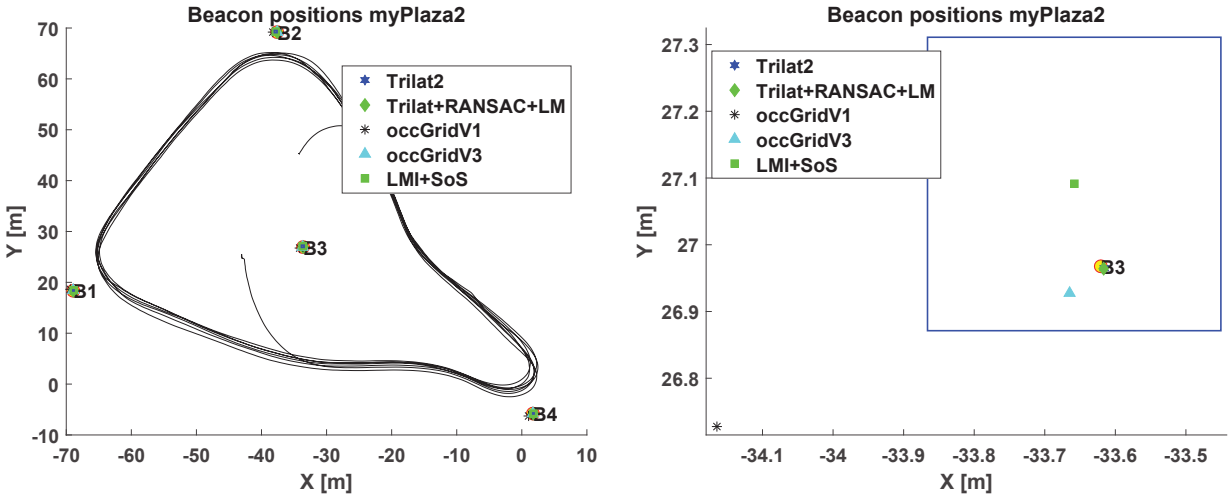

(a) Algorithmes de trilatération

(b) Algorithmes robustes

FIGURE 3.13 – Évolution de l'erreur moyenne en fonction du pourcentage de mesures erronées pour le jeu de données Plaza2

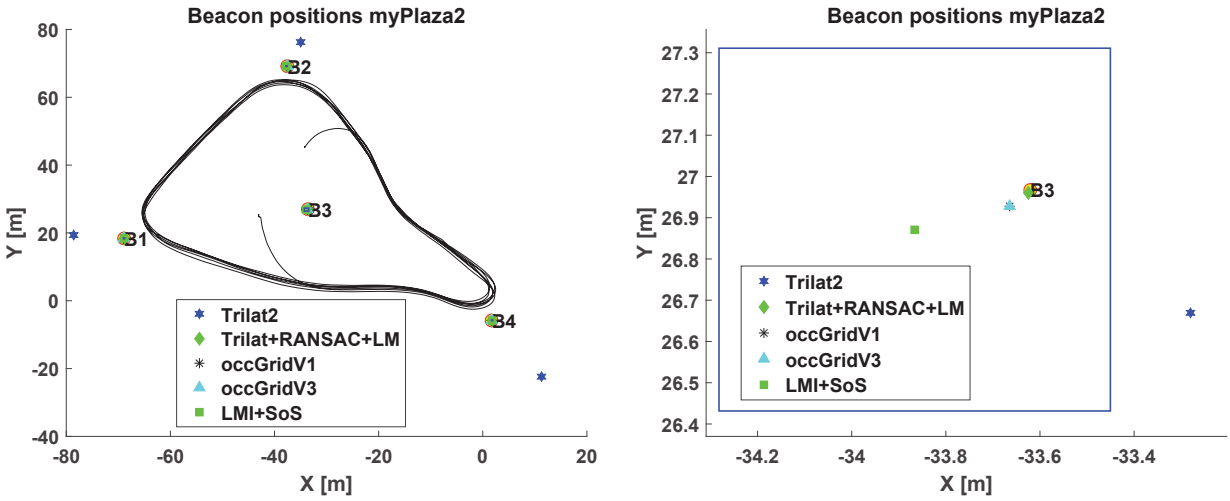
avec comme auparavant une légère amélioration supplémentaire quand l'on ajoute l'algorithme d'optimisation non-linéaire *Trilat+RANSAC+LM*. Les méthodes basées sur les grilles d'occupations *occGrid1* et *occGrid3* donnent des erreurs moyennes plus faibles que l'approche proposée *LMI+SoS* pour le jeu de données Plaza1, mais donnent des erreurs comparables pour le jeu de données Plaza2. Une explication au fait que les erreurs moyennes retournées pour l'algorithme *LMI+SoS* sont légèrement plus élevées que pour les autres algorithmes robustes est que la position estimée est calculée en prenant le milieu de la boîte retournée par l'algorithme comme déjà vu précédemment. Ainsi, si la vraie position est éloignée du centre de la boîte, l'erreur sera légèrement plus importante. Cependant, le fait que l'algorithme retourne une boîte est un avantage par rapport aux autres méthodes, car il permet de savoir dans quelles zones précisément se trouve la vraie position de la balise. En regardant les figures 3.14b, 3.15b, 3.16b on peut mieux visualiser le phénomène. Ces figures montrent un zoom sur la position estimée de la balise 3 pour le jeu de données Plaza2 pour 0%, 20% et 50% de mesures erronées respectivement. On peut ainsi voir que même si le centre de la boîte retournée par l'algorithme *LMI+SoS* (carré vert) est plus loin que les positions estimées par les autres algorithmes robustes *occGrid1*, *occGrid3* et *Trilat+RANSAC+LM*, la boîte, elle, englobe bien la vraie solution.

En regardant les figures 3.14, 3.15 et 3.16, on peut résumer les principales observations de la comparaison de la robustesse aux mesures erronées des différents algorithmes de localisation. En effet, comme on le voit sur la Figure 3.14 montrant les positions estimées par différents algorithmes pour le jeu de données Plaza2 sans mesures erronées, on constate que toutes les méthodes retournent une bonne position pour les 4 balises, seul un zoom permet de faire une légère différence entre les positions estimées. Par contre, en présence de mesures erronées, 20% pour la Figure 3.15 et 50% pour la Figure 3.16, on remarque que les positions estimées (étoiles bleues) par l'algorithme de trilatération classique *Trilat2* s'éloignent d'autant plus des vraies positions que le pourcentage est élevé. En revanche, pour les algorithmes robustes : *occGrid1*, *occGrid3*, *Trilat+RANSAC+LM* et *LMI+SoS*, il faut à nouveau zoomer pour distinguer les différences. Parmi ces algorithmes, celui présentant les meilleurs résultats est l'algorithme de trilatération basé sur RANSAC (*Trilat+RANSAC+LM*). Puis viennent les méthodes utilisant les grilles d'occupation (*occGrid1*, *occGrid3*) et enfin l'approche *LMI+SoS*. Malgré une moins bonne estimation de la position, l'algorithme *LMI+SoS* a l'avantage fournir une boîte qui englobe presque toujours la vraie solution. Le "presque toujours" en raison de quelques contre-




(a) Ensemble des balises (b) Zoom balise 3

FIGURE 3.14 – Comparaison des positions obtenues par différents algorithmes pour le jeu de données Plaza2 avec 0% de mesures erronées

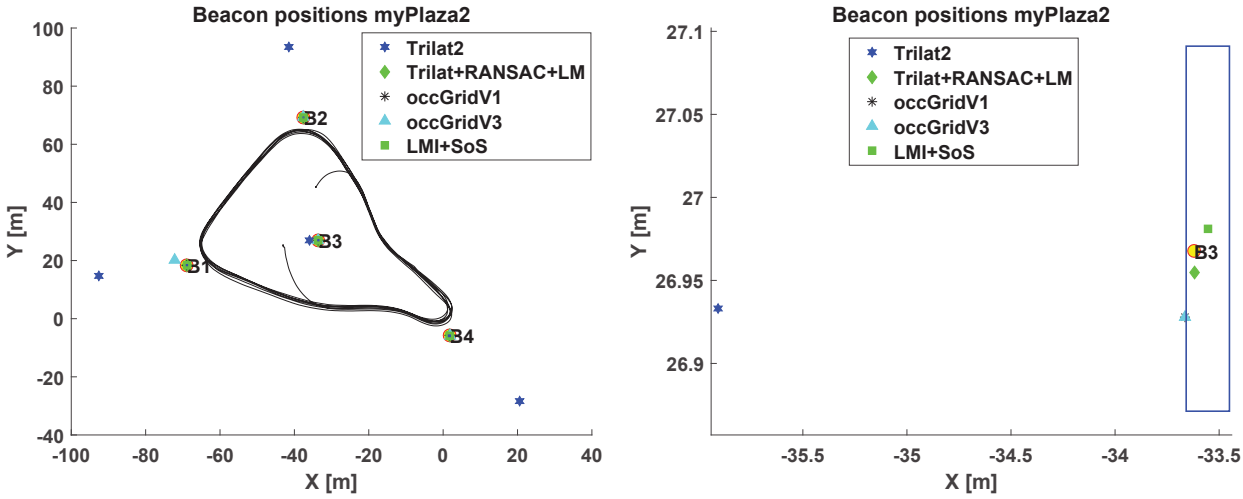



(a) Ensemble des balises (b) Zoom balise 3

FIGURE 3.15 – Comparaison des positions obtenues par différents algorithmes pour le jeu de données Plaza2 avec 20% de mesures erronées

exemples observés durant les expériences sans pouvoir en expliquer la raison. Enfin, en termes de rapidité, on peut souligner que l'algorithme *Trilat+RANSAC+LM* est de loin le plus rapide pour obtenir une estimation (moins d'une seconde), comparé aux plusieurs secondes voire minutes pour l'approche *LMI+SoS* et surtout pour les méthodes basées sur une grille d'occupation.

Après avoir vu différents algorithmes de localisation utilisant uniquement des mesures de distance, nous allons maintenant présenter comment il est possible d'utiliser et d'intégrer des informations supplémentaires provenant d'autres capteurs. Nous allons tout d'abord voir en quoi le fait de ne travailler qu'avec les mesures de distance par rapport aux balises en complément des mesures d'odométrie présente une difficulté supplémentaire par rapport au cas général où l'on dispose des mesures de distance et d'angle par rapport à des repères fixes. Pour cela nous allons procéder à une analyse de l'observabilité du problème de localisation par mesures de distance.




(a) Ensemble des balises

(b) Zoom balise 3

FIGURE 3.16 – Comparaison des positions obtenues par différents algorithmes pour le jeu de données Plaza2 avec 50% de mesures erronées

## 3.5 Analyse de l'observabilité

Quand on cherche à estimer les variables d'un système dont les équations d'évolution ainsi que les mesures sont connues, il est important de déterminer s'il est possible de retrouver complètement les variables et de converger vers les bonnes valeurs. Ainsi, une analyse d'observabilité, [83], [2], [121], permet de définir si toutes les variables du système peuvent être estimées sans ambiguïtés. Un système est dit observable s'il est possible de calculer de manière unique l'état du système à partir d'une séquence d'actions de commande et d'observations (ou mesures). Il existe plusieurs moyens de déterminer l'observabilité d'un système. En théorie de l'estimation, la condition d'observabilité est évaluée en examinant la matrice d'information de Fisher (FIM) [2], [133]. Du point de vue de l'automatique, un système linéaire à temps continu est dit observable, si sa matrice d'observabilité $\mathcal{O}$ est de rang plein, ou autrement dit, son rang doit être égal à la dimension de l'état du système [83], [58]. Quand le système étudié est non-linéaire, ce qui est le cas ici, l'étude devient plus complexe. Les conditions d'observabilité pour un système non-linéaire sont établies dans [58]. L'observabilité étant un concept global, dans le cas d'un système non-linéaire on définit plutôt le concept d'observabilité locale faible qui est une condition plus faible de l'observabilité. Le concept d'observabilité locale faible peut être interprété comme le fait de pouvoir distinguer instantanément chaque vecteur d'état, $\mathbf{x}$, de son voisinage. Ainsi, afin de prouver qu'un système non-linéaire est localement faiblement observable au voisinage d'un point $\mathbf{x}_0$, il est nécessaire de construire la matrice d'observabilité à partir des gradients des dérivées de Lie successives de la fonction de mesure du système. L'espace d'observation doit alors être égal à l'espace des dérivées de Lie successives. La dérivée de Lie d'une fonction $h(\mathbf{x})$ le long d'un champ vectoriel $\mathbf{f}$ est donnée par :

$$L_{\mathbf{f}}[h] = (dh)\mathbf{f}$$

avec $dh$ le gradient de $h$ par rapport au vecteur $\mathbf{x}$. Si l'on définit $h(\mathbf{x})$ comme la fonction d'observation du système dont l'état est donné par $\mathbf{x}$, alors les dérivées de Lie sont définies comme les dérivées temporelles de $h(\mathbf{x})$ le long de la trajectoire $\mathbf{x}$ du système.

La dérivée de Lie à l'ordre 0 est donnée par la fonction $h$ elle-même :

$$L^0_{\mathbf{f}}[h] = h(\mathbf{x})$$

Le gradient de la dérivée de Lie à l'ordre 0 est donc :

$$dL^0_{\mathbf{f}}[h] = \frac{\partial L^0_{\mathbf{f}}[h]}{\partial \mathbf{x}} = \frac{\partial h(\mathbf{x})}{\partial \mathbf{x}} = dh$$

À l'ordre 1 on obtient :

$$L^1_{\mathbf{f}}[h] = \frac{\partial h}{\partial \mathbf{x}}\frac{d\mathbf{x}}{dt} = \frac{\partial h}{\partial \mathbf{x}}\mathbf{f}(\mathbf{x}) = (dh)\mathbf{f}(\mathbf{x})$$

À l'ordre 2 on obtient :

$$L^2_{\mathbf{f}}[h] = L_{\mathbf{f}}[L^1_{\mathbf{f}}[h]] = \frac{\partial}{\partial \mathbf{x}}\Big(L^1_{\mathbf{f}}[h]\Big)\mathbf{f}(\mathbf{x}) = \frac{\partial}{\partial \mathbf{x}}\Big(\frac{\partial h}{\partial \mathbf{x}}\mathbf{f}(\mathbf{x})\Big)\mathbf{f}(\mathbf{x})$$

À l'ordre $k+1$ on obtient :

$$L^{k+1}_{\mathbf{f}}[h] = L_{\mathbf{f}}[L^k_{\mathbf{f}}[h]] = \frac{\partial}{\partial \mathbf{x}}\Big(L^k_{\mathbf{f}}[h]\Big)\mathbf{f}(\mathbf{x})$$

La matrice d'observabilité est alors construite à partir de l'espace traversé par les gradients de toutes les dérivées de Lie d'ordre $k$, $dL^k_{\mathbf{f}}[h]$ :

$$\mathcal{O} = \begin{pmatrix} dL^0_f[h(\mathbf{x})] \\ dL^1_f[h(\mathbf{x})] \\ dL^2_f[h(\mathbf{x})] \\ dL^3_f[h(\mathbf{x})] \\ \vdots \end{pmatrix} \tag{3.33}$$

Dans le cas de la localisation, on cherche à déterminer la pose du robot. Le vecteur d'état du système, $\mathbf{x}$, est donc donné par :

$$\mathbf{x} = (x, y, \theta)^T$$

Le vecteur d'état est donc de dimension $n = 3$ : $\mathbf{x} \in \mathcal{R}^3$.

L'entrée du système, ou vecteur de contrôle, $\mathbf{u}$, est donnée par le module de la vitesse linéaire $v$ et par la vitesse angulaire $\omega$ du robot :

$$\mathbf{u} = (v, \omega)^T$$

Le vecteur de contrôle est donc de dimension $m = 2$ : $\mathbf{u} \in \mathcal{R}^2$.

Pour un robot mobile de type unicycle, les équations d'évolution de l'état en temps continu sont données par (2.5) :

$$\begin{cases} \dot{x} = v\cos\theta \\ \dot{y} = v\sin\theta \\ \dot{\theta} = \omega \end{cases}$$

Ce système peut se réécrire sous la forme :

$$\dot{\mathbf{x}} = f_0(\mathbf{x}) + \sum_{i=1}^{m} f_i(\mathbf{x})u_i \tag{3.34}$$

avec : $u_1 = v$, $u_2 = \omega$, $f_0 = (0,0,0)^T$, $f_1 = (\cos\theta, \sin\theta, 0)^T$, $f_2 = (0,0,1)^T$.

Le vecteur de mesure, $\mathbf{z}$, est composé des mesures de distance entre le robot et les différentes balises disposées sur le terrain :

$$\mathbf{z} = \mathbf{h}(\mathbf{x}) = (z_1, \ldots, z_i, \ldots, z_l)^T$$

avec $z_i = d_i = \sqrt{(x_i - x)^2 + (y_i - y)^2}$ la mesure de distance entre la position du robot $(x, y)$ et la position de la $i^{\text{ème}}$ balise $(x_i, y_i)$. Le vecteur de mesure est donc de dimension $l$ : $\mathbf{z} \in \mathcal{R}^l$, où $l$ est le nombre de balises.

Nous allons maintenant étudier l'observabilité du problème de localisation pour différents nombres de balises $l$, en commençant tout d'abord par le cas où l'on ne dispose que d'une seule mesure de distance par rapport à une balise.

### 3.5.1 Avec une balise

L'analyse d'observabilité avec une balise présentée ici, reprend le développement effectué par Martinelli dans le cours *Mooc on Mobile Robots and Autonomous Vehicles* [81]. Comme on ne considère qu'une seule balise, on peut sans perte de généralité supposer que celle-ci se situe à l'origine du repère dans lequel le robot est localisé. L'équation de mesure devient alors :

$$h(\mathbf{x}) = \sqrt{x^2 + y^2}$$

Pour simplifier les calculs, on effectue un changement de coordonnées, de Cartésien à polaire, $(x, y) \rightarrow (\rho, \phi)$, avec :

$$\begin{cases} x = \rho \cos\phi \\ y = \rho \sin\phi \end{cases} \Leftrightarrow \begin{cases} \rho = \sqrt{x^2 + y^2} \\ \phi = \text{atan}(\frac{x}{y}) \end{cases}$$

L'orientation du robot $\theta$ en coordonnées cartésiennes reste inchangée en polaire. Le nouveau vecteur d'état a donc la forme suivante :

$$\mathbf{x} = (\rho, \phi, \theta)^T$$

et reste de dimension $n = 3$.

En réécrivant les dérivées temporelles de $x$ et $y$ en polaire :

$$\begin{cases} \dot{x} = \dot{\rho}\cos\phi - \rho\dot{\phi}\sin\phi \\ \dot{y} = \dot{\rho}\sin\phi + \rho\dot{\phi}\cos\phi \end{cases} \tag{3.35}$$

dans le système d'équations (2.5), on obtient :

$$\begin{cases} \dot{\rho}\cos\phi - \rho\dot{\phi}\sin\phi = v\cos\theta & (1) \\ \dot{\rho}\sin\phi + \rho\dot{\phi}\cos\phi = v\sin\theta & (2) \end{cases} \tag{3.36}$$

En faisant $(1) \times \cos\phi + (2) \times \sin\phi$ et $-(1) \times \sin\phi + (2) \times \cos\phi$, le modèle cinématique du robot en polaire devient :

$$\begin{cases} \dot{\rho} = v\cos(\theta - \phi) \\ \dot{\phi} = \frac{v}{\rho}\sin(\theta - \phi) \\ \dot{\theta} = w \end{cases}$$

En identifiant à nouveau les composantes $f_i$ de (3.34), on trouve :

$$\begin{aligned} f_0(\mathbf{x}) &= (0, 0, 0)^T \\ f_1(\mathbf{x}) &= (\cos(\theta - \phi), \frac{1}{\rho}\sin(\theta - \phi), 0)^T \\ f_2(\mathbf{x}) &= (0, 0, 1)^T \end{aligned}$$

On considère ici pour l'analyse que le robot n'est jamais au repos et donc que $v \neq 0$ et $\omega \neq 0$. L'équation de mesure devient quant à elle linéaire par rapport à l'état du système :

$$h(\mathbf{x}) = \rho$$

On peut alors calculer les dérivées de Lie successives :

$$L_{\mathbf{f}}^0 h = h = \rho$$

D'où :

$$d(L_{\mathbf{f}}^0 h) = dh = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}$$

$$L^1_{\mathbf{f}_1} h = d(L^0_{\mathbf{f}} h).\mathbf{f}_1 = (dh).\mathbf{f}_1 = \cos(\theta - \phi)$$

D'où :

$$d(L^1_{\mathbf{f}_1} h) = \begin{pmatrix} 0 & \sin(\theta-\phi) & -\sin(\theta-\phi) \end{pmatrix}$$

$$L^1_{\mathbf{f}_2} h = d(L^0_{\mathbf{f}} h).\mathbf{f}_2 = (dh).\mathbf{f}_2 = 0$$

D'où :

$$d(L^1_{\mathbf{f}_2} h) = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}$$

$$\begin{aligned} L^2_{11} h &= d(L^1_{\mathbf{f}_1} h).\mathbf{f}_1 \\ &= \begin{pmatrix} 0 & \sin(\theta-\phi) & -\sin(\theta-\phi) \end{pmatrix} \begin{pmatrix} \cos(\theta-\phi) \\ \frac{\sin(\theta-\phi)}{\rho} \\ 0 \end{pmatrix} \\ &= \frac{\sin(\theta-\phi)^2}{\rho} \end{aligned}$$

D'où :

$$d(L^2_{11} h) = \begin{pmatrix} -\frac{\sin(\theta-\phi)^2}{\rho^2} & -\frac{2}{\rho}\sin(\theta-\phi)\cos(\theta-\phi) & \frac{2}{\rho}\sin(\theta-\phi)\cos(\theta-\phi) \end{pmatrix}$$

$$\begin{aligned} L^2_{12} h &= d(L^1_{\mathbf{f}_1} h).\mathbf{f}_2 \\ &= \begin{pmatrix} 0 & \sin(\theta-\phi) & -\sin(\theta-\phi) \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \\ &= -\sin(\theta-\phi) \end{aligned}$$

D'où :

$$d(L^2_{12} h) = \begin{pmatrix} 0 & \cos(\theta-\phi) & -\cos(\theta-\phi) \end{pmatrix}$$

$$L^2_{21} h = d(L^1_{\mathbf{f}_2} h).\mathbf{f}_1 = 0$$

D'où :

$$d(L^2_{21} h) = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}$$

$$L^2_{22} h = d(L^1_{\mathbf{f}_2} h).\mathbf{f}_2 = 0$$

D'où :

$$d(L^2_{22} h) = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}$$

Si l'on construit la matrice d'observabilité :

$$\mathcal{O} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \sin(\theta-\phi) & -\sin(\theta-\phi) \\ -\frac{\sin(\theta-\phi)^2}{\rho^2} & -\frac{2}{\rho}\sin(\theta-\phi)\cos(\theta-\phi) & \frac{2}{\rho}\sin(\theta-\phi)\cos(\theta-\phi) \\ 0 & \cos(\theta-\phi) & -\cos(\theta-\phi) \\ \vdots & \vdots & \vdots \end{pmatrix}$$

on observe que les 2ème et 3ème colonnes sont de la forme $[*, -*]$, où $*$ représente la même quantité. Si l'on continue à calculer les dérivées de Lie suivantes : $L^3_{111} h$, $L^3_{112} h$, $L^3_{121} h$, $L^3_{122} h$ ainsi que leurs gradients, la matrice d'observabilité conservera cette propriété.

$$L^3_{111} h = (dL^2_{11} h).\mathbf{f}_1 = \frac{3}{\rho^2}\sin(\theta-\phi)^2\cos(\theta-\phi)$$

$$L^3_{112}h = (dL^2_{11}h).\mathbf{f}_2 = \frac{2}{\rho}\sin(\theta-\phi)\cos(\theta-\phi)$$

$$L^3_{121}h = (dL^2_{12}h).\mathbf{f}_1 = \frac{1}{\rho}\sin(\theta-\phi)\cos(\theta-\phi)$$

$$L^3_{122}h = (dL^2_{12}h).\mathbf{f}_2 = -\cos(\theta-\phi)$$

On a donc seulement deux colonnes linéairement indépendantes (colonne 1 et 2). Ainsi, le rang de la matrix d'observabilité $\mathcal{O}$ est égale à 2 alors que la dimension de l'état est de 3 :

$$\text{rang}(\mathcal{O}) = 2 < \dim(\mathbf{x}) = 3$$

On peut donc conclure que le système n'est pas observable avec seulement une seule mesure de distance par rapport à une balise. Et donc qu'il n'est pas possible de localiser de façon unique le robot avec une seule balise, même si l'on dispose d'une série de mesures le long d'une trajectoire. En effet, on obtient les mêmes mesures en faisant tourner la position initiale du robot d'un angle quelconque par rapport à la balise. Nous allons maintenant procéder à l'analyse dans le cas de plusieurs mesures de distance par rapport à des balises.

### 3.5.2 Avec deux balises

Si l'on considère le cas de deux balises, on peut sans perte de généralité, considérer que la première se situe à l'origine et que la seconde se situe sur l'axe $x$ à une distance $a$. On a donc $\mathbf{x}_1 = (0,0)^T$ et $\mathbf{x}_2 = (a,0)^T$. Ce qui donne les fonctions de mesures suivantes :

$$\begin{aligned} h_1(\mathbf{x}) &= x^2 + y^2 \\ h_2(\mathbf{x}) &= (a-x)^2 + y^2 \end{aligned}$$

On considère donc ici que les mesures sont données par la distance au carré, ce qui ne change en rien les propriétés d'observabilité du système, mais permet de simplifier les calculs.

Les dérivées de Lie à l'ordre 0 sont donc :

$$\begin{aligned} dL^0_{\mathbf{f}}h_1(\mathbf{x}) &= \begin{pmatrix} 2x & 2y & 0 \end{pmatrix} \\ dL^0_{\mathbf{f}}h_2(\mathbf{x}) &= \begin{pmatrix} -2(a-x) & 2y & 0 \end{pmatrix} \end{aligned}$$

Pour calculer les dérivées suivantes, on utilise l'expression suivante [83] :

$$dL^i_{\mathbf{f}}h(\mathbf{x}) = dL^{i-1}_{\mathbf{f}}h(\mathbf{x})\frac{\partial \mathbf{f}}{\partial \mathbf{x}} + \mathbf{f}^T\frac{\partial}{\partial \mathbf{x}}(dL^{i-1}_{\mathbf{f}}h(\mathbf{x}))$$

Les dérivées de Lie à l'ordre 1 sont :

$$\begin{aligned} dL^1_{\mathbf{f}}h(\mathbf{x}) &= \begin{pmatrix} 2v\cos\theta & 2v\sin\theta & 2v(-x\sin\theta + y\cos\theta) \end{pmatrix} \\ dL^1_{\mathbf{f}}h_2(\mathbf{x}) &= \begin{pmatrix} 2v\cos\theta & 2v\sin\theta & 2v((a-x)\sin\theta + y\cos\theta) \end{pmatrix} \end{aligned}$$

Les dérivées de Lie à l'ordre 2 sont :

$$\begin{aligned} dL^2_{\mathbf{f}}h(\mathbf{x}) &= \begin{pmatrix} -2\omega v\sin\theta & 2\omega v\cos\theta & -2\omega v(x\cos\theta + y\sin\theta) \end{pmatrix} \\ dL^2_{\mathbf{f}}h_2(\mathbf{x}) &= \begin{pmatrix} -2\omega v\sin\theta & 2\omega v\cos\theta & 2\omega v((a-x)\cos\theta - y\sin\theta) \end{pmatrix} \end{aligned}$$

En construisant la matrice d'observabilité, $\mathcal{O}$, comme suit :

$$\mathcal{O} = \begin{pmatrix} dL_{\mathbf{f}}^0 h_1(\mathbf{x}) \\ dL_{\mathbf{f}}^0 h_2(\mathbf{x}) \\ dL_{\mathbf{f}}^1 h_1(\mathbf{x}) \end{pmatrix} = \begin{pmatrix} 2x & 2y & 0 \\ -2(a-x) & 2y & 0 \\ 2v\cos\theta & 2v\sin\theta & 2v(-x\sin\theta + y\cos\theta) \end{pmatrix}$$

on constate que son rang est égal à 3, sauf pour plusieurs cas particuliers qui sont déterminés en regardant où le déterminant de la matrice $\mathcal{O}$ s'annule : $\det(\mathcal{O}) = 0$. L'expression de ce déterminant est donné par :

$$\det(\mathcal{O}) = 8avy\Big(y\cos(\theta) - x\sin(\theta)\Big)$$

Celui-ci s'annule quand :

— $y = 0$, c'est-à-dire que le robot se déplace uniquement sur l'axe $x$, formé par les balises,

— $v = 0$, c'est-à-dire si le robot reste sur place,

— $a = 0$, qui correspond au cas où les deux balises sont confondues,

— $y = -\tan\theta a + \tan\theta x$, qui correspond à une trajectoire spéciale dépendant de l'orientation du robot.

Ainsi, à part les cas particuliers cités précédement, le système avec deux balises est localement faiblement observable puisque : $\text{rang}(\mathcal{O}) = \dim(\mathbf{x}) = 3$.

### 3.5.3 Avec plusieurs balises

Dans le cas de plusieurs balises $l \geq 3$, la fonction de mesure $\mathbf{h}(\mathbf{x})$ devient :

$$\mathbf{h}(\mathbf{x}) = \begin{pmatrix} d_1(\mathbf{x}) \\ d_2(\mathbf{x}) \\ \vdots \\ d_l(\mathbf{x}) \end{pmatrix} = \begin{pmatrix} \sqrt{(x_1-x)^2 + (y_1-y)^2} \\ \sqrt{(x_2-x)^2 + (y_2-y)^2} \\ \vdots \\ \sqrt{(x_l-x)^2 + (y_l-y)^2} \end{pmatrix}$$

Les dérivées de Lie successives sont alors calculées en utilisant une représentation cartésienne de la pose du robot. À l'ordre 0, on obtient :

$$L_{\mathbf{f}}^0 \mathbf{h} = \mathbf{h}(\mathbf{x})$$

D'où :

$$dL_{\mathbf{f}}^0 \mathbf{h} = d\mathbf{h} = \begin{pmatrix} -\frac{\delta_{x_1}}{d_1} & -\frac{\delta_{y_1}}{d_1} & 0 \\ -\frac{\delta_{x_2}}{d_2} & -\frac{\delta_{y_2}}{d_2} & 0 \\ \vdots & \vdots & \vdots \\ -\frac{\delta_{x_l}}{d_l} & -\frac{\delta_{y_l}}{d_l} & 0 \end{pmatrix} = \begin{pmatrix} -\cos(\phi_1) & -\sin(\phi_1) & 0 \\ -\cos(\phi_2) & -\sin(\phi_2) & 0 \\ \vdots & \vdots & \vdots \\ -\cos(\phi_l) & -\sin(\phi_l) & 0 \end{pmatrix}$$

avec $\delta_{x_i} = (x_i - x)$, $\delta_{y_i} = (y_i - y)$, $i \in [1, ..., l]$, et $\phi_i$ l'angle entre le repère global et la balise $i$ (angle de sa représentation polaire), voir Figure 3.17.

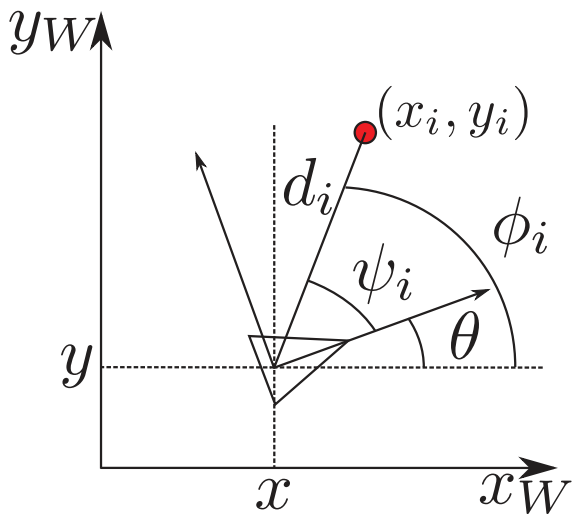



FIGURE 3.17 – Schéma de la configuration robot/balise

Pour les dérivées de Lie à l'ordre 1, on calcule :

$$\begin{aligned}
dL^1_{\mathbf{f}}h_i(\mathbf{x}) &= dL^0_{\mathbf{f}}h_i(\mathbf{x})\frac{\partial \mathbf{f}}{\partial \mathbf{x}} + \mathbf{f}^T\frac{\partial}{\partial \mathbf{x}}(dL^0_{\mathbf{f}}h_i(\mathbf{x})) \\
&= \Big(-\tfrac{v\delta_{y_i}}{d_i^3}(\sin(\theta)\delta_{x_i} - \cos(\theta)\delta_{y_i}) \quad \tfrac{v\delta_{x_i}}{d_i^3}(\sin(\theta)\delta_{x_i} - \cos(\theta)\delta_{y_i}) \quad \tfrac{v}{d_i}(\sin(\theta)\delta_{x_i} - \cos(\theta)\delta_{y_i})\Big) \\
&= \Big(-\tfrac{vs(\phi_i)}{d_i}(s(\theta)c(\phi_i) - c(\theta)s(\phi_i)) \quad \tfrac{vc(\phi_i)}{d_i}(s(\theta)c(\phi_i) - c(\theta)s(\phi_i)) \quad v(s(\theta)c(\phi_i) - c(\theta)s(\phi_i))\Big) \\
&= \Big(\tfrac{v}{d_i}\sin(\phi_i)\sin(\psi_i) \quad -\tfrac{v}{d_i}\cos(\phi_i)\sin(\psi_i) \quad -v\sin(\psi_i)\Big)
\end{aligned}$$

avec $\psi_i$ l'angle relatif entre le repère du robot et la balise $i$. En effet, on a :

$$\begin{aligned}
&\frac{\delta_{x_i}}{d_i}\sin(\theta) - \frac{\delta_{y_i}}{d_i}\cos(\theta) \\
&= \sin(\theta)\cos(\phi_i) - \cos(\theta)\sin(\phi_i) \\
&= \sin(\theta - \phi_i) \\
&= -\sin(\psi_i)
\end{aligned}$$

On obtient donc à l'ordre 1 :

$$dL^0_{\mathbf{f}}\mathbf{h} = \begin{pmatrix}
\frac{v}{d_1}\sin(\phi_1)\sin(\psi_1) & -\frac{v}{d_1}\cos(\phi_1)\sin(\psi_1) & -v\sin(\psi_1) \\
\frac{v}{d_2}\sin(\phi_2)\sin(\psi_2) & -\frac{v}{d_2}\cos(\phi_2)\sin(\psi_2) & -v\sin(\psi_2) \\
\vdots & \vdots & \vdots \\
\frac{v}{d_l}\sin(\phi_l)\sin(\psi_l) & -\frac{v}{d_l}\cos(\phi_l)\sin(\psi_l) & -v\sin(\psi_l)
\end{pmatrix}$$

Si l'on s'intéresse au cas de trois balises ou plus et que l'on commence à construire la matrice d'observabilité avec les premières dérivées de Lie (ordre 0 et 1) calculées précédemment, on obtient :

$$\mathcal{O} = \begin{pmatrix}
-\cos(\phi_1) & -\sin(\phi_1) & 0 \\
-\cos(\phi_2) & -\sin(\phi_2) & 0 \\
\frac{v}{d_3}\sin(\phi_3)\sin(\psi_3) & -\frac{v}{d_3}\cos(\phi_3)\sin(\psi_3) & -v\sin(\psi_3)
\end{pmatrix}$$

En regardant les trois premières lignes de la matrice $\mathcal{O}$, on constate dans le cas général que son rang est de 3, qui est égal à la dimension de l'état. Afin de déterminer les cas particuliers où le rang de $\mathcal{O}$ n'est plus égal à 3, on cherche les conditions pour lesquelles son déterminant s'annule :

$$\det(\mathcal{O}) = v\sin(\psi_3)\Big(\sin(\phi_1)\cos(\phi_2) - \cos(\phi_1)\sin(\phi_2)\Big)$$

Ce dernier s'annule donc quand :

— $v = 0$ : le robot ne bouge pas,

— $\psi_3 = \{0, \pi\}$ : le robot est toujours orienté vers la balise 3 (ou s'en éloigne),

— $\phi_1 = \phi_2$ : les balises 1 et 2 sont alignées.

Ainsi, en dehors de certaines configurations de balises et de trajectoires du robot, le système est localement faiblement observable pour $l \geq 3$ balises.

Maintenant que nous avons établi que le problème de localisation d'un robot mobile avec des mesures de distance uniquement est observable uniquement à partir de deux balises, nous allons voir dans la section suivant comment il est possible de fusionner les informations de plusieurs capteurs afin de localiser le robot.

## 3.6 Localisation par fusion de données

Comme cela a déjà été souligné dans la section consacrée à l'état de l'art, les méthodes classiques de fusion de données reposent principalement sur l'estimation d'état. Bien que d'autres méthodes existent, avec parfois de meilleures performances, les techniques de filtrage restent une valeur sûre pour de nombreuses utilisations. Parmi les algorithmes de filtrage, deux méthodes se dégagent en robotique mobile : le filtre de Kalman et le filtre particulaire. Dans ce travail, un bon nombre d'algorithmes développés sont basés sur le filtre de Kalman. Ainsi, avant de présenter plus en détails les algorithmes de localisation fusionnant les mesures de distance avec d'autres capteurs, nous allons d'abord rappeler brièvement les équations générales du filtre de Kalman dans le cas d'un système linéaire, puis d'un système non-linéaire. Dans un deuxième temps, nous appliquerons les différents modèles présentés au chapitre 2 au filtre de Kalman non-linéaire ou étendu (EKF) afin d'obtenir des algorithmes de localisation en 2D et en 3D. Enfin, les performances du filtre EKF développé seront analysées à travers les résultats de simulations et de jeux de données réelles, puis comparées à d'autres algorithmes de fusion de données.

### 3.6.1 Filtre de Kalman

Le filtre de Kalman, introduit dans [71], est devenu un outil très utilisé en robotique mobile. Le détail de son fonctionnement et de ses propriétés, ainsi que son application au cas de la localisation et du SLAM en robotique mobile sont expliqués dans le livre *Probabilistic Robotics* [126]. Nous ne présentons ici qu'un rappel des principales équations régissant son fonctionnement. Le filtre de Kalman est un observateur permettant d'estimer les variables d'état d'un système à partir des équations du modèle du système et de mesures faites sur des grandeurs liées aux variables d'état. L'hypothèse principale du filtre de Kalman est que l'ensemble des variables ainsi que les différents bruits sont modélisés par des distributions normales qui peuvent être facilement représentés pas deux paramètres : le vecteur des moyennes $\mathbf{m}$ et la matrice de covariance $\mathbf{P}$. Pour rappel la distribution normale d'une variable $\mathbf{x}$ est donnée par :

$$p(\mathbf{x}) = \frac{1}{\sqrt{2\pi det(\mathbf{P})}} \exp\Big( -\frac{1}{2}(\mathbf{x}-\mathbf{m})^T \mathbf{P}^{-1}(\mathbf{x}-\mathbf{m})\Big) \tag{3.37}$$

La matrice de covariance $\mathbf{P}$ est une matrice symétrique définie positive.

#### Filtre de Kalman linéaire (KF)

Comme montré à la Figure 3.1, le filtre de Kalman est composé de deux phases principales : la prédiction et la correction. Dans le cas d'un système linéaire, les équations modélisant ces

deux étapes sont linéaires par rapport au vecteur d'état $\mathbf{x}$. L'étape de prédiction, qui modélise l'évolution du système, est représentée par :

$$\mathbf{x}_t = \mathbf{F}_t\mathbf{x}_{t-1} + \mathbf{B}_t\mathbf{u}_t + \epsilon_t \tag{3.38}$$

avec $\mathbf{x}_t \in \mathcal{R}^n$ le vecteur d'état à l'instant t, $\mathbf{u}_t \in \mathcal{R}^m$ le vecteur de contrôle du système à l'instant t et $\epsilon_t$ un bruit blanc Gaussien modélisant l'incertitude du processus de transition d'état de moyenne nulle et de matrice de covariance $\mathbf{Q}_t$. $\mathbf{F}_t$ est une matrice de taille $[n \times n]$, la matrice $\mathbf{B}_t$ est de taille $[n \times m]$.

Durant l'étape de correction, les mesures sur l'état sont fusionnées afin d'estimer les variables d'état. Une équation de mesure est définie par :

$$\mathbf{z}_t = \mathbf{H}_t\mathbf{x}_t + \delta_t \tag{3.39}$$

avec $\mathbf{z}_t$ le vecteur des mesures, $\mathbf{H}_t$ une matrice de taille $[l \times n]$, $l$ étant le nombre de mesures. De plus $\delta_t$ est un bruit blanc Gaussien représentant l'incertitude du processus de mesure de moyenne nulle et de matrice de covariance $\mathbf{R}_t$.

Les équations du filtre de Kalman linéaire pour la prédiction sont données pas :

$$\mathbf{x}_t^- = \mathbf{F}_t\mathbf{x}_{t-1} + \mathbf{B}_t\mathbf{u}_t \tag{3.40}$$
$$\mathbf{P}_t^- = \mathbf{F}_t^T\mathbf{P}_t\mathbf{F}_t + \mathbf{Q}_t \tag{3.41}$$

Pour la correction on a :

$$\nu_t = \mathbf{z}_t - \mathbf{H}_t\mathbf{x}_t \tag{3.42}$$
$$\mathbf{S}_t = \mathbf{H}_t^T\mathbf{P}_t^-\mathbf{H}_t + \mathbf{R}_t \tag{3.43}$$
$$\mathbf{K}_t = \mathbf{P}_t^-\mathbf{H}_t^T\mathbf{S}_t^{-1} \tag{3.44}$$
$$\mathbf{x}_t = \mathbf{x}_t^- + \mathbf{K}_t\nu_t \tag{3.45}$$
$$\mathbf{P}_t = (\mathbf{I} - \mathbf{K}_t\mathbf{H}_t)\mathbf{P}_t^- \tag{3.46}$$

Le vecteur $\nu_t$ est appelé l'innovation et correspond simplement à la différence entre la mesure et la prédiction de la mesure. La matrice d'innovation associée, $\mathbf{S}_t$, correspond, elle, à la confiance que l'on peut accorder aux mesures qui ont été faites. La matrice $\mathbf{K}_t$ est appelée gain de Kalman, et représente la confiance avec laquelle les variables d'état peuvent être corrigées avec les mesures. Le filtre de Kalman permet alors, en répétant ces deux étapes, d'obtenir une estimation des variables d'état qui minimise la variance de l'état.

### Filtre de Kalman étendu (EKF)

Dans le cas où le modèle du système et/ou des mesures est non-linéaire, il est nécessaire d'adapter la méthode. Dans cette nouvelle version, appelée filtre de Kalman étendu, les différentes équations non-linéaires sont linéarisées avec un développement de Taylor afin de se ramener à un système linéaire.

L'équation de prédiction de l'état est modélisée par la fonction $\mathbf{f}$ :

$$\mathbf{x}_k^- = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_k) + \epsilon_k$$

La prédiction de la matrice de covariance devient alors :

$$\mathbf{P}_k^- = \mathbf{F}_k\mathbf{P}_{k-1}\mathbf{F}_k^T + \mathbf{W}_k\mathbf{Q}_k\mathbf{W}_k^T$$

avec :

$$\mathbf{F}_k = \frac{\partial\mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_k)}{\partial\mathbf{x}_{k-1}}$$

$$\mathbf{W}_k = \frac{\partial \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)}{\partial \mathbf{u}_k}$$

les Jacobiens de la fonction de prédiction $\mathbf{f}$ par rapport à l'état et au bruit respectivement. L'équation de correction est modélisée par la fonction $\mathbf{h}$ :

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k^-) + \delta_k$$

Les équations de correction sont donc données par :

$$\begin{aligned}
\nu_k &= \mathbf{z}_k - \mathbf{h}(\mathbf{x}_k^-) \\
\mathbf{S}_k &= \mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k \\
\mathbf{K}_k &= \mathbf{P}_k^- \mathbf{H}_k^T \mathbf{S}_k^{-1} \\
\mathbf{x}_k &= \mathbf{x}_k^- + \mathbf{K}_k \nu_k \\
\mathbf{P}_k &= (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_k^- (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)^T + \mathbf{K}_k \mathbf{R}_k \mathbf{K}_k^T
\end{aligned}$$

La matrice $\mathbf{H}_k$ correspondant au Jacobien de la fonction de mesure $\mathbf{h}$ par rapport à l'état est donnée par :

$$\mathbf{H}_k = \frac{\partial \mathbf{h}(\mathbf{x}_k^-)}{\partial \mathbf{x}_k^-}$$

Maintenant que les équations générales du filtre de Kalman et du filtre de Kalman étendu ont été présentées, nous allons les appliquer au problème de localisation par mesures de distance.

### 3.6.2 Filtre EKF de localisation 2D

On se place ici dans le cas où l'on souhaite localiser un robot mobile évoluant dans un monde plan (2D) à l'aide de mesures de distance par rapport à des balises. De plus, le robot de type unicycle est aussi équipé de codeurs incrémentaux sur ses deux roues motrices. La première chose à faire dans le cas de l'utilisation d'un filtre de Kalman est de définir les variables du vecteur d'état à estimer. Ici, il s'agit de la pose du robot, le vecteur d'état est donc défini par :

$$\mathbf{x} = (x, y, \theta)^T \tag{3.47}$$

Ensuite, il faut définir les équations de prédiction et de correction qui seront utilisées par le filtre afin d'estimer le vecteur d'état. La prédiction, consiste ici, à simplement prédire la prochaine pose du robot à partir de la pose courante et des commandes de vitesse $\mathbf{u} = (v, \omega)^T$. Les équations pour un robot de type unicycle ont été présentées au chapitre précédent dans (2.5), et sont rappelées ici :

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \Leftrightarrow \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} v \cos\theta \\ v \sin\theta \\ \omega \end{pmatrix}$$

L'équation de correction est basée sur l'équation (2.9) du modèle de mesure de distance présentée au chapitre précédent dans la partie sur les balises (cf. 2.3.1). Ainsi, l'équation de correction pour une mesure de distance par rapport à une balise $i$ est donnée par :

$$z = h(\mathbf{x}) = \sqrt{(x_i - x)^2 + (y_i - y)^2} + b_i + \epsilon_i \tag{3.48}$$

avec $\mathbf{x}_i = (x_i, y_i)^T$ la position de la balise $i$, $b_i$ et $\epsilon_i$ le biais et le bruit associés à la balise $i$. Si le biais est non nul, on distingue deux cas. Dans le premier cas, on suppose que celui-ci est obtenu par calibration et donc connu pour chaque balise. Dans le second cas, on suppose qu'il

n'est pas connu à priori et qu'il est nécessaire de l'estimer aussi. Dans ce cas, il suffit d'ajouter $N$ variables dans le vecteur d'état du filtre, $N$ étant le nombre de balises :

$$\mathbf{x} = (x, y, \theta, b_1, b_2, ..., b_N)^T$$

Ces variables $b_i$ représenteront alors le biais à estimer pour chaque balise. L'augmentation de l'état ainsi effectuée ne modifie aucunement les équations de prédiction, seule l'équation de correction change. En effet, le biais utilisé dans l'équation de correction ne proviendra plus d'une valeur prédéfinie, mais bien de la variable correspondante dans le vecteur d'état. De plus, le Jacobien de la fonction $h(\mathbf{x})$ sera légèrement différent, comme cela sera détaillé par la suite.

Les équations de prédiction et de correction du filtre ont été données dans le cas continu. En pratique, l'algorithme EKF sera implémenté en discret dans un calculateur, et il est donc nécessaire de discrétiser les équations correspondantes. En réalisant cette discrétisation pour la prédiction, il est possible de faire apparaître les mesures des codeurs incrémentaux à la place des commandes de vitesse. Ces équations (2.7), ont été présentées au chapitre précédent, dans la partie sur l'odométrie des roues (cf. 2.2.2), et sont rappelées ici :

$$\mathbf{x}_{k+1}^{-} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) + \epsilon_k \Leftrightarrow \begin{cases} x_{k+1} = x_k + (\delta U_k + \epsilon_{\delta U_k}) \cos(\theta_k + \frac{(\delta\Theta_k + \epsilon_{\delta\Theta_k})}{2}) \\ y_{k+1} = y_k + (\delta U_k + \epsilon_{\delta U_k}) \sin(\theta_k + \frac{(\delta\Theta_k + \epsilon_{\delta\Theta_k})}{2}) \\ \theta_{k+1} = \theta_k + (\delta\Theta_k + \epsilon_{\delta\Theta_k}) \end{cases}$$

avec $\mathbf{x}_k = (x_k, y_k, \theta_k)^T$ la pose du robot à l'instant $t_k = k\Delta T$, $(\delta U_k, \delta\Theta_k)$ les mesures de déplacement linéaire et angulaire des codeurs incrémentaux, $(\epsilon_{\delta U_k}, \epsilon_{\delta\Theta_k})$ le bruit sur les mesures des codeurs incrémentaux, et $\Delta T$, la période d'échantillonnage des mesures d'odométrie.

L'équation de mesure en discret ne change pas, et se réécrit simplement :

$$z_{k,i} = h_{k,i}(\mathbf{x}_k) = \sqrt{(x_i - x_k)^2 + (y_i - y_k)^2} + b_i + \epsilon_i$$

en remplaçant $t$ par $k$.

Les équations de prédiction et de correction étant non-linéaires par rapport à l'état, il est nécessaire de calculer les différents Jacobiens pour réaliser les calculs sur le matrice de covariance du filtre. Ainsi pour la prédiction :

$$\begin{aligned} \mathbf{F}_k &= \frac{\partial \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)}{\partial \mathbf{x}_k} \\ &= \begin{pmatrix} 1 & 0 & -\delta U_k \sin\theta_k \\ 0 & 1 & \delta U_k \cos\theta_k \\ 0 & 0 & 1 \end{pmatrix} \end{aligned}$$

$$\begin{aligned} \mathbf{W}_k &= \frac{\partial \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)}{\partial \mathbf{u}_k} \\ &= \begin{pmatrix} \cos(\theta_k + \frac{\delta\Theta_k}{2}) & -\frac{1}{2}\delta U_k \sin(\theta_k + \frac{\delta\Theta_k}{2}) \\ \sin(\theta_k + \frac{\delta\Theta_k}{2}) & \frac{1}{2}\delta U_k \cos(\theta_k + \frac{\delta\Theta_k}{2}) \\ 0 & 1 \end{pmatrix} \end{aligned}$$

Pour la correction, on a dans le cas où le biais n'est pas estimé :

$$\begin{aligned} \mathbf{H}_k &= \frac{\partial \mathbf{h}(\mathbf{x}_k)}{\partial \mathbf{x}_k} \\ &= \begin{pmatrix} \frac{-(x_i - x)}{d_{i,k}} & \frac{-(y_i - y)}{d_{i,k}} & 0 \end{pmatrix} \end{aligned}$$

avec $d_{i,k} = \sqrt{(x_i - x_k)^2 + (y_i - y_k)^2}$.

Si le biais est estimé, on obtient alors :

$$\mathbf{H}_k = \begin{pmatrix} \frac{-(x_i - x)}{d_{i,k}} & \frac{-(y_i - y)}{d_{i,k}} & 0 & \dots & 0 & 1 & 0 & \dots \end{pmatrix}$$

où l'on complète simplement la partie correspondant aux variables du biais des balises. Les valeurs sont égales à zéro, sauf pour la mesure correspondant à la $i^{\text{ème}}$ balise.

Afin d'assurer une bonne estimation par le filtre EKF, il est nécessaire de régler correctement les différents bruits affectant les étapes de prédiction et de correction. Pour cela, il s'agit de régler les coefficients des matrices $\mathbf{Q}_k$ et $\mathbf{R}_k$. Comme le filtre de Kalman suppose que les bruits sont modélisés par des distributions gaussiennes, ces deux matrices représentent des matrices de covariance. Dans le cas de la prédiction, on suppose que le bruit $(\epsilon_{\delta U_k}, \epsilon_{\delta\Theta_k})$ provient de la mesure des codeurs incrémentaux $(\delta U_k, \delta\Theta_k)$. On considère que ces deux composantes sont non-corrélées et donc que la matrice $\mathbf{Q}_k$ à la forme suivante :

$$\mathbf{Q}_k = \begin{pmatrix} \sigma^2_{\delta U_k} & 0 \\ 0 & \sigma^2_{\delta\Theta_k} \end{pmatrix} \tag{3.49}$$

où l'on suppose que les écarts-types $\sigma_*$ sont proportionnels à la distance et à l'angle mesuré :

$$\begin{cases} \sigma_{\delta U_k} = k_{\delta U_k} |\delta U_k| \\ \sigma_{\delta\Theta_k} = k_{\delta\Theta_k} |\delta\Theta_k| \end{cases} \tag{3.50}$$

avec $k_{\delta U_k}$ et $k_{\delta\Theta_k}$ des coefficients de proportionnalité déterminés expérimentalement par des tests sur l'odométrie ou obtenus par réglage afin d'obtenir les performances souhaitées par le filtre EKF.

Concernant le bruit sur les mesures de distance, représenté par la matrice $\mathbf{R}_k$, il s'agit tout simplement de la variance du bruit de la mesure de distance. Dans ce travail, la variance est modélisée par une dépendance à la distance, ce qui se traduit par l'équation suivante :

$$\mathbf{R}_k = \sigma^2_{k,i} = (1 + \eta_i d_{i,k})^2 \sigma^2_{i0}$$

avec $\eta_i$ le paramètre de dépendance à la distance et $\sigma^2_{i0}$ la variance initiale de la balise $i$.

En définissant les équations de prédiction et de correction, ainsi que leurs bruits et Jacobiens associées, toutes les informations nécessaires à l'implémentation du filtre EKF sont réunies. Nous allons maintenant voir comme il est possible de détecter et rejeter d'éventuelles mesures de distance aberrantes.

### Rejet des mesures erronées

Comme pour les algorithmes de localisation utilisant uniquement des mesures de distance, le filtre de Kalman est également sensible aux mesures erronées. En effet, tant que le bruit affectant les mesures de distance reste conforme au bruit spécifié par la matrice $\mathbf{R}$, alors le filtre devrait estimer correctement les variables du vecteur d'état. Cependant, dès que les mesures sont entachées d'un bruit non modélisé ou s'éloignant trop du modèle, le filtre aura de fortes chances d'être perturbé et de donner une estimation de position erronée. Heureusement, dans le cas du filtre de Kalman, il existe un test facile à mettre en place permettant de rejeter une mesure s'éloignant trop de la prédiction faite à partir de l'état courant. Ce test est basé sur la distance de Mahalanobis [90] et utilise directement les quantités calculées pendant la phase de correction du filtre. Ainsi, au moment de l'étape de correction, une fois le vecteur $\nu_k$ et la matrice $\mathbf{S}_k$ d'innovation calculés, la distance de Mahalanobis $d_{\text{maha}}$ se calcule de la façon suivante :

$$d_{\text{maha}} = \nu_k^T \mathbf{S}_k^{-1} \nu_k \tag{3.51}$$

La distance de Mahalanobis calcule donc le carré de l'erreur entre la mesure et la prédiction, pondérée par les incertitudes de la mesure et de la prédiction associées. Afin de rejeter une mesure aberrante, il suffit alors de comparer cette distance $d_{\text{maha}}$ à un seuil $s_{\text{maha}}$ fixé. Si la distance est supérieure au seuil, alors la mesure est considérée comme erronée. Dans le cas contraire, la mesure est acceptée et l'on poursuit normalement la fusion de la mesure à partir des équations de correction. Il est bien sûr évident que ce test ne fonctionnera correctement que si l'estimation courante du vecteur d'état est bonne. Ce test peut donc servir à indiquer si l'estimation se déroule bien en comptant le nombre de mesures de distance rejetées par les tests de Mahalanobis. Si une fraction importante des mesures est rejetée à partir d'un instant donné, on peut fortement supposer que l'estimation n'est plus bonne et que le filtre a divergé et enclencher une procédure de relocalisation du robot.

Nous allons maintenant évaluer les performances du filtre présenté et le comparer à d'autres algorithmes de localisation utilisant le principe de fusion de données.

### Performance et comparaison du filtre EKF de localisation

Afin de valider le bon fonctionnement du filtre EKF de localisation, celui-ci est testé sur les jeux de données Gesling2 et Plaza1 publiés dans [32]. Pour le jeu de données Gesling2, les vraies mesures d'odométrie et de distance sont extraites de la vérité terrain, puis les différents bruits testés y sont ajoutés. Pour le jeu de données Plaza1, les données réelles sont directement utilisées. Les différents bruits et niveaux de bruit appliqués sur les mesures de distance pour le premier jeu de données, Gesling2, sont :

- Sans bruit : les mesures correspondent à la vérité terrain,
- Std1 : ajout d'un bruit blanc Gaussien de moyenne nulle et d'écart-type égal à 0.0333 m,
- Std2 : comme Std1, mais avec un écart-type de 0.167 m,
- Std1+biais : ajout d'un biais de 0.2 m en plus du bruit Std1,
- Std1+$x$% outliers : bruit Std1 dont $x$% des mesures sont erronées ($x$=10%,20%,30%,40%,50%).

Pour les mesures d'odométrie, celles-ci sont bruitées en accord avec le modèle défini pour le filtre EKF par les équations (3.49) et (3.50). Ainsi les coefficients de proportionnalités permettant de générer les bruits sur les mesures des encodeurs sont fixés à : $k_{\delta U} = 0.01$ et $k_{\delta\Theta} = 0.05$.

Dans les simulations qui suivent, les paramètres utilisés pour le filtre EKF sont les suivants :

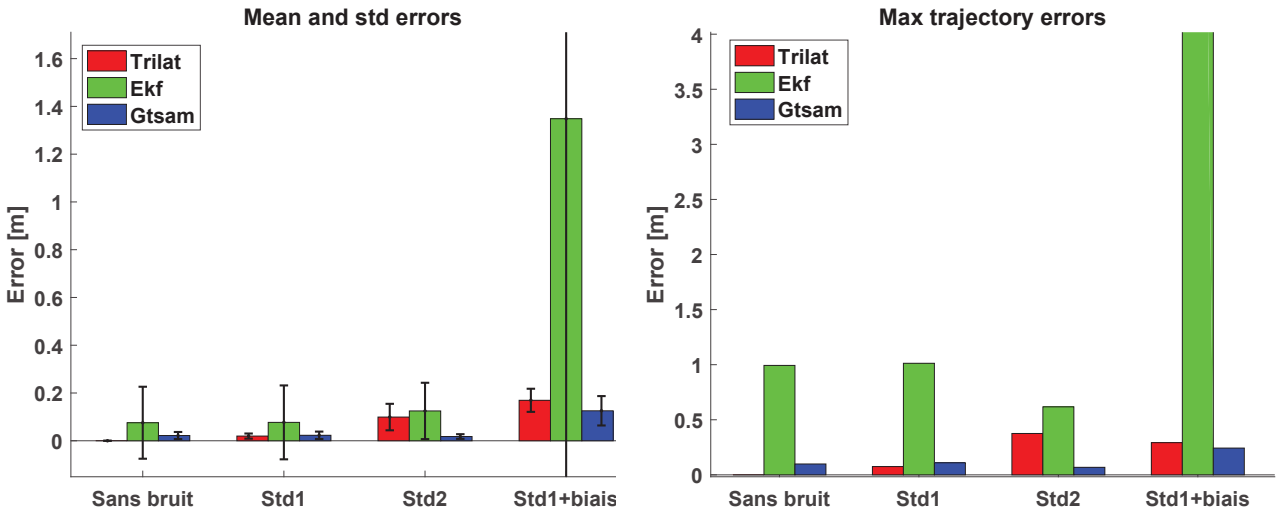
- Bruit sur l'odométrie : $k_{\delta U} = 0.011$, $k_{\delta\Theta} = 0.055$,
- Écart-type des mesures de distance : $\sigma_{i0}$ = Std1 ou Std2 en fonction du bruit utilisé,
- Seuil de Mahalanobis $s_{\text{maha}} = 100$.

Les performances du filtre EKF sont aussi comparées à celles d'autres algorithmes de localisation :

- Algorithme de trilatération Trilat+RANSAC+LM présenté dans la section précédente,
- Algorithme d'optimisation GTSAM présenté dans [27],

Afin de comparer les différents algorithmes, les erreurs entre la position estimée et la vérité terrain sont calculées tout au long de la trajectoire. La moyenne, la déviation standard et le maximum des erreurs sont ensuite calculés. Les figures 3.18, 3.19, 3.20, regroupent l'ensemble des résultats obtenus pour le jeu de données Gesling2 sous la forme de graphes en barres pour représenter les erreurs. Dans chaque figure, les erreurs des différents algorithmes sont comparées. La Figure 3.18 regroupe les résultats correspondant aux différents bruits sans mesures erronées, alors que les figures 3.19 et 3.20 présentent les résultats pour un nombre croissant de mesures erronées.

En regardant pour l'instant uniquement les résultats du filtre EKF (barres vertes), on constate sur la Figure 3.18 qu'il n'y a pas une très grande différence entre le bruit Std1 et




(a) Erreurs moyennes et écarts-types

(b) Erreurs maximales

FIGURE 3.18 – Comparaison des erreurs de trajectoires moyennes, des écarts-types et des erreurs maximales pour différents algorithmes de localisation et différents bruits pour le jeu de données Gesling2

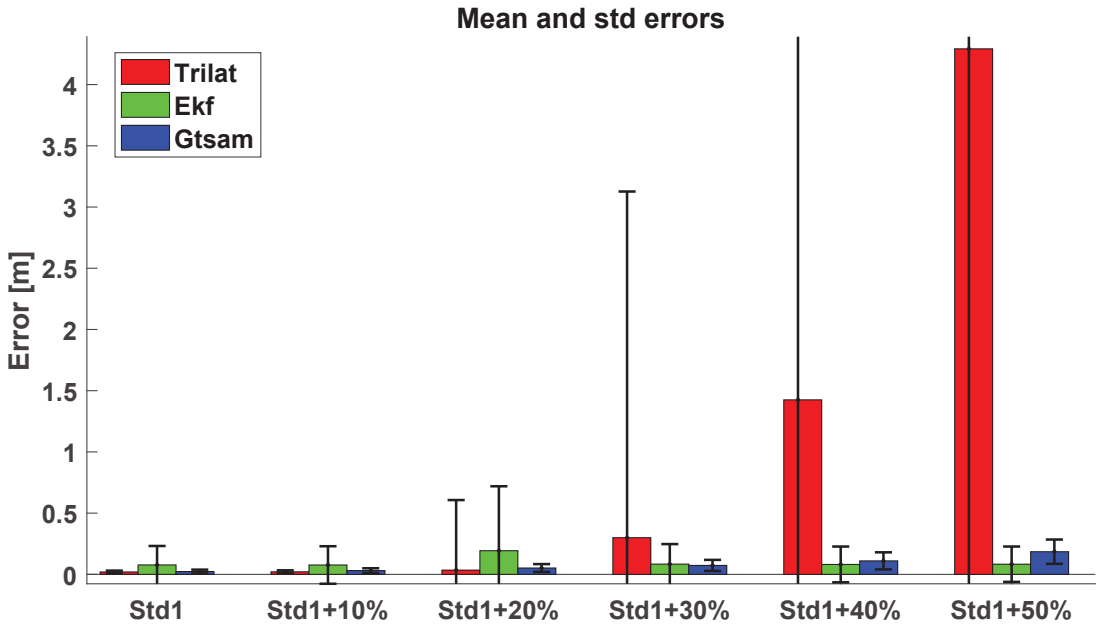
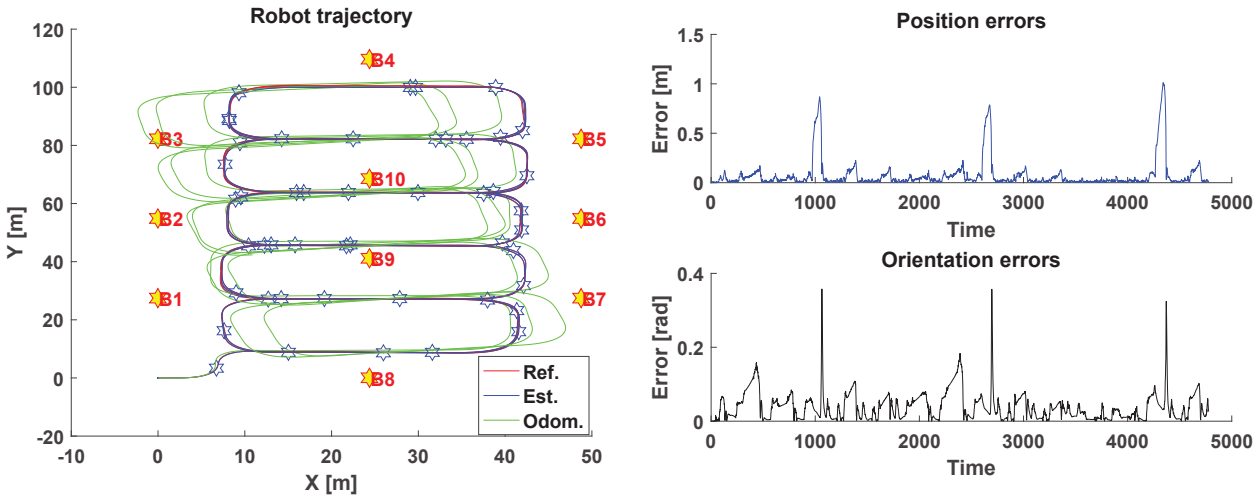



FIGURE 3.19 – Comparaison des erreurs de trajectoires moyennes et des écarts-types pour différents algorithmes de localisation et pour différents pourcentages de mesures erronées pour le jeu de données Gesling2

sans bruit, ce qui montre la bonne robustesse du filtre de Kalman au bruit de mesure. En augmentant la déviation standard de Std1 à Std2, la moyenne des erreurs augmente, mais par contre l'erreur maximale diminue. En ajoutant du biais (Std1+biais), la moyenne, la déviation standard ainsi que le maximum des erreurs augmentent encore plus et de manière assez significative. Ceci provient du fait qu'en fin de trajectoire, pour une raison inexpliquée, le filtre EKF commence à diverger. Par contre, en présence des mesures aberrantes sur les données, on voit sur les figures 3.19 et 3.20 que le filtre EKF garde une bonne estimation de la trajectoire et


FIGURE 3.20 – Comparaison des erreurs de trajectoires maximales pour différents algorithmes de localisation et pour différents pourcentages de mesures erronées pour le jeu de données Gesling2




(a) Trajectoire estimée

(b) Évolution de l'erreur de trajectoire

FIGURE 3.21 – Trajectoire finale et évolution de l'erreur pour le filtre EKF avec le jeu de données Gesling2 pour le bruit Std1

n'est que modérément affecté. On constate juste pour le cas Std1+20% de mesures erronées que le filtre semble avoir plus de mal à s'adapter, notamment en regardant l'erreur maximale qui fait un bon à plus de 3 mètres. Si le filtre EKF garde une bonne estimation même en présence de mesures erronées, c'est en raison du test de Mahalanobis qui est réalisé comme cela a déjà été expliqué. En effet, si l'on observe les valeurs de la Figure 3.22, on remarque que le pourcentage des mesures rejetées par le filtre EKF, lors de l'étape de correction, est en accord avec le pourcentage des mesures aberrantes présentes dans le jeu de données. On constate donc que le test de Mahalanobis est une sécurité efficace pour rejeter les mesures aberrantes et garantir une certaine robustesse au filtre EKF. Une illustration de la trajectoire estimée par le filtre

EKF pour le jeu de données Gesling2 avec le bruit Std1 est visible à la Figure 3.21a. Sur la Figure 3.21b à côté, l'évolution des erreurs de la trajectoire et de l'orientation du robot est aussi montrée. En observant les trajectoires, on peut voir que la trajectoire estimée par le filtre EKF (en bleu) ne dérive pas par rapport à la vraie trajectoire (rouge) au contraire de la trajectoire de l'odométrie seule (vert). Concernant l'évolution des erreurs, on constate que la plupart du temps l'erreur de localisation se situe aux alentours d'une dizaine de centimètres, ce qui est en accord avec la plupart des résultats obtenus avec d'autres bruits. De plus, on constate qu'à chaque fois que l'erreur sur la position du robot augmente, cela entraîne presque toujours immédiatement une augmentation de l'erreur sur l'orientation. En effet, comme le filtre EKF ne dispose que de mesures de distance donnant une information sur la position du robot, et pas de mesure directe sur l'orientation cette dernière est plus difficile à estimer. Ainsi, l'estimation de l'orientation du robot n'est faite qu'implicitement par le filtre en intégrant les données de l'odométrie et en utilisant les mesures de distance pour valider que l'orientation corresponde bien aux changements de position.

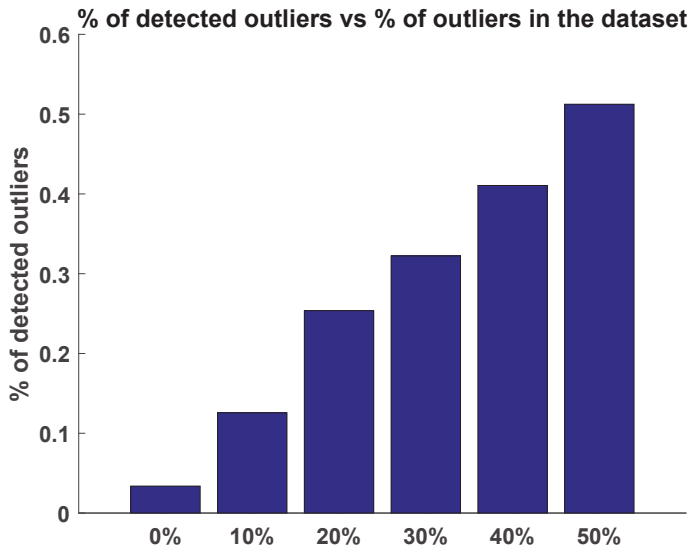
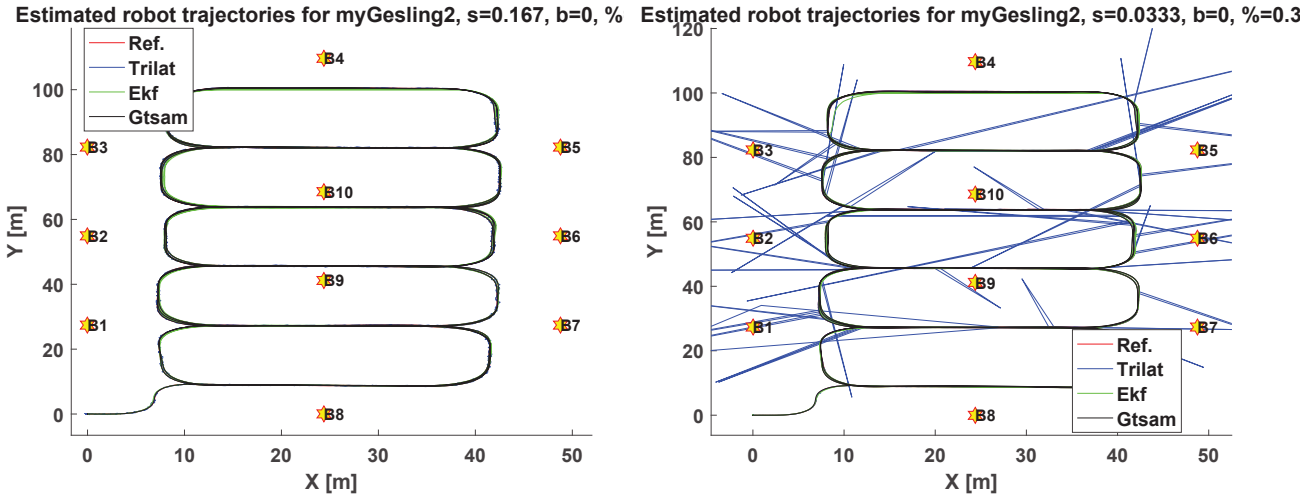



Figure 3.22 – Pourcentages moyens de mesures erronées détectées avec le test de Mahalanobis du filtre EKF pour différents pourcentages de mesures erronées affectant le jeu de données Gesling2

En comparant les différents algorithmes à partir des figures 3.18, 3.19, 3.20, on constate d'une manière générale que le filtre EKF présente des erreurs moyennes et maximales plus élevées que les autres algorithmes. En regardant l'algorithme de trilatération, on remarque que celui-ci présente même pour certains bruits, Sans bruit et Std1 par exemple, des erreurs inférieures à l'algorithme GTSAM. Il faut cependant souligner trois points concernant l'algorithme de trilatération. Le premier étant qu'il n'utilise que les mesures de distance pour calculer une estimation de position, et par conséquent qu'il ne fournit pas dans sa version classique d'estimation sur l'orientation du robot. Le deuxième point étant que les mesures d'odométrie bruitées utilisées par les autres algorithmes n'affectent pas l'algorithme de trilatération. Troisièmement, on rappelle ici que l'algorithme de trilatération a besoin impérativement d'au moins trois mesures de distance pour obtenir une estimation de la position, ce qui n'est pas le cas des autres algorithmes qui peuvent fusionner un nombre quelconque de mesures. Si les performances de l'algorithme de trilatération sont plutôt bonnes en simulation, celles-ci risquent cependant d'être inférieures dans le cas de jeux de données réelles. Ainsi, pour le jeu de données Plaza1 présenté par la suite, il est nécessaire de regrouper les mesures de distance par intervalles de temps

afin d'avoir assez de mesures pour obtenir une estimation, ce qui a pour effet d'introduire des erreurs supplémentaires. Concernant l'algorithme GTSAM, on constate qu'il s'agit de l'algorithme présentant les erreurs moyennes et maximales les plus faibles pour la plupart des bruits. Pour les cas suivants : Sans bruit, Std1, Std2 et Std1+biais, les erreurs de l'algorithme GTSAM sont ainsi généralement de l'ordre de 5 à 10 fois inférieures à celles du filtre EKF. Seul un rééquilibrage des erreurs moyennes apparaît lorsque le pourcentage des mesures erronées augmente (Std1+30%, Std1+40%, Std1+50%). En regardant les écarts-types (lignes noires) pour l'algorithme de trilatération sur la Figure 3.19, on constate que les valeurs deviennent de plus en plus grandes avec le pourcentage de mesures erronées. Ceci est dû au fait que l'algorithme RANSAC ne parvient pas à rejeter correctement les mesures erronées induisant ainsi une estimation complètement faussée. L'algorithme de trilatération classique n'utilisant pas les données d'odométrie, aucune continuité des positions successives n'est imposée. Ceci peut amener au phénomène visible à la Figure 3.23b, avec la courbe bleue, et qui explique les valeurs élevées des écarts-types. En regardant la Figure 3.23a qui montre les trajectoires estimées par les différents algorithmes pour le bruit Std2, on peut difficilement distinguer les trajectoires. De plus, elles sont toutes très proches de la trajectoire de référence (rouge), ce qui illustre pour ce bruit, les bonnes performances de chaque algorithme.



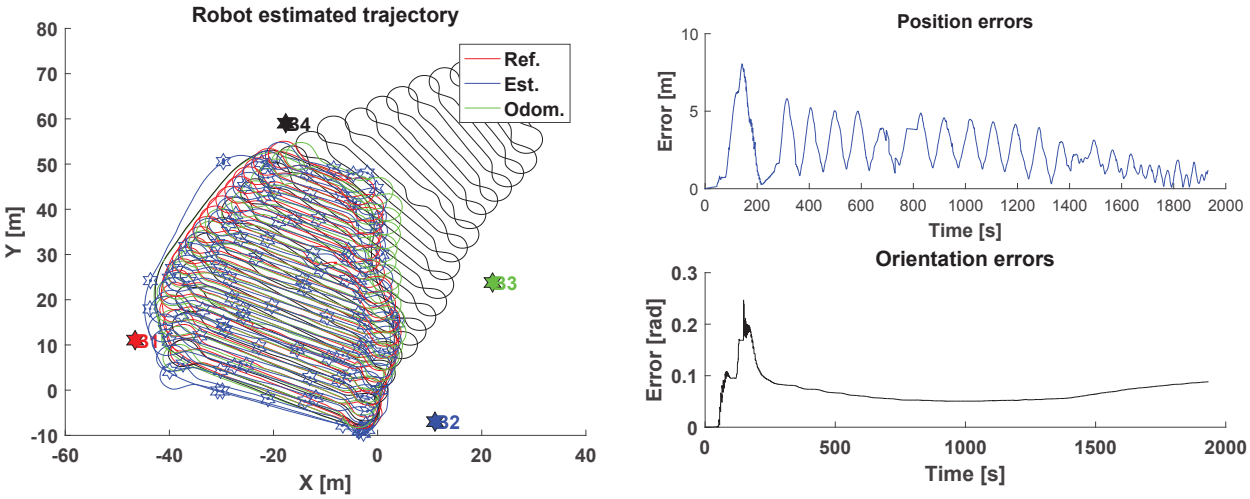

(a) Bruit Std2

(b) Bruit Std1+30% outliers

FIGURE 3.23 – Comparaison des trajectoires estimées par les différents algorithmes pour deux bruits différents sur le jeu de données Gesling2

Les performances du filtre EKF sur le jeu de données Plaza1 avec des conditions réelles de bruit, sont visibles à la Figure 3.24 où la trajectoire finale estimée ainsi que l'évolution des erreurs sont affichées. Comme on peut le voir, les erreurs d'estimation sont plus importantes que précédemment et ceci en raison notamment d'un niveau de bruit sur les mesures de distance plus important. En effet, les mesures de distance sont entachées pour ce jeu de données d'un bruit avec un écart-type de l'ordre du mètre. Cependant, en regardant la Figure 3.24b, on s'aperçoit que l'erreur diminue progressivement au fil de la trajectoire.

En regardant la Figure 3.26 qui donne les erreurs moyennes, maximales et les écarts-types des algorithmes de localisation sur le jeu de données Plaza1, on aperçoit ici que le filtre EKF présente des erreurs moyennes et maximales légèrement plus faibles que l'algorithme GTSAM. La trilatération présente des erreurs plus élevées que les autres algorithmes. De plus, il faut noter que par rapport aux autres algorithmes qui fournissent une estimation pour toutes les positions, l'algorithme de trilatération, lui, ne fournit une estimation que pour des positions




(a) Trajectoire estimée

(b) Évolution de l'erreur de trajectoire

FIGURE 3.24 – Trajectoire et évolution de l'erreur pour le filtre EKF avec le jeu de données Plaza1

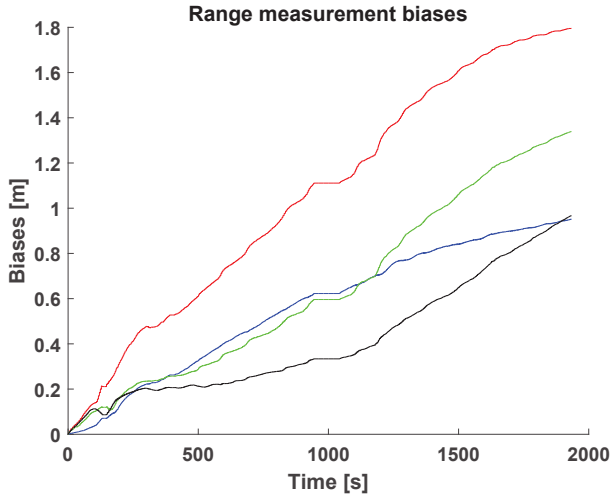
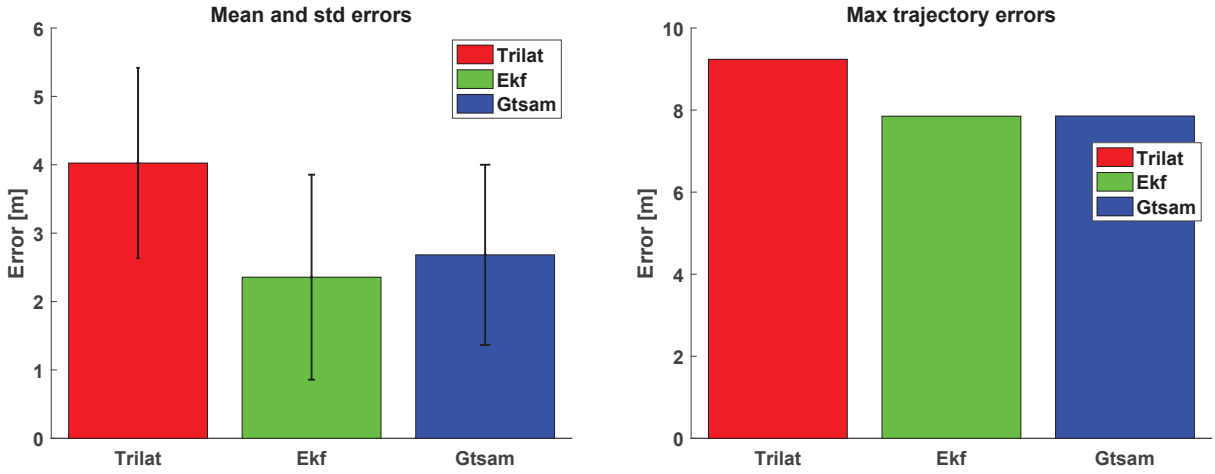



FIGURE 3.25 – Évolution de l'estimation du biais des 4 balises pour le jeu de données Plaza1

ou l'on a pu accumuler assez de mesures de distance. Ainsi, en pratique, l'algorithme de trilatération n'est efficace qu'à la condition d'être sûr d'avoir toujours au moins trois mesures de distance par rapport à trois balises différentes pour obtenir une estimation de position. Dans le cas contraire, il vaut mieux privilégier un algorithme de fusion de données et exploiter les informations complémentaires d'autres capteurs.

Le filtre EKF, contrairement à un filtre particulaire par exemple, n'est pas un algorithme de localisation globale, mais plutôt de suivi de la position. Il est donc nécessaire pour un bon fonctionnement d'initialiser son état avec des valeurs proches de la vérité terrain. Ceci peut être un inconvénient dans le cas d'une utilisation en conditions réelles. Cependant, grâce aux mesures de distance par rapport à des balises, il est possible d'obtenir une première estimation de la position du robot en utilisant un algorithme de trilatération. Ainsi, au démarrage du filtre EKF, on peut imposer que le robot attende d'avoir collecté un minimum de trois mesures de distance par rapport à trois balises différentes afin d'obtenir une estimation de sa position. Cette position est alors utilisée pour initialiser la position du robot dans l'état du filtre. Il ne reste donc plus que l'orientation initiale du robot à connaître pour initialiser correctement le




(a) Erreurs moyennes et écarts-types

(b) Erreurs maximales

FIGURE 3.26 – Comparaison des erreurs de trajectoires moyennes, des écarts-types et des erreurs maximales pour différents algorithmes de localisation pour le jeu de données Plaza1

filtre. Ce même principe peut être appliqué afin de réinitialiser le filtre EKF, dans le cas où celui-ci perd le suivi de la position. Cette technique, bien que ne permettant pas de retrouver l'orientation du robot, permet d'améliorer la robustesse de l'algorithme en cas de perte de suivi de la pose du robot.

Pour l'orientation du robot, il existe une solution permettant de faciliter son initialisation. Si l'on considère le cas où le robot est localisé dans le repère des balises, il est possible de fixer arbitrairement la position des balises. On peut ainsi imposer que la balise n° 1 corresponde à l'origine du repère, et que la droite passant par les positions des balises n° 1 et n° 2 définisse l'axe $x$ du repère. Dans cette situation, il est alors aisé, par une intervention humaine, de déposer le robot sur le terrain avec une orientation parallèle à la direction de l'axe $x$ et ainsi initialiser l'orientation du robot à zéro.

Le filtre de Kalman qui vient d'être présenté et testé, repose sur l'hypothèse que le robot évolue dans un monde plan. Ainsi, toutes les composantes suivant l'axe $z$, que ce soit pour le robot ou les balises sont supposées nulles. Dans les situations où cette hypothèse n'est plus exacte, il est alors nécessaire de prendre en compte et d'effectuer les compensations nécessaires. Une autre solution, plus appropriée, mais ajoutant un niveau de complexité supplémentaire, consiste à réaliser l'estimation des paramètres directement dans un environnement 3D. Nous allons donc voire maintenant l'extension en 3D du filtre EKF qui d'être présenté.

### 3.6.3 Filtre EKF de localisation 3D

Précédemment, un algorithme de localisation en 2D basé sur un filtre de Kalman étendu a été présenté. Cependant, dans le cas où le robot mobile est amené à évoluer dans un environnement extérieur, les hypothèses d'un monde 2D ne sont souvent plus valables. Il est donc nécessaire d'estimer les différentes variables (pose, vitesses, etc) en 3D. Pour cela, un nouvel algorithme de localisation, lui aussi basé sur un filtre de Kalman étendu, a été développé. Ce nouvel algorithme est un peu plus qu'une simple extension du cas 2D précédent. En effet, le filtre de Kalman développé utilise une formulation basée sur l'erreur d'état (ESKF), [114], [120], qui estime l'erreur de l'état au lieu de l'état lui-même, et permet aussi de bénéficier d'une représentation minimale des paramètres à estimer. De plus, le filtre permet de fusionner les données d'une centrale inertielle (accéléromètre, gyroscope, magnétomètre) et éventuellement

d'un GPS, afin de bénéficier d'informations supplémentaires sur son orientation et sa vitesse dans l'espace 3D.

Le vecteur d'état du filtre contenant les variables à estimer voit donc sa taille augmenter pour prendre en compte le passage à la 3D et l'ajout de nouveaux capteurs. Les variables estimées sont donc :

- La pose du robot : position $\mathbf{p} = (x, y, z)^T$ et orientation (quaternion) $\mathbf{q} = (q_w, q_x, q_y, q_z)^T$,
- Les vitesse linéaire $\mathbf{v} = (v_x, v_y, v_z)^T$, et angulaire $\omega = (\omega_x, \omega_y, \omega_z)^T$,
- Les biais de la centrale inertielle : accéléromètre $\mathbf{b}_a = (b_{a,x}, b_{a,y}, b_{a,z})^T$, gyroscope $\mathbf{b}_g = (b_{g,x}, b_{g,y}, b_{g,z})^T$ et magnétomètre $\mathbf{b}_m = (b_{m,x}, b_{m,y}, b_{m,z})^T$,
- Optionnel : le vecteur du champ magnétique terrestre dans le repère de la Terre : $\mathbf{m} = (m_x, m_y, m_z)^T$,
- Optionnel : le vecteur de la gravité terrestre dans le repère de la Terre : $\mathbf{g} = (g_x, g_y, g_z)^T$,
- Optionnel : les biais associés aux mesures de distance des balises : $\mathbf{b} = (b_1, ..., b_i, ..., b_N)^T$.

Le vecteur d'état du filtre est donc donné par :

$$\mathbf{x} = (\mathbf{p}, \mathbf{q}, \mathbf{v}, \omega, \mathbf{b}_a, \mathbf{b}_g, \mathbf{b}_m, \mathbf{m}, \mathbf{g}, \mathbf{b})^T \tag{3.52}$$

Si l'on définit l'erreur d'estimation $\delta x$ d'une variable $x$ par :

$$x = \hat{x} \boxplus \delta x$$

avec : $x$ la vraie valeur, $\hat{x}$ la valeur estimée et $\delta x$ l'erreur d'estimation, alors le vecteur d'erreur d'état du filtre est défini par :

$$\delta\mathbf{x} = (\delta\mathbf{p}, \delta\theta, \delta\mathbf{v}, \delta\omega, \delta\mathbf{b}_a, \delta\mathbf{b}_g, \delta\mathbf{b}_m, \delta\mathbf{m}, \delta\mathbf{g}, \delta\mathbf{b})^T \tag{3.53}$$

avec :

- $\delta\mathbf{p} = (\delta x, \delta y, \delta z)^T$,
- $\delta\theta = (\delta\theta_x, \delta\theta_y, \delta\theta_z)^T$,
- $\delta\mathbf{v} = (\delta v_x, \delta v_y, \delta v_z)^T$,
- $\delta\omega = (\delta\omega_x, \delta\omega_y, \delta\omega_z)^T$,
- $\delta\mathbf{b}_a = (\delta b_{a,x}, \delta b_{a,y}, \delta b_{a,z})^T$,
- $\delta\mathbf{b}_g = (\delta b_{g,x}, \delta b_{g,y}, \delta b_{g,z})^T$,
- $\delta\mathbf{b}_m = (\delta b_{m,x}, \delta b_{m,y}, \delta b_{m,z})^T$,
- $\delta\mathbf{m} = (\delta m_x, \delta m_y, \delta m_z)^T$,
- $\delta\mathbf{g} = (\delta g_x, \delta g_y, \delta g_z)^T$,
- $\delta\mathbf{b} = (\delta\mathbf{b}_1, ..., \delta\mathbf{b}_N)^T$.

L'opérateur $\boxplus$ correspond simplement à l'addition classique $(+)$ pour toutes les variables, excepté pour le quaternion, où l'opérateur $\boxplus$ est défini par :

$$\begin{aligned}\mathbf{q} &= \hat{\mathbf{q}} \boxplus \delta\theta \\ &= \hat{\mathbf{q}} \otimes \delta\theta\end{aligned}$$

avec l'opérateur $\otimes$ qui correspond à la multiplication entre deux quaternions [120].

Les équations de prédiction du filtre sont basées sur le modèle cinématique d'un robot de type unicycle et ses mesures des codeurs incrémentaux, mais étendu à la 3D.

L'étape de correction du filtre se fait grâce aux mesures des capteurs suivants :

— Balises : donnent les mesures de distance entre le robot et $N$ balises disposées sur le terrain,

— Accéléromètre : donne la direction du vecteur gravité (en supposant l'accélération du robot nulle),

— Gyroscope : mesure directement la vitesse angulaire du robot,

— Magnétomètre : mesure l'intensité du champ magnétique dans le repère du robot et donne la direction du Nord magnétique,

— Optionnel : un GPS qui donne la position du robot dans le repère de la Terre,

— Pas de mouvement : quand on sait que le robot ne se déplace pas (commandes de vitesse nulles), on peut forcer les vitesses linéaire et angulaire du robot à zéro en faisant une correction sur les vitesses. Ceci a pour effet d'améliorer sensiblement les dérives liées à l'intégration de certaines mesures comme l'odométrie et de la centrale inertielle.

Nous allons maintenant voir en détails les équations de prédiction et de correction des différents capteurs.

### Prédiction

Les équations de prédiction du filtre 3D étant basées sur les équations cinématiques en 3D d'un robot unicycle, la fonction de prédiction du filtre, $\mathbf{f}$, est donnée dans le cas continu par les équations différentielles suivantes :

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \epsilon \Leftrightarrow \begin{cases} \dot{\mathbf{p}} = \mathbf{R}(\mathbf{q})(\mathbf{v}_u + \epsilon_{v,u}) \\ \dot{\mathbf{q}} = \frac{1}{2}\mathbf{q} \otimes (\omega_u + \epsilon_{\omega,u}) \\ \dot{\mathbf{v}} = \epsilon_v \\ \dot{\omega} = \epsilon_\omega \\ \dot{\mathbf{b}}_a = \epsilon_{\mathbf{b}_a} \\ \dot{\mathbf{b}}_g = \epsilon_{\mathbf{b}_g} \\ \dot{\mathbf{b}}_m = \epsilon_{\mathbf{b}_m} \\ \dot{\mathbf{m}} = \epsilon_{\mathbf{m}} \end{cases}$$

avec : $\epsilon_{v,u}$, $\epsilon_{\omega,u}$, $\epsilon_v$, $\epsilon_\omega$, $\epsilon_{\mathbf{b}_*}$, $\epsilon_{\mathbf{m}}$ des bruits blanc Gaussien de moyenne nulle, et $\mathbf{R}(\mathbf{q})$ la matrice de rotation construite à partir de l'estimation courante du quaternion. Le passage de la représentation en quaternion à la matrice de rotation est donné par :

$$\mathbf{R}(\mathbf{q}) = \begin{pmatrix} 2(q_w^2 + q_x^2) - 1 & 2(q_x q_y - q_w q_z) & 2(q_w q_y + q_x q_z) \\ 2(q_w q_z + q_x q_y) & 2(q_w^2 + q_y^2) - 1 & 2(q_y q_z - q_w q_x) \\ 2(q_x q_z - q_w q_y) & 2(q_w q_x + q_y q_z) & 2(q_w^2 + q_z^2) - 1 \end{pmatrix}$$

Si l'on suppose que la vitesse linéaire du robot, exprimée dans le repère du robot, comprend deux composantes non-nulles suivant $x$ et $y$ :

$$\mathbf{v}_u = \begin{pmatrix} v_{ux} & v_{uy} & 0 \end{pmatrix}^T$$

on obtient alors l'évolution en continu de la position du robot suivante :

$$\dot{\mathbf{p}} = \mathbf{R}(\mathbf{q}) \begin{pmatrix} v_{ux} + \epsilon_{vux} \\ v_{uy} + \epsilon_{vuy} \\ 0 \end{pmatrix}$$

et en discret :

$$\mathbf{p}_{k+1} = \mathbf{p}_k + \mathbf{R}(\mathbf{q}) \begin{pmatrix} \delta U_x \\ \delta U_y \\ 0 \end{pmatrix}$$
$$\Leftrightarrow \begin{pmatrix} x_{k+1} \\ y_{k+1} \\ z_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ y_k \\ z_k \end{pmatrix} + \begin{pmatrix} r_{00}\delta U_x + r_{01}\delta U_y \\ r_{10}\delta U_x + r_{11}\delta U_y \\ r_{20}\delta U_x + r_{21}\delta U_y \end{pmatrix}$$

avec :

$$\begin{cases} \delta U_x = \delta U \cos(\delta\Theta/2) \\ \delta U_y = \delta U \sin(\delta\Theta/2) \end{cases}$$

Pour la vitesse angulaire du robot, exprimée dans le repère du robot, la seule composante est la rotation suivant l'axe $z$ du robot :

$$\omega_u = \begin{pmatrix} 0 & 0 & \omega_{uz} \end{pmatrix}^T$$

on obtient alors l'évolution en continu de l'orientation du robot suivante :

$$\dot{\mathbf{q}} = \frac{1}{2}\mathbf{q} \otimes (\omega_{uz} + \epsilon_{\omega uz})$$

et en discret :

$$\mathbf{q}_{k+1} = \mathbf{q}_k \otimes \mathbf{q} \left\{ \delta\Theta \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \right\} = \mathbf{q}_k \otimes \begin{pmatrix} \cos(\delta\Theta/2) \\ 0 \\ 0 \\ \sin(\delta\Theta/2) \end{pmatrix}$$
$$\Leftrightarrow \begin{pmatrix} q_{w,k+1} \\ q_{x,k+1} \\ q_{y,k+1} \\ q_{z,k+1} \end{pmatrix} = \begin{pmatrix} q_{w,k}\cos(\delta\Theta/2) - q_{z,k}\sin(\delta\Theta/2) \\ q_{x,k}\cos(\delta\Theta/2) + q_{y,k}\sin(\delta\Theta/2) \\ q_{y,k}\cos(\delta\Theta/2) - q_{x,k}\sin(\delta\Theta/2) \\ q_{z,k}\cos(\delta\Theta/2) + q_{w,k}\sin(\delta\Theta/2) \end{pmatrix}$$

L'évolution des erreurs d'état liées à la pose du robot en continu sont données par :

$$\begin{cases} \delta\dot{\mathbf{p}} = -\mathbf{R}\lfloor \mathbf{v}_u \rfloor_\times \delta\theta \pm \mathbf{R}\epsilon_{vu} \\ \delta\dot{\theta} = -\lfloor \omega_u \rfloor_\times \delta\theta \pm \epsilon_{\omega u} \end{cases}$$

et en discret :

$$\begin{cases} \delta\mathbf{p}_{k+1} = \delta\mathbf{p}_k - \mathbf{R}\lfloor \begin{pmatrix} \delta U_x \\ \delta U_y \\ 0 \end{pmatrix} \rfloor_\times \delta\theta_k - \mathbf{R}\Delta T \begin{pmatrix} \epsilon_{vux} \\ \epsilon_{vuy} \\ 0 \end{pmatrix} \\ \delta\theta_{k+1} = \mathbf{R}^T \left\{ \mathbf{q} \left\{ \begin{pmatrix} 0 \\ 0 \\ \delta\Theta \end{pmatrix} \right\} \right\} \delta\theta_k - \Delta T \begin{pmatrix} 0 \\ 0 \\ \epsilon_{\omega uz} \end{pmatrix} \end{cases}$$

avec :

$$-\mathbf{R}\lfloor \begin{pmatrix} \delta U_x \\ \delta U_y \\ 0 \end{pmatrix} \rfloor_\times = -\mathbf{R} \begin{pmatrix} 0 & 0 & \delta U_y \\ 0 & 0 & -\delta U_x \\ -\delta U_y & \delta U_x & 0 \end{pmatrix} = \begin{pmatrix} r_{02}\delta U_y & -r_{02}\delta U_x & -r_{00}\delta U_y + r_{01}\delta U_x \\ r_{12}\delta U_y & -r_{12}\delta U_x & -r_{10}\delta U_y + r_{11}\delta U_x \\ r_{22}\delta U_y & -r_{12}\delta U_x & -r_{20}\delta U_y + r_{21}\delta U_x \end{pmatrix}$$

et

$$\mathbf{R}^T \left\{ \mathbf{q} \left\{ \begin{pmatrix} 0 \\ 0 \\ \delta\Theta \end{pmatrix} \right\} \right\} = \mathbf{R}^T \left\{ \begin{pmatrix} \cos(\delta\Theta/2) \\ 0 \\ 0 \\ \sin(\delta\Theta/2) \end{pmatrix} \right\} =$$

$$\begin{pmatrix} 2\cos(\delta\Theta/2)^2 - 1 & -2\cos(\delta\Theta/2)\sin(\delta\Theta/2) & 0 \\ 2\cos(\delta\Theta/2)\sin(\delta\Theta/2) & 2\cos(\delta\Theta/2)^2 - 1 & 0 \\ 0 & 0 & 2(\cos(\delta\Theta/2)^2 + \sin(\delta\Theta/2)^2) - 1 \end{pmatrix}^T = \tag{3.54}$$

$$\begin{pmatrix} \cos(\delta\Theta) & \sin(\delta\Theta) & 0 \\ -\sin(\delta\Theta) & \cos(\delta\Theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.55}$$

Les Jacobiens des erreurs d'état, liés à la pose du robot, par rapport à l'état et au bruit sont donnés par :

$$\mathbf{F} = \frac{\partial \delta\mathbf{x}_{k+1}}{\partial \delta\mathbf{x}_k} = \begin{pmatrix} \mathbf{I}_{3\times3} & -\mathbf{R}\lfloor \begin{pmatrix} \delta U_x \\ \delta U_y \\ 0 \end{pmatrix} \rfloor_\times \\ \mathbf{O}_{3\times3} & \mathbf{R}^T \left\{ \mathbf{q} \left\{ \begin{pmatrix} 0 \\ 0 \\ \delta\Theta \end{pmatrix} \right\} \right\} \end{pmatrix}$$

et

$$\mathbf{W} = \frac{\partial \delta\mathbf{x}_{k+1}}{\partial \epsilon} = \begin{pmatrix} \begin{pmatrix} r_{00} \\ r_{10} \\ r_{20} \end{pmatrix} & \mathbf{O}_{3\times1} \\ \mathbf{O}_{3\times1} & \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \end{pmatrix}$$

Dans l'étape de prédiction, seules les variables d'état liées à la pose du robot sont concernées. Les autres variables sont supposées constantes et l'équation d'évolution en discret de chacune des variables prend la forme suivante :

$$x_{k+1} = x_k$$

Comme durant l'étape de prédiction, seules les variables liées à la pose du robot sont mises à jour, le calcul de la matrice de covariance nécessite uniquement le calcul du Jacobien par rapport à la pose, et seuls les termes de la matrice de covariance liés à la pose sont affectés. Il est donc possible de faire le calcul de prédiction de la matrice de covariance uniquement sur la sous-matrice contenant la pose afin de réduire les temps de calculs.

### Équations de correction

Nous allons maintenant présenter les équations de mesures ainsi que leurs Jacobiens correspondant aux différents capteurs utilisés pour l'étape de correction du filtre EKF.

Commandes de vitesse nulles (pas de mouvement) :

Quand le robot ne se déplace pas (vitesses de commandes nulles $\mathbf{v}_u = \omega_u = 0$), on peut forcer les vitesses linéaires et angulaires du robot à zéro, ce qui a tendance à améliorer les performances du filtre. Les mesures correspondantes sont donc :

$$\begin{cases} \mathbf{z}_{v0} = \mathbf{0}_3 = \mathbf{H}_{v0}\mathbf{x} \\ \mathbf{z}_{\omega0} = \mathbf{0}_3 = \mathbf{H}_{\omega0}\mathbf{x} \end{cases}$$

avec :

$$\begin{cases} \mathbf{H}_{v0} = (\mathbf{0}_{3\times3}\mathbf{0}_{4\times4}\mathbf{I}_{3\times3}\mathbf{0}_{3\times3}\mathbf{0}_{3\times3}\mathbf{0}_{3\times3}\mathbf{0}_{3\times3}\mathbf{0}_{3\times3}) \\ \mathbf{H}_{v0} = (\mathbf{0}_{3\times3}\mathbf{0}_{4\times4}\mathbf{0}_{3\times3}\mathbf{I}_{3\times3}\mathbf{0}_{3\times3}\mathbf{0}_{3\times3}\mathbf{0}_{3\times3}\mathbf{0}_{3\times3}) \end{cases}$$

Équation de mesure de l'accéléromètre :

L'équation de mesure de l'accéléromètre provient du modèle de l'accéléromètre, présenté au chapitre précédent (cf. section 2.3.2), et qui est rappelée ici :

$$\mathbf{z}_a = \mathbf{R}^T(\mathbf{a} + \mathbf{g}) + \mathbf{b}_a + \epsilon_a$$

En supposant que l'accélération du robot dans le repère d'inertie, $\mathbf{a}$, est nulle, on obtient :

$$\begin{aligned} \mathbf{z}_a &= \mathbf{R}^T\mathbf{g} + \mathbf{b}_a + \epsilon_a \\ &= \begin{pmatrix} r_{00} & r_{10} & r_{20} \\ r_{01} & r_{11} & r_{21} \\ r_{02} & r_{12} & r_{22} \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ g \end{pmatrix} + \begin{pmatrix} b_{a,x} \\ b_{a,y} \\ b_{a,z} \end{pmatrix} \\ &= \begin{pmatrix} r_{20} \\ r_{21} \\ r_{22} \end{pmatrix} g + \begin{pmatrix} b_{a,x} \\ b_{a,y} \\ b_{a,z} \end{pmatrix} \\ &= \begin{pmatrix} 2(q_x q_z - q_w q_y)g + b_{a,x} \\ 2(q_w q_x + q_y q_z)g + b_{a,y} \\ [2(q_w^2 + q_z^2) - 1]g + b_{a,z} \end{pmatrix} \end{aligned}$$

avec $g = \pm 9.81$ m.s$^{-2}$ suivant l'orientation de l'axe $z$ ($z$ vers le haut : $g = -9.81$ m.s$^{-2}$, $z$ vers le bas : $g = 9.81$ m.s$^{-2}$).

Les Jacobiens de l'équation de mesure, $\mathbf{z}_a$, sont donnés par :

$$\frac{\partial \mathbf{z}_a}{\partial \mathbf{q}} = \begin{pmatrix} -2q_y g & 2q_z g & -2q_w g & 2q_x g \\ 2q_x g & 2q_w g & 2q_z g & 2q_y g \\ 4q_w g & 0 & 0 & 4q_z g \end{pmatrix}$$

$$\frac{\partial \mathbf{z}_a}{\partial \mathbf{b}_a} = \mathbf{I}_{3\times3} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Comme la magnitude de l'accélération n'est pas une information exploitée ici, il est possible de normaliser la mesure d'accélération, $\mathbf{z}_a^* = \frac{\mathbf{z}_a}{||\mathbf{z}_a||_2}$, et de remplacer $g$ par $\pm 1$, ce qui évite de connaître précisément la valeur de l'accélération de pesanteur. On obtient alors (avec la convention de l'axe $z$ vers le haut $g = -1$) :

$$\begin{aligned} \mathbf{z}_a^* &= \begin{pmatrix} r_{00} & r_{10} & r_{20} \\ r_{01} & r_{11} & r_{21} \\ r_{02} & r_{12} & r_{22} \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix} + \begin{pmatrix} b_{a,x} \\ b_{a,y} \\ b_{a,z} \end{pmatrix} \\ &= \begin{pmatrix} -r_{20} \\ -r_{21} \\ -r_{22} \end{pmatrix} + \begin{pmatrix} b_{a,x} \\ b_{a,y} \\ b_{a,z} \end{pmatrix} \\ &= \begin{pmatrix} -2(q_x q_z - q_w q_y) + b_{a,x} \\ -2(q_w q_x + q_y q_z) + b_{a,y} \\ -2(q_w^2 + q_z^2) + 1 + b_{a,z} \end{pmatrix} \end{aligned}$$

Le Jacobien de l'équation de mesure $\mathbf{z}_a^*$ est donné par :

$$\frac{\partial \mathbf{z}_a^*}{\partial \mathbf{q}} = \begin{pmatrix} 2q_y & -2q_z & 2q_w & -2q_x \\ -2q_x & -2q_w & -2q_z & -2q_y \\ -4q_w & 0 & 0 & -4q_z \end{pmatrix}$$

Équation de mesure du gyroscope :

Le gyroscope fournit directement une information sur la vitesse angulaire du robot, $\omega$, dans le repère du robot. L'équation de mesure s'écrit donc d'après le modèle dérivé au chapitre précédent (cf. section 2.3.3) :

$$\mathbf{z}_g = \omega + \mathbf{b}_g + \epsilon_g$$

Les Jacobiens de l'équation de mesure $\mathbf{z}_g$ sont donc simplement :

$$\frac{\partial \mathbf{z}_g}{\partial \omega} = \mathbf{I}_{3\times 3}$$

et

$$\frac{\partial \mathbf{z}_g}{\partial \mathbf{b}_g} = \mathbf{I}_{3\times 3}$$

Équation de mesure du magnétomètre :

Le magnétomètre fournit une information sur l'orientation du robot par rapport au Nord magnétique (et Nord géographique si l'angle de déclinaison est connu). L'équation de mesure du magnétomètre, dérivée d'après le modèle présenté au chapitre précédent (cf. section 2.3.4), est donc donnée par :

$$\mathbf{z}_m = \mathbf{R}^T \mathbf{m} + \mathbf{b}_m + \epsilon_m$$

avec :

— $\mathbf{z}_m$ : la valeur du champ magnétique mesurée par le capteur,

— $\mathbf{R}$ : la matrice de rotation liée à l'orientation du robot,

— $\mathbf{m}$ : la valeur du champ magnétique terrestre dans le repère de la Terre,

— $\mathbf{b}_m$ : le biais du capteur (champ magnétique local) dans le repère du robot,

— $\epsilon_m$ : le bruit blanc Gaussien de moyenne nulle associé à la mesure.

Les Jacobiens de l'équation de mesure $\mathbf{z}_m$ sont donnés par :

$$\frac{\partial \mathbf{z}_m}{\partial \mathbf{q}} = \begin{pmatrix} 4m_x q_w + 2m_y q_z - 2m_z q_y & 4m_x q_x + 2m_y q_y + 2m_z q_z & 2m_y q_x - 2m_z q_w & 2m_y q_w + 2m_z q_x \\ 4m_y q_w - 2m_x q_z + 2m_z q_x & 2m_x q_y + 2m_z q_w & 4m_y q_y + 2m_x q_x + 2m_z q_z & 2m_z q_y - 2m_x q_w \\ 4m_z q_w + 2m_x q_y - 2m_y q_x & 2m_x q_z - 2m_y q_w & 2m_x q_w + 2m_y q_z & 4m_z q_z + 2m_x q_x + 2m_y q_y \end{pmatrix}$$

$$\frac{\partial \mathbf{z}_m}{\partial \mathbf{m}} = \mathbf{R}^T$$

$$\frac{\partial \mathbf{z}_m}{\partial \mathbf{b_m}} = \mathbf{I}_{3\times 3}$$

La valeur du champ magnétique terrestre $\mathbf{m}$ dans le repère de la Terre dépendant de la position géographique et du temps, il est possible d'obtenir ces valeurs sur le site de la National Oceanic and Atmospheric Administration (NOAA) [3]. Pour cela, il suffit de fournir les coordonnées GPS de la position, ainsi que la date ou la période de temps considérée. Cependant, comme indiqué précédemment, il est aussi possible d'estimer cette valeur dans l'état du filtre. De plus, dans le cas de l'utilisation du magnétomètre, il est important de prendre en compte la

3. https://www.ngdc.noaa.gov/geomag-web

transformation (rotation) entre le repère des balises et le Nord magnétique (géographique).

Équation de mesure du lacet :

Le magnétomètre fournissant la valeur du champ magnétique terrestre, cela revient aussi à obtenir une mesure de la direction du Nord magnétique (et donc du Nord géographique en connaissant l'angle de déclinaison). Ceci permet donc d'obtenir une information sur l'orientation du robot (angle dans le plan perpendiculaire à la surface de la Terre) par rapport à un repère géographique global (North-East-Down ou North-East-Up par exemple). Le magnétomètre permet donc d'obtenir l'angle de lacet $\psi$ du robot. Pour cela, il faut aligner le champ magnétique mesurée $\mathbf{z}_m$ avec l'horizontale :

$$\mathbf{z}_m^* = \mathbf{R}(\phi, \theta, 0)\mathbf{z}_m$$

Les angles de roulis, $\phi$, et tangage, $\theta$, sont obtenus à partir du quaternion estimé par le filtre. Il est alors possible d'obtenir l'angle de lacet, $\psi$, comme suit :

$$\psi = \text{atan}(\frac{z_{m,y}^*}{z_{m,x}^*}) - \psi_0$$

avec $\psi_0$ l'angle de lacet calculé à partir du magnétomètre à l'initialisation du filtre.

L'équation de mesure est alors donnée par :

$$z_\psi = \text{atan}(\frac{2(q_w q_z + q_x q_y)}{1 - 2(q_y^2 + q_z^2)})$$

Le Jacobien de l'équation de mesure est donné par :

$$\frac{\partial z_\psi}{\partial \mathbf{q}} = \begin{pmatrix} \frac{\partial z_\psi}{\partial q_w} & \frac{\partial z_\psi}{\partial q_x} & \frac{\partial z_\psi}{\partial q_y} & \frac{\partial z_\psi}{\partial q_z} \end{pmatrix}$$

avec :

$$\begin{aligned} \frac{\partial z_\psi}{\partial q_w} &= \frac{2q_z}{b(1 + (a/b)^2)} \\ \frac{\partial z_\psi}{\partial q_x} &= \frac{2q_y}{b(1 + (a/b)^2)} \\ \frac{\partial z_\psi}{\partial q_y} &= \frac{2q_x/b + 4q_y a/b^2}{1 + (a/b)^2} \\ \frac{\partial z_\psi}{\partial q_z} &= \frac{2q_w/b + 4q_z a/b^2}{1 + (a/b)^2} \end{aligned}$$

et $a = 2(q_w q_z + q_x q_y)$ et $b = 1 - 2(q_y^2 + q_z^2)$. La même remarque que pour la fusion du magnétomètre est aussi valable ici. C'est-à-dire, qu'il est nécessaire de tenir compte de l'éventuelle rotation entre le repère des balises et celui du Nord magnétique (ou géographique).

Équation des mesures de distance robot/balises :

Grâce aux $N$ balises positionnées sur le terrain dont la position 3D est connue : $\mathbf{x}_i = (x_i, y_i, z_i)^T$, $i \in [1, N]$, le robot peut mesurer sa distance par rapport à chaque balise. L'équation de mesure de la distance entre le robot et la balise $i$ est donnée par (cf. section 2.3.1) :

$$z_{r,i} = d_i + b_i + \epsilon_{r,i}$$

avec :

— $d_i = \sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2}$ : la vraie distance,

— $b_i$ : le biais,

— $\epsilon_{r,i}$ : le bruit blanc Gaussien de moyenne nulle et dont la variance dépend de la distance : $\sigma_{r,i}^2 = (1 + \eta_i d_i)^2 \sigma_i^2$. $\eta_i$ étant le paramètre de dépendance à la distance et $\sigma_i^2$ la variance initiale associée à la balise $i$.

Le Jacobien de l'équation de mesure par rapport à la position du robot $\mathbf{p}$ est donné par :

$$\frac{\partial z_{r,i}}{\partial \mathbf{p}} = \begin{pmatrix} -\frac{dx_i}{d_i} & -\frac{dy_i}{d_i} & -\frac{dz_i}{d_i} \end{pmatrix}$$

avec $dx_i = (x_i - x)$, $dy_i = (y_i - y)$, $dz_i = (z_i - z)$. Dans le cas où l'on estime aussi le biais de la balise dans l'état du filtre, le Jacobien est donné par :

$$\frac{\partial z_{r,i}}{\partial b_i} = 1$$

D'où le Jacobien par rapport à l'état du filtre :

$$\mathbf{H} = \frac{\partial z_{r,i}}{\partial \mathbf{x}} = \begin{pmatrix} -\frac{dx_i}{d_i} & -\frac{dy_i}{d_i} & -\frac{dz_i}{d_i} & 0 & \dots & 1 & \dots \end{pmatrix}$$

Équation de mesure de la position donnée par le GPS :

Comme pour le magnétomètre, le GPS donne la position du robot dans un repère global géographique (NED ou ENU par exemple), alors que les balises donnent la position par rapport au repère des balises. Il est donc ici aussi nécessaire, soit de connaître la position des balises dans le repère global, soit de retrouver la transformation (translation) entre le repère géographique du GPS et celui des balises.

### Initialisation du filtre

Comme dans le cas 2D, il est nécessaire d'initialiser les variables du filtre EKF avec des valeurs proches de la vérité terrain pour que le filtre converge. Pour cela, au démarrage du filtre, on laisse le robot statique pendant un certain temps en enregistrant les données des différents capteurs. À partir de l'accéléromètre qui mesure le vecteur gravité, on peut estimer les angles de roulis et de tangage. À partir du magnétomètre, on peut estimer l'angle de lacet (par rapport au Nord magnétique et/ou géographique). À partir des mesures de distances, si on a des mesures par rapport à au moins 4 balises, on peut utiliser un l'algorithme de trilatération pour estimer la position du robot par rapport au repère des balises. Si l'on dispose d'un GPS, on peut obtenir la position du robot par rapport à un repère géographique. Le robot étant statique, les valeurs des vitesses linéaire et angulaire sont bien évidements initialisées à zéro. Il ne reste donc plus que les biais de la centrale inertielle (accéléromètre, gyroscope, magnétomètre) à initialiser. Sans informations à priori d'une procédure de calibration, ces valeurs peuvent être initialisées à zéro.

Maintenant que nous avons vu les différentes équations permettant d'implémenter le filtre EKF en 3D, ainsi que la procédure d'initialisation, nous allons voir si l'algorithme proposé permet effectivement d'estimer correctement la pose d'un robot dans le cas d'un monde 3D.

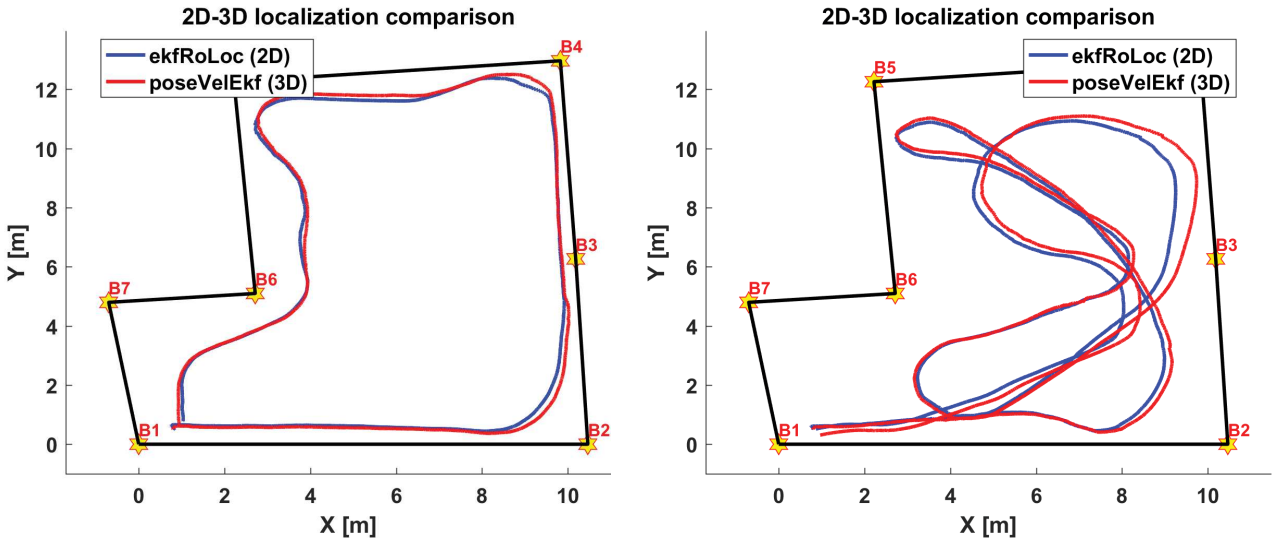
### Expériences sur le filtre EKF 3D

Le bon fonctionnement du filtre EKF 3D que nous venons de présenter est validé en utilisant les jeux de données réalisés, avec le prototype du robot, sur un terrain légèrement pentu

avec 6 balises positionnées sur le pourtour. Lors de ces expériences, le robot est piloté manuellement à l'intérieur du terrain définit par les balises. Les jeux de données ne comprenant pas de vérité terrain, l'estimation du filtre 3D est comparée à l'estimation du filtre EKF 2D, présenté précédemment. Par manque de temps, tant au niveau de la réalisation des expériences avec le prototype que de l'implémentation du filtre 3D, seuls des résultats préliminaires sur la trajectoire estimée dans le plan 2D sont présentés ici.

La Figure 3.27 montre la comparaison des trajectoires estimées par les filtre EKF 2D (ligne bleue) et 3D (ligne rouge) pour deux des jeux de données réalisés avec le prototype. On s'aperçoit que les deux trajectoires sont très proches l'une de l'autre, surtout dans le cas du premier jeu de données de la Figure 3.27a. Pour le second, à la Figure 3.27b, où le robot tourne beaucoup plus, on constate des écarts plus importants dans les courbes. Lors des expériences, les positions de départ et d'arrivée du robot étaient toujours approximativement les mêmes. Au regard des figures, on constate pour les deux jeux de données et pour les deux filtres, que c'est bien le cas pour les positions estimées. On peut donc penser que le filtre parvient bien à limiter les dérives de l'odométrie grâce à la fusion des mesures de distance, et de la centrale inertielle pour le filtre 3D.

Ainsi, de ces premières observations, il semblerait que le filtre EKF 3D permet bien d'estimer la trajectoire dans le plan, malgré la prise en compte de la 3$^{\text{ème}}$ dimension. Cependant, il reste encore à confirmer les résultats sur l'axe $z$, et sur les vitesses linéaire et angulaire du robot. Pour cela, il est nécessaire de poursuivre les expériences, et surtout d'obtenir une vérité terrain avec qui comparer les résultats.




(a) Jeu de données 1

(b) Jeu de données 2

FIGURE 3.27 – Comparaison des trajectoires estimées par les filtres EKF 2D et 3D pour deux jeux de données réalisés avec le prototype

## 3.7 Conclusion

Ce chapitre dédié au problème de la localisation d'un robot mobile dans un environnement connu au préalable, a permis de présenter toute une série de solutions basées sur l'utilisation de mesures de distance par rapport à des balises fixes. Dans un premier temps, différentes techniques utilisant uniquement ces mesures de distance ont été présentées. À partir de l'algorithme classique de trilatération, différentes versions plus ou moins précises et robustes ont été présentées. D'après les simulations réalisées, il ressort que l'algorithme de trilatération avec

RANSAC suivi d'un algorithme d'optimisation non-linéaire de type Levenberg-Marquardt (LM) présente les meilleures performances en termes de précision et de robuste aux mesures aberrantes. L'utilisation d'un algorithme RANSAC permet d'obtenir une meilleure robustesse aux mesures erronées, mais de par sa nature aléatoire, il ne garantit pas d'obtenir la solution globale. Dans cette optique, une nouvelle formulation basée sur l'utilisation de sommes de polynômes aux carrés et d'inégalités linéaires matricielles est présentée. La méthode proposée permet de retourner une boîte (intervalle 2D) qui, dans certains cas, contient la vraie solution à estimer, et ce même en présence d'un nombre important de mesures erronées. L'algorithme proposé est beaucoup plus lent et moins précis que la méthode basée sur l'algorithme de trilatération avec RANSAC suivit d'un algorithme d'optimisation non-linéaire de type Levenberg-Marquardt. Cependant, il présente l'avantage de garantir que la vraie position se trouve dans l'intervalle retourné par l'algorithme.

Dans un deuxième temps, différents algorithmes de filtrage, basés pour la plupart sur le filtre de Kalman, et permettant de fusionner les informations provenant de multiples capteurs sont présentés. Juste avant cela, une analyse d'observabilité du problème de la localisation en 2D d'un robot mobile de type unicycle a permis de déterminer qu'avec une seule balise, le problème n'est pas observable, et donc qu'il n'est pas possible de reconstruire de façon unique la trajectoire du robot. Par contre, le système devient observable dès qu'une deuxième balise est utilisée. Ainsi, en s'assurant de disposer d'aux moins deux balises sur le terrain, il est possible en utilisant un filtre de Kalman 2D d'estimer la pose du robot. Après une rapide description générale des équations du filtre de Kalman, deux implémentations en 2D et 3D sont présentés. Des simulations réalisées avec l'implémentation 2D du filtre, il ressort que le filtre implémenté fonctionne correctement et permet d'obtenir une très bonne précision de la position estimée dans la plupart des scénarios testés. De plus, grâce à l'ajout d'un test de Mahalanobis, il est possible de rendre le filtre beaucoup plus robuste aux mesures erronées très préjudiciable au bon fonctionnement du filtre si elles ne sont pas détectées. En comparant le filtre EKF avec d'autres implémentations comme l'algorithme GTSAM, il ressort que le filtre EKF présente généralement une moins bonne précision. Cependant le filtre EKF présente l'avantage d'une implémentation aisée et d'un temps d'exécution lui permettant d'être intégré sans problème dans un système embarqué. Enfin, concernant le filtre 3D implémenté, en regard des observations qualitatives faites à partir des jeux de données réels du prototype, il semble que la solution développée permet d'estimer correctement la pose 3D du robot. De plus, la trajectoire estimée en 3D ramenée dans le plan est très proche de celle estimée par le filtre 2D et par l'algorithme de trilatération. Ces résultats restent cependant à confirmer avec l'utilisation d'une vérité terrain d'une précision suffisante, et ainsi obtenir des données quantitatives sur les performances réelles du filtre. Ces données permettront alors de valider si la précision de localisation souhaitée pour l'application est bien atteinte ou non.

Comme cela a déjà été énoncé dans l'introduction, il n'est pas toujours possible d'obtenir une représentation, au préalable, de la carte de l'environnement dans lequel le robot évoluera. Ainsi, dans notre cas du déploiement d'un système permettant à un robot d'opérer de façon autonome dans une zone donnée, l'utilisateur doit, après avoir disposé les balises sur le pourtour du terrain, apprendre au système les limites de la zone. La position des balises n'étant pas connue par le système, l'utilisateur devra aussi, lors de cette étape d'apprentissage, apprendre au système où se situe les balises. On se retrouve ainsi dans la situation où l'on cherche à reconstruire la trajectoire de l'utilisateur (du robot), tout en devant estimer la position des balises préalablement disposées sur le terrain. Cette situation correspond au problème du SLAM, qui est présenté dans le chapitre suivant.

CHAPITRE 4

# SLAM AVEC MESURES DE DISTANCE

## 4.1 Introduction

Dans le chapitre précédent, différentes méthodes de localisation d'un robot mobile à partir de mesures de distance par rapport à des balises fixes ont été présentées. Cependant, dans le cas de l'utilisation de balises, l'inconvénient majeur est la nécessité d'avoir une connaissance précise de leurs positions. Ainsi, pour différentes raisons, la procédure permettant de déterminer la position des balises peut être plus ou moins fastidieuse, et cela d'autant plus que le nombre de balises est élevé. De plus, en fonction des outils de mesure utilisés pour calculer la position des balises, la précision obtenue sera sujette à des erreurs plus ou moins grandes qui auront un impact sur les performances de localisation du robot. Il semble donc intéressant et pratique de disposer d'une méthode alternative permettant à la fois de localiser le robot et les balises. Ce problème d'auto-localisation d'un robot mobile est communément appelé localisation et cartographie simultanées (SLAM). Le terme localisation se réfère comme précédemment à la localisation du robot dans son environnement, alors que le terme cartographie, lui, implique la construction de la carte de l'environnement du robot. Par carte, on entend généralement une représentation 2D ou 3D, continue ou discrète, de l'espace dans lequel évolue le robot. Mais dans le cas présent, il s'agira plus simplement de la position (2D ou 3D) des balises.

Le SLAM est un sujet de recherche bien connu en robotique mobile. Depuis ses débuts dans les années 1980, beaucoup d'avancées ont été faites, permettant ainsi l'arrivée à maturation de certaines techniques et l'incorporation de celles-ci dans des applications industrielles comme les robots aspirateurs 360 eye de Dyson [1] et Roomba 980 d'iRobot [2]. Cependant, malgré les nombreux progrès qui ont permis de mieux comprendre et consolider les aspects théoriques, il reste encore des perspectives de recherche et d'améliorations comme la robustesse, la fiabilité à long terme et l'adaptation à grande échelle des algorithmes. Un état des lieux récent et complet sur le passé, le présent et le futur du SLAM est présenté par Cadena et al. [18]. Ces dernières années, avec l'émergence des caméras, les travaux sur le SLAM ont eu tendance à se rapprocher et à s'inspirer des travaux réalisés en vision sur la reconstruction de scène et de position de la caméra par le mouvement (SfM). En effet, le problème de départ est le même : localiser le robot ou la caméra tout en reconstruisant l'environnement, seuls les capteurs utilisés changent.

Dans ces travaux, comme cela a déjà été expliqué dans l'introduction, l'utilisation de caméras se prêtait mal aux exigences du cahier des charges de l'application. Pour cela, l'utilisation

1. http://spectrum.ieee.org/automaton/robotics/home-robots/dyson-the-360-eye-robot-vacuum
2. http://spectrum.ieee.org/automaton/robotics/home-robots/review-irobot-roomba-980

de balises RF UWB fournissant des mesures de distance a été retenue. Le SLAM par vision et par mesures de distance partagent tous les deux une difficulté principale : l'observabilité partielle. En effet, une caméra peut être vue comme un capteur d'angle, car elle donne la direction d'un point 3D de l'environnement à partir de sa projection extraite dans l'image. De manière similaire, une balise RF est un capteur de distance, car elle fournit une mesure de distance par rapport à un robot ou à une autre balise. Dans le cas classique du SLAM, on considère généralement que l'on possède un capteur permettant d'obtenir les mesures d'angle et de distance par rapport à un point de repère fixe. Cette situation est illustrée à la Figure 4.1, sur le schéma de gauche, où $a$ et $d$ correspondent respectivement à la mesure d'angle et de distance entre le robot et le point de repère $B_i$. À partir de la mesure combinée d'angle et de distance, il est alors possible de calculer la position du point de repère par rapport au robot. Ceci n'est plus le cas quand on possède uniquement la mesure d'angle ou de distance. Dans le cas d'un système fournissant des mesures d'angle uniquement, il est nécessaire de réaliser au minimum deux mesures par rapport au point de repère pour pouvoir déterminer sa position relative par rapport au robot, comme cela est illustré sur le schéma du milieu de la Figure 4.1. De façon similaire, quand on dispose uniquement de mesures de distance, il faut au minimum trois mesures, comme le montre le schéma de droite de la Figure 4.1. On constate ainsi qu'avec les mesures d'angle ou de distance uniquement, l'initialisation de la position des points de repère n'est pas possible avec une seule mesure, mais nécessite l'intégration d'un certain nombre d'entre elles. Ceci a pour conséquence directe de retarder la cartographie des points de repère. L'observabilité partielle entraîne donc un niveau de difficulté supplémentaire dans la résolution du SLAM, et implique de développer de nouvelles méthodes permettant de gérer le problème d'initialisation. Le SLAM avec mesures de distance uniquement (RO-SLAM) ayant eu beaucoup moins d'attention que le SLAM avec vision (V-SLAM), ce dernier a déjà atteint un bon niveau de maturité aux niveaux des techniques proposées. Ceci est moins le cas pour le RO-SLAM, où le champ des perspectives de recherche reste encore très vaste.

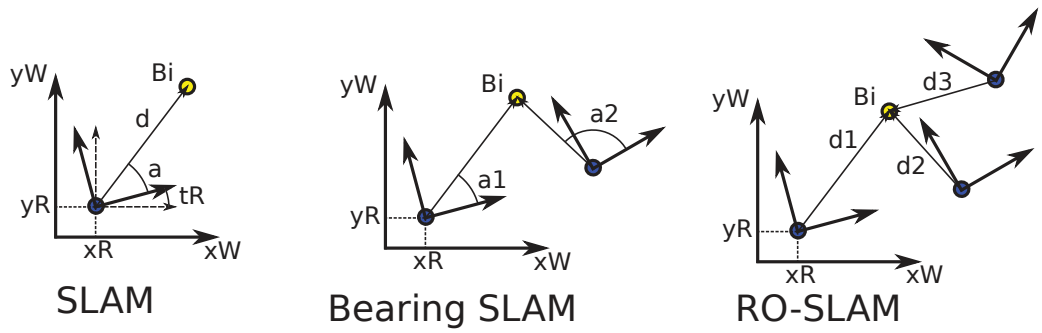



Figure 4.1 – Principales catégories de SLAM en fonction des mesures retournées par le capteur extéroceptif

Ainsi, ce chapitre qui est consacré au RO-SLAM, débutera par un état de l'art détaillé des différents algorithmes utilisés pour résoudre le problème du SLAM, mais aussi des techniques d'initialisation des balises spécifiques à l'utilisation des mesures de distance uniquement. Ensuite, une nouvelle méthode d'initialisation de la position des balises dans le vecteur d'état d'un filtre de Kalman étendu (EKF) est présentée. L'approche proposée est alors testée et comparée avec différentes techniques d'initialisation basées également sur l'utilisation d'un filtre EKF, mais aussi avec d'autres types d'algorithmes. Enfin, une conclusion du chapitre résumera les principales contributions et listera les différentes perspectives d'améliorations.

## 4.2 État de l'art

Le problème du SLAM dans le cas général ayant déjà fait l'objet de présentations dans les parties précédentes, nous allons nous concentrer ici sur le cas plus spécifique du SLAM avec des mesures de distance uniquement. Bien sûr, la plupart des techniques et algorithmes utilisés pour résoudre le SLAM classique peuvent être réutilisés pour le RO-SLAM. Le principe de base ne changeant pas, seules quelques adaptations sont nécessaires dues à l'observabilité partielle inhérente aux mesures de distance.

Comparé au SLAM classique, où l'on a accès à la mesure d'angle et de distance par rapport à des repères naturels ou artificiels de l'environnement, le RO-SLAM lui n'utilise que l'information de distance. Cette particularité du RO-SLAM entraîne un problème d'observabilité partielle qui rend l'initialisation des repères plus difficile. Ce problème d'observabilité partielle vient du fait qu'il n'est pas possible d'initialiser la position d'un repère (ou d'une balise) avec une seule mesure de distance. Il est nécessaire de collecter au minimum trois mesures de distance prises à trois positions différentes pour pouvoir reconstruire la position de la balise. Une bonne partie des contributions dans le domaine du RO-SLAM se sont donc attachées à trouver des solutions à ce problème. On distingue ainsi dans la littérature, deux principales approches : les méthodes d'initialisation avec délai et celles sans délai. Concrètement, une méthode d'initialisation avec délai, va attendre d'avoir suffisamment d'informations (de mesures de distance) pour initialiser la balise. Au contraire, les méthodes sans délai vont initialiser la balise dès la première mesure tout en tenant compte du fait que les paramètres modélisant la position de la balise ne sont pas tous connus.

### 4.2.1 Méthodes d'initialisation avec délai

L'un des premiers articles traitant de l'observabilité partielle pour le problème du SLAM a été écrit par Leonard et al. [85]. Dans ces travaux, un filtre de Kalman étendu est utilisé afin de résoudre le problème pour un robot équipé de capteurs à ultrasons. Différentes techniques d'initialisation de caractéristiques de l'environnement (point, ligne...) extraites à partir des mesures de distance des sonars sont proposées. Le principe consiste à incorporer les positions du robot associées aux mesures de distance dans le vecteur d'état du filtre, jusqu'à avoir assez de mesures pour calculer et initialiser les paramètres de la caractéristique dans le filtre. L'incertitude des positions du robot associées aux mesures des sonars, ainsi que l'incertitude des mesures elles-mêmes servent alors à initialiser l'incertitude des paramètres de la caractéristique. Les anciennes positions du robot n'étant plus nécessaires sont ensuite retirées du filtre. Pour initialiser un point à partir des mesures d'un sonar, il suffit de collecter deux mesures de distance seulement au lieu de trois avec une balise RF. En effet, le sonar possédant un cône d'émission/réception, qui même si l'ouverture est large, permet de lever l'ambiguïté sur l'une des deux solutions de l'intersection de deux cercles à partir de l'information des angles d'ouverture.

Dans le cas de balises RF qui est le nôtre, l'onde se propage de façon omnidirectionnelle et plus aucune information d'angle n'est disponible. Dans ce cas, la méthode la plus classique pour obtenir une estimation de la position de la balise consiste à attendre au minimum trois mesures de distance et d'utiliser un algorithme de trilatération, [138], [134]. Ce processus est illustré à la Figure 4.2d où $(x_i, y_i)$, $(x_j, y_j)$, $(x_k, y_k)$ représentent les positions du robot et $(x, y)$ est la solution de l'algorithme de trilatération qui servira comme initialisation de la position de la balise. Ainsi, dans Menegatti et al. [97] un algorithme de trilatération est utilisé afin d'initialiser la position des balises dans un filtre de Kalman étendu. Cette méthode, bien que très simple, est assez sensible au bruit. Une solution serait donc d'attendre et d'utiliser plus que trois mesures de distance afin d'obtenir une meilleure initialisation. L'accumulation de mesures devrait normalement améliorer l'estimation, cependant ce n'est pas toujours le cas. En

effet, si l'on attend trop longtemps, l'erreur d'odométrie du robot peut devenir trop grande, ce qui deviendrait préjudiciable pour l'initialisation de la balise. Il existe donc un compromis pas toujours évident à trouver entre initialiser rapidement ou attendre plus de mesures pour obtenir éventuellement de meilleurs résultats.

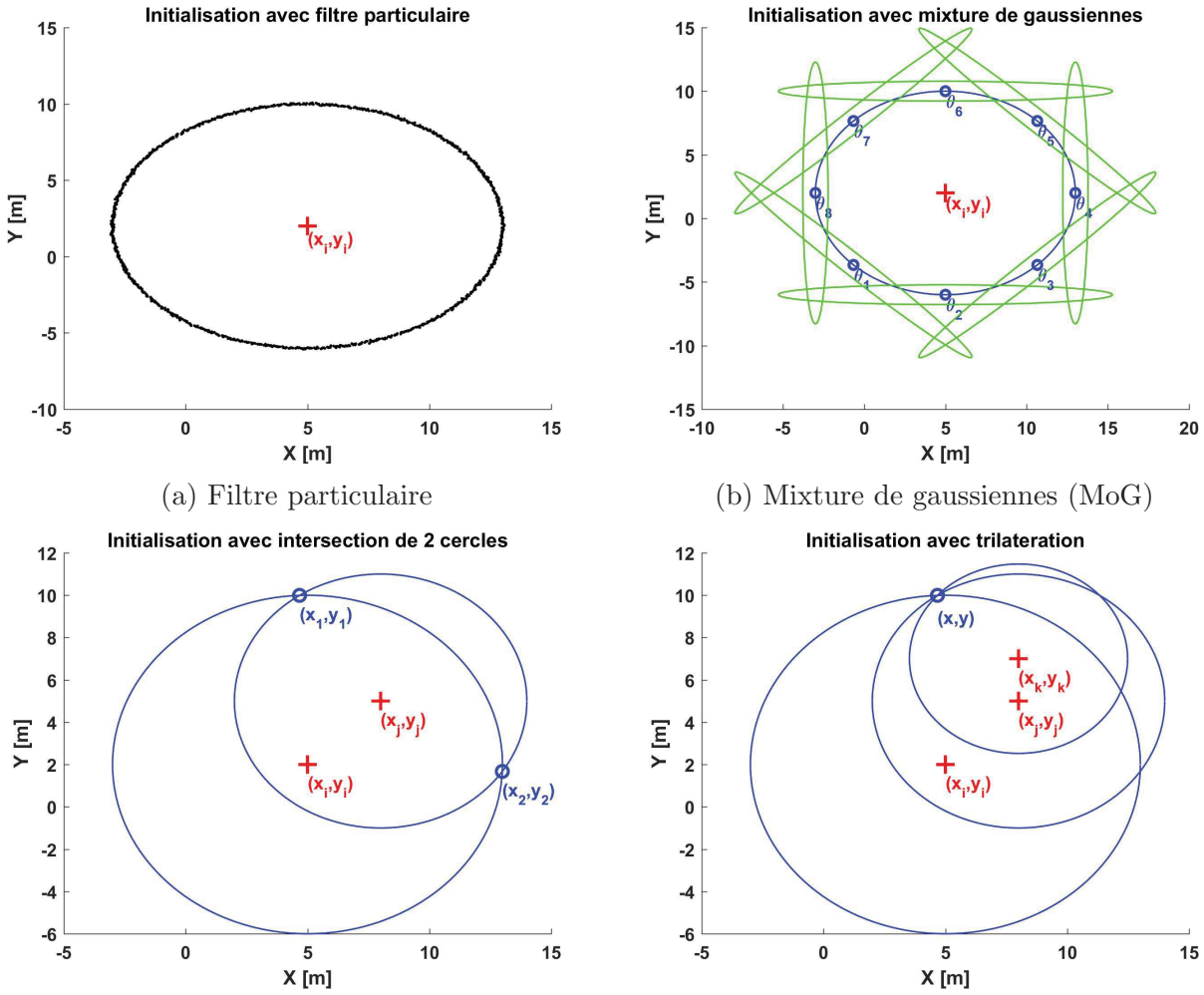
Olson et al. [104] dans le cas d'un robot sous-marin, ainsi que Djugash et al. plus tard dans [34] et [31] pour un robot mobile, utilisent quant à eux une grille de probabilités 2D afin d'initialiser la position des balises dans un filtre EKF. L'environnement du robot est donc discrétisé en une grille 2D où chaque cellule contient le support (nombre de votes) correspondant à la possibilité que la balise se trouve à cette position. Le principe consiste à utiliser deux mesures de distance pour calculer les deux positions possibles de la balise. Puis, pour chaque position trouvée, la valeur correspondante dans la grille est incrémentée d'une unité. Après un certain temps ou un certain nombre de votes, la cellule ayant le plus grand support (la plus grande valeur) est considérée comme étant la position initiale de la balise et va servir pour initialiser l'état correspond dans le filtre EKF.

Une autre solution suivie par Blanco et al. [11] et par Yang [135] consiste à utiliser un filtre particulaire pour représenter l'incertitude initiale de la position d'une balise. Dans les deux cas, l'algorithme principal est un filtre particulaire Rao-Blackwellized (RBPF) qui permet grâce à l'estimation de toute la trajectoire du robot, d'estimer de manière indépendante la position des balises pour chaque particule. Quand on reçoit la première mesure de distance par rapport à une balise, la seule chose que l'on sait est que la position de la balise se situe sur un cercle centré sur la position du robot et de rayon égal à la mesure de distance. Cette information peut alors être exploitée pour représenter les positions possibles de la balise à l'aide d'une densité de probabilité. Au début, cette densité de probabilité sera uniformément distribuée autour du cercle. Le principe est illustré à la Figure 4.2a où les particules (points noirs) sont distribuées uniformément le long du cercle centrée sur la position $(x_i, y_i)$ du robot à l'instant de la mesure et de rayon égal à la mesure de distance. Puis, en accumulant d'autres mesures de distance par rapport à la balise correspondante, cette densité de probabilité va progressivement converger vers un même point représentant la position de la balise. Afin de représenter cette densité de probabilité, il est possible d'utiliser un filtre particulaire dont les particules vont approximer la distribution. Chaque particule possède une pondération associée qui est mise à jour à chaque nouvelle mesure de distance. Les particules dont la pondération devient négligeable sont alors remplacées par d'autres plus probables grâce à un processus de rééchantillonnage. Quand les particules ont convergé autour d'une position, il est possible d'approximer la distribution par une loi normale, centrée sur la moyenne des particules, et de variance correspondant à la dispersion des particules. Cette nouvelle distribution gaussienne peut alors servir à initialiser la position de la balise dans le filtre de Kalman. L'approche qui vient d'être décrite est aussi utilisée dans [96] avec un filtre de Kalman étendu et dans [127] avec un filtre d'information. Même si les techniques d'initialisation avec délai présentées ci-dessous permettent généralement d'obtenir des résultats d'estimation corrects, l'inconvénient majeur de ces techniques est la dissociation complète de l'étape d'initialisation de la position des balises avec le filtre principal. Ceci a pour principale conséquence que les informations fournies par les mesures de distance ne sont que partiellement exploitées. Pour cela, des méthodes d'initialisation sans délai permettent de pallier à cet inconvénient.

### 4.2.2 Méthodes d'initialisation sans délai

Une méthode d'initialisation sans délai de la position des balises permet, dès la première mesure de distance, d'exploiter la totalité des informations fournies par les mesures de distance dans l'algorithme d'estimation.

Djugash et al. [33] ont été parmi les premiers à proposer l'utilisation d'une nouvelle para-




(a) Filtre particulaire

(b) Mixture de gaussiennes (MoG)

(c) Intersection de deux cercles

(d) Trilatération

FIGURE 4.2 – Représentation de différentes méthodes d'initialisation de la position d'une balise. On distingue des méthodes sans délai (a), (b) et des méthodes avec délai (c), (d).

métrisation pour représenter la position des balises dans un filtre EKF. Leur approche, appelée ROP-EKF pour Relative Over Parametrization EKF, représente la position du robot et des balises en coordonnées polaires et non plus en coordonnées cartésiennes. Cette paramétrisation permet une meilleure modélisation des formes telles que les cercles et les anneaux que l'on retrouve typiquement dans le RO-SLAM. Ceci permet ainsi de mieux gérer les non-linéarités du problème et le caractère multimodal de l'observabilité partielle. La position d'une balise $i$ est sur-paramétrée par :

— La position du robot dans le repère global lors de la première mesure de distance par rapport à la balise : $(x_i, y_i)$. Cette position définit ainsi l'origine du repère polaire,

— Les coordonnées polaires $(\rho_i, \theta_i)$ : dont la distance $\rho_i$ correspond à la mesure de distance.

Cette représentation permet d'insérer les paramètres liés à la balise dès la première mesure de distance, en utilisant la position courante du robot, la mesure de distance et en initialisant l'angle des coordonnées polaires avec une très grande variance, permettant d'obtenir une probabilité de distribution quasi uniforme entre $]-\pi, \pi]$. Quand une nouvelle mesure de distance permet de calculer l'intersection de deux cercles et d'obtenir une nouvelle distribution bimodale avec les deux solutions, une représentation multi-hypothèses est adoptée en dupliquant le filtre afin de suivre les deux modes créés. Il est alors possible de suivre l'évolution des deux modes jusqu'à l'obtention d'une seule et unique solution correspondant à la position de la balise.

Par la suite, Caballero et al. [17] en 2D, puis Fabresse et al. [40] en 3D, ont repris le principe de la paramétrisation en polaire/sphérique qui, combinée à une mixture de gaussiennes permet, au sein d'un filtre EKF, d'obtenir un algorithme de RO-SLAM avec initialisation sans délai très efficace. Comparé à ce qui est fait dans [33] ou l'on initialise l'angle de la représentation polaire avec une grande variance, l'approche proposée ici initialise plusieurs hypothèses d'angle avec une mixture de gaussiennes. En effet, avec la représentation en polaire, le caractère multimodal de la position de la balise se retrouve uniquement suivant l'angle polaire, car la position du repère polaire et la distance sont connues. Seul l'angle, qui donne la position précise sur le cercle, est inconnu. En utilisant un produit pondéré de $k$ gaussiennes, il est donc possible de représenter cette incertitude angulaire autour du cercle en initialisant $k$ hypothèses d'angle $\theta_{ij}$, $j \in [1, k]$. De plus, l'emploi d'une mixture de gaussiennes fait que chaque hypothèse d'angle possède sa propre moyenne et variance, ce qui permet d'inclure directement les paramètres de la position de la balise dans le vecteur d'état du filtre EKF et d'appliquer normalement les différentes étapes du filtre de Kalman. La Figure 4.2b permet de visualiser cette étape d'initialisation. La croix rouge en $(x_i, y_i)$ représente la position du robot au moment de la mesure, et les petits cercles bleus correspondent aux positions des différentes hypothèses angulaires $\theta_{ij}$ le long du cercle de rayon égal à la distance mesurée. Les ellipses vertes qui leurs sont associées correspondent à l'incertitude en position de la balise au moment de l'initialisation. Chaque hypothèse possède une pondération qui évolue en fonction de la probabilité que les mesures de distance proviennent bien de cette hypothèse. Quand la pondération d'une hypothèse devient plus faible qu'un certain seuil, l'hypothèse est supprimée de l'état du filtre. Si tout se passe bien, la dernière hypothèse restante doit correspondre à la position de la balise. L'inconvénient principal de la méthode est le nombre de paramètres à estimer. Normalement la position d'une balise est représentée par deux paramètres $(x, y)$. Pour cette approche, au moment de l'insertion des paramètres dans le filtre on a un nombre total de $2 + 1 + k$ paramètres (centre du repère polaire, distance et $k$ hypothèses d'angle), qui tombe à 4 après la suppression de toutes les mauvaises hypothèses. Il y a donc un compromis à faire pour le choix du nombre d'hypothèses $k$ à initialiser, mais d'après les expériences un nombre entre 8 et 16 donne de bons résultats. On note toutefois que généralement le nombre d'hypothèse d'une balise va très rapidement décroître jusqu'à deux, puis mettre un peu plus de temps pour converger vers une solution unique.

Cette méthode d'initialisation avec mixture de gaussiennes peut aussi s'utiliser au sein d'un filtre particulaire "Rao-Blackwellisé". Ainsi, Blanco et al. [10] ont amélioré leur précédents travaux [11] en utilisant une paramétrisation en sphérique avec une mixture de gaussiennes afin d'initialiser la position des balises dans un filtre EKF au sein d'un filtre RBPF. Cette approche permet pour chaque particule de maintenir un filtre EKF par balises, et ainsi de suivre l'évolution des différentes hypothèses angulaires de façon indépendante pour chaque balise. Le fait de disposer d'un filtre EKF par balise permet d'accélérer les calculs, mais la méthode reste cependant encore très coûteuse en temps de calcul puisqu'il faut réaliser le processus pour chaque particule. En effet, dans le cas d'un filtre particulaire avec $n_p$ particules et $n_b$ balises, le nombre de filtre de Kalman à mettre à jour à chaque mesure de distance est de $n_p \times n_b$.

Jusqu'à présent, la plupart des méthodes présentées étaient basées sur deux algorithmes très populaires : le filtre de Kalman et le filtre particulaire. Il existe cependant beaucoup d'autres techniques pour résoudre le RO-SLAM.

### 4.2.3 Autres algorithmes

Parmi les alternatives aux classiques EKF et RBPF, on retrouve tout d'abord des dérivées du filtre EKF. Ainsi dans Torres-González et al. [127], [131], un filtre d'information étendu (SEIF) est utilisé afin d'exploiter les mesures inter-balises et les capacités de calcul des balises. L'approche qui est proposée permet de distribuer les calculs de l'étape de correction du filtre aux

processeurs des balises. Les balises RF pouvant être alimentées par des batteries, une stratégie de gestion est proposée dans [131] afin de limiter leurs consommations en ressources tout en maintenant une bonne précision de l'estimation.

Lourenço et al. [88] résolvent le problème d'initialisation des balises en concevant un filtre globalement exponentiellement stable (GES). Pour cela, ils combinent l'approche robot-centrique introduite dans Castellanos et al. [21] et la technique d'augmentation d'état proposée par Batista [6] pour réaliser un filtre avec une formulation linéaire à temps invariant (LTI). En réalisant une analyse d'observabilité de leur formulation, ils obtiennent des conditions sur la trajectoire du robot afin de rendre le système observable. En pratique le filtre est implémentée avec un filtre de Kalman qui possède alors une dynamique de l'erreur GES.

Une alternative aux techniques de filtrage qui est de plus en plus utilisée grâce à l'amélioration des capacités de calculs est l'utilisation d'algorithmes d'optimisation. Kehagias et al. [73] sont parmi les premiers à proposer une solution au RO-SLAM sous forme d'un problème d'optimisation. Leur méthode permet de retrouver la trajectoire du robot (position mais pas orientation) et la position des balises en utilisant les mesures d'odométrie du robot et les mesures de distance robot-balise et balise-balise. Le problème d'optimisation formulé étant non-linéaire, un algorithme de Gauss-Newton est utilisé pour réaliser la minimisation. Plus tard, d'autres algorithmes d'optimisation populaires comme le GraphSLAM [62], ISAM [59] puis GTSAM [27] ont été adaptés au cas particulier du RO-SLAM.

Boots et al. [12] proposent une approche différente pour résoudre le problème du RO-SLAM faisant appel à l'apprentissage spectral. Cette approche pouvant s'appliquer en temps réel ou après collecte des données, consiste à former une matrice à partir des mesures de distance robot-balise au carrée, d'une façon similaire à la matrice des distances euclidiennes (EDM) [24]. Chaque ligne de la matrice correspond à une balise, alors que les colonnes sont associées aux instants auxquels les mesures sont prises. Cette méthode requiert qu'en chaque instant les mesures de distance par rapport à toutes les balises soient connues. En pratique cela n'est pas toujours le cas, mais il est possible d'utiliser une technique d'interpolation afin d'estimer les valeurs manquantes. À partir de la matrice ainsi construire, on réalise alors une décomposition en valeurs singulières (SVD) pour retrouver les positions du robot et des balises, qui sont obtenues à une transformation orthogonale (translation, rotation et réflexion) près. Pour retrouver la solution complète, deux solutions sont proposées. La technique qui vient d'être brièvement décrite utilisait uniquement les mesures de distance. Cependant, une autre version est aussi présentée, qui, à partir des informations de l'odométrie du robot, permet aussi de retrouver l'orientation du robot en plus de sa position.

Le problème du RO-SLAM peut aussi être traité par des approches ensemblistes. Bien que beaucoup moins populaires, ces méthodes présentent l'avantage de fournir la garantie de toujours contenir la solution estimée, au prix parfois d'une moins bonne précision et d'un temps de calcul plus long que d'autres approches plus traditionnelles telle que le filtre EKF. En général, pour appliquer ces approches, on fait l'hypothèse que les erreurs sur le modèle et les mesures sont inconnues mais que leurs bornes le sont. Ainsi, Di Marco et al. [93], [30], [29], proposent une méthode reprenant les principales étapes d'un filtre de Kalman, mais en les adaptant au cas où les variables sont représentées par des ensembles. La méthode est présentée dans le cas du SLAM avec des mesures de distance et d'angle, mais peut parfaitement s'appliquer au cas du RO-SLAM. Les variables à estimer étant représentées par des ensembles, les équations de prédiction et de mesure de distance permettent de formuler des contraintes qui se traduisent alors par des opérations sur les ensembles, comme des intersections, des unions... En appliquant ces opérateurs, il est alors possible de réduire progressivement la taille des ensembles et d'affiner l'estimation des variables. Différentes représentations approximatives des ensembles sont

proposées afin de faciliter les calculs des opérateurs et accélérer les temps de calcul. Parmi ces approximations on retrouve notamment les boîtes (intervalle 2D) et les parallélotopes ou polygones.

Sur le même principe, mais avec une approche d'analyse par intervalles, Jaulin présente dans [63] (et à l'excercice 11 de son cours en ligne [3]) une méthode permettant de résoudre le RO-SLAM. Le RO-SLAM est transformé en un problème de satisfaction de contraintes, qui, à l'aide d'algorithmes de propagation d'intervalles et de contracteurs permet de progressivement réduire la taille des intervalles jusqu'à converger vers une solution englobant la trajectoire du robot et la position des balises.

Toujours dans le cadre de l'hypothèse des erreurs inconnues mais bornées, Spletzer et al. [123] proposent une solution au problème du RO-SLAM basée sur la résolution de problèmes d'optimisation convexes. L'approche proposée permet de transformer les mesures d'odométrie et de distance en contraintes convexes afin de formuler des problèmes d'optimisation linéaire (LP) et semi-définie positif (SDP) respectivement. En résolvant récursivement les problèmes d'optimisation LP et SDP au fur à l'arrivée de chaque nouvelle mesure d'odométrie ou de distance, il est alors possible d'obtenir une estimation de la position du robot et des balises. L'inconvénient majeur de l'approche est que la solution proposée suppose que l'orientation globale du robot est accessible, par la mesure d'un magnétomètre par exemple. Afin d'améliorer la précision de l'estimation des variables, une méthode basée sur l'approximation par des polytopes est présentée. La validité de l'approche est testée en simulation et des solutions permettant de gérer les mesures aberrantes ainsi que des contraintes non-convexes sont présentées.

### 4.2.4 Mesures inter-balises

Il est important de noter que l'utilisation d'un système de balises RF pour obtenir les mesures de distance possède l'avantage de s'affranchir du problème d'association de données qui apparaît avec d'autres types de capteurs, et qui complique grandement la résolution du SLAM. En effet, comme il est possible de transmettre des données lors de la réalisation de la mesure, il est donc tout à fait logique d'attribuer un identifiant unique à chaque balise et de fournir celui-ci lors des échanges. Cette simple information supplémentaire évite alors le travail fastidieux de chercher de quelle balise provient la mesure. En outre, les balises peuvent être vues comme un réseau de capteurs dont il possible d'exploiter certaines propriétés. Ainsi, la plupart du temps les balises sont aussi capables de réaliser des mesures de distance entre elles, en plus des mesures robot-balise. L'intégration de ces mesures inter-balises permet d'intégrer plus d'information et de contraintes au problème et ainsi d'améliorer les performances du système. Comme cela est écrit dans [127] : "intégrer les mesures inter-balises réduit l'incertitude de l'estimation de la position des balises et indirectement améliore aussi la précision de localisation du robot". Ainsi parmi les travaux utilisant ces propriétés on peut citer : [31], [129], [128], [127] et [131].

### 4.2.5 Analyse de l'observabilité

Comme dans le cas de la localisation, il est important de connaître les propriétés d'observabilité du problème du SLAM, et en particulier ici, du RO-SLAM. Guoquan et al. [61] ont réalisé une analyse d'observabilité du SLAM dans le cas d'un robot mobile de type unicycle, capable de réaliser des mesures de distance et d'angle par rapport à des balises. La conclusion de l'étude est que le système n'est pas observable dans le cas d'une ou plusieurs mesures de distance et d'angle. Le lemme 4.2 dans [61] montre ainsi que l'espace traversé par le gradient

3. https://www.ensta-bretagne.fr/jaulin/iamooc.html

de toutes les dérivées de Lie d'ordre $k$ est donné par :

$$d\mathbb{G} = \text{span}_{\text{row}} \begin{pmatrix} \sin\theta_R & -\cos\theta_R & -\cos\theta_R\delta_x - \sin\theta_R\delta_y & -\sin\theta_R & \cos\theta_R \\ \cos\theta_R & \sin\theta_R & \sin\theta_R\delta_x - \cos\theta_R\delta_y & -\cos\theta_R & -\sin\theta_R \end{pmatrix} \tag{4.1}$$

avec $\delta_x = x_B - x_R$, $\delta_y = y_B - y_R$, $(x_R, y_R, \theta_R)$ la pose du robot et $(x_B, y_B)$ la position de la balise.

La matrice $d\mathbb{G}$ correspond également à la matrice d'observabilité du système du SLAM non-linéaire. Or, cette matrice est seulement de rang 2 (les 3 dernières colonnes étant des combinaisons linéaires des deux premières). Le SLAM n'est donc pas observable et en particulier, les coordonnées globales du système ne sont pas observables. En effet, pour n'importe quelle translation ou rotation dans le plan, il est impossible de distinguer un changement du vecteur d'état avec les mesures.

Même si le problème du RO-SLAM n'est pas observable pour le cas classique d'un système avec un robot unicycle, il est possible de modifier ou de changer le système ainsi que les mesures afin d'obtenir un système observable. Ainsi, dans [88] une nouvelle formulation du RO-SLAM est proposée grâce à une approche basée capteur et une augmentation d'état. Une analyse d'observabilité du nouveau système dynamique est réalisée et permet de valider la garantie de convergence des erreurs.

Dans le même esprit Bayat, et al. [8], [7], proposent un nouvel observateur pour le RO-SLAM qui permet, sous certaines conditions, de faire converger les erreurs d'estimation vers zéro. La description de l'observateur basé sur un estimateur à énergie minimum est suivie par une étude de ses propriétés de convergence. Les résultats expérimentaux obtenus avec un AUV équipé d'une centrale inertielle, d'un appareil de mesure de distance par rapport à des balises fixes et d'un capteur de profondeur, confirment les résultats théoriques de l'étude convergence de l'observateur.

### 4.2.6 Jeux de données

Afin de tester les algorithmes de RO-SLAM et comparer les différentes méthodes, il est intéressant de disposer de jeux de données réalisés en conditions réelles et disponibles en libre accès. Ceci permet de tester les algorithmes développés sans avoir à construire de prototype ou avant de les implémenter sur le système, afin de valider en amont leur bon fonctionnement. Pour le problème spécifique du RO-SLAM, peu de jeux de données sont disponibles. Le plus populaire est celui proposé par Djugash et al. [4] dans [32]. Cinq jeux de données ont été réalisés en extérieur avec un robot mobile et des balises. Chaque jeu est composé des données d'odométrie du robot, des mesures de distance, et de la vérité terrain de la trajectoire du robot et de la position des balises. Ces jeux de données ont déjà été utilisés dans le chapitre précédent sur la localisation et seront réutilisés dans ce chapitre pour réaliser certaines des expériences.

Dans Herranz et al. [59] qui comparent l'algorithme de ROP-EKF [33] avec l'algorithme de lissage SAM [26], des jeux de données de l'université d'Alcalá [5] sont utilisés en plus des jeux de données précédents. Ces jeux de données comportent deux expériences réalisées en environnement intérieur et en conditions réelles à l'aide d'un robot mobile équipé d'un laser pour obtenir la vérité terrain et de codeurs incrémentaux pour l'odométrie. Les mesures de distance sont collectées grâce à des balises WiFi. Le format des données est semblable à celui des jeux de données proposés dans [32].

4. http://www.frc.ri.cmu.edu/projects/emergencyresponse/RangeData
5. http://www.robesafe.es/repository/UAHWiFiDataset

D'autres jeux de données[6] utilisés dans [45] notamment ont été réalisés avec un quadricoptère dans des environnements intérieurs et extérieurs. En plus des mesures de distance, les données d'un altimètre barométrique, d'une caméra et de vérité terrain (VICON ou DGPS) sont fournies. Cependant, l'utilisation d'un drone et l'absence de données d'odométrie, provenant d'une centrale inertielle par exemple, ne permettent pas d'utiliser directement ces jeux de données dans notre cas.

### 4.2.7 Résumé des différents algorithmes

En résumé, le Tableau 4.1 regroupe les différents algorithmes recensés qui ont été utilisés pour résoudre le RO-SLAM, alors que le Tableau 4.2, lui, regroupe les différentes techniques d'initialisation des balises.

| Algorithmes | Articles |
|---|---|
| EKF | [103], [104], [17], [40], [45], [41], [44], [43], [128], [59], [130], [97], [72], [96], [49] |
| KF | [87], [86], [88] |
| PF | [10], [11], [135] |
| SEIF | [127], [131] |
| GraphSlam | [62] |
| Spectral learning | [12] |
| GTSAM, iSAM | [59], [27] |
| Approches ensemblistes | [93], [30], [29] |
| Approches ensemblistes 2 | [64], [28], [66] |
| SDP programming | [123] |
| LM, GN, optimisation | [73] |
| Observateur | [7] |

Tableau 4.1 – Comparatif des algorithmes utilisés pour le RO-SLAM

| Algorithmes | Articles |
|---|---|
| Trilateration | [97] |
| PF | [130], [11], [128], [127], [131], [135], [96] |
| MoG | [17], [40], [45], [41], [44], [43], [10] |
| ROP-EKF | [33], [59] |
| Composite | [49] |
| Grille d'occupation | [103], [104], [34], [31] |

Tableau 4.2 – Comparatif des algorithmes utilisés pour initialiser les balises dans le RO-SLAM

## 4.3 Nouvelle méthode d'initialisation des balises

Comme cela a été vu précédemment dans la section dédiée à l'état de l'art, la principale difficulté du RO-SLAM concerne l'étape d'initialisation de la position des balises. Dans le cas d'une solution basée sur un filtre EKF, l'initialisation d'une balise correspond à l'ajout des paramètres modélisant la position de la balise dans le vecteur d'état du filtre. Afin d'exploiter entièrement

6. https://grvc.us.es/staff/felramfab/roslam_datasets

les informations fournies par les mesures de distance, cette étape d'initialisation doit se faire le plus tôt possible, idéalement dès la première mesure. De plus, cette étape d'initialisation doit être la plus robuste possible aux différents types de bruits, et fournir une estimation correcte de la position initiale de la balise afin de permettre la convergence de ses paramètres vers les bonnes valeurs. Sinon, si l'estimation initiale de la position d'une balise est trop éloignée de la vraie solution, le filtre risque fortement de diverger et de fournir une mauvaise estimation. Enfin, l'étape d'initialisation doit aussi permettre de bonnes performances de calcul, en gardant une faible complexité et en utilisant un nombre de paramètres aussi faible que possible. Aucune des méthodes présentées dans l'état de l'art ne répond entièrement à tous ces critères. Les méthodes avec délai comme la trilatération, les grilles d'occupation et le filtre à particules retardent l'initialisation et n'exploitent pas totalement les premières mesures de distance. De plus, les méthodes d'initialisation basées une grille d'occupation ou un filtre particulaire (suivant le nombre de particules utilisées), bien qu'assez robustes, sont souvent lentes en temps de calcul. Quant aux méthodes sans délai, comme avec la mixture de gaussiennes, elles utilisent bien souvent une sur-paramétrisation de la position des balises ce qui a pour effet d'agrandir le vecteur d'état et ainsi d'augmenter le temps de calcul de manière quadratique par rapport au nombre de variables d'état à estimer.

Au regard de ces constatations, une nouvelle méthode d'initialisation des balises est proposée ici, qui est basée sur l'initialisation de deux hypothèses d'une mixture de gaussiennes obtenues après calcul de l'intersection de deux cercles. Cette méthode repose sur l'observation qu'avec la méthode de la mixture de gaussiennes proposée dans [17], le nombre d'hypothèses converge assez rapidement vers deux solutions. De plus, dans les travaux d'Olson et al. [103] et Djugash et al. [33], l'utilisation de deux mesures de distance pour calculer l'intersection de deux cercles est aussi employée afin d'obtenir des solutions possibles pour la position d'une balise. Deux méthodes permettant de calculer l'intersection de deux cercles sont proposés dans [134] et [85]. Suivant la fréquence des mesures de distance, attendre deux mesures pour calculer l'intersection peut se faire rapidement et permet donc d'avoir un compromis entre les approches sans et avec délai. De plus, par rapport à l'approche avec délai basée sur la trilatération, le calcul de l'intersection de deux cercles est beaucoup moins sensible aux bruits des mesures que l'algorithme de trilatération.

Ainsi, l'idée de la solution proposée consiste à enregistrer ou incorporer dans l'état du filtre, la position du robot au moment de la première mesure de distance et d'attendre la seconde mesure de distance afin de calculer l'intersection de deux cercles. Le calcul de l'intersection de deux cercles fournit alors généralement deux solutions (ou 1, ou 0), qui sont alors regroupées dans une mixture de gaussiennes à deux hypothèses. Ces hypothèses sont insérées dans le vecteur d'état du filtre EKF, de la même façon que dans [17], à la différence qu'ici les balises sont paramétrées en coordonnées cartésiennes, ce qui permet de réduire le nombre de paramètres à estimer.

Dans la suite, nous allons tout d'abord introduire les différentes variables du problème. Puis, les différentes étapes de l'algorithme sont présentées en détail. Et pour finir, les performances de la méthode proposée sont comparées à d'autres algorithmes de la littérature grâce à des simulations et à un jeu de données réelles.

### 4.3.1 Modélisation du problème

#### Robot

On considère toujours un robot mobile à conduite différentielle modélisé par un unicycle évoluant dans un environnement plan. Sa pose dans le repère du monde à l'instant $t_k = k\Delta T$,

$\Delta T$ étant la période d'échantillonnage, est donc représentée par :

$$\mathbf{p}_k = (x_k, y_k, \theta_k)^T$$

On suppose que le robot est équipé de codeurs incrémentaux sur les moteurs des roues qui permettent d'obtenir la distance $\Delta U_k$, en mètres, et l'angle $\Delta\Theta_k$, en radians, parcourus entre deux périodes d'échantillonnage $k-1$ et $k$. La mesure d'odométrie correspondante est donnée par :

$$\mathbf{U}_k = (\Delta U_k, \Delta\Theta_k)^T + \epsilon_k$$

où $\epsilon_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k)$ représente un bruit blanc gaussien de moyenne nulle et de covariance :

$$\mathbf{Q}_k = \text{diag}(\sigma^2_{\Delta U}, \sigma^2_{\Delta\Theta})$$

avec $\sigma_{\Delta U} = k_{\Delta U}|\Delta U_k|$ et $\sigma_{\Delta\Theta} = k_{\Delta\Theta}|\Delta\Theta_k|$. Les écarts-type du bruit sur les incréments de distance et d'angle sont donc proportionnels à la distance et à l'angle parcourus. Les facteurs de proportionnalités $k_{\Delta U}$ et $k_{\Delta\Theta}$ sont déterminés expérimentalement afin de correspondre aux performances des codeurs incrémentaux des roues.

### Balises

On suppose qu'un nombre $N$ de balises statiques et identifiables sont positionnées sur le terrain dans lequel le robot évolue. La position de la balise $i$ dans le repère du monde est notée :

$$\mathbf{p}_i = (x_i, y_i)^T$$

Par rapport au chapitre précédent sur la localisation, on suppose ici que la position des balises n'est pas connue à l'avance. Ceci implique d'ajouter ces variables supplémentaires dans le vecteur d'état du filtre EKF afin de les estimer en plus de la pose du robot.

Les balises fournissent des mesures de distance entre elles et le robot. Bien qu'il soit possible et même recommandé d'exploiter les mesures inter-balises (cf. section 4.2.4), cette information supplémentaire n'est pas prise en compte ici. Ainsi, seules les mesures entre les balises et le robot seront utilisées par la suite. En partant du modèle défini dans la section 2.3.1, l'équation de mesure de distance $h(\mathbf{x})$ entre la position du robot à l'instant $k$ et la balise $i$ est donnée par :

$$h(\mathbf{x}) = d_{i,k} + b_i + \epsilon_i \tag{4.2}$$

avec $d_{i,k} = \sqrt{(x_i - x_k)^2 + (y_i - y_k)^2}$, et $\epsilon_i = \mathcal{N}(0, \sigma_i^2(d_i))$ la réalisation d'un bruit blanc gaussien de moyenne nulle et de variance $\sigma_i^2(d_i)$. L'expression de la variance est donc propre à chaque balise et dépendante de la distance mesurée. $b_i$ représente le biais de la mesure. Ce dernier peut être obtenu au préalable par une procédure de calibration ou peut être inclus dans l'état du filtre EKF afin d'estimer sa valeur en temps réel. Par la suite, on suppose que le biais a été déterminé à l'avance par une méthode de calibration (cf. annexe A), et que sa valeur est identique pour toutes les balises : $b = b_i$, $i \in [1, N]$.

### Filtre de Kalman

Comme la position des balises n'est pas connue à l'avance, il est nécessaire d'estimer ces paramètres en plus de la pose du robot dans le vecteur d'état du filtre EKF. Le nombre de variables à estimer est donc plus important que dans le cas de la localisation et est proportionnel au nombre de balises déployées. La taille du vecteur d'état du filtre EKF dans le cas du RO-SLAM en 2D est en effet de $3+2\times N$. Bien que le nombre de balises à utiliser dépend de la taille et de la géométrie du terrain, l'utilisation de balises RF UWB permet de bénéficier d'une portée

assez élevée pouvant atteindre plus de 100 m. Ceci permet ainsi généralement d'employer moins d'une dizaine de balises. Ce chiffre est à comparer à la centaine voir au millier de caractéristiques visuelles communément employées dans le cas du SLAM visuel. Ainsi, la complexité quadratique par rapport à la taille du vecteur d'état de l'étape de correction du filtre EKF restera limitée et permettra d'obtenir un algorithme efficace en temps de calcul.

Bien qu'une paramétrisation de la position des balises en coordonnées polaires permet de mieux gérer les non-linéarités et le caractère multimodal inhérents aux mesures de distance [33], cette dernière requiert l'utilisation de deux variables supplémentaires par balise par rapport aux coordonnées cartésiennes. Ici, nous avons fait le choix de garder une paramétrisation en coordonnées cartésiennes afin de garder un nombre minimal de variables à estimer dans le filtre. Ainsi, le vecteur d'état du filtre à l'instant $k$, $\mathbf{x}_k$, après l'initialisation de toutes les balises est donné par :

$$\mathbf{x}_k = (\mathbf{p}_k, \mathbf{p}_1, \ldots, \mathbf{p}_i, \ldots, \mathbf{p}_N)^T$$

Dans le cas où la balise $i$ n'est pas encore totalement initialisée, il existe encore deux hypothèses possibles pour la position de la balise provenant du calcul de l'intersection de deux cercles. Le vecteur $\mathbf{p}_i$ contient donc les deux hypothèses :

$$\mathbf{p}_i = (x_{i1}, y_{i1}, x_{i2}, y_{i2})^T$$

avec $(x_{ij}, y_{ij})^T$ la position de la j$^{\text{ème}}$ hypothèse de la balise $i$.

### 4.3.2 Description de l'algorithme

L'algorithme utilisé afin d'obtenir une estimation de la pose du robot et de la position des balises est un filtre de Kalman étendu dont le principe de fonctionnement a déjà été présenté au chapitre précédent sur la localisation (cf. section 3.6.1). Nous allons maintenant reprendre les différentes étapes du filtre en donnant uniquement les équations nécessaires à l'implémentation de l'algorithme dans le cas du RO-SLAM.

#### Étape de prédiction

Les balises étant statiques dans l'environnement :

$$\mathbf{p}_{i,k+1} = \mathbf{p}_{i,k}, \quad \forall i \in [1, N]$$

les seules variables affectées par l'étape de prédiction sont celles de la pose du robot. Ainsi, de façon similaire au cas de la localisation la pose du robot évolue suivant les équations suivantes :

$$\begin{cases} x_{k+1} = x_k + \Delta U_k \cos(\theta_k + \frac{\Delta\Theta_k}{2}) \\ y_{k+1} = y_k + \Delta U_k \sin(\theta_k + \frac{\Delta\Theta_k}{2}) \\ \theta_{k+1} = \theta_k + \Delta\Theta_k \end{cases}$$

#### Étape de correction

Pour l'étape de correction du filtre EKF, on distingue deux cas en fonction du nombre d'hypothèses restant pour la position d'une balise. Quand la position de la balise a convergé vers une unique solution, on peut appliquer normalement les équations de correction. Tout d'abord, on prédit la mesure de distance en utilisant l'équation de mesure donnée par (4.2). Puis, on calcule le Jacobien, $\mathbf{H}$, de la fonction de mesure, $h(\mathbf{x})$, par rapport aux variables d'état du filtre :

$$\mathbf{H} = \frac{\partial h(\mathbf{x})}{\partial \mathbf{x}} = \begin{pmatrix} -\frac{d_x}{d_{ik}} & -\frac{d_y}{d_{ik}} & 0 & \ldots & 0 & \frac{d_x}{d_{ik}} & \frac{d_y}{d_{ik}} & 0 & \ldots \end{pmatrix} \tag{4.3}$$

avec $d_x = x_i - x_k$, et $d_y = y_i - y_k$.

Il est alors possible de calculer le vecteur $\boldsymbol{\nu}$ et la matrice $\mathbf{S}$ d'innovation. On peut alors enfin calculer le gain de Kalman $\mathbf{K}$ qui permet de corriger le vecteur d'état, $\mathbf{x}_k$, et la matrice de covariance, $\mathbf{P}_k$, du filtre.

Dans le cas où il y a encore deux hypothèses pour la position d'une balise, les équations de correction décrites dans [17] sont appliquées. Les pondérations, $\omega_{ij}$, de chaque hypothèse $j$ sont aussi mises à jour. Pour cela, on calcule la probabilité $p_j$ que la mesure de distance provienne bien de l'hypothèse $j$ :

$$p_j = p\Big(r_{i,k}\Big|(x_k, y_k), (x_{ij}, y_{ij})\Big) = \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\Big(-\frac{(r_{i,k} - d_{ijk})^2}{2\sigma_i^2}\Big) \tag{4.4}$$

avec $d_{ijk} = \sqrt{(x_{ij} - x_k)^2 + (y_{ij} - y_k)^2}$ la distance prédite pour l'hypothèse $j$ de la balise $i$, et $r_{i,k}$ la mesure de distance. L'étape de correction complète du filtre est résumée dans l'Algorithme 4.

Comme dans le cas de la localisation, un test de Mahalanobis est réalisé avant de réaliser la correction afin de rejeter les mesures de distance erronées qui seraient trop éloignées de la valeur prédite. Si la distance de Mahalanobis, donnée par la formule (3.51), est supérieure à un seuil, $t_{\text{maha}}$, fixé entre 100 et 1000 dans les expériences réalisées par la suite, la mesure est rejetée et l'étape de correction n'est pas appliquée. L'avantage d'un tel test est double. Tout d'abord, il permet d'augmenter la robustesse du filtre aux mesures erronées. Deuxièmement, il permet aussi, en comptant le nombre de mesures rejetées, d'avoir une indication sur la bonne initialisation d'une balise. En effet, si les mesures de distance par rapport à une balise sont systématiquement rejetées, cela peut indiquer que celle-ci n'a pas correctement convergé vers la bonne position et que l'initialisation ne s'est pas déroulée correctement. Cette procédure d'initialisation est maintenant détaillée dans la section qui suit.

### Initialisation d'une balise

Une bonne estimation initiale de la position des balises est nécessaire au bon fonctionnement du filtre EKF, afin de permettre la convergence des variables vers les bonnes valeurs. La mauvaise initialisation d'une seule balise peut être préjudiciable à l'ensemble des variables estimées dans le filtre et entraîner sa divergence. En utilisant une seule mesure de distance, l'initialisation d'une balise aura un caractère multimodal en raison du manque de l'information angulaire. Afin d'obtenir une représentation précise de cette incertitude angulaire, il est souvent nécessaire d'employer un grand nombre d'hypothèses. Cependant, en utilisant une deuxième mesure de distance, le nombre d'hypothèses tombe directement au maximum à deux. Il est donc possible, avec un délai de deux mesures de distance, d'initialiser la position d'une balise en utilisant seulement deux hypothèses et cela tout en gardant une paramétrisation en coordonnées cartésiennes. La méthode d'initialisation proposée repose sur cette idée et se déroule de la façon suivante.

Dès la première mesure de distance par rapport à une balise i, la position courante estimée du robot, $\mathbf{p}_o = \mathbf{p}_k = (x_k, y_k)^T$, est gardée en mémoire, ainsi que la mesure, $r_o = r_i$. Une alternative plus précise consiste à augmenter l'état du filtre avec la position courante du robot [85]. Cette façon de procéder permet ainsi de garder la corrélation entre les variables et les incertitudes liées à la position du robot qui seront utilisées pour calculer l'incertitude initiale de la position de la balise. Le vecteur d'état augmenté de la pose courante du robot devient donc :

$$\mathbf{x}_k = (\mathbf{p}_k, \mathbf{p}_o, \mathbf{p}_1, \ldots, \mathbf{p}_i, \ldots, \mathbf{p}_N)^T$$

**Input** : $\mathbf{x}_k$, $\mathbf{P}_k$, $r_{ik}$, $\sigma_i$, $t_{\text{maha}}$
**Output** : $\mathbf{x}_k$, $\mathbf{P}_k$

$n_i$ = nombre d'hypothèses de la balise $i$ ;
**if** $n_i == 1$ **then**
  Calculer $d_{ik}$ ;
  Calculer $\nu = r_{ik} - d_{ik}$ ;
  Calculer $\mathbf{H}$ d'après (4.3) ;
  $\mathbf{S} = \mathbf{H}\mathbf{P}_k\mathbf{H}^T + \sigma_i^2$ ;
  /* Test de Mahalanobis */
  Calculer $d_{\text{maha}}$ d'après (3.51) ;
  **if** $d_{\text{maha}} \leq t_{\text{maha}}$ **then**
    $\mathbf{K} = \mathbf{P}_k\mathbf{H}^T\mathbf{S}^{-1}$ ;
    $\mathbf{x}_k = \mathbf{x}_k + \mathbf{K}\nu$ ;
    $\mathbf{P}_k = (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P}_k$ ;
  **end**
**else**
  **for** $j = 1$ **to** $n_i$ **do**
    Calculer $d_{ijk}$ ;
    /* Probabilité de l'hypothèse $j$ */
    Calculer $p_j$ d'après (4.4) ;
  **end**
  /* Calcul des pondérations des hypothèses */
  **for** $j = 1$ **to** $n_i$ **do**
    $\lambda_j = \frac{p_j}{\sum_j p_j}$ ;
    $\sigma_j^2 = \frac{\sigma_j^2}{\lambda_j}$ ;
    /* Pondération de l'hypothèse $j$ */
    $\omega_{ij} = \omega_{ij}p_j$ ;
  **end**
  /* Normalisation des pondérations */
  **for** $j = 1$ **to** $n_i$ **do**
    $\omega_{ij} = \frac{\omega_{ij}}{\sum_j \omega_{ij}}$ ;
  **end**
  /* Fusion des hypothèses */
  **for** $j = 1$ **to** $n_i$ **do**
    $\nu_j = r_{ik} - d_{ijk}$ ;
    Calculer $\mathbf{H}_j$ d'après (4.3) ;
    $\mathbf{S}_j = \mathbf{H}_j\mathbf{P}_k\mathbf{H}_j^T + \sigma_j^2$ ;
    $\mathbf{K}_j = \mathbf{P}_k\mathbf{H}_j^T\mathbf{S}_j^{-1}$ ;
    $\mathbf{x}_k = \mathbf{x}_k + \mathbf{K}_j\nu_j$ ;
    $\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_j\mathbf{H}_j)\mathbf{P}_k$ ;
  **end**
**end**

**Algorithme 4 :** Étape de correction du filtre EKF pour une mesure de distance provenant de la balise $i$

Après le traitement de la première mesure de distance, il faut attendre une seconde mesure afin de réaliser l'initialisation des paramètres de la position de la balise dans le filtre. Une fois la deuxième mesure de distance, $r_i$, disponible, il est possible de calculer l'intersection de deux cercles centrées sur les positions courante $(x_k, y_k)$ et enregistrée $(x_o, y_o)$ du robot et de rayons $r_i$ et $r_o$ respectivement. Pour ce faire, il suffit de résoudre une équation du second ordre à partir du système à deux équations suivant [134], [85] :

$$\begin{cases} (x_o - x_i)^2 + (y_o - y_i)^2 = r_o^2 \\ (x_k - x_i)^2 + (y_k - y_i)^2 = r_i^2 \end{cases} \tag{4.5}$$

La résolution du système permet d'obtenir deux, une, ou aucune solution. Dans le cas où aucune solution n'est trouvée, on attend simplement la mesure suivante afin de répéter la procédure. S'il y a une unique solution, celle-ci est directement insérée dans le vecteur d'état du filtre. Dans le cas général avec deux solutions, celles-ci sont ajoutées dans le vecteur d'état du filtre sous la forme d'une mixture de gaussiennes à deux hypothèses. Les pondérations, $\omega_{ij}$, associées à chaque hypothèse sont initialisées à $\frac{1}{2}$. Concernant l'initialisation des matrices de covariance associées aux positions initiales des balises, deux solutions sont possibles. Si le vecteur d'état a été augmenté avec la position du robot correspondant à la première mesure de distance, la méthode employée dans [85] peut être appliquée afin de propager les incertitudes des positions du robot et des mesures de distance aux positions des balises. La matrice de covariance du filtre juste après l'insertion des deux hypothèses de la position de la balise i prend la forme suivante :

$$\mathbf{P}_k^+ = \begin{pmatrix} \mathbf{P}_k & \mathbf{P}_k\mathbf{G}_x^T & \mathbf{P}_k\mathbf{G}_x^T \\ \mathbf{G}_x\mathbf{P}_k & \mathbf{P}_1 & \mathbf{0} \\ \mathbf{G}_x\mathbf{P}_k & \mathbf{0} & \mathbf{P}_2 \end{pmatrix} \tag{4.6}$$

avec

$$\begin{aligned} \mathbf{P}_j &= \mathbf{G}_x\mathbf{P}_k\mathbf{G}_x^T + \mathbf{G}_z\mathbf{R}\mathbf{G}_z^T, \quad j = \{1, 2\} \\ \mathbf{R} &= \mathrm{diag}(\sigma_{\mathrm{i}}^2, \sigma_{\mathrm{i}}^2) \\ \mathbf{G}_x &= -\mathbf{H}_y^{-1}\mathbf{H}_x \\ \mathbf{G}_z &= \mathbf{H}_y^{-1} \end{aligned}$$

et $\mathbf{H}_x$ et $\mathbf{H}_y$ sont les Jacobiens du système d'équations (4.5) de l'intersection de deux cercles par rapport à l'état et aux balises respectivement :

$$\mathbf{H}_x = \begin{pmatrix} \mathbf{H}_{x,1} & \mathbf{0}_{2\times 2} & \dots & \mathbf{0}_{2\times 2} & \mathbf{H}_{x,2} & \mathbf{0}_{2\times 2} & \dots \end{pmatrix} \tag{4.7}$$

$$\mathbf{H}_{x,1} = \begin{pmatrix} \frac{-2(x_i - x_o)}{r_o} & \frac{-2(y_i - y_o)}{r_o} \\ 0 & 0 \end{pmatrix} \tag{4.8}$$

$$\mathbf{H}_{x,2} = \begin{pmatrix} 0 & 0 \\ \frac{-2(x_i - x_k)}{r_i} & \frac{-2(y_i - y_k)}{r_i} \end{pmatrix} \tag{4.9}$$

$$\mathbf{H}_y = \begin{pmatrix} \frac{2(x_i - x_o)}{r_o} & \frac{2(y_i - y_o)}{r_o} \\ \frac{2(x_i - x_k)}{r_i} & \frac{2(y_i - y_k)}{r_i} \end{pmatrix} \tag{4.10}$$

Si plus aucune mesure de distance n'est associée à l'ancienne pose du robot, $\mathbf{p}_o$, celle-ci est supprimée du vecteur d'état.

L'autre alternative, consiste à initialiser les matrices de covariances en additionnant les incertitudes de la position courante du robot et les incertitudes des mesures de distance. Les différentes étapes de la procédure d'initialisation sont résumés dans Algorithme 5.

À noter qu'il peut être utile d'imposer une distance minimale $d_{\min}$ entre les positions du robot au moment de la première et de la seconde mesure :

$$\sqrt{(x_k - x_o)^2 + (y_k - y_o)^2} \geq d_{\min}$$

Ceci permet d'éviter les cas où des positions trop rapprochées du robot ne permettraient pas de résoudre correctement le système d'équations de l'intersection de deux cercles. Cette distance minimale peut être choisie en rapport avec l'écart-type des balises afin d'éviter que le bruit de mesure n'influence trop les résultats obtenus.

**Input** : $\mathbf{x}_k$, $\mathbf{P}_k$, $r_i$, $\sigma_i$, $d_{min}$
**Output** : $\mathbf{x}_k$, $\mathbf{P}_k$

**if** *Balise $i$ observée pour la 1ème fois* **then**
  Augmenter le vecteur d'état du filtre avec la pose courante du robot ;
  Augmenter la matrice de covariance ;
  Enregistrer la mesure de distance $r_i$ ;
**else**
  Récupérer la 1ère mesure de distance $r_o$ correspondant à la balise $i$ ;
  Récupérer la position du robot associée : $\mathbf{p}_o$ ;
  Calculer la distance $d$ entre la position courante du robot $\mathbf{p}_k$ et $\mathbf{p}_o$ ;
  **if** $d \geq d_{min}$ **then**
    Résoudre le système d'équations (4.5) pour obtenir $n$ hypothèses ;
    **if** $n > 0$ **then**
      **for** $j \leftarrow 1$ **to** $n$ **do**
        Initialiser les pondérations : $\omega_{ij} = \frac{1}{n}$ ;
        Augmenter le vecteur d'état $\mathbf{x}_k$ avec : $(x_{ij}, y_{ij})$ ;
        Calculer les matrices $\mathbf{G}_x$ et $\mathbf{G}_z$ avec (4.7), (4.8), (4.9), (4.10) ;
        Augmenter $\mathbf{P}_k$ en utilisant (4.6) et $\sigma_i$ ;
      **end**
      Si nécessaire, supprimer $\mathbf{p}_o$ du vecteur d'état $\mathbf{x}_k$ ;
    **end**
  **end**
**end**

**Algorithme 5 :** Procédure d'initialisation de la position d'une balise $i$ dans le vecteur d'état du filtre EKF

### Suppression des hypothèses

L'emploi d'une mixture de gaussiennes permet de suivre l'évolution et la pertinence de chaque hypothèse grâce à sa pondération associée. Après un certain nombre de mesures de distance, les pondérations des hypothèses vont augmenter ou diminuer en fonction de la probabilité que les mesures proviennent bien de la position associée à l'hypothèse. Une faible pondération indique une faible probabilité que l'hypothèse associée corresponde à la bonne solution, alors qu'une pondération élevée signifie une forte probabilité que l'hypothèse soit la bonne. Comme il y a peu d'intérêt à suivre l'évolution d'une mauvaise hypothèse, et qu'il est préférable de maintenir le vecteur d'état le plus petit possible, il est utile de mettre en place un mécanisme de suppression des mauvaises hypothèses. Pour cela, différents tests sont réalisés pour déterminer si une hypothèse doit être supprimée ou non.

Le premier test permet de supprimer une hypothèse $j$, associée à la balise $i$, si sa pondération, $\omega_{ij}$, est inférieure à un seuil $t_1$ fixé à $1e^{-3}$ par la suite.

Le second test permet de fusionner les deux hypothèses d'une balise en une seule, si les positions associées sont proches l'une de l'autre. Pour déterminer si les positions des deux

hypothèses sont proches, la distance Euclidienne entre les deux positions est calculée, puis comparée à un deuxième seuil $t_2$ proportionnel à l'écart-type de la mesure de distance. Si la distance est inférieure au seuil $t_2$, l'hypothèse ayant la pondération la plus faible est supprimée du filtre au profit de celle avec la pondération la plus élevée.

Si l'initialisation des balises se déroule correctement et que les mauvaises hypothèses sont toutes bien supprimées, le vecteur d'état du filtre aura une forme minimale et sera composé uniquement de la pose du robot et de la position des balises en coordonnées cartésiennes. Dans la prochaine section, la validité de l'approche est testée, puis comparée à d'autres procédures d'initialisation des balises.

## 4.4 Expériences

Afin de tester le bon fonctionnement de l'approche proposée et de comparer ses performances avec d'autres techniques d'initialisation, deux types d'expériences sont réalisés de façon similaire à ce qui a été fait dans le chapitre précédent sur la localisation (cf. 3.6.2). Ainsi, les jeux de données Gesling 2 et Plaza 1 de [32] [7] sont à nouveau utilisés afin de réaliser les expériences. Le premier jeu, Gesling 2, est utilisé afin de tester différents niveaux de bruits et différents pourcentages de mesures erronées en ajustant les paramètres de bruits perturbant les mesures parfaites. Les différentes configurations de bruit sont les mêmes qu'à la section 3.6.2. Le second jeu de données, Plaza 1, est utilisé avec les vraies mesures provenant de l'expérience réelle.

Afin de comparer la méthode d'initialisation des balises dans le filtre EKF présentée dans la section précédente, appelée *Comp* par la suite, les méthodes suivantes ont aussi été implémentées :

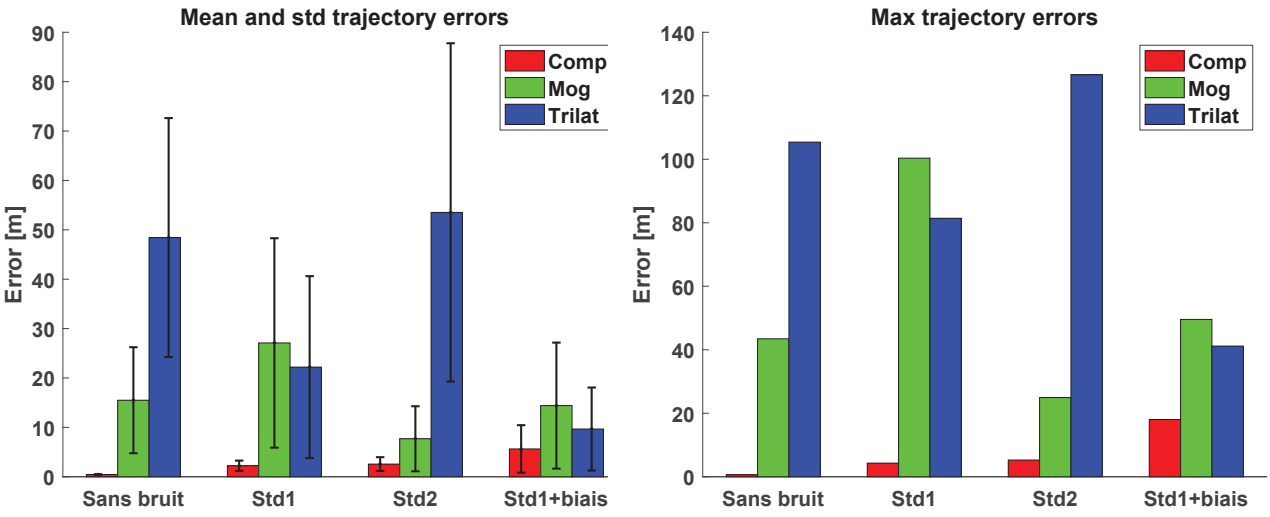
- *Trilat* : méthode d'initialisation avec délai basée sur la trilatération [97], 5 mesures de distance sont utilisées pour l'algorithme de trilatération,
- *Grid* : méthode d'initialisation avec délai basée sur une grille d'occupation. Au contraire de la méthode proposée dans [104], la version implémentée incorpore les mesures de distance les unes après les autres et non deux par deux,
- *Pf* : méthode d'initialisation avec délai utilisant un filtre particulaire [11], avec 1000 particules,
- *Mog* : méthode d'initialisation sans délai utilisant une mixture de 16 gaussiennes [17],
- *Rop* : méthode d'initialisation sans délai basée sur les travaux de [33].

Les performances des différentes méthodes d'initialisation sont comparées à partir de trois critères : les erreurs sur la trajectoire, les erreurs sur la position des balises, le temps de calcul, ainsi que l'itération à partir de laquelle toutes les balises sont initialisées dans le filtre. Les erreurs sur les positions de la trajectoire et sur les positions des balises sont calculées en utilisant la distance entre les positions données par la vérité terrain et celles estimées par le filtre EKF. La moyenne, l'écart-type et la valeur maximale des erreurs sont alors calculés pour la trajectoire et les balises et utilisés pour réaliser les comparaisons. Le temps d'exécution est calculé à partir des commandes *cputime* et *tic*/*toc* de *MatLab* [8].

### 4.4.1 Résultats des simulations sur Gesling2

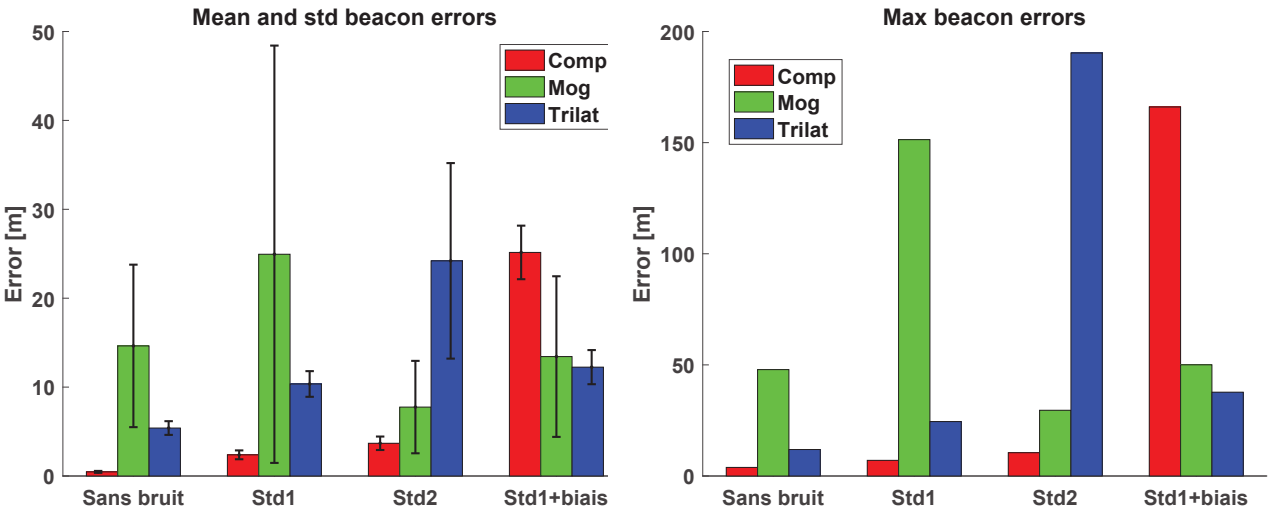
Les simulations menées sur le jeu de données Gesling2 sont réalisées afin d'analyser le comportement du filtre EKF développé en présence de différents types et niveaux de bruits. Pour ces premières simulations, seules trois méthodes d'initialisation des balises sont comparées, à

7. http://www.frc.ri.cmu.edu/projects/emergencyresponse/RangeData
8. https://fr.mathworks.com




(a) Erreurs moyennes et écarts-types (b) Erreurs maximales

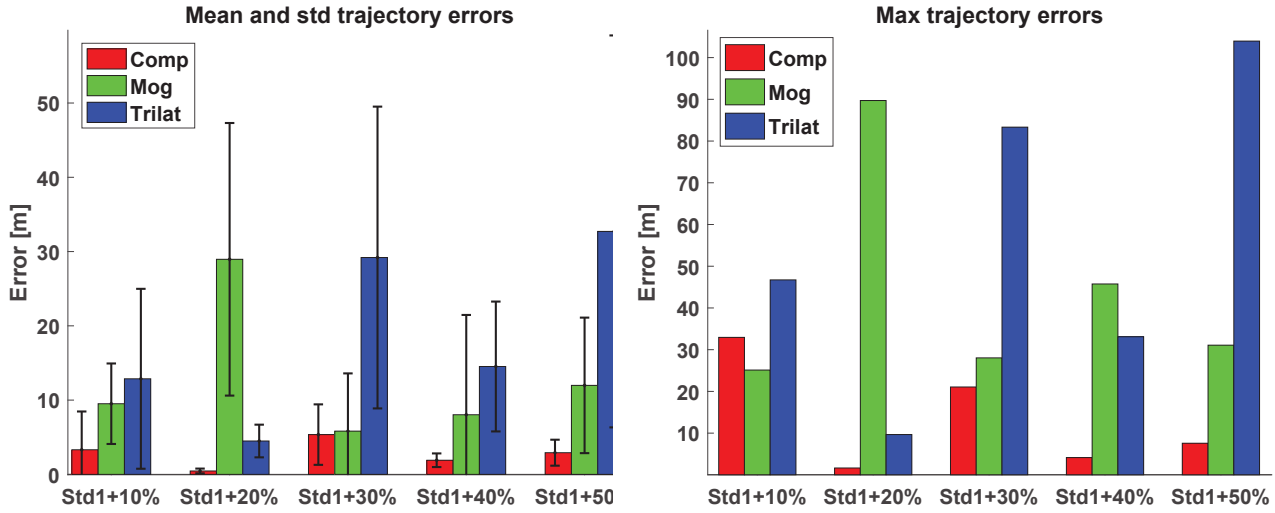
FIGURE 4.3 – Comparaison des erreurs moyennes, des écarts-types (a) et des erreurs maximales (b) de la trajectoire du robot pour différents bruits sur le jeu de données Gesling2




(a) Erreurs moyennes et écarts-types (b) Erreurs maximales

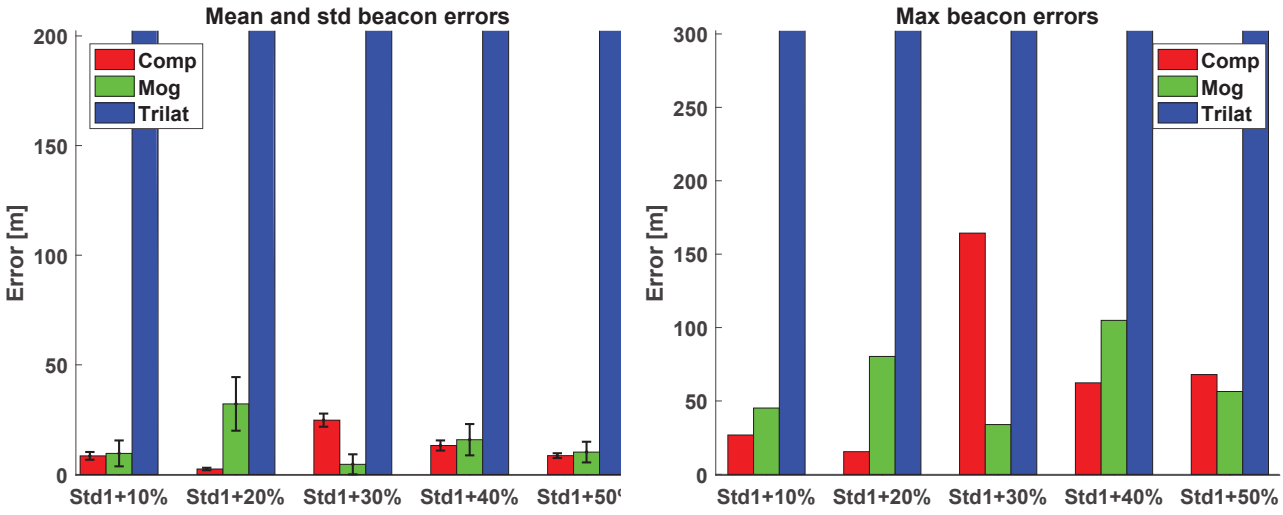
FIGURE 4.4 – Comparaison des erreurs moyennes, des écarts-types (a) et des erreurs maximales (b) de la position des balises pour différents bruits sur le jeu de données Gesling2

savoir : *Comp*, *Mog* et *Trilat*. Les autres étant comparées sur le jeu de données Plaza1 présenté par la suite. Les paramètres du filtre EKF, dont la matrice $\mathbf{Q}$ sur le bruit de l'odométrie et $\mathbf{R} = \sigma_i$ sur le bruit des mesures de distance, sont réglés ici avec les mêmes valeurs que les paramètres des bruits utilisés pour perturber les données parfaites de la simulation. Les erreurs moyennes, écarts-types et erreurs maximales sur la trajectoire du robot pour les différents bruits sont visibles aux figures 4.3 et 4.5. De même, les résultats pour les erreurs sur la position des balises sont visibles aux figures 4.4 et 4.6. En s'intéressant tout d'abord à la méthode d'initialisation *Comp* proposée (barres rouges), on constate sur les figures 4.3a et 4.3b que plus le bruit est important, plus les erreurs moyennes et maximales sont élevées. Ceci est aussi valable pour les erreurs sur la position des balises visibles aux figures 4.4a et 4.4b.




(a) Erreurs moyennes et écarts-types (b) Erreurs maximales

FIGURE 4.5 – Comparaison des erreurs moyennes, des écarts-types (a) et des erreurs maximales (b) de la trajectoire du robot pour différents pourcentages de mesures aberrantes sur le jeu de données Gesling2




(a) Erreurs moyennes et écarts-types (b) Erreurs maximales

FIGURE 4.6 – Comparaison des erreurs moyennes, des écarts-types (a) et des erreurs maximales (b) de la position des balises pour différents pourcentages de mesures aberrantes sur le jeu de données Gesling2

Pour la trajectoire, les erreurs moyennes et maximales sont aux alentours de quelques mètres, sauf pour l'erreur maximale avec le bruit Std1+Biais qui monte jusqu'à environ 20 mètres. Cette erreur assez importante, signifie une perte progressive ou temporaire du suivi de la trajectoire du robot. En regardant la Figure 4.4 sur les erreurs de position des balises pour le même bruit, on constate que la moyenne est supérieure à 20 mètres et que l'erreur maximale dépasse les 100 mètres. On peut ainsi déduire pour ce scénario, que l'initialisation de plusieurs des balises s'est mal déroulée, ce qui a eu pour conséquence de fausser l'estimation de la position du robot.

Excepté pour le bruit Std1+Biais, l'ensemble des erreurs moyennes, maximales et des écarts-

types des autres bruits montre que la méthode *Comp* est capable d'estimer correctement la trajectoire du robot ainsi que la position des balises. Si l'on regarde la Figure 4.7 montrant la trajectoire finale (ligne bleue) ainsi que les position des balises (étoiles bleues) estimées par le filtre EKF avec la méthode *Comp* pour le bruit Std1, on constate visuellement que les positions des balises sont proches de la vérité terrain (étoiles jaunes) et que la trajectoire estimée ne dérive pas comme l'odométrie seule (ligne verte) et suit d'assez près la vérité terrain (ligne rouge). En effet, comme le montre la Figure 4.8 avec l'évolution des erreurs de trajectoire au cours du temps, on constate que l'erreur navigue entre 0.5 et 4 mètres environ. L'erreur sur l'orientation du robot reste quant à elle généralement inférieure à 0.2 radians soit une dizaine de degrés. En regardant la Figure 4.9 montrant cette fois-ci l'évolution des erreurs des différentes balises au cours du temps, on constate que les erreurs sont généralement comprises entre 1 et 5 mètres avec de fortes disparités suivant les balises. On constate aussi que l'évolution de l'erreur varie d'une balise à l'autre. Ainsi pour la balise 1, l'erreur diminue progressivement au cours du temps, alors que pour la balise 6, l'erreur augmente progressivement jusqu'à stagner autour de 1 mètre et enfin pour les balises 5 et 7, les erreurs sont à peu près constantes autour de 1 et 5 mètres respectivement. Ces variations s'expliquent en grande partie par la configuration des positions du robot ainsi que le bruit affectant les mesures de distance associées, qui sont utilisés pour l'initialisation de la balise. En fonction de ces paramètres, le calcul de l'intersection de deux cercles donnera une estimation plus ou moins proche de la vérité terrain.

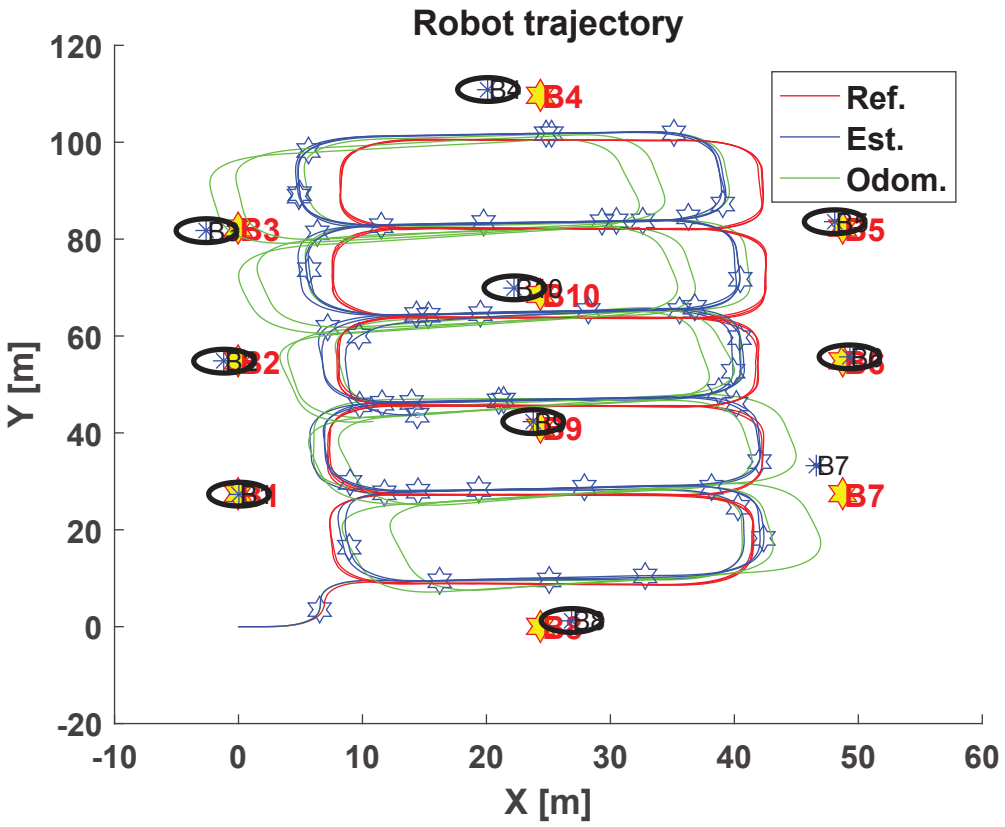



FIGURE 4.7 – Trajectoire du robot et positions des balises estimées par le filtre EKF avec l'approche *Comp* dans le cas du bruit Std1 pour le jeu de données Gesling2

En observant les résultats pour des pourcentages de mesures aberrantes croissants sur la Figure 4.5 pour la trajectoire et sur la Figure 4.6 pour les balises, on constate que les erreurs n'augmentent pas forcément en fonction du pourcentage de mesures aberrantes. Ainsi, pour la trajectoire Figure 4.5, les erreurs moyennes et maximales pour Std1+10% et Std1+30%

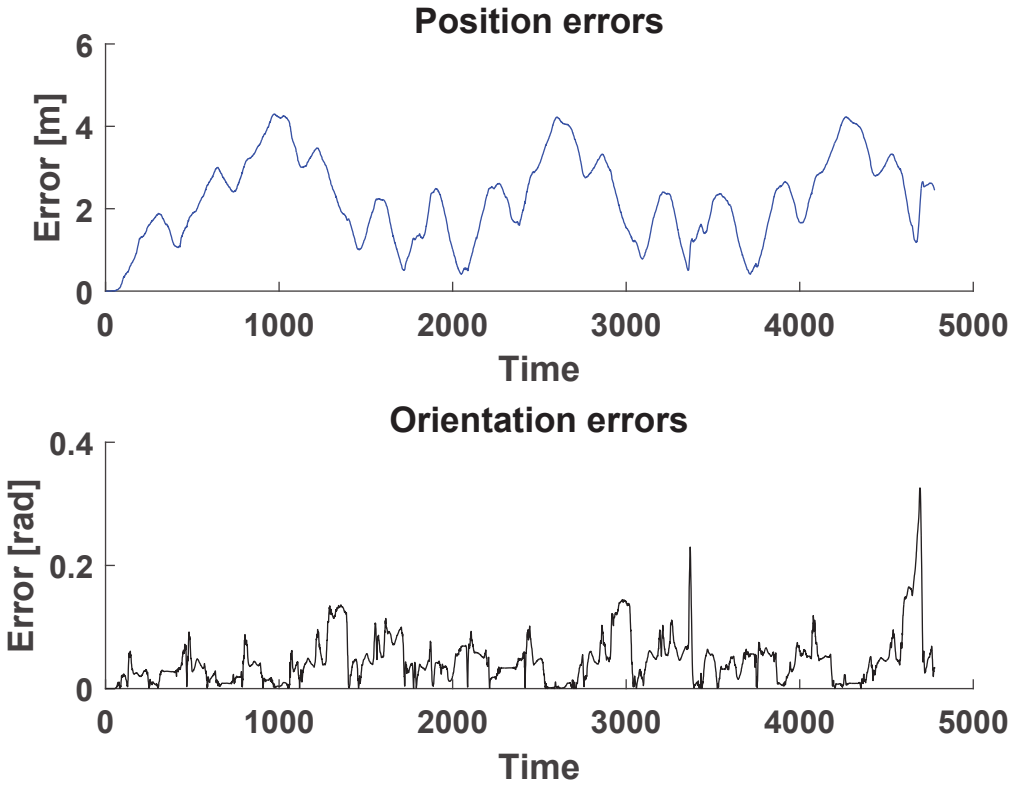



FIGURE 4.8 – Évolution des erreurs de trajectoire avec la méthode *Comp* dans le cas du bruit Std1 pour le jeu de données Gesling2

sont plus élevées que pour les autres pourcentages (20, 40, et 50%). L'explication principale à ce comportement provient du processus d'initialisation des balises. Quel que soit la méthode utilisée, le processus d'initialisation des balises est très sensible au bruit sur les mesures de distance et notamment en présence de mesures aberrantes. En effet, aucune des méthodes comparées ne possède un mécanisme permettant de sélectionner les "bonnes" mesures pour réaliser l'initialisation. Seules les méthodes d'initialisation *Grid* et dans une moindre mesure *Pf* seraient en mesure d'atténuer l'effet des mesures aberrantes de par leur nature probabiliste. Ainsi, comme on peut le voir aux figures 4.5 et 4.6 sur les erreurs de trajectoire et de position des balises, toutes les méthodes *Comp*, *Mog* (barres vertes) et *Trilat* (barres bleues) sont sensibles aux mesures aberrantes et ce quel que soit le pourcentage. On note cependant que pour les méthodes *Mog* et *Comp* qui utilisent respectivement une et deux mesures de distance pour initialiser des hypothèses dans le filtre EKF, les erreurs sont généralement plus faibles que pour la méthode *Trilat* nécessitant 5 mesures. On voit en effet sur la Figure 4.6, que les erreurs sur la position des balises sont à chaque fois très élevées (supérieures à 300 mètres) pour la méthode *Trilat*. Ceci indique une mauvaise initialisation et donc sûrement l'utilisation d'une ou plusieurs mesures aberrantes dans le calcul de trilatération permettant d'estimer la position de la balise. De cette observation, on peut ainsi déduire que plus les balises seront insérées tôt dans l'état du filtre, meilleures seront les chances pour le filtre de rejeter les mesures aberrantes.

En regardant la Figure 4.10c qui montre le numéro de l'itération à partir duquel toutes les balises sont insérées dans le vecteur d'état du filtre, on se rend bien compte que la méthode *Mog* est la plus rapide (itération 1), suivie de l'approche *Comp* (itération 60) et enfin de *Trilat* (itération 100). Ceci confirme bien que les méthodes nécessitent respectivement 1, 2 et 5 mesures de distance pour réaliser l'insertion des paramètres des balises dans le vecteur d'état du

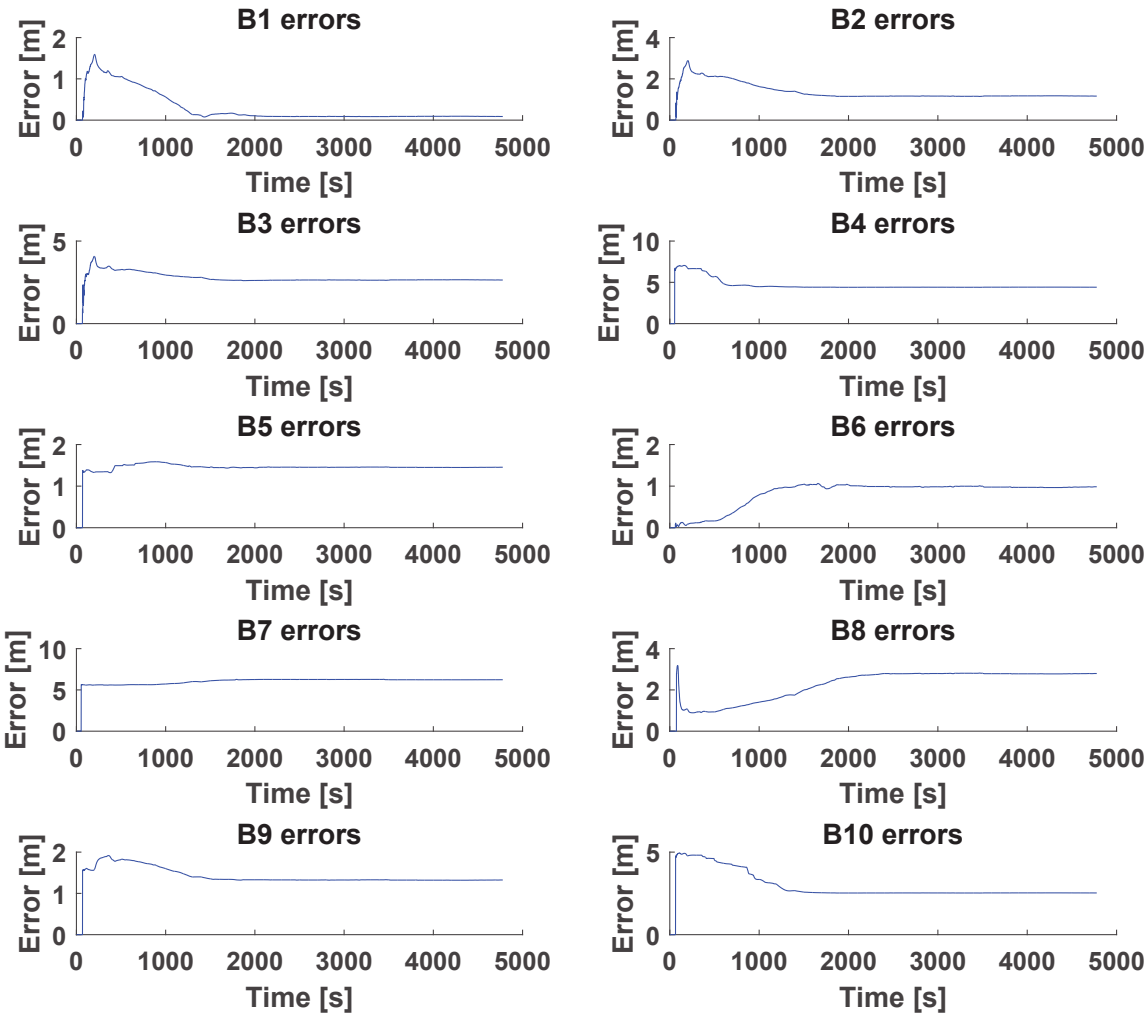


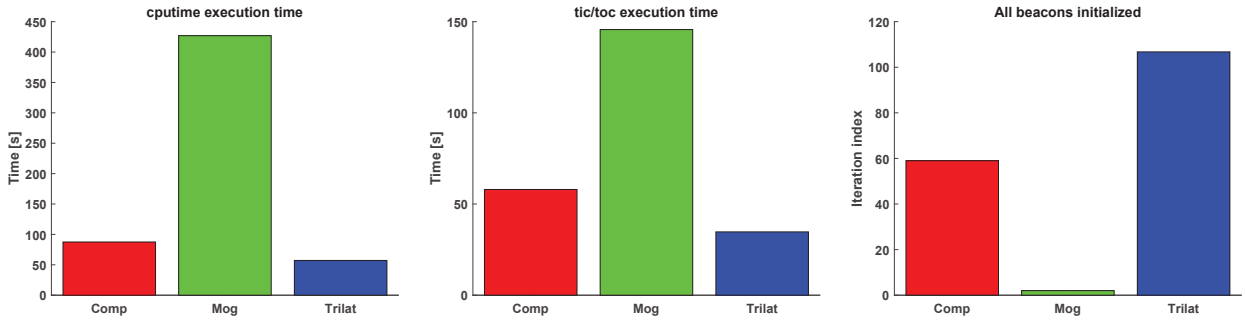

FIGURE 4.9 – Évolution des erreurs de position des balises avec la méthode *Comp* dans le cas du bruit Std1 pour le jeu de données Gesling2

filtre. Il faut cependant souligner que pour les méthodes *Mog* et *Comp*, les balises sont insérées dans le filtre sous forme d'une mixture de gaussiennes à plusieurs hypothèses, et qu'il faudra encore un certain nombre d'itérations avant la convergence vers une unique solution. L'itération correspondant à la convergence de toutes les balises pouvant alors être supérieure à l'itération de l'approche *Trilat* où toutes les balises sont initialisées. Cependant, comme vu précédemment avec les mesures aberrantes, il est préférable d'insérer le plus tôt possible les paramètres des balises dans le filtre afin de corréler au plus vite les variables et d'être plus robuste aux bruits des mesures, quitte à converger plus lentement par la suite.

Concernant le temps d'exécution des différentes méthodes, les résultats des commandes MatLab *cputime* et *tic/toc* sont regroupés aux figures 4.10a et 4.10b. Des deux figures, on constate que l'approche *Trilat* est la plus rapide, suivie de près par la méthode *Comp* et en dernier l'approche *Mog* qui est environ trois fois plus lente que la méthode *Comp*. Ces résultats confirment que plus la taille du vecteur d'état du filtre est élevée, plus l'algorithme prend du temps. La méthode *Trilat* ne nécessitant aucune hypothèse et insérant directement la position des balises en coordonnées cartésiennes dans le filtre, possède ainsi un vecteur d'état de taille minimale $[3 + 2 \times N]$, N étant le nombre de balises, tout au long de l'estimation. Ce qui n'est pas le cas pour les méthodes *Comp* et *Mog* qui utilisent respectivement 2 et 16 hypothèses. Ainsi, pour l'approche *Comp* le vecteur d'état a une taille de $[3 + 4 \times N]$ quand toutes les balises sont

insérées et qu'aucune des hypothèses n'a été supprimée, alors que pour l'approche *Mog*, la taille est de $[3 + 19 \times N]$. Les méthodes *Comp* et particulièrement *Mog* seront donc beaucoup plus lentes lors de la phase d'initialisation des balises et cela jusqu'à ce que toutes les mauvaises hypothèses soient supprimées. Une fois toutes les balises initialisées, la méthode *Comp* aura un vecteur d'état de taille minimale, $[3 + 2 \times N]$, comme pour la méthode *Trilat*, alors que la méthode *Mog* aura un vecteur d'état de taille légèrement plus élevée due à la paramétrisation en coordonnées polaires, $[3 + 4 \times N]$.

Avec ces premiers résultats issus de simulations, nous avons montré la validité de la méthode d'initialisation *Comp* proposée, ainsi qu'établit les premières comparaisons avec les méthodes *Trilat* et *Mog*. Nous allons maintenant confirmer les résultats sur un jeu de données réel et étendre la comparaison à d'autres méthodes d'initialisations.




(a) *cputime* (b) *tic/toc* (c) Itération balises insérées

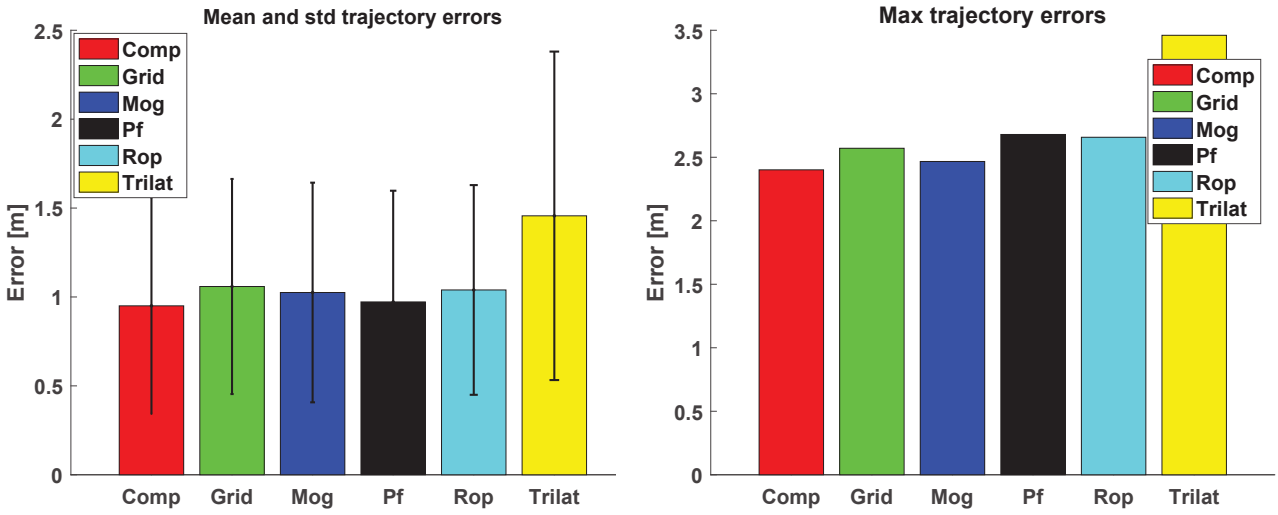
FIGURE 4.10 – Comparaison des temps d'exécution avec les commandes MatLab *cputime* (a) et *tic/toc* (b), et du numéro de l'itération à partir duquel toutes les balises sont insérées dans le filtre (c) pour le jeu de données Gesling2

### 4.4.2 Résultats des données expérimentales sur Plaza1

Afin de valider le fonctionnement en conditions réelles du filtre EKF et des différentes méthodes d'initialisation des balises, le jeu de données Plaza1 est utilisé avec les mesures du dispositif expérimental. Ici, les paramètres du filtre EKF ($\mathbf{Q}$, $\mathbf{R}$) sont déterminés au préalable en estimant les bruits sur l'odométrie et les mesures de distance par comparaison entre les mesures bruitées et la vérité terrain fournit par le jeu de données. À la Figure 4.11, on retrouve les erreurs moyennes, les écarts-types ainsi que les erreurs maximales de la trajectoire du robot qui sont comparés pour les six méthodes d'initialisation implémentées. La même chose est visible à la Figure 4.12 pour la position des balises. De plus, comme précédemment les résultats des temps d'exécution données par les commandes MatLap *cputime* et *tic/toc* ainsi que le numéro de l'itération pour lequel toutes les balises sont insérées dans le filtre sont regroupés sur la Figure 4.13.

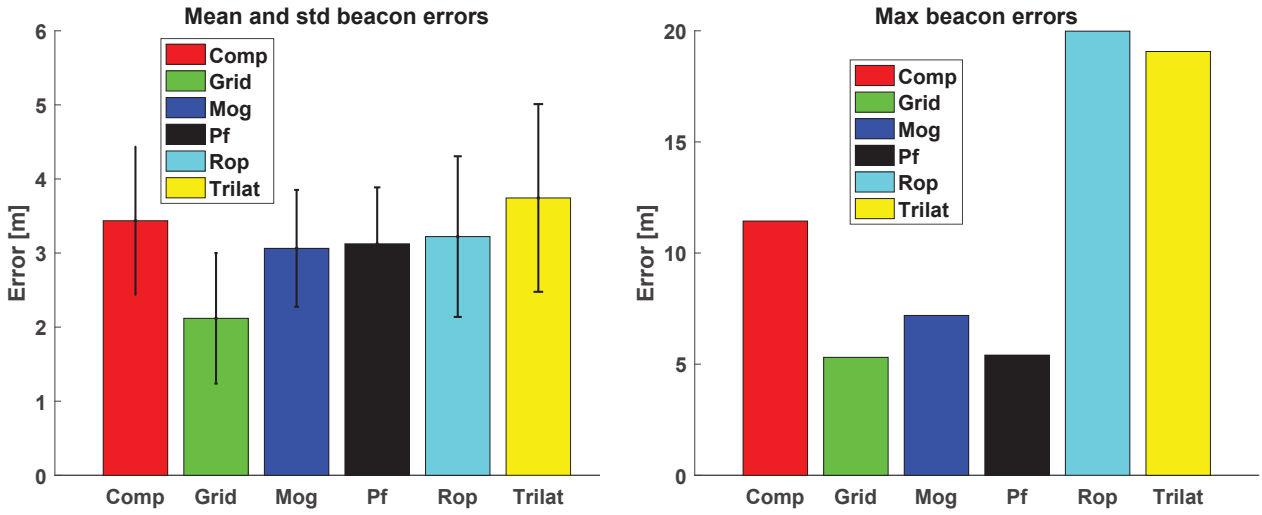
En comparant les valeurs sur la Figure 4.11, on remarque pour l'ensemble des méthodes que l'erreur de trajectoire moyenne se situe aux alentours de 1 mètre sauf pour la méthode *Trilat* où l'erreur est plutôt de 1.5 mètres. Le même constat est visible à la Figure 4.12 avec les erreurs de trajectoire maximales ou la méthode *Trilat* possède une valeur plus élevée d'environ 1 mètre que les autres méthodes, qui se situent aux alentours de 2.5 mètres.

Par contre, en comparant les résultats pour la position des balises, on constate une plus grande variation des erreurs en fonction des méthodes. Ainsi, la méthode *Grid* possède les erreurs moyennes et maximales les plus faibles, qui sont d'environ 2 et 5 mètres respectivement. Ensuite, viennent les méthodes *Mog* et *Pf* avec un léger avantage pour la méthode *Pf* au vu de




(a) Erreurs moyennes et écarts-types (b) Erreurs maximales

FIGURE 4.11 – Comparaison des erreurs moyennes, des écarts-types (a) et des erreurs maximales (b) de la trajectoire du robot pour les différentes méthodes d'initialisation sur le jeu de données Plaza1



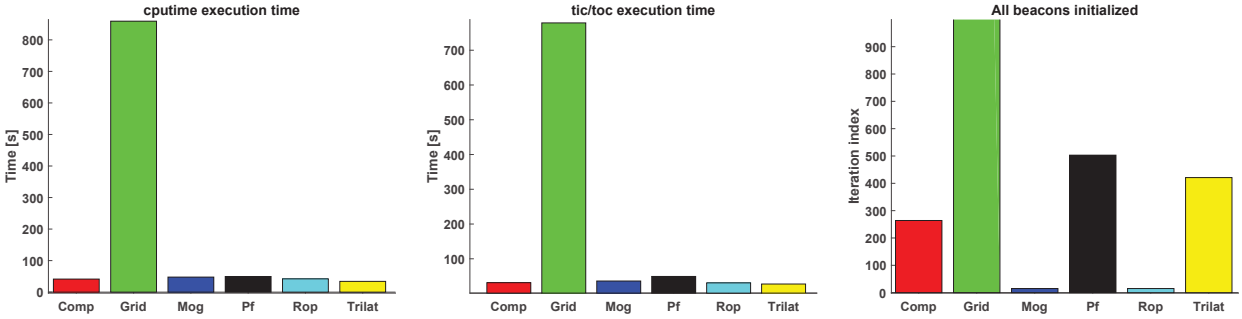

(a) Erreurs moyennes et écarts-types (b) Erreurs maximales

FIGURE 4.12 – Comparaison des erreurs moyennes, des écarts-types (a) et des erreurs maximales (b) de la position des balises pour les différentes méthodes d'initialisation sur le jeu de données Plaza1

son erreur maximale plus faible. Enfin, on retrouve les méthodes *Comp*, *Rop* et *Trilat* avec des erreurs moyennes comprises entre 3 et 4 mètres. Mais les deux dernières méthodes possédant une valeur maximale beaucoup plus élevée et proche de 20 mètres comparé à la méthode *Comp* où l'erreur maximale est d'environ 12 mètres.

Malgré une meilleure estimation de la position des balises, la méthode *Grid* est aussi la plus lente et celle insérant le plus tardivement toutes les balises dans le filtre, comme on peut le voir sur la Figure 4.13 avec les barres vertes. Pour les autres méthodes qui semblent assez proches au niveau du temps d'exécution, on remarque quand même que la méthode *Pf* est légèrement plus lente que les autres et qu'elle initialise toutes les balises seulement après la 500$^{\text{ème}}$ itération. Pour les méthodes *Comp*, *Mog* et *Trilat*, on retrouve les mêmes résultats que

pour les simulations présentées précédemment. C'est-à-dire, la méthode *Trilat* est la plus rapide en temps d'exécution, mais insère toutes les balises en dernier, alors que la méthode *Mog* insère le plus tôt toutes les balises, mais est plus lentes que les autres méthodes. La méthode *Comp* étant à chaque fois un compromis entre les deux.




(a) *cputime* (b) *tic/toc* (c) Itération balises insérées

FIGURE 4.13 – Comparaison des temps d'exécution avec les commandes MatLab *cputime* (a) et *tic/toc* (b), et du numéro de l'itération à partir duquel toutes les balises sont insérées dans le filtre (c) sur le jeu de données Plaza1

Le résultat final de l'estimation de la trajectoire du robot et de la position des balises avec la méthode *Comp*, est visible à la Figure 4.14 où l'on peut voir que la trajectoire estimée (en bleue) est beaucoup plus proche de la vérité terrain (rouge) que la trajectoire estimée par l'odométrie seule (verte). On s'aperçoit aussi que la position estimée des balises (bleue) se trouvent dans le voisinage des positions réelles des balises. Il est aussi possible de visualiser ces résultats sur la Figure 4.15 qui montre l'évolution des erreurs de trajectoire, d'orientation et de la position des balises. Ainsi, si visuellement les positions estimées des balises semblent proches de la vérité terrain, on s'aperçoit à la Figure 4.15b que les erreurs convergent quand même vers des valeurs de l'ordre de 3 à 5 mètres. On peut aussi noter que l'erreur sur la position de la balise est assez élevée au moment de l'initialisation, entre 5 et 12 mètres, mais que malgré tout cela la position estimée arrive à converger vers une position plus proche de la vérité terrain. En regardant l'erreur initiale pour la balise 1, qui est d'environ 12 mètres, on retrouve l'erreur maximale donnée à la Figure 4.12b pour la méthode *Comp*. Au niveau de la trajectoire et de l'orientation du robot visibles à la Figure 4.15a, on remarque une erreur plus élevée au début et à la fin de la trajectoire. La diminution des erreurs après un certain temps, correspond à la fin de l'initialisation des balises dans le filtre où chaque balise a convergé vers une seule hypothèse. En ayant une meilleure estimation de la position des balises, il est alors possible de localiser le robot plus précisément. On s'aperçoit aussi que vers l'itération 1400, l'erreur de position augmente brutalement, ce qui entraîne une augmentation progressive de l'erreur d'orientation du robot. Ce phénomène est sans doute dû à une mesure erronée plus importante que la moyenne et souligne la sensibilité de l'algorithme et le manque de robustesse à certains bruits.

De ces résultats expérimentaux, on peut donc conclure que la méthode d'initialisation *Grid* permet d'obtenir une meilleure estimation de la position des balises, mais pas forcément de la trajectoire du robot, en raison notamment du temps nécessaire pour insérer et initialiser l'ensemble des balises dans le vecteur d'état du filtre. Dans le cas d'un robot qui posséderait une moins bonne odométrie, il n'est pas sûr que cette méthode soit à nouveau la meilleure. De plus, son temps d'exécution qui dépend de la taille de l'environnement et de la résolution de la grille étant très important, cela peut devenir un facteur limitant dans le cas d'un filtre de Kalman fonctionnant en temps réel. Au vu des résultats obtenus aussi en simulation, on constate que la méthode d'initialisation *Trilat* est celle qui présente les moins bonnes performances en

termes de localisation du robot et d'estimation de la position des balises. De plus, l'emploi de l'algorithme de trilatération et l'attente de 5 mesures de distance rend cette méthode particulièrement sensible aux erreurs des mesures de distance et aux mauvaises configurations des positions du robot. Le seul avantage de cette méthode résidant dans sa rapidité d'exécution. Ensuite, au vu des expériences réalisées, les méthodes d'initialisation *Comp*, *Mog*, *Pf* et *Rop*, présentent des performances similaires. Chacune possédant des avantages et des inconvénients sur les autres méthodes. Ainsi, pour la méthode *Comp* proposée, on constate que même si les erreurs sur la trajectoire du robot et sur la position des balises sont comparables aux autres méthodes, cette dernière propose un compromis intéressant entre complexité de calcul/rapidité d'exécution et délai d'insertion des balises dans le vecteur d'état du filtre. Finalement, il est important de souligner qu'aucune des méthodes d'initialisation implémentées n'est vraiment robuste aux bruits des mesures de distance de distance et d'odométrie du robot. Les résultats finaux de l'estimation par le filtre dépendent encore beaucoup de la trajectoire du robot et des bruits affectant les mesures au cours de la phase d'initialisation des balises. De plus, suivant les scénarios, la mauvaise initialisation d'une seule balise peut être préjudiciable pour la suite de l'estimation du filtre.

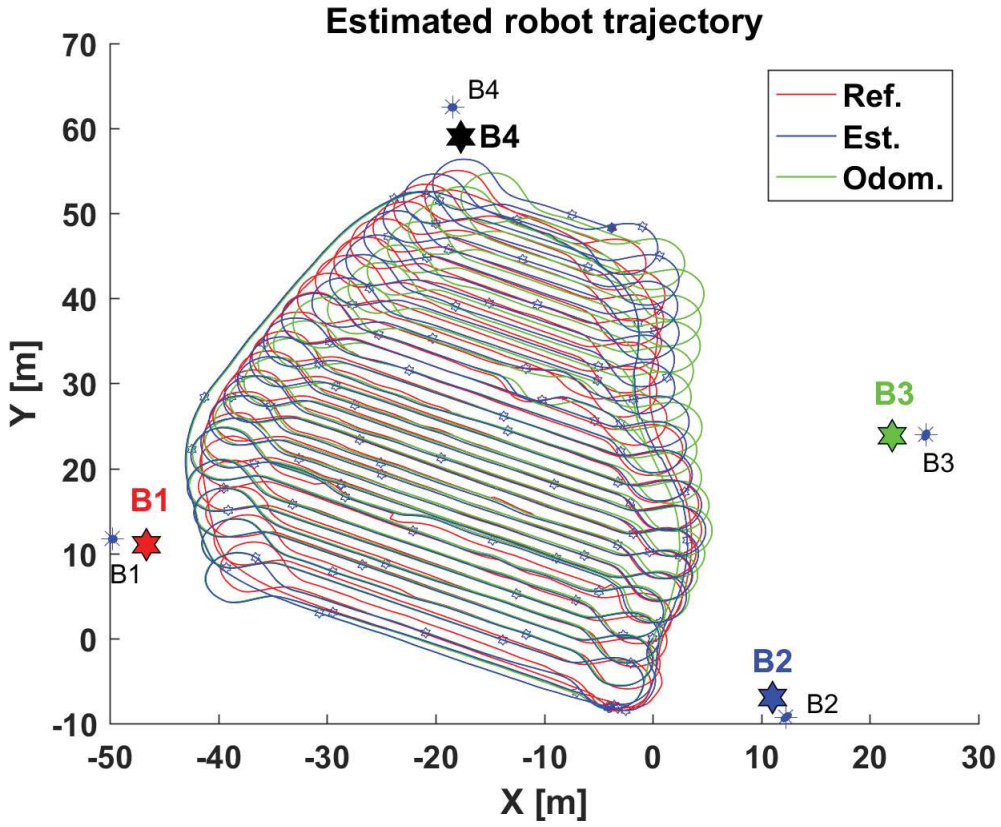
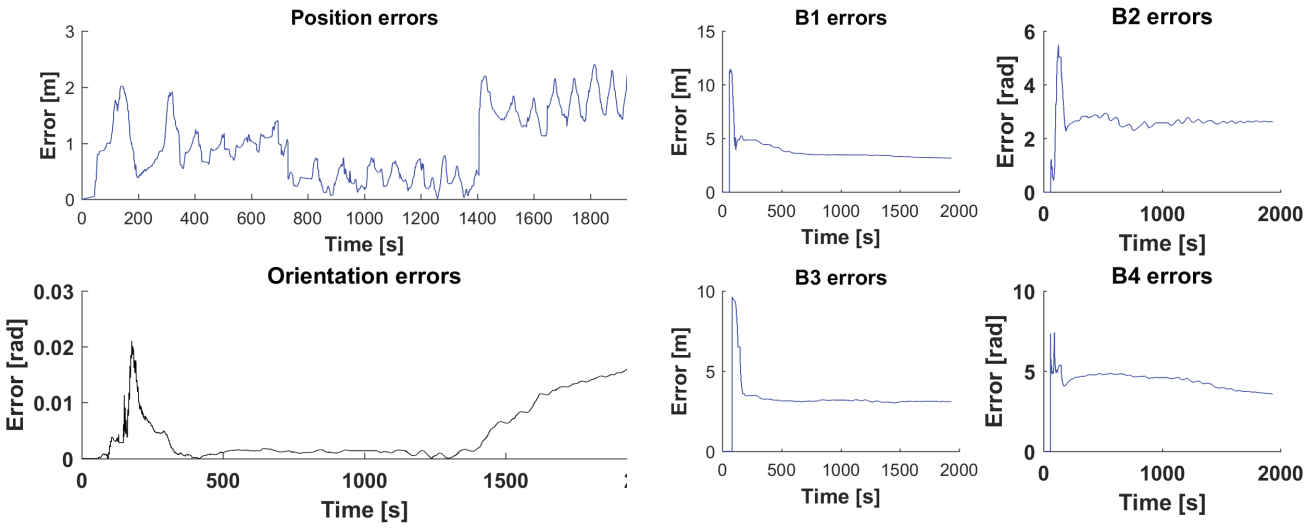



FIGURE 4.14 – Trajectoire du robot et positions des balises estimées par le filtre EKF avec la méthode *Comp* sur le jeu de données Plaza1




(a) Trajectoire et orientation (b) Balises

FIGURE 4.15 – Évolution des erreurs de trajectoire et d'orientation (a) et de position des balises (b) pour la méthode *Comp* sur le jeu de données Plaza1

## 4.5 Conclusion

Dans ce chapitre consacré au SLAM avec des mesures distance uniquement, un état de l'art exhaustif sur les différents algorithmes développés afin de résoudre le problème a été fait. L'accent a été plus particulièrement mis sur les différentes techniques permettant d'initialiser les paramètres correspondant aux balises dans le l'algorithme d'estimation.

Comme dans le cas de la localisation, un algorithme de fusion de données basé sur un filtre de Kalman étendu a été développé afin de résoudre le problème du RO-SLAM. Dans ce cadre-là, une nouvelle méthode d'initialisation de la position des balises dans le vecteur d'état du filtre est proposée.

Cette méthode basée sur le calcul de l'intersection de deux cercles à partir de deux mesures de distance, permet d'initialiser deux hypothèses pour la position d'une balise. Le choix d'utiliser une paramétrisation en coordonnées cartésiennes comparé à d'autres méthodes de la littérature permet d'obtenir un compromis entre le délai d'insertion des balises dans le filtre et le temps d'exécution/complexité de calcul de la solution.

Les expériences menées sur des simulations avec différents niveaux et types de bruits ainsi que sur des données réelles ont permis de valider le bon fonctionnement de l'approche proposée. De plus, en comparant l'approche proposée avec d'autres méthodes d'initialisation, il ressort qu'en terme de précision de localisation du robot et des balises, la plupart des méthodes présentent des performances similaires sauf pour la méthode utilisant un algorithme de trilatération qui est clairement déconseillée.

Même si les résultats présentés ici permettent de valider le bon fonctionnement des différentes méthodes d'initialisation, ces dernières restent assez sensibles aux conditions dans lesquelles la phase d'initialisation se déroule. Ainsi, l'un des principaux axes d'améliorations concerne la robustification des méthodes d'initialisation en réduisant la sensibilité aux mesures aberrantes. Il serait donc intéressant de développer un mécanisme permettant d'identifier les mesures aberrantes ou impropres à l'initialisation de la balise. Une autre piste serait de développer une méthode capable de détecter la mauvaise initialisation d'une balise, et de recommencer la procédure jusqu'à obtenir une bonne solution. Comme souligné dans l'état de l'art, un moyen d'obtenir une convergence plus rapide du filtre et une meilleure estimation consiste à exploi-

ter les mesures de distance inter-balises. Cette piste d'amélioration a l'avantage de nécessiter très peu de modifications aux niveaux des algorithmes. Enfin, en s'inspirant de l'utilisation croissante des algorithmes d'optimisation dans le cas du SLAM visuel, il serait intéressant de combiner un algorithme de fusion de données chargé d'obtenir une première estimation de la trajectoire du robot et d'initialiser la position avec un algorithme d'optimisation chargé d'améliorer l'estimation des différentes variables à partir d'un certain nombre de poses clés du robot et de la position des balises. En effet, dans le cadre de ce projet, le SLAM étant utilisé pour l'apprentissage des limites du terrain et de la position des balises, il important d'obtenir l'estimation la plus précise possible au cours de cette phase, afin que la localisation ultérieure du robot se déroule au mieux. Toute technique permettant ainsi de raffiner l'estimation, comme c'est le cas des algorithmes d'optimisation, est donc essentielle pour atteindre la meilleure précision possible.

Comme on vient de le dire, le bon déroulement de la phase d'apprentissage et l'obtention de l'estimation la plus précise possible de la position des balises est un facteur important pour obtenir une bonne localisation du robot lorsque celui-ci opérera dans sa zone. Cependant comme cela est développé dans le chapitre qui suit, le placement de la position des balises a aussi une influence sur la qualité de localisation du robot.

CHAPITRE 5

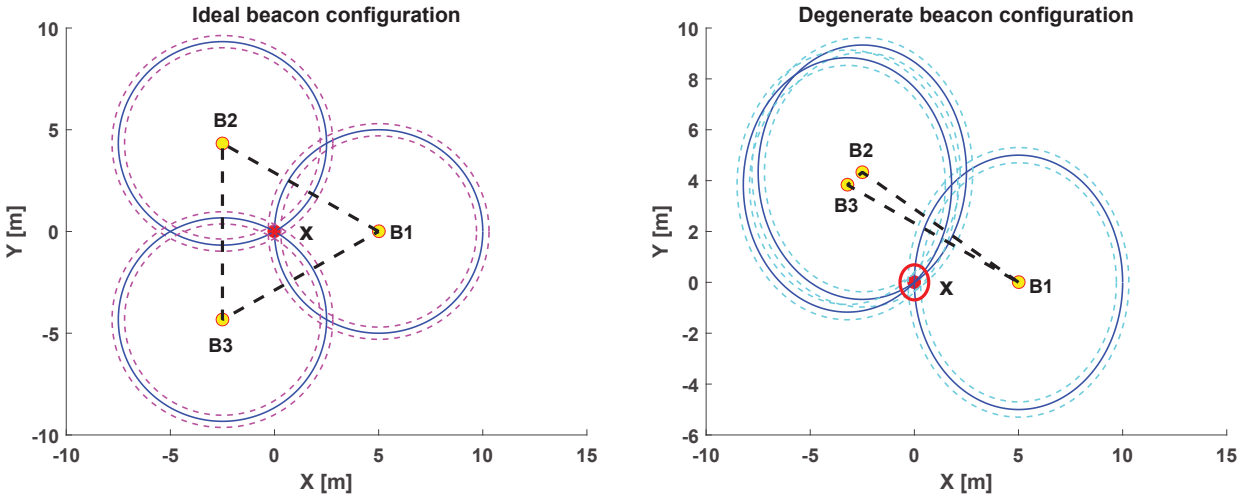
# PLACEMENT OPTIMISÉ DES BALISES

## 5.1 Introduction

Afin de faciliter la localisation d'un robot mobile, on a vu qu'il est possible de déployer des balises, fournissant des mesures de distance, à des endroits fixes de l'espace de travail du robot. D'un point de vue pratique, il est bien sûr avantageux de recourir à un nombre minimal de balises et cela pour des raisons de coût, de rapidité et de facilité d'installation du système. D'un autre côté, il est nécessaire de garder un nombre suffisant de balises afin de garantir une bonne couverture et une bonne localisation du robot sur l'ensemble de son environnement. Tout l'enjeu de ce chapitre est donc de déterminer combien de balises et où les positionner afin de satisfaire aux critères concurrents précités. Comme cela a déjà été observé dans les chapitres précédents sur la localisation et le SLAM, la qualité de la localisation du robot dépend aussi de la qualité du positionnement des balises.

Afin d'illustrer cela, deux configurations de trois balises utilisées pour estimer une position située à l'origine du repère par trilatération sont représentées à la Figure 5.1. Dans la première configuration, visible à la Figure 5.1a, la position à estimer se trouve au milieu du triangle inscrit par les balises. Alors que dans la seconde configuration de la Figure 5.1b, deux balises (B2 et B3) sont très proches l'une de l'autre. En traçant l'incertitude liée à l'estimation de position par trilatération (cercle rouge), calculée à partir de la formule (5.4) présentée plus loin, pour les deux configurations, on constate que l'incertitude de la deuxième configuration est plus élevée que pour la première. En effet, le cercle rouge de la Figure 5.1b est beaucoup plus grand que celui de la Figure 5.1a où ce dernier est à peine visible. On se rend ainsi compte que le positionnement des balises a une influence sur l'incertitude de localisation et qu'une configuration où la position à estimer est située à l'intérieur du triangle formé par les balises permettra d'obtenir une bonne localisation au contraire d'une configuration où deux balises sont très rapprochées l'une de l'autre.

Grâce à ces exemples, on s'aperçoit de l'importance du positionnement des balises sur la qualité de la localisation du robot. Dans la suite de ce chapitre, nous verrons qu'il est possible de relier la précision de la localisation avec le positionnement des balises. Outre l'importance du placement des balises sur le terrain pour garantir une bonne précision de localisation, on peut aussi facilement comprendre l'importance du placement pour obtenir une bonne couverture du terrain. Les balises ayant une portée maximale, et suivant la taille de la zone à couvrir, le positionnement des balises va permettre une bonne couverture du terrain et ainsi assurer qu'en chaque position du terrain le robot puisse accéder à un nombre désiré de mesures de distance par rapport aux balises.




(a) Bonne configuration (b) Mauvaise configuration

FIGURE 5.1 – Exemple de deux configurations l'une bonne (a) et l'autre mauvaise (b) pour estimer la position du point $\mathbf{x} = (0, 0)^T$ à partir de 3 mesures de distance par rapport aux balises $B_1$, $B_2$ et $B_3$ avec un algorithme de trilatération. Dans la mauvaise configuration, le cercle rouge d'incertitude est bien visible au contraire de la bonne configuration ou celui-ci est à peine visible.

Finalement, comme dans la plupart des applications, il est souvent souhaitable de réduire, voire d'éliminer totalement, l'utilisation de dispositifs annexes afin de faciliter le déploiement du système et de réduire le coût total. On cherchera ainsi dans notre cas à minimiser le nombre de balises utilisées pour couvrir et localiser le robot.

Au vu de ces trois arguments, on se rend compte de l'importance du choix du nombre et du positionnement des capteurs dans l'environnement du robot afin de maximiser les performances du système. Ce chapitre est consacré au développement de solutions permettant d'obtenir le placement des balises sur les bords de la zone à couvrir, en optimisant un critère défini à priori. Le placement des balises se ramène donc à résoudre un problème d'optimisation. Dans notre cas, il s'agira de résoudre un problème d'optimisation avec des fonctions de coût non-convexes, non-linéaires et non-dérivables. Pour ces raisons, l'algorithme employé pour la résolution du problème est basé sur un algorithme génétique (GA), [51].

L'approche proposée dans ce chapitre a pour avantage de fonctionner avec des environnements complexes, pouvant être de forme non-convexe, et contenir d'éventuels obstacles à l'intérieur. Comme dans le cadre de ce travail, la position des balises est restreinte aux bords extérieurs du terrain ainsi qu'aux bords des obstacles intérieurs, l'utilisation d'une paramétrisation en abscisse curviligne de la position des balises le long des bordures, permet de réduire le nombre de variables pour une balise (de deux à un). Afin de définir un bon placement, une fonction de coût permettant de contrôler la précision voulue en chaque position du terrain a été développée en obtenant une relation entre la précision de localisation et le déterminant de la matrice de Fisher. Enfin, après une évaluation des performances de différentes fonctions de coût sur différents environnements, une comparaison de l'impact du placement optimisé des balises sur deux types d'algorithmes de localisation (trilatération et filtre de Kalman Étendu) souligne les limitations des fonctions de coût actuelles basées sur la trilatération lorsque la localisation est effectuée avec un algorithme de filtrage.

Ce chapitre est organisé comme suit. Tout d'abord, un état de l'art sur le placement des capteurs est donné. Ensuite, la modélisation du problème utilisé dans le cadre de ce travail est présenté. Dans une troisième partie, l'algorithme génétique utilisé est brièvement présenté

ainsi que les expressions des différents critères à optimiser et les fonctions de coût. Dans une quatrième partie, les résultats obtenus pour le placement des balises sont présentés. Puis, les limitations des fonctions de coût basées sur la trilatération sont exposées ainsi que les simulations correspondantes. Finalement, une conclusion sur le placement des balises ainsi que des perspectives d'améliorations clôturent le chapitre.

## 5.2 État de l'art

Pour résoudre le problème d'optimisation du placement de capteurs, il est nécessaire de définir quels sont les coûts à minimiser. Laguna et al. proposent trois critères [78] :

— Minimiser le nombre de balises,

— Maximiser la couverture de l'espace de travail du robot,

— Maximiser le pourcentage de l'aire couverte avec des valeurs de dilution de précision (DOP) admissibles.

Le dernier point peut être reformulé d'une façon plus générale en : maximiser les performances du système de localisation sur l'ensemble de l'espace de travail du robot. Ces trois critères peuvent être combinés dans un seul objectif qui peut se définir comme suit : couvrir entièrement l'espace de travail avec suffisamment de mesures pour s'assurer qu'en chaque position du terrain, la précision de localisation soit maximisée ou atteigne un critère prédéfini et ceci en utilisant le moins de capteurs (balises) possible. Suivant les besoins, ces critères peuvent être minimisés séparément ou combinés de différentes manières. L'idéal serait bien sûr de résoudre tous les critères simultanément. Cependant, certains critères étant contradictoires, trouver une formulation les combinant tous s'avère difficile, et il est bien souvent nécessaire de faire des compromis.

### 5.2.1 Critères de couverture

Un critère essentiel pour le placement de capteurs est la disponibilité d'un nombre suffisant de mesures en chaque position du terrain pour permettre de localiser correctement le robot. Pour cela, Allen et al. [1] proposent une fonction d'utilité de localisation permettant de quantifier le bénéfice d'un placement de balises ultrasons en 2D dans le cas où le chemin du robot est connu à l'avance. La fonction d'utilité attribue un score dépendant du nombre de mesures de distance à une position donnée, mais aussi de la géométrie du placement des balises. Le Tableau 5.1 présente les différents scores attribués par la fonction. Le "Well-heard" et le "Well-seen" correspondent tous les deux a au moins 3 balises disponibles. La différence étant que le second présente une géométrie du placement des balises plus favorable. En notant $u_i$ la fonction d'utilité calculée à la position $\mathbf{p}_i = (x_i, y_i)^T$, le problème d'optimisation à résoudre est alors défini par :

$$\max \frac{1}{N} \sum_{i=1}^{N} u_i$$

avec $N$ le nombre de positions $\mathbf{p}_i$ dans l'espace de travail du robot. Différents algorithmes sont proposés dans [1] pour résoudre le problème d'optimisation, dont un algorithme d'escalade (hill-climbing), de recuit simulé (simulated annealing), de descentes de coordonnées (coordinate descent) et de glouton (greedy heuristic).

| Coverage | Localization utility |
|---|---|
| 0 beacon | 0 |
| 1 beacon | 1 |
| 2 beacons | 2 |
| Well-heard | 3 |
| Well-seen | 5 |

TABLEAU 5.1 – Fonction d'utilité de localisation attribuant un score à une position donnée en fonction du nombre de balises et de la géométrie du placement des balises (reproduit d'après [1])

### 5.2.2 Critères de précision de localisation

Dans le cas de la localisation par mesures de distance, l'un des algorithmes le plus souvent utilisé est celui de la trilatération. Pour cette raison, ce dernier est à l'origine de différents critères de précision de localisation. Ainsi, dans Burke at al. [15], l'incertitude moyenne de la position estimée par l'algorithme de trilatération est utilisée comme critère à minimiser pour résoudre le problème du placement des balises. Les incertitudes des mesures de distance sont propagées grâce aux équations de l'algorithme de trilatération pour obtenir l'incertitude sur la position estimée : $\mathrm{cov}(x, y, z)$. L'erreur de mesure de distance est modélisée par un bruit blanc Gaussien de moyenne nulle et dont la variance dépend de la distance. Le critère d'erreur en position proposé, $e$, est défini comme la racine carrée de la somme des valeurs propres de la matrice d'incertitude en position (matrice de covariance de l'erreur en position) :

$$e = \sqrt{\sum_{i=1}^{3} \mathrm{Eig}\Big[\mathrm{cov(x, y, z)}\Big]}$$

La fonction de coût qui est minimisée est alors donnée par la moyenne de $e$ sur l'ensemble de l'espace de travail du robot. Le problème d'optimisation non-convexe est résolu avec l'algorithme du simplexe de Nelder-Mead. Les résultats obtenus pour deux environnements, l'un avec 4 balises et sans obstacle, l'autre avec 8 balises et 5 obstacles montrent une réduction de l'incertitude moyenne en position de 14% et 65% respectivement après optimisation. Ceci montre bien l'intérêt d'un algorithme de positionnement des balises, surtout dans le cas où l'environnement devient plus complexe. Grâce au calcul de l'erreur en chaque position du terrain, il est possible d'obtenir une carte de précision de localisation. Cette carte permet alors de facilement visualiser les performances de localisation sur l'ensemble du terrain et ainsi d'identifier des zones plus ou moins critiques. Outre la visualisation, Burke at al. [15] présentent un autre avantage de cette carte. Dans leurs travaux, cette dernière est utilisée comme heuristique pour améliorer un algorithme de planification de chemin et obtenir ainsi un chemin bénéficiant d'une bonne localisation du robot.

La dilution de précision (DOP) [136], utilisée couramment pour le système GPS a aussi été employée comme critère à minimiser dans le cas du problème de placement de capteurs. La dilution de précision géométrique en position (PDOP), dont le calcul sera détaillé plus loin, est définie comme le ratio entre la racine carrée de la somme des variances de l'incertitude en position suivant les axes $x$, $y$ et $z$ et la déviation standard de l'incertitude des mesures de distance. Ce critère fournit une information sur l'effet de la configuration géométrique des balises sur la précision de l'estimation de position. Ainsi, plus basse sera la valeur de PDOP, et meilleure est censée être la précision de la localisation.

Dans [22], il est démontré que la dilution de précision géométrique (GDOP) est un minorant de la borne inférieure de Cramer Rao (CRLB) si les erreurs des mesures de distance sont modélisées par un bruit blanc Gaussien de moyenne nulle et de même variance pour chaque

mesures.

Ainsi dans [74], la GDOP est utilisée pour caractériser l'incertitude introduite par le positionnement des capteurs. Deux formulations sont proposées pour résoudre le problème d'optimisation, l'une discrète et l'autre continue. L'approche discrète est alors résolue en utilisant des algorithmes de Binary Integer Programming et de Mixed Integer Programming, alors que l'approche continue est résolue avec un algorithme de Nonlinear Programming.

Rao et al. [113] proposent quant à eux de combiner la dilution de précision avec un terme de non-disponibilité des mesures de distance, pour créer une fonction de coût composée de deux termes pondérés. Le premier terme est défini par la moyenne des valeurs de DOP sur l'ensemble de l'espace de travail : $\overline{\mathrm{DOP}}$. Le second terme correspond au pourcentage de l'aire de l'espace de travail où aucune mesure de distance n'est disponible : NAR. La fonction de coût complète à minimiser est ainsi donnée par :

$$\alpha_1 \overline{\mathrm{DOP}} + \alpha_2 \mathrm{NAR}$$

avec $\alpha_1$ et $\alpha_2$ les pondérations permettant de régler d'influence de chacun des deux termes. Le problème de minimisation est alors résolu avec un algorithme d'optimisation méta-heuristique appelé diversified local search (DLS).

Plus récemment, Rajagopal et al. [112] ont proposé une version améliorée de la GDOP incorporant le principe de localisation unique, et qui permet dans certaines conditions, de fournir une position à partir de deux balises seulement. La méthode est présentée dans le cadre de la localisation en intérieur où un plan du bâtiment est disponible à priori. Le placement des balises optimisant le critère proposé est obtenu en utilisant un algorithme de recherche exhaustive.

Lorsque le modèle utilisé pour les mesures de distance devient plus élaboré (paramètres différents pour chaque balise, variance dépendante de la distance ...), le critère de précision de localisation est généralement dérivé à partir de la matrice d'information de Fisher (FIM). L'expression générique de la FIM est donnée par [94] :

$$\mathbf{FIM} = \mathrm{E}\Big[\Big(\frac{\partial \ln \mathrm{p}(\mathbf{r}|\mathbf{p},\mathbf{b})}{\partial \mathbf{p}}\Big)^{\mathrm{T}}\Big(\frac{\partial \ln \mathrm{p}(\mathbf{r}|\mathbf{p},\mathbf{b})}{\partial \mathbf{p}}\Big)\Big]_{\mathbf{p}} \tag{5.1}$$

avec $\mathbf{r}$, le vecteur des mesures de distance, $\mathbf{p}$, la position à estimer, $\mathbf{b}$, le vecteur des positions des balises correspondant aux mesures. $p(\mathbf{r}|\mathbf{p},\mathbf{b})$ est définie comme la probabilité d'obtenir les mesures $\mathbf{r}$ sachant les positions des balises $\mathbf{b}$ et la position $\mathbf{p}$ à estimer. Enfin, E, représente l'opérateur d'espérance mathématique. Avec les hypothèses appropriées, la matrice d'information de Fisher correspond à l'inverse de la borne inférieure de Cramer-Rao, c'est-à-dire la borne inférieure de la matrice de covariance de l'estimateur de position non-biaisé [94]. Comme la minimisation de la CRLB entraîne la diminution de l'incertitude de l'estimation, la FIM, à l'inverse, doit être maximisée.

Ainsi, parmi les travaux basés sur la FIM, Jourdan et al. [69] utilisent un algorithme de descente de coordonnées appelé RELOCATE pour minimiser la moyenne de la borne de l'erreur de position (PEB) sur un espace de travail 2D. La PEB est définie comme la racine carrée de la trace de l'inverse de la matrice d'information de Fisher :

$$\mathrm{PEB} = \sqrt{\mathrm{trace}\big(\mathbf{FIM}^{-1}\big)}$$

L'algorithme proposé fonctionne en optimisant la position d'un capteur à la fois, jusqu'à convergence de la fonction à minimiser. Les mesures de distance sont modélisées avec un biais et un bruit blanc Gaussien de moyenne nulle dont la variance dépend de la distance. Cette dernière étant modélisée par l'équation :

$$\sigma^2(d) = \sigma_0^2 d^{\alpha}$$

avec $\alpha \geq 0$, l'exposant d'atténuation, et $\sigma_0^2$ la variance évaluée pour une distance de un mètre. Le nombre de capteurs est fixe durant l'optimisation et ceux-ci sont contraints aux bords de l'espace de travail considéré comme convexe. L'algorithme est aussi capable de traiter d'éventuels obstacles situés à l'intérieur. De plus, le bénéfice de l'algorithme proposé est aussi comparé par rapport à un placement uniforme et un placement aléatoire, dans le cas où le chemin du robot dans l'environnement est connu.

Dans Martínez et Bullo [94], le déterminant de la FIM, $\left|\mathrm{FIM}\right|$, est utilisé comme critère de précision de localisation pour le placement de capteurs dans le contexte du suivi de cibles. Les expressions du déterminant de la FIM sont données dans les cas 2D et 3D, en considérant que les mesures de distance sont affectées par une erreur modélisée avec un bruit blanc Gaussien de moyenne nulle et de variance identique pour toutes les mesures.

Bishop et al. [9] ont analysé et identifié les géométries du positionnement des capteurs permettant de minimiser l'ellipse d'incertitude associée à l'estimation de position. Pour ce faire, le déterminant de la FIM est aussi utilisé, et son expression est calculée dans le cas 2D. Parmi les résultats fournis, il est notamment montré que pour deux balises, le placement optimal est unique, et intervient quand l'angle sous-tendu par les deux balises et la position à estimer est égal à $\frac{\pi}{2}$.

Moreno-Salinas et al. [98] étudient le problème du placement de capteurs à la surface de l'océan afin de localiser un véhicule sous-marin autonome (AUV), dans le cas où la position de celui-ci n'est pas connue précisément. L'erreur sur la mesure de distance est modélisée par un bruit blanc Gaussien de moyenne nulle et de variance qui dépend de la distance :

$$\sigma^2(d) = (1 + \eta d)^2 \sigma_0^2 \tag{5.2}$$

avec $\eta$, le paramètre de dépendance à la distance, et $\sigma_0^2$, la variance du bruit de mesure. La configuration géométrique optimale des capteurs est alors obtenue en maximisant le déterminant de la FIM, dont l'expression est donnée dans le cas 3D par :

$$|\mathbf{FIM}| = \frac{(1 + 3\eta^2)}{\sigma^2} \sum_{\mathrm{i} \leq \mathrm{j} \leq \mathrm{k}}^{\mathrm{n}} \frac{\left[(\mathrm{v_i} \times \mathrm{v_j}).\mathrm{v_k}\right]^2}{(1 + \eta \mathrm{r_i})^2 (1 + \eta \mathrm{r_j})^2 (1 + \eta \mathrm{r_k})^2} \tag{5.3}$$

avec :

$$v_l = \Big(\frac{(x_l - x)}{r_l}, \frac{(y_l - y)}{r_l}, \frac{(z_l - z)}{r_l}\Big)^T, \quad l \in [1, n]$$

$(x_l, y_l, z_l)^T$ étant la position de la balise $l$, $(x, y, z)^T$ la position du véhicule et $r_l$ la mesure de distance entre le véhicule et la balise $l$. Cette formulation sera employée dans ce travail pour réaliser les différentes fonctions de coût utilisant ce critère.

Perez-Ramirez et al. [109] étudient le placement optimal de capteurs hétérogènes pour la localisation d'un véhicule autonome guidé (AGV). Le critère optimisé est le déterminant de la FIM moyenné sur des régions élémentaires de l'espace de travail du robot. Le modèle des mesures de distance est le même que celui proposé dans [69]. Aux différences que le biais n'est pas pris en compte, et que les paramètres du modèle sont différents pour chaque capteur. Avec ces hypothèses, l'expression du déterminant de la FIM est donnée pour les cas 2D et 3D. Le problème d'optimisation est défini avec une formulation *minimax* permettant une optimisation sur l'ensemble du terrain en maximisant le critère de la région possédant le plus faible critère. L'espace de recherche des balises est considéré comme convexe, et une technique de lissage exponentiel est employée pour résoudre le problème.

Enfin, un autre critère de précision de localisation appelé probabilité d'erreur circulaire (CEP) est proposé dans [111]. Le CEP est défini comme le rayon du cercle ayant pour centre

la moyenne et contenant la moitié des réalisations d'un vecteur aléatoire. Le CEP peut être approximé par la formule suivante :

$$\text{CEP} = 0.75\sqrt{\sum_{i=1}^{D}\sigma_i^2} \tag{5.4}$$

avec $D = 2, 3$ la dimension du vecteur position à estimer et $\sigma_i^2$ la variance de la $i^{\text{ème}}$ coordonnée estimée.

Dans le cadre de ce travail, un algorithme génétique [51] est utilisé pour résoudre le problème d'optimisation. Ce choix a été fait pour plusieurs raisons. Tout d'abord, à cause de la complexité du problème, et de la forte non-linéarité des fonctions de coût à optimiser. Ensuite, pour sa capacité à optimiser des critères concurrents. Parmi les critères de précision de localisation présentés ci-dessus, le choix du déterminant de la FIM a été fait, notamment pour pouvoir utiliser un modèle des mesures de distance plus élaboré. Avant de présenter en détails les fonctions de coût développées, nous allons d'abord introduire, dans les sections suivantes, la modélisation du problème ainsi que la formulation de l'algorithme génétique utilisé.

## 5.3 Modélisation du problème du placement des balises

Dans cette section, la modélisation du problème du placement des balises est présentée. Dans un premier temps la modélisation de l'espace du travail est décrite, puis le modèle des mesures de distance est présenté ainsi que les différentes hypothèses utilisées pour le fonctionnement et la réalisation de l'algorithme.

### 5.3.1 Modélisation de l'espace de travail

Le problème considéré est celui du positionnement d'un ensemble de balises fournissant des mesures de distance par rapport à un robot mobile dans le but de localiser ce dernier dans son espace de travail. Pour des raisons de simplicité, l'espace de travail est ramené à un environnement à deux dimensions, en considérant que l'antenne du robot et des balises sont à la même hauteur ou que les différences de hauteur sont connues et déjà compensées.

L'espace de travail du robot est modélisé par un polygone (pas nécessairement convexe) représentant les bordures extérieures, ainsi que par d'éventuels obstacles (aussi modélisés par des polygones) se trouvant à l'intérieur.

Afin de calculer les différents critères à optimiser, l'espace de travail est discrétisé avec une grille carrée, dont la taille des cellules peut être ajustée en fonction de la précision de localisation souhaitée et des capacités de calcul. Les différentes fonctions de coût seront donc évaluées pour chaque point (milieu de cellule) de la grille se trouvant dans l'espace atteignable par le robot, c'est-à-dire à l'intérieur du polygone des bordures extérieures et en dehors des polygones des obstacles.

### 5.3.2 Modélisation des mesures de distance

Dans le problème considéré, on cherche à positionner un nombre $N$, variable ou fixe, de balises dont les positions sont données par : $\mathbf{p}_i = (x_i, y_i)^T$, $i \in [1, N]$, de manière à localiser au mieux un robot mobile dont sa position est notée $\mathbf{p} = (x, y)^T$. Dans ce travail, un seul type de balise est considéré. Chacune permettant d'obtenir des distances en mesurant le temps de vol d'une onde radio UWB. Le même modèle de mesure est donc utilisé pour toutes les balises,

mais en s'autorisant la possibilité d'attribuer un niveau de bruit propre à chaque balise. Une mesure de distance entre le robot et la balise $i$ est donc modélisée par l'équation :

$$r_i = d_i + b_i + \epsilon_i$$

avec $r_i$ la distance mesurée, $d_i = \sqrt{(x_i - x)^2 + (y_i - y)^2}$ la vraie distance, $b_i$ le biais supposé nul dans la suite, et $\epsilon_i \sim \mathcal{N}(0, \sigma_i^2(d_i))$ un bruit blanc Gaussien de moyenne nulle et de variance $\sigma_i^2(d_i)$ qui dépend de la distance. La dépendance en fonction de la distance de la variance est modélisée selon la formule proposée dans [98], et est donnée par l'équation (5.2). Les balises possèdent aussi une portée maximale, notée $r_{i,max}$, limitant la disponibilité des mesures dans un cercle de rayon $r_{i,max}$ centré sur la position de la balise.

Afin de simplifier le problème, seules les mesures de distance ayant une ligne de vue directe (LOS) entre le robot et la balise sont comptées et utilisées pour le calcul de l'estimation de position. Plus simplement, si un obstacle se trouve sur la ligne de visée directe entre le robot et la balise, la mesure provenant de la balise n'est pas considérée. Cette hypothèse est plus ou moins réaliste suivant la technologie des balises. Par exemple, pour des balises ultrasons cette hypothèse est parfaitement justifiée car les ondes ultrasons ne peuvent pas traverser les obstacles. Par contre, pour des balises RF, comme dans le cas présent, les obstacles n'empêchent pas les ondes radios de se propager. Par contre, suivant le type de matériau traversé, cela va entraîner un ralentissement plus ou moins important du signal, ce qui aura pour conséquence de biaiser la mesure. Cependant la modélisation de ce biais est complexe et dépend du matériau de l'obstacle rencontré. Pour cette raison, toutes les mesures qui ne sont pas en ligne de vue directe sont éliminées dans nos simulations.

### 5.3.3 Hypothèses de modélisation

Dans le cadre de ce travail, on considère que la position des balises est restreinte aux bords extérieurs du terrain ainsi qu'aux bords des obstacles intérieurs. Une explication simple à cette hypothèse est que le robot va opérer dans la zone de travail, et ne doit donc pas être gêné par une balise se trouvant au beau milieu du terrain. En tenant compte de cette hypothèse, le périmètre extérieur et l'ensemble des périmètres des obstacles sont mis bout à bout afin de représenter la position d'une balise par une abscisse curviligne normalisée, $s_i \in [0, 1[$. On peut alors définir une transformation réciproque entre l'abscisse curviligne, $s_i$, et la position, $\mathbf{p}_i$, de cette balise. Cette nouvelle paramétrisation de la position permet de réduire le nombre de variables d'une balise de deux $\mathbf{p}_i = (x_i, y_i)^T$ à un $(s_i)$, ce qui a pour conséquence directe de réduire le nombre de variables du problème d'optimisation à résoudre. Précisons cependant que cette représentation comporte des discontinuités spatiales dans la position des balises quand l'espace de travail contient des obstacles. En effet, des discontinuités apparaissent au passage d'un périmètre à l'autre, par exemple lorsque l'on quitte le périmètre extérieur pour passer à celui du premier obstacle. Du fait de l'introduction de ces discontinuités, il apparaît d'autant plus intéressant d'utiliser un algorithme génétique afin de résoudre le problème d'optimisation.

Afin d'améliorer la répartition des balises sur le terrain et d'éviter certaines configurations singulières comme dans le cas de balises alignées, il est possible d'introduire des contraintes sur la position relative des balises dans le problème d'optimisation. On peut par exemple imposer :

— Une distance minimale entre deux balises : $d(B_i, B_j) = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \geq d_s$,

— Une distance minimale entre les coordonnées de deux balises : $|x_j - x_i| \geq x_s$, $|y_j - y_i| \geq y_s$.

Il faut toutefois noter que l'ajout de telles contraintes dans le problème d'optimisation peut avoir un impact non-négligeable sur le temps d'exécution.

Maintenant que le problème est modélisé, l'algorithme génétique utilisé pour résoudre le problème d'optimisation du placement des balises est présenté dans la section suivante.

## 5.4 Algorithme génétique et fonctions de coût

Dans cette partie, l'algorithme génétique utilisé pour résoudre le problème d'optimisation du placement des balises, puis les différentes fonctions de coût à minimiser sont présentées en détails.

### 5.4.1 Choix de l'algorithme d'optimisation

Il existe un grand nombre d'algorithmes permettant de résoudre un problème d'optimisation. Le choix d'un algorithme est généralement fortement influencé par la formulation du problème considéré, et plus précisément par la fonction à optimiser. Les principaux critères de choix concernant la fonction de coût sont la :

— Linéarité,

— Convexité,

— Dérivabilité,

— Continuité.

Ainsi dans le cas où la fonction de coût est convexe, on s'orientera vers des algorithmes d'optimisation convexe (moindres carrées, programmation linéaire, programmation quadratique, optimisation conique, programmation semi-définie positive, SOCP...).

Si la fonction est continue et dérivable, il est possible d'utiliser des algorithmes basés sur le calcul de la dérivée (Jacobien, Hessien) pour leur rapidité de convergence :

— Newton,

— Gauss-Newton,

— Quasi-Newton,

— Gradient,

— Gradient conjugué,

— Levenberg-Marquardt,

— Régions de confiance.

Il existe également des algorithmes d'optimisation sans calcul de dérivées :

— Méthode de Nelder-Mead,

— Descente par coordonnées (coordinate descent).

Dans les cas plus complexes où la fonction de coût n'est pas linéaire, convexe, dérivable ni continue, il existe une classe d'algorithme d'optimisation, appelée métaheuristique, permettant d'obtenir une solution approchée au problème avec moins de risques de converger vers un minimum local. Parmi ces algorithmes on retrouve notamment :

— Algorithmes évolutionnaires dont font partie les algorithmes génétiques,

— Recuit simulé,

— Algorithme de colonies de fourmis,

— Algorithme par essaims particulaires,

— Recherche tabou,

— Méthode GRASP.

Comme dans notre cas les fonctions de coût développées ne sont ni linéaires, ni convexes, ni dérivables, ni continues, c'est donc un algorithme de type métaheuristique qui a été choisi afin de résoudre le problème d'optimisation. Plus précisément, le choix s'est porté sur un algorithme génétique.

### 5.4.2 Algorithme génétique

Afin de résoudre le problème d'optimisation non-linéaire du placement des balises, un algorithme génétique a été développé en C++. Cette classe d'algorithme a pour objectif de trouver assez rapidement une bonne solution à un problème se posant comme une fonction à minimiser. De plus, ce type d'algorithme est particulièrement efficace quand l'ensemble des solutions est de grande dimension et quand la fonction à minimiser est fortement non-linéaire. Les méthodes classiques de type descente de gradient sont inefficaces dans ces situations, car souvent l'expression du gradient n'est pas disponible, et par ailleurs les minimums locaux sont très nombreux.

Dans notre cas, l'algorithme nécessite comme arguments d'entrée, la description de l'espace de travail du robot, la fonction de coût à optimiser, ainsi que du nombre de balises. Nous supposons ici que le nombre de balises est fixé à l'avance, et ne fait pas partie des variables à optimiser.

Le principe de l'algorithme génétique est basé sur la théorie de l'évolution, avec l'idée de la survie des individus les mieux adaptés et de leur amélioration génération après génération via des croisements entre les individus les plus adaptés. En pratique, un individu est ici une solution potentielle au problème, c'est-à-dire l'ensemble des positions de $N$ balises représentées avec leurs abscisses curvilignes croissantes. Les gènes d'un individu sont donc codés par $N$ nombres croissants entre 0 et 1.

Un algorithme génétique commence par initialiser une population de taille donnée $N_{pop}$ avec des individus générés aléatoirement ou de façon uniforme. Ensuite, 4 étapes se succèdent à chaque itération :

1. Test du critère d'arrêt : vérifier si l'algorithme a convergé,
2. Évaluation de la fonction de coût pour les individus de la population courante,
3. Sélection de $N_{pop}/2$ individus qui resteront dans la nouvelle génération,
4. Création de $N_{pop}/2$ individus par croisement et mutation des individus sélectionnés, afin de compléter la nouvelle population.

Nous détaillons maintenant les choix effectués pour les différentes étapes.

#### Initialisation de la population aléatoire

Dans l'algorithme implémenté, la taille de la population est fixée à 100 individus. La population de départ est initialisée avec des placements des balises aléatoires, sauf pour un individu dont la position des balises est initialisée de façon uniforme sur l'abscisse curviligne des bords du terrain. Il est aussi possible d'initialiser toute la population de départ avec des placements uniformes, en décalant légèrement chaque placement afin d'obtenir des positions de balises différentes pour chaque individu. Le fait d'ajouter au moins un individu avec un placement uniforme à la population de départ, vient de l'observation que dans certains scénarios assez simples, la solution optimale est souvent proche du placement uniforme.

#### Critère d'arrêt

En pratique, avec l'algorithme générique, il est généralement impossible de déterminer si le minimum global de la fonction à minimiser est atteint. Pour cette raison, l'algorithme est arrêté au bout d'un certain nombre d'itérations, ou bien quand la solution n'évolue plus pendant un certain nombre d'itérations. En pratique, le nombre maximal d'itérations est fixé à 100, et le nombre maximal d'itérations où le meilleur individu n'a pas changé est fixé à 10. Pour éviter les minimums locaux, il est souvent plus efficace de faire davantage d'essais rapides à partir de populations aléatoires différentes, que d'itérer longtemps à partir d'une seule population aléatoire.

#### Sélection

Une fois la fonction de coût évaluée pour les individus de la population courante, une idée naïve de la sélection serait de garder simplement la meilleure moitié de la population. Cela aboutit pourtant à de moins bons résultats car de mauvaises solutions peuvent, via croisements et mutation, être à l'origine de bonnes solutions après quelques générations. Nous avons donc opté pour une sélection en deux phases :

1. Une première phase, dite d'élitisme, où les 10 meilleurs individus sont conservés.
2. Une deuxième phase, dite de tournoi, pour déterminer les 40 autres individus qui seront conservés. Dans cette phase, deux individus sont choisis aléatoirement et le meilleur des deux est conservé. L'opération se répète jusqu'à compléter la première moitié de la nouvelle population. On observe que les bons individus ont plus de chance d'être sélectionnés, mais de mauvaises solutions peuvent également passer si elles sont en tournoi contre une solution encore moins bonne. Il paraît évident toutefois que la moins bonne solution d'une génération n'est jamais sélectionnée car elle ne peut gagner aucun tournoi.

#### Croisement et mutation

La deuxième moitié de la nouvelle population est créée à partir de croisements et de mutations des individus sélectionnés dans la première moitié de la nouvelle population. Pour créer un nouvel individu par croisement, deux individus (appelés les parents) sont choisis au hasard parmi les sélectionnés. Puis, l'opération de croisement à proprement parler, est effectuée de la façon suivante : les balises de la nouvelle solution reprennent alternativement les positions de celles de son premier et de son deuxième parent. Cette méthode, appelée croisement uniforme, n'est qu'une possibilité parmi d'autres méthodes plus ou moins complexes. Un autre choix aurait été de conserver un nombre aléatoire de balises de chacun des deux parents, sans forcément alterner.

Afin de ne pas se limiter aux positions générées initialement, certains individus subissent également une mutation qui modifie légèrement la position curviligne d'une de ses balises. Au cours de ces deux opérations, on prend soin, si nécessaire, de bien renuméroter les balises par ordre croissant d'abscisse curviligne.

Nous détaillons maintenant les différentes fonctions de coût envisagées pour le placement.

### 5.4.3 Fonctions de coût

Dans une première version de l'algorithme génétique, on ne considère que deux objectifs à optimiser dans la fonction de coût : la couverture de l'espace de travail, et la précision de localisation induite par la configuration des balises. La deuxième version de l'algorithme génétique qui considère en plus le nombre de balises à utiliser est présentée dans la section 5.7.

#### Critère de couverture du terrain

Le critère de couverture du terrain est défini comme la disponibilité d'un certain nombre de mesures de distance provenant de balises distinctes à une position donnée du terrain. Ainsi, une première solution basique d'une fonction de coût serait de chercher à maximiser le pourcentage de position du terrain ayant accès à au moins une mesure de distance.

Dans le cas d'une localisation dans le plan par un algorithme de trilatération, on sait qu'il est nécessaire d'avoir au minimum trois mesures de distance par rapport à trois balises différentes pour pouvoir calculer une estimation de la position. La couverture optimale de l'espace de travail

correspond donc à la configuration des balises permettant d'avoir accès sur la plus grande partie du terrain à aux moins trois mesures de distance par rapport à trois balises différentes.

Ainsi, en reprenant l'idée proposée dans [1], un premier critère appelé par la suite *nRanges* a été développé. Ce dernier retourne la moyenne du nombre de mesures de distance disponibles en chaque point de l'espace de travail avec une saturation à 3, s'il y a plus de mesures disponibles. Cette saturation permet d'obtenir une couverture du terrain plus uniforme, en évitant les configurations de balises qui vont couvrir seulement une partie du terrain avec un nombre important de mesures de distance, et couvrir l'autre partie avec seulement une ou deux mesures de distance. Ce critère consiste ainsi à résoudre le problème d'optimisation suivant :

$$\max \frac{1}{M} \sum_{i=1}^{M} (\text{nRanges})_i \tag{5.5}$$

avec $M$ le nombre de positions du terrain, et $(\text{nRanges})_i$ pouvant prendre les valeurs 1, 2 ou 3 suivant le nombre de mesures de distance disponibles. Une variante possible de cette fonction de coût serait de calculer le pourcentage de terrain couvert par au moins trois mesures de distance.

Comme la présence d'un minimum de trois mesures de distance est un prérequis nécessaire pour effectuer la localisation par trilatération, l'emploi de ce seul critère dans la fonction de coût à optimiser, donne déjà de bons résultats comme cela sera montré dans la section suivante. Cependant comme proposé dans [113], il est intéressant d'ajouter à la fonction de coût un critère lié à la précision de la localisation. En effet, comme cela a déjà été montré dans [9] et précisé dans l'introduction de ce chapitre, le placement des balises a un impact sur la précision de localisation pouvant être obtenue.

### Critère de précision de localisation

Pour optimiser la précision de localisation, l'un des premiers critères de choix pour réaliser la fonction de coût est l'emploi de la dilution de précision en position (PDOP) [136]. Ce critère est obtenu par la propagation des incertitudes des mesures de distance en calculant le Jacobien des équations de mesure. Les mesures de distance étant modélisées par l'équation suivante :

$$r_i = \sqrt{(x_i - x)^2 + (y_i - y)^2} + \epsilon$$

avec $\epsilon \sim \mathcal{N}(0, \sigma^2)$, et $\sigma^2$ la variance du bruit de mesure, identique pour toutes les balises. Ce calcul, n'est bien sûr possible, que quand il y a au minimum trois mesures de distance (dans le cas 2D).

Dans le cas de $N \geq 3$ mesures de distance, le Jacobien du vecteur des mesures $\mathbf{r} = (r_1, \ldots, r_N)^T$ par rapport à la position à estimer $\mathbf{p} = (x, y)^T$ est donné par :

$$\mathbf{J} = \frac{\partial \mathbf{r}}{\partial \mathbf{p}} = \begin{pmatrix} -\frac{(x_1 - x)}{r_1} & -\frac{(y_1 - y)}{r_1} \\ -\frac{(x_2 - x)}{r_2} & -\frac{(y_2 - y)}{r_2} \\ \vdots & \vdots \\ -\frac{(x_N - x)}{r_N} & -\frac{(y_N - y)}{r_N} \end{pmatrix}$$

La matrice de covariance de l'erreur en position au point $\mathbf{p}$ est alors donnée par :

$$\mathbf{C} = \mathbf{J}^{\dagger} \Sigma (\mathbf{J}^{\dagger})^T$$

avec $\mathbf{J}^{\dagger} = \mathbf{J}^T (\mathbf{J}\mathbf{J}^T)^{-1}$ la pseudo-inverse de $\mathbf{J}$, et $\Sigma = \text{diag}(\sigma^2, \ldots, \sigma^2)$.

Si l'on définit $\mathbf{C}$ par :

$$\mathbf{C} = \text{diag}(\sigma_x^2, \sigma_y^2)$$

alors la PDOP est donnée par :

$$\text{PDOP} = \frac{\sqrt{\sigma_{\text{x}}^2 + \sigma_{\text{y}}^2}}{\sigma^2}$$

Cependant, comme mentionné plus haut, dans le cas où le modèle des mesures de distance est plus élaboré et que les bruits de mesure sont différents pour chaque balise, il est plus avantageux d'utiliser une formulation utilisant la matrice d'information de Fisher. Ainsi, dans les fonctions de coût proposées par la suite, au lieu d'utiliser un critère basé sur la dilution de précision comme dans [113], le terme lié à la précision de localisation est calculé en utilisant le déterminant de la matrice de Fisher, $\left|\text{FIM}\right|$, dont l'expression reprise de [98] a été rappelée dans l'équation (5.3).

Cependant, à la différence d'autres travaux, au lieu de maximiser la moyenne du $\left|\text{FIM}\right|$ sur tout le terrain ou sur des régions du terrain comme dans [109], le critère de précision de localisation est défini comme le pourcentage du terrain où le $\left|\text{FIM}\right|$ est supérieur à un seuil $\delta_{\text{FIM}}$ donnée :

$$\left|\text{FIM}\right| \geq \delta_{\text{FIM}}$$

En effet, il est possible de relier le $\left|\text{FIM}\right|$ à la précision de localisation, moyennant quelques hypothèses. Tout d'abord, on suppose que l'erreur de position donnée par l'algorithme de localisation est modélisée par une densité de probabilité Gaussienne de matrice de covariance $\mathbf{C}$. Ensuite, si l'on considère que $\sigma_x = \sigma_y = \sigma$, alors l'expression de $\mathbf{C}$ devient :

$$\mathbf{C} = \begin{pmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{pmatrix}$$

et son déterminant est donnée par $\left|\mathbf{C}\right| = \sigma^4$.

D'un point de vue géométrique, $\left|\mathbf{C}\right|$ peut être interprété comme l'aire de l'incertitude liée à la position estimée. Maintenant, en sachant que la matrice d'information de Fisher est l'inverse de la matrice de covariance, $\text{FIM} = \mathbf{C}^{-1}$ [94], on obtient que :

$$\left|\text{FIM}\right| = \left|\mathbf{C}^{-1}\right| = \frac{1}{\sigma^4}$$

En utilisant la règle des $3\sigma$, illustrée à la Figure 5.2, et disant que 99.7% des données sont comprises dans l'intervalle $\mu \pm 3\sigma$ ($\mu$ étant la moyenne), on définit la précision $d$ telle que $d = 3\sigma$, et on obtient donc la relation suivante entre la précision $d$ et le $\left|\text{FIM}\right|$ :

$$\left|\text{FIM}\right| \simeq \frac{81}{d^4}$$

En utilisant cette relation, on peut définir le seuil $\delta_{\text{FIM}}$ sur $\left|\text{FIM}\right|$ en le reliant à la précision de localisation $d$ voulue. Des exemples de correspondance sont donnés dans le Tableau 5.2.

Par exemple, en imposant que le robot puisse être localisé avec une précision $d = \pm 0.25$ m, le critère de précision de localisation cherchera à maximiser le pourcentage de positions du terrain où le $\left|\text{FIM}\right|$ sera supérieur au seuil $\delta_{\text{FIM}} = 20736$.

Dans les figures 5.3, 5.4, 5.5, il est possible de voir les valeurs du $\left|\text{FIM}\right|$ et de la précision de localisation correspondante pour trois configurations de balises différentes. Les deux premières configurations correspondent à la bonne et à la mauvaise configuration de balises présentées en début de chapitre, alors que la troisième configuration correspond à quatre balises disposées en carré. Dans le cas des figures montrant le $\left|\text{FIM}\right|$, les zones en jaune correspondent aux positions qui auront une bonne précision de localisation. À l'inverse, pour les figures sur la

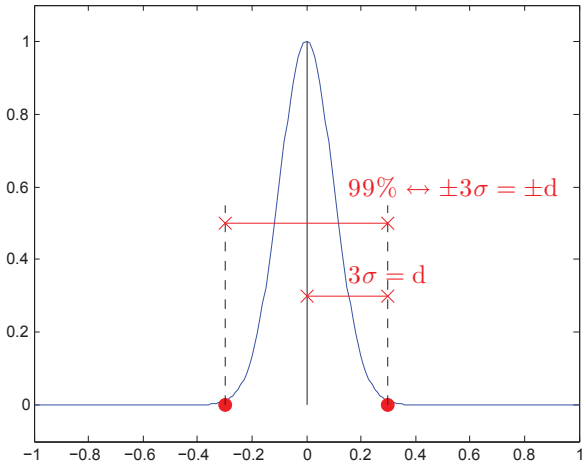
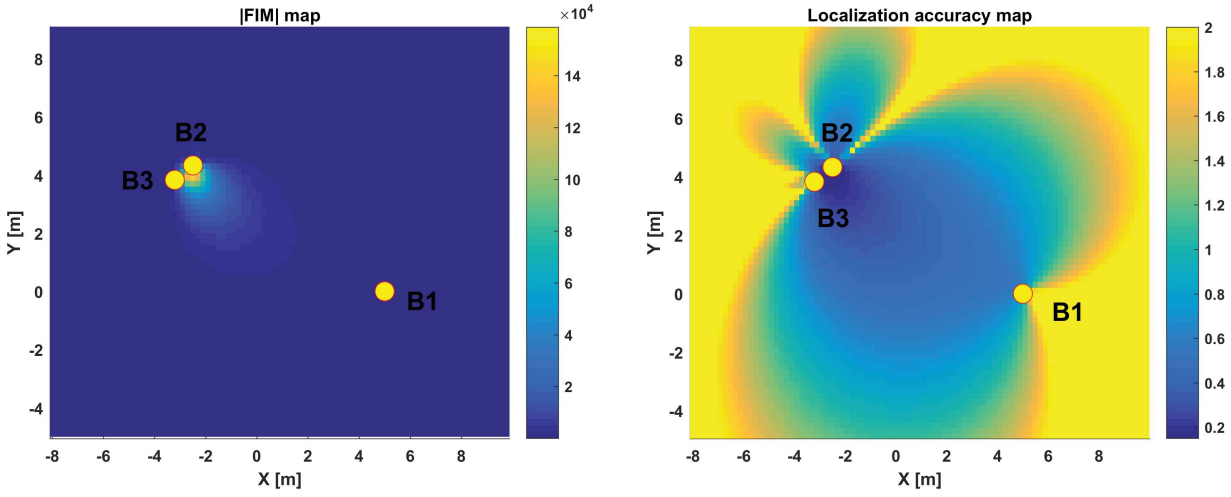



FIGURE 5.2 – Illustration du choix de la précision de localisation, $d$, en rapport avec la règle des $3\sigma \leftrightarrow 99.7\%$ de la fonction Gaussienne

| $d$ [m] | \|FIM\| |
|---|---|
| 0.1 | 810000 |
| 0.2 | 50625 |
| 0.25 | 20736 |
| 0.3 | 10000 |
| 0.5 | 1296 |

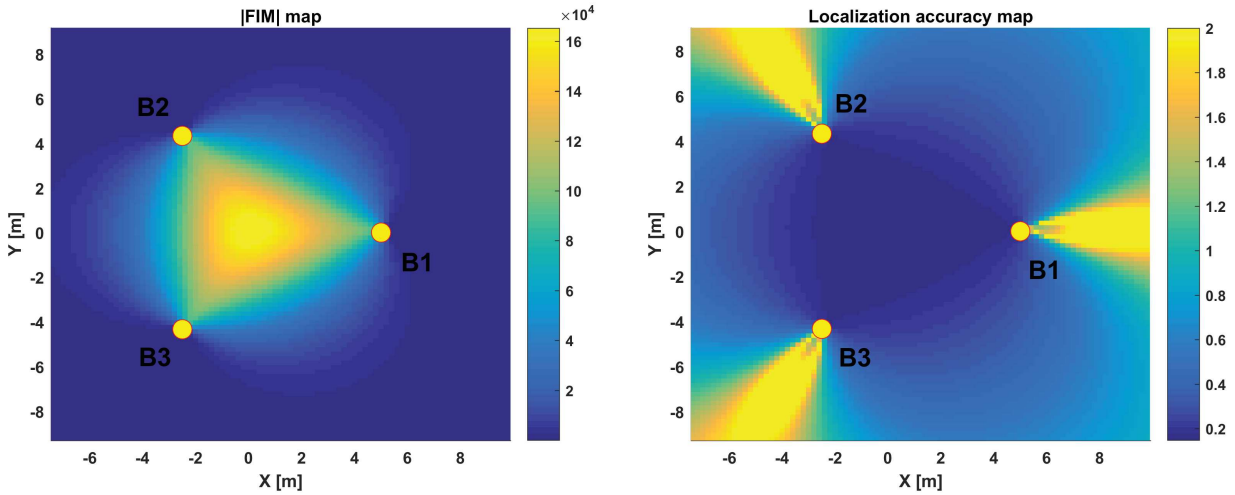
TABLEAU 5.2 – Relation entre la précision de localisation $d$ et $|\text{FIM}|$




(a) $|\text{FIM}|$ (b) $\pm d$

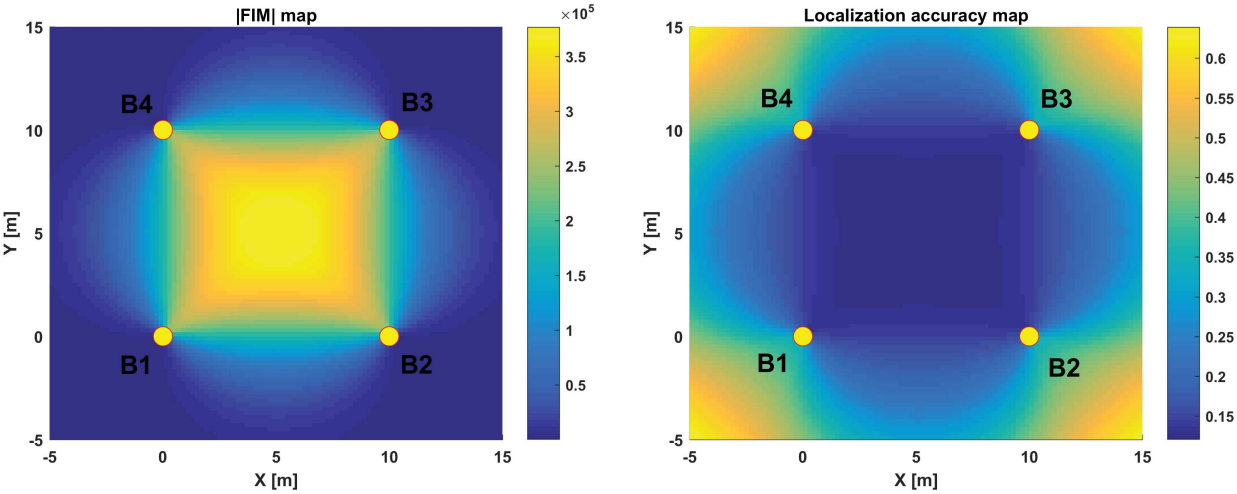
FIGURE 5.3 – Valeurs de $|\text{FIM}|$ (a) et précision de localisation (b) dans le cas d'un mauvais placement de 3 balises

précision de localisation, les zones en bleu foncé correspondent aux positions avec une variance de l'estimation en position plus faible, et ayant donc une meilleure précision de localisation. Comme attendu, on constate pour la mauvaise configuration des balises, donnée à la Figure 5.3, que très peu de zones possèdent une bonne précision de localisation (presque pas de jaune pour le $|\text{FIM}|$). Alors que pour la bonne configuration, donnée à la Figure 5.4, les zones possédant une bonne précision de localisation sont bien situées à l'intérieur du triangle formé par les trois




(a) $|\mathrm{FIM}|$ (b) $\pm d$

FIGURE 5.4 – Valeurs de $|\mathrm{FIM}|$ (a) et précision de localisation (b) dans le cas d'un placement idéal de 3 balises




(a) $|\mathrm{FIM}|$ (b) $\pm d$

FIGURE 5.5 – Valeurs de $|\mathrm{FIM}|$ (a) et précision de localisation (b) dans le cas d'un placement de 4 balises

balises (bien visible pour le $\left|\mathrm{FIM}\right|$).

À partir de la formule qui vient d'être dérivée, il est possible de définir un critère de précision de localisation, appelé *DetFimThresh* par la suite, qui consiste à maximiser le pourcentage des positions du terrain dont la précision de localisation est inférieure à un seuil $t_{\mathrm{FIM}}$ donné. Le problème d'optimisation correspondant peut donc être écrit comme :

$$\max \frac{1}{M} \sum_{i=1}^{M} \left( \left|\mathrm{FIM}\right| \geq t_{\mathrm{FIM}} \right)_i \tag{5.6}$$

avec $M$ le nombre de positions du terrain.

Comparée à l'utilisation de la moyenne du $\left|\mathrm{FIM}\right|$ sur l'espace de travail, la formulation proposée a plusieurs avantages. Tout d'abord, elle permet d'améliorer l'homogénéité de la couverture du terrain avec la précision de localisation désirée, et évite donc la formation de zones

avec de fortes valeurs de $\left|\text{FIM}\right|$ au détriment de zones avec de faibles valeurs. De plus, cette formulation permet de contrôler le niveau de précision de localisation désiré, qui peut être plus ou moins important suivant les applications, ou à l'intérieur même de l'espace de travail.

On peut ainsi envisager d'imposer un seuil différent en chaque position du terrain en fonction de la configuration du terrain. On peut en effet imaginer, qu'il est préférable d'avoir une bonne précision de localisation près des bordures du terrain, des obstacles, ou de passages étroits, alors que dans une zone bien dégagée une bonne précision serait moins critique. Bien que cette idée n'ait pas été testée, il est très facile de la mettre en œuvre. Pour cela, il suffit d'attribuer une pondération pour chaque cellule de l'espace de travail discrétisé, qui correspondra à l'importance du critère de précision de localisation associée à la position.

Le principal désavantage du critère proposé, mais qui est aussi le cas pour tout critère utilisant le $\left|\text{FIM}\right|$, est qu'il nécessite déjà au minimum trois mesures de distance pour être calculé. Quand cette condition n'est pas respectée, le critère ne peut être calculé pour la position correspondante, et il peut s'avérer nécessaire de lui attribuer une valeur particulière. Ceci peut engendrer une mauvaise couverture du terrain, en favorisant les zones où il est plus facile de calculer ce critère, tout en délaissant les autres régions. Une façon d'éviter ce problème serait de s'assurer ou d'imposer au préalable que l'ensemble de l'espace de travail soit déjà couvert avec le nombre requis de mesures de distance.

Nous venons de voir deux critères possibles pour la réalisation de la fonction de coût à minimiser. Ces critères peuvent être utilisés en l'état pour la minimisation, mais afin de garantir à la fois une bonne couverture et une bonne précision, il peut être plus avantageux de les combiner.

### Combinaisons de critères

Maintenant que les deux premiers critères de la fonction de coût ont été définis, il reste à définir comment les combiner afin de former la fonction de coût finale qui sera optimisée dans l'algorithme génétique. Les deux termes peuvent être combinés linéairement avec des pondérations associées à chaque terme comme cela a été fait dans [113]. Mais il est aussi possible les multiplier, ce qui évite alors l'introduction des deux coefficients de pondération. La fonction de coût finale, appelée *MixedCosts* par la suite, prend donc la forme suivante :

$$\Big[\%\big(\text{NbRanges} \geq 3\big)\Big] \times \Big[\%\big(|\text{FIM}| \geq \delta_{\text{FIM}}\big)\Big] \tag{5.7}$$

Le premier terme est défini comme le pourcentage du terrain couvert par au moins trois mesures de distance. Alors que le second terme correspond au pourcentage du terrain où le $\left|\text{FIM}\right|$ est supérieur au seuil $\delta_{\text{FIM}}$. La fonction de coût proposée va donc avoir pour effet de maximiser l'aire de l'espace de travail où il y a au minimum trois mesures de distance (premier critère) et où la précision de localisation atteint un seuil désiré (second critère).

Comme noté précédemment, pour une position du terrain donnée, le second critère de l'équation (5.7) ne peut être calculé qu'à condition que le premier critère soit déjà satisfait. En effet, le calcul du déterminant de la matrice de Fisher n'est possible que si au moins trois mesures de distance sont disponibles. Ainsi, la fonction de coût va d'abord favoriser la couverture du terrain avant d'imposer le critère de précision de localisation. En effet, il est préférable d'avoir une bonne couverture du terrain avec peu de zones ayant une bonne précision de localisation, que d'avoir une zone ayant une bonne précision de localisation et le reste sans information de position disponible, comme cela est schématisé à la Figure 5.6. Dans le cas du scénario bleu, on a 50 % de terrain couvert et ayant la précision de localisation souhaitée, ce qui signifie que tout le terrain couvert est à la précision de localisation souhaitée. Cependant le scénario rouge avec 90 % de couverture et 30 % du terrain avec la précision souhaitée est préférable car plus de

zones du terrain seront couvertes et donc susceptibles d'avoir une estimation de position même si celle-ci ne sera qu'approximative. Dans la majorité des cas, il est donc préférable de privilégier d'abord une bonne couverture du terrain avant d'imposer une précision de localisation souhaitée. La zone hachurée de la figure correspond aux situations impossibles où le premier critère n'est pas remplit pour permettre de calculer le second critère.

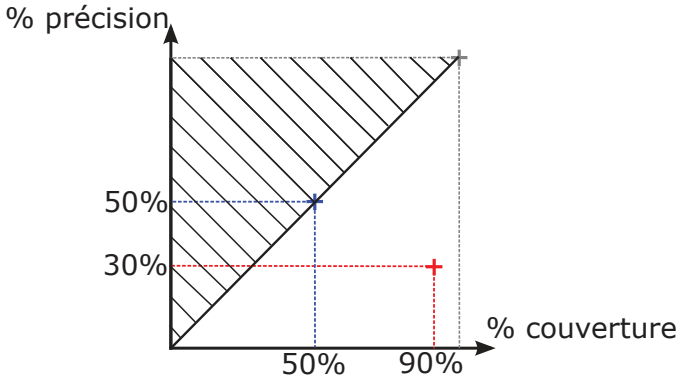



FIGURE 5.6 – Compromis entre les critères de pourcentage de couverture du terrain et de pourcentage de terrain ayant la précision de localisation souhaitée

### Résumé des fonctions de coût implémentées

Voici un récapitulatif des différentes fonctions de coût implémentées lors de ce travail et qui ont été comparées lors des simulations :

- *Couverture* : moyenne sur le terrain du nombre de positions où au moins une mesure de distance est disponible (pourcentage du terrain couvert par au moins une mesure de distance),
- *nRanges* : moyenne du nombre de mesures de distance disponibles en chaque position du terrain avec une saturation au-delà de 3, voir équation (5.5),
- *GDOP* : moyenne de la GDOP, donnée par (5.6), sur l'ensemble du terrain : min $\frac{1}{M}\sum_{i=1}^{M}\text{PDOP}_i$,
- *DetFim* : moyenne du déterminant de la matrice d'information de Fisher sur l'ensemble du terrain : max $\frac{1}{M}\sum_{i=1}^{M}|\text{FIM}|_i$,
- *DetFimThresh* : pourcentage des positions du terrain dont la précision de localisation est inférieure à un seuil, voir équation (5.6),
- *MixedCosts* : produit des fonctions de coût *nRanges* et *DetFimThresh*, comme présenté par l'équation (5.7),
- *MixedCosts1* : combinaison linéaire des fonctions de coût *nRanges* et *DetFimThresh*,
- *MixedCosts2* : combinaison linéaire pondérée des fonctions de coût *nRanges* et *DetFimThresh* : $\alpha(nRanges)+\beta(DetFimThresh)$.

Pour les expériences présentées par la suite, le seuil sur le déterminant de la FIM est fixé à $t_{\text{FIM}} = 50625$, ce qui correspond d'après le Tableau 5.2 à une précision de localisation de $\pm 25$ cm. Pour la fonction de coût *MixedCosts1*, les pondérations pour les critères *nRanges* et *DetFimThresh* sont égaux et fixés à 0.5. Alors que pour la fonction de coût *MixedCosts2*, les pondérations sont égales à $\alpha = 0.7$ et $\beta = 0.3$.

En plus des fonctions de coût précédemment citées, deux autres formulations ont été implémentées, mais n'ont pas été testées par manque de temps. L'idée derrière ces deux fonctions de coût est de maximiser à la fois le pourcentage de zone couverte et le pourcentage de zone couverte par une précision de localisation souhaitée dans une même formule au lieu de deux

comme avec la fonction *MixedCosts*. Ainsi, la première fonction cherche à maximiser la plus petite valeur du déterminant de la matrice d'information de Fisher sur le terrain, tandis que la seconde a pour but de minimiser l'aire de la plus grande région non couverture ou dont la précision de localisation n'atteint pas le seuil fixé. L'atout de la seconde fonction de coût étant de travailler sur des régions du terrain et non plus sur sa globalité, ce qui devrait théoriquement permettre une meilleure homogénéisation des résultats sur le terrain. On espère ainsi favoriser les petites zones mal couvertes au profit des régions plus grandes et mal couvertes.

Maintenant que l'algorithme génétique ainsi que les différentes fonctions de coût pouvant être utilisées pour résoudre le problème du placement des balises ont été présentés, nous allons dans la prochaine section valider en simulation l'approche proposée.

## 5.5 Optimisation de la position des balises

Pour évaluer le bon fonctionnement de l'algorithme génétique, ainsi que les performances des fonctions de coût implémentées, deux expériences provenant de simulations ont été menées. La première permet de visualiser l'évolution de la fonction de coût en fonction du nombre de balises, car rappelons-le, dans cette première version de l'algorithme génétique, le nombre de balises utilisées n'est pas une variable optimisée. La deuxième expérience quant à elle, évalue l'impact du placement des balises obtenu après optimisation sur la précision de localisation d'un robot avec un algorithme de trilatération. L'objectif des simulations menées est double. On souhaite tout d'abord vérifier l'utilité d'un algorithme de placement de balises par rapport à d'autres stratégies de placement plus simples. Puis, on cherche à comparer les performances des différentes fonctions de coût proposées afin d'identifier celles qui fonctionnent le mieux.

Les différentes stratégies de placement des balises comparées sont :

— Placement uniforme (UNIF),

— Placement aléatoire (RAND),

— Placement après optimisation (GA).

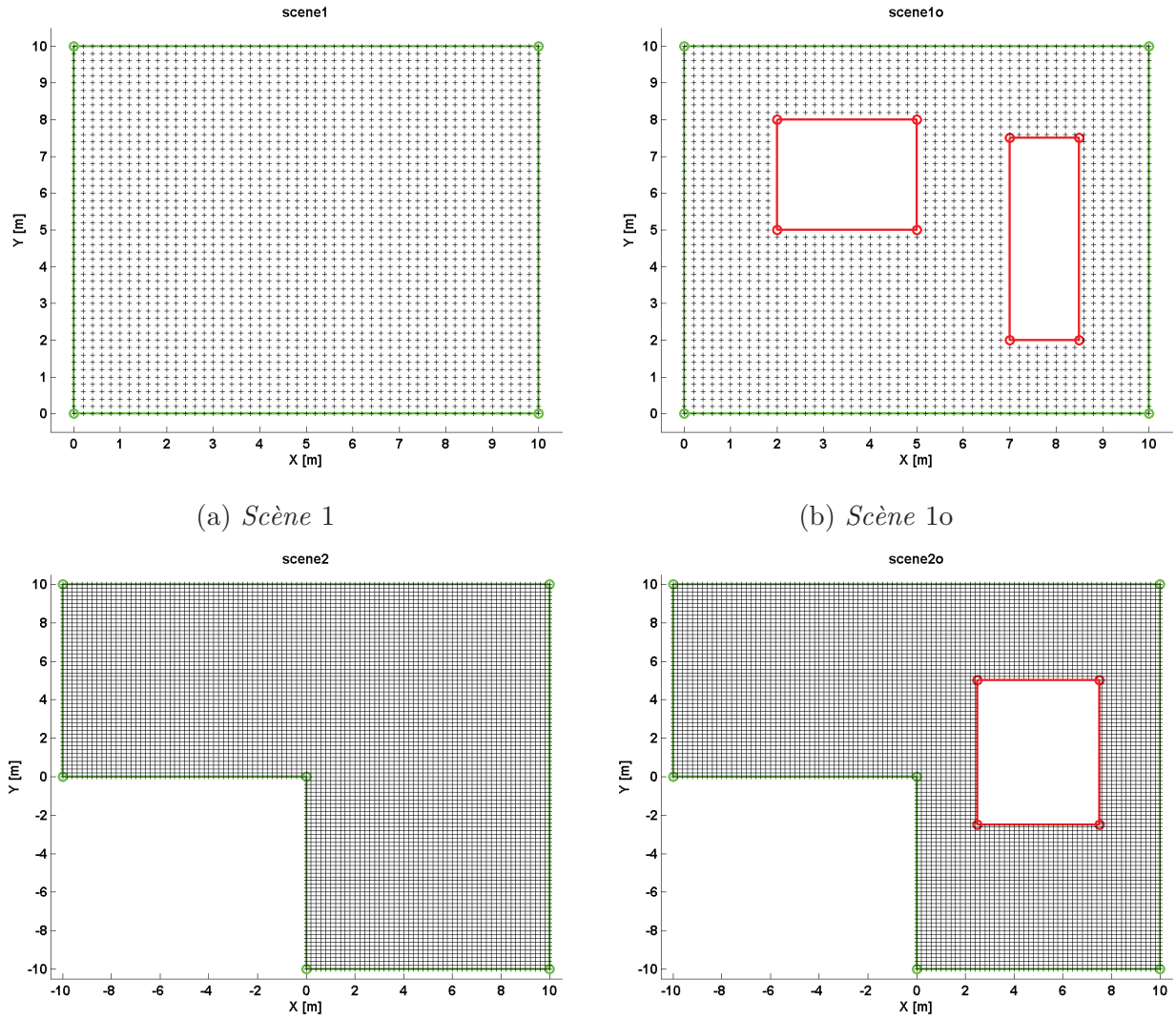
Pour le placement uniforme, les balises sont espacées régulièrement le long des bordures du terrain et des éventuels obstacles, comme cela est fait pour l'initialisation de l'un des individus de la population de départ de l'algorithme génétique. Pour le placement aléatoire, la position de chaque balise du terrain est tirée aléatoirement d'une distribution uniforme couvrant toute l'abscisse curviligne (intervalle $[0, 1[$). Afin de caractériser le processus aléatoire, 1000 placements sont générés, et les fonctions de coût correspondantes évaluées. La moyenne et la déviation standard des fonctions de coût sont alors calculées et utilisées pour la comparaison avec les autres placements. Une alternative serait de sélectionner le "meilleur" placement aléatoire, c'est-à-dire celui optimisant le mieux la fonction de coût et d'utiliser la valeur de sa fonction de coût pour la comparaison.

Différents environnements de simulation ont été créés avec des formes géométriques assez simples et contenant ou non des obstacles. Certaines des scènes utilisées lors des simulations sont présentées à la Figure 5.7. Les différents environnements sont notés par la suite *Scène* X, avec X le numéro de la scène. Quand il s'agit d'une variante contenant des obstacles, elle sera notée *Scène* Xo.

Pour toutes les expériences menées dans la suite de ce chapitre, les paramètres des mesures de distance des balises sont les suivants :

— Biais : $b = 0$ m,

— Déviation standard : $\sigma_r = 0.1/3 = 0.033$ m,




(a) *Scène* 1

(b) *Scène* 1o

(c) *Scène* 2

(d) *Scène* 2o

FIGURE 5.7 – Exemple de différentes scènes utilisées pour les simulations. Les traits en pointillé vert représentent les bordures du terrain ne bloquant pas les mesures de distances, alors que ceux en trait continu rouge le sont. Les petits signes plus rouges à l'intérieur du terrain et en-dehors des obstacles, représentent les positions issues de la discrétisation du terrain et utilisées pour le calcul de la fonction de coût.

— Dépendance à la distance : $\eta = 1e^{-3}$,

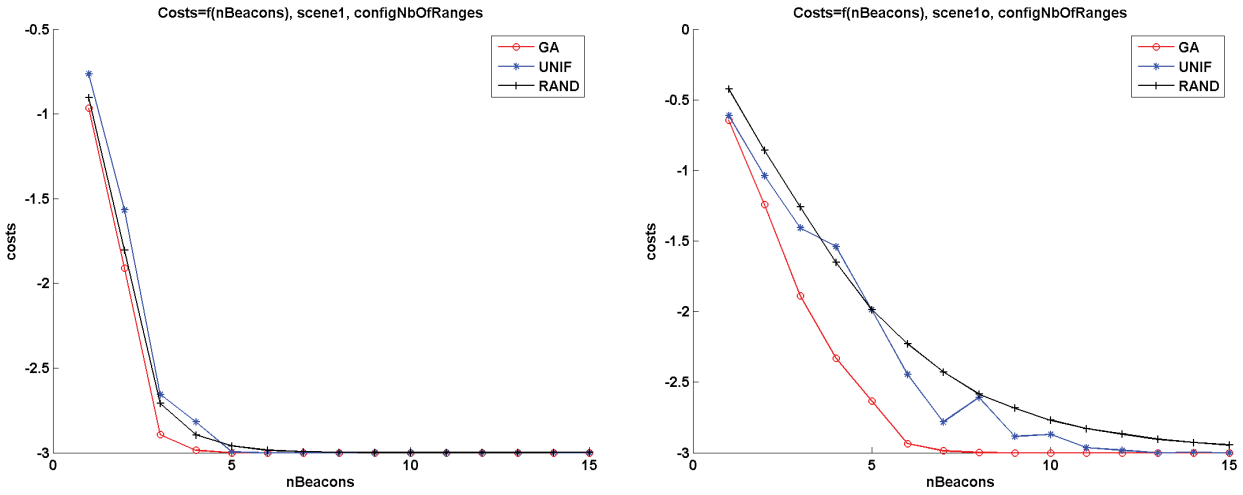
— Portée maximale : $r_{max} = 10$ m.

Dans les deux paragraphes suivants, nous allons valider en simulation le bénéfice du placement optimisé des balises pour les différentes fonctions de coût, ainsi que l'impact sur la précision de localisation d'un robot par un algorithme de trilatération.

### 5.5.1 Évolution des fonctions de coût en fonction du nombre de balises

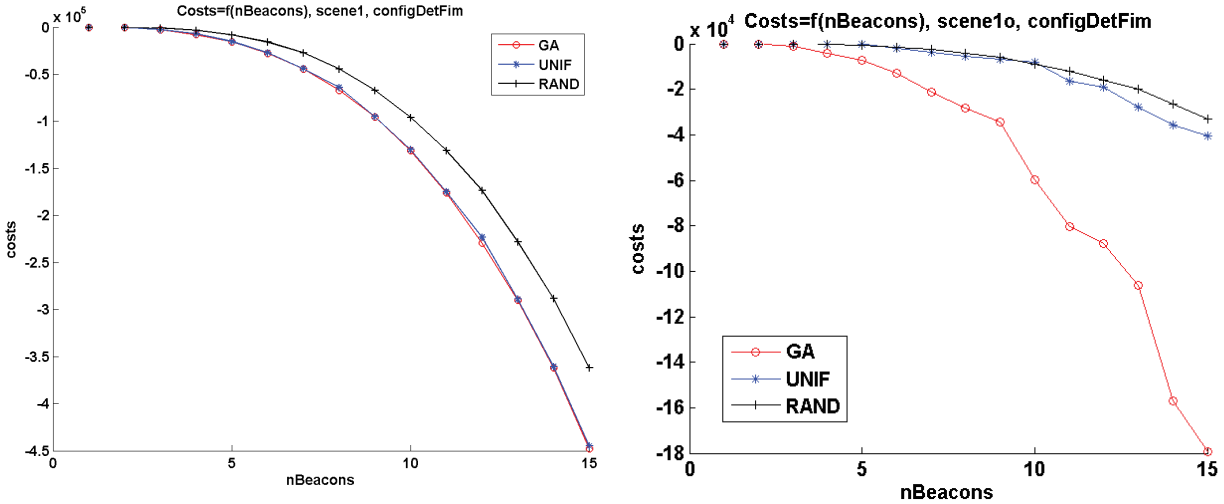
Dans cette première partie consacrée aux simulations, nous allons analyser l'évolution des différentes fonctions de coût en fonction du nombre de balises et comparer les résultats avec un placement uniforme et un placement aléatoire. Pour cela, les fonctions de coût obtenues pour les trois placements RAND, UNIF et GA sont calculées pour un nombre de balises allant de 1

à 15. Pour le placement aléatoire, en plus de la moyenne et de la déviation standard estimées à partir de 1000 tirages, la médiane et les 1$^{er}$ et 3$^{ème}$ quartiles sont aussi calculés. Parmi l'ensemble des résultats obtenus, ceux pour les *Scènes* 1 et 1o et les fonctions de coût *nRanges*, *DetFim*, *GDOP*, *DetFimThresh* et *MixedCosts1* sont présentés respectivement aux figures 5.8, 5.9, 5.10, 5.11 et 5.12. Dans l'ensemble des figures précitées, les courbes correspondantes au placement RAND sont obtenues à partir de la moyenne des 1000 tirages aléatoires.




(a) *Scène* 1, *nRanges* (b) *Scène* 1o, *nRanges*

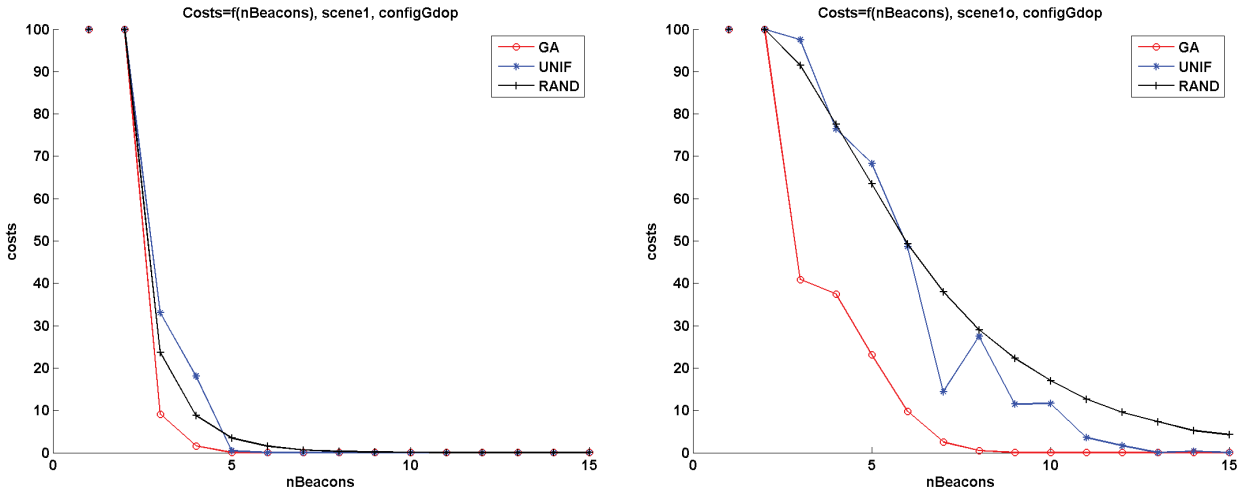
FIGURE 5.8 – coût = $f$(nBalises) pour la fonction de coût *nRanges* et les *Scènes* 1 et 1o




(a) *Scène* 1, *DetDim* (b) *Scène* 1o, *DetDim*

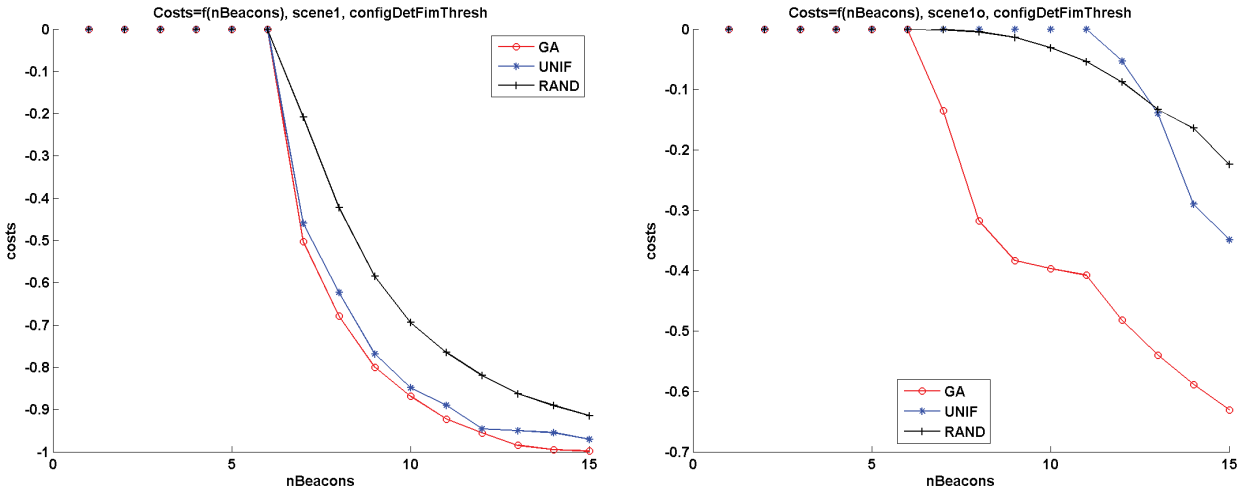
FIGURE 5.9 – coût = $f$(nBalises) pour la fonction de coût *DetFim* et les *Scènes* 1 et 1o

Comme attendu, les différents coûts diminuent avec l'augmentation du nombre de balises, et ceci indépendamment de la stratégie de placement utilisée, de la scène ou de la fonction de coût considérée. Cependant, à partir d'un certain nombre de balises qui dépend de la scène et de la fonction de coût, on constate en général une stagnation de l'évolution du coût. Le coût plafonne et aucune amélioration nette n'est constatée. Cette stagnation s'explique par le fait qu'à partir d'un certain nombre de balises, on atteint les limites de la fonction de coût, et donc que le critère optimisé est entièrement satisfait. De manière générale, on constate que quel que




(a) *Scène* 1, *GDOP* (b) *Scène* 1o, *GDOP*

FIGURE 5.10 – coût = $f$(nBalises) pour la fonction de coût *GDOP* et les *Scènes* 1 et 1o



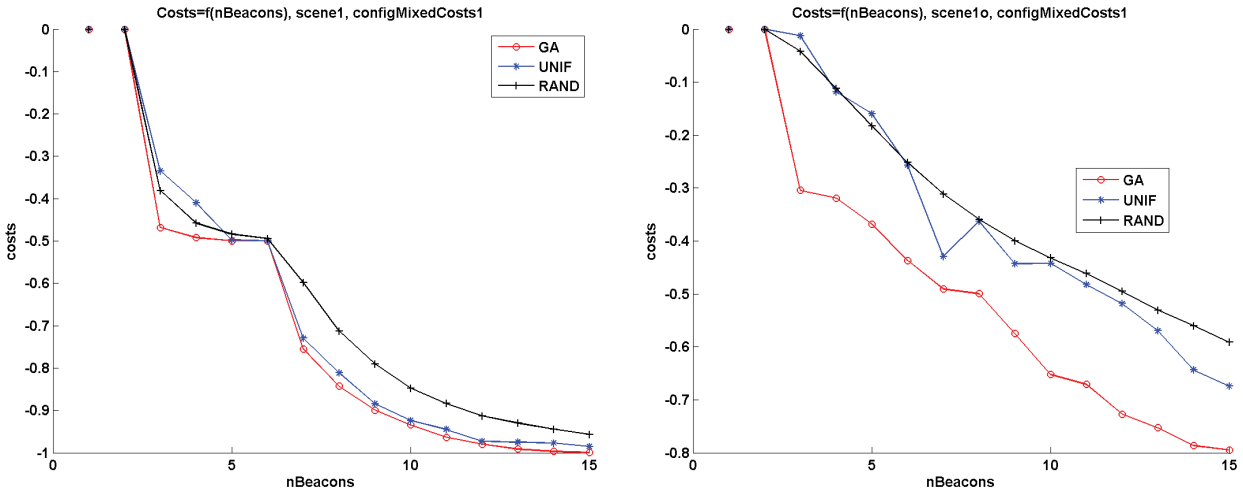

(a) *Scène* 1, *DetFimThresh* (b) *Scène* 1o, *DetFimThresh*

FIGURE 5.11 – coût = $f$(nBalises) pour la fonction de coût *DetFimThresh* et les *Scènes* 1 et 1o

soit la fonction de coût utilisée, le placement après optimisation (GA), retourne un coût plus faible que le placement uniforme (UNIF) et aléatoire (RAND). De même, le placement uniforme obtient bien souvent des coûts plus faibles que l'espérance du placement aléatoire. On constate même que le coût du placement GA est la plupart du temps bien meilleur que la moyenne du coût du placement aléatoire plus trois fois sa déviation standard.

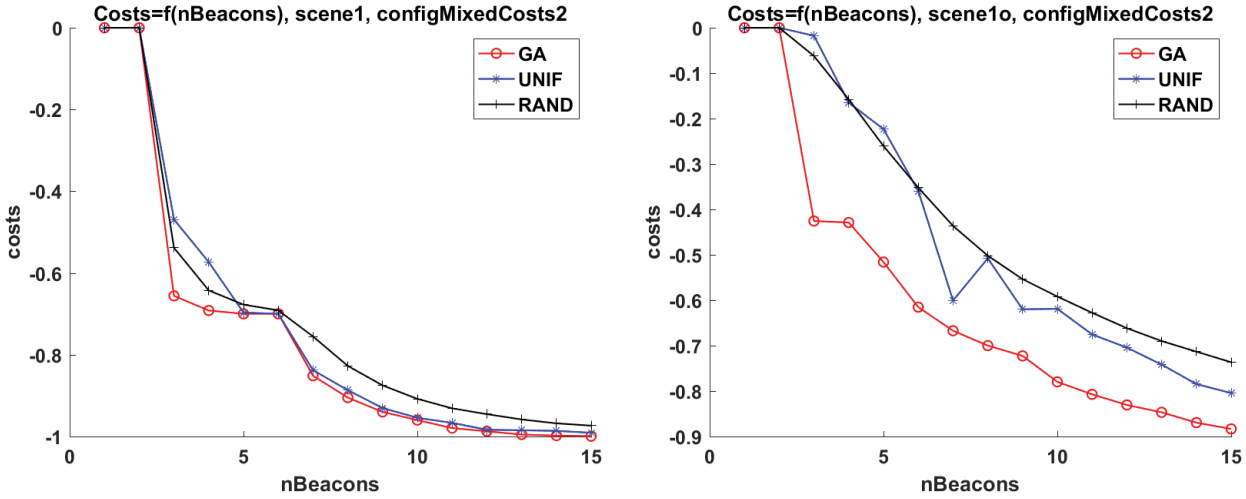
De plus, avec le placement GA, le palier de stagnation du coût est souvent atteint pour un nombre de balises plus faible que pour les placements UNIF et RAND, ce qui signifie que le critère optimisé est généralement satisfait avec moins de balises pour le placement GA comparé aux placements UNIF et RAND.

On constate aussi que les différences de coût entre le placement GA et les deux autres placements sont plus importantes pour un nombre de balises assez faible que pour un grand nombre de balises. Ainsi, l'intérêt et l'avantage d'un algorithme de placement des balises est bien plus important quand le nombre de balises est faible que quand celui-ci est grand. Ainsi, on voit par exemple sur les figures 5.8 et 5.10, que pour un nombre de balises compris entre 3 et 5 le placement GA est bien meilleur que les autres placements mais qu'après 6 balises




(a) *Scène* 1, *MixedCosts1* (b) *Scène* 1o, *MixedCosts1*

FIGURE 5.12 – coût = $f$(nBalises) pour la fonction de coût *MixedCosts1* et les *Scènes* 1 et 1o




(a) *Scène* 1, *MixedCosts2* (b) *Scène* 1o, *MixedCosts2*

FIGURE 5.13 – coût = $f$(nBalises) pour la fonction de coût *MixedCosts2* et les *Scènes* 1 et 1o

les coûts stagnent et convergent vers la même valeur pour les 3 placements. L'apport d'un algorithme de placement des balises est ainsi maximal dans la zone juste avant la stagnation des coûts. Malheureusement, il n'est pas possible de connaître cette zone à l'avance avec la méthode proposée, car celle-ci change en fonction de la scène considérée et de la fonction de coût. Une solution possible serait de résoudre le problème d'optimisation en ajoutant le nombre de balises comme une variable.

En comparant les résultats obtenus entre les scènes sans (*Scène*1, *Scène* 2) et avec obstacles (*Scène* 1o, *Scène* 2o), on constate que les coûts des scènes sans obstacles sont généralement plus faibles que pour celles avec obstacles. De plus, les paliers de stagnation de la fonction de coût apparaissent en général pour un nombre plus petit de balises dans le cas sans obstacles par rapport au cas avec. Ceci montre donc une plus grande difficulté à trouver un bon placement en présence des obstacles. L'hypothèse qui empêche les mesures de distance de traverser les obstacles favorise l'augmentation des zones non-couvertes et implique généralement d'utiliser un plus grand nombre de balises pour obtenir la même couverture qu'avec une scène sans obstacles. De plus, l'ajout d'obstacles augmente aussi le nombre de possibilités pour positionner

les balises. Bien que le fait d'avoir plus de choix pour positionner les balises peut s'avérer être un avantage pour obtenir de meilleurs placements, cela a aussi pour conséquences de ralentir la convergence de l'algorithme et d'augmenter potentiellement le nombre de minima locaux.

On remarque aussi que, plus la scène est complexe (géométrie du terrain, présence d'obstacles), plus la différence entre le coût du placement GA et des placement RAND et UNIF est importante. Ceci suggère que l'algorithme de placement des balises est d'autant plus intéressant que l'environnement est complexe. En effet, pour des scènes avec une géométrie simple comme la *Scène* 1, le placement uniforme sera déjà très proche d'un placement optimal et l'algorithme génétique devrait retourner une solution assez proche du placement uniforme, ce qui limite ainsi son intérêt. Par contre pour des scènes plus complexes, il y a moins de chances que le placement uniforme soit proche du placement optimal, qui devient moins intuitif et qui justifie alors l'emploi d'un algorithme de placement. Ce fait s'illustre très bien en comparant les courbes rouges et bleues des placement GA et UNIF respectivement, dans les différentes figures 5.8, 5.9, 5.10, 5.11 et 5.12. Ainsi pour les scènes sans obstacles (a), les deux courbes sont souvent très proches avec un léger avantage pour les courbes rouges du placement GA. Par contre, pour les scènes avec obstacles (b), les écarts entre les courbes rouges et bleues sont bien plus grands, ce qui montre les limitations du placement uniforme et l'intérêt du placement optimisé.

Ces premiers résultats permettent déjà de confirmer certaines idées intuitives et de fournir des conclusions sur le bénéfice d'un algorithme de placement optimal des balises. Cependant, ils sont uniquement basés sur l'évolution du coût des différents critères à optimiser. Ainsi, il reste à confirmer en pratique qu'un algorithme de placement permet bien d'améliorer les performances du système pour lequel les balises sont utilisées, qui dans le cas présent, consiste à localiser un robot mobile. Dans les simulations présentées par la suite, les résultats de localisation d'un algorithme de trilatération sont comparés pour les différents placements et fonctions de coût, afin de valider le bénéfice réel d'un algorithme de placement sur les performances de la localisation.

### 5.5.2 Impact du placement des balises sur la localisation par trilatération

Afin de valider l'intérêt de l'algorithme de placement optimal des balises pour la localisation d'un robot dans son espace de travail, des simulations permettant de comparer les performances de localisation des trois stratégies de placement ont été réalisées. Lors de ces simulations, différentes fonctions de coût sont aussi évaluées afin de déterminer si l'une d'entre elles permet d'améliorer significativement les performances. Comme la plupart des fonctions de coût sont basées et/ou dérivées de la localisation par trilatération, l'algorithme de localisation utilisé dans cette partie sera un simple algorithme linéaire de trilatération. Une comparaison du placement des balises sur différents algorithmes de localisation est présentée dans la section 5.6 suivante.

Le principe des simulations est le suivant. Pour chacune des scènes, une trajectoire de référence quelconque est générée pour le robot. En utilisant les positions des balises données par les différents placements, les vraies mesures de distance entre la position courante du robot sur la trajectoire et la position des balises en ligne de vue sont calculées. Les mesures de distances sont alors affectées d'un bruit blanc Gaussien de moyenne nulle et de variance $\sigma^2$. Les paramètres du bruit ajouté aux mesures de distance sont les mêmes que ceux utilisés dans le modèle de mesure, c'est-à-dire : $\sigma = \sigma_r = 0.033$ m et $\eta = 1e^{-3}$.

L'algorithme de trilatération est alors utilisé afin d'estimer la position du robot à partir des mesures de distance bruitées précédemment générées. Pour chaque placement, la localisation par trilatération est simulée $S = 100$ fois afin de tenir compte des propriétés statistiques des bruits ajoutés aux mesures. La moyenne, la déviation standard, le minimum et le maximum

de la métrique de localisation sont alors calculés et utilisés pour la comparaison des différents placements et fonctions de coût.

Afin de comparer les performances de localisation des différents placements de balises, différentes métriques peuvent être utilisées. La première étant l'erreur quadratique moyenne (RMSE) entre la trajectoire de référence et la trajectoire estimée. Celle-ci est calculée grâce à la formule suivante :

$$e_{\text{RMSE}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} e_i^2} \tag{5.8}$$

avec $e_i = \sqrt{(\bar{x}_i - \hat{x}_i)^2 + (\bar{y}_i - \hat{y}_i)^2}$ l'erreur de la trajectoire à l'instant $i$ entre la position de référence $(\bar{x}_i, \bar{y}_i)^T$ et la position estimée $(\hat{x}_i, \hat{y}_i)^T$, et $N$ étant le nombre total de points de la trajectoire.

La deuxième métrique correspond au calcul de l'erreur moyenne ou distance moyenne entre les positions de références et estimées de la trajectoire :

$$e_{\text{DIST}} = \frac{1}{N}\sum_{i=1}^{N} e_i \tag{5.9}$$

Au cours des expériences qui suivent, les deux erreurs de trajectoire précédentes sont calculées, mais seule l'erreur RMS sera donnée dans les différents résultats présentés.

### Résultats

Les résultats des simulations sur la localisation par trilatération sont rassemblés dans le Tableau 5.3 pour la *Scène* 1 avec trois balises et dans le Tableau 5.4 pour la *Scène* 1o avec sept balises. En plus, l'évolution des erreurs de la trajectoire est donnée à la Figure 5.14 pour les différents placements (RAND, UNIF et GA) et pour trois fonctions de coût : *nRanges*, *DetFim* et *MixedCosts2*. La Figure 5.16 montre une comparaison des cartes de coût de la *Scène* 1o obtenues avec la fonction *MixedCosts2* pour les trois stratégies de placement, alors que la Figure 5.15 compare les cartes de coût du placement GA pour les fonctions de coût *MixedCosts2* et *DetFim*. Dans les deux figures précitées, trois trajectoires sont à chaque fois visibles. Celles en bleu correspondent à la trajectoire de référence alors que celles en rouge et noire correspondent respectivement aux meilleures et pires trajectoires estimées.

| Cost function | Placement | Best cost | Mean [m] | Std dev. [m] | Max. [m] | Min. [m] | Time [s] |
|---|---|---|---|---|---|---|---|
| *nRanges* | GA | 2.92 | 0.064 | 0.002 | 0.068 | 0.060 | 54 |
| | RAND | 2.87 | 0.095 | 0.006 | 0.110 | 0.086 | 2 |
| *DetFim* | GA | 8.1*e*4 | 0.101 | 0.005 | 0.115 | 0.089 | 204 |
| | RAND | 8.0*e*4 | 0.082 | 0.004 | 0.093 | 0.069 | 2 |
| *MixedCosts2* | GA | 0.67 | 0.033 | 0.002 | 0.038 | 0.029 | 110 |
| | RAND | 0.66 | 0.054 | 0.003 | 0.062 | 0.049 | 3 |
| - | UNIF | - | 0.324 | 0.018 | 0.381 | 0.284 | 0 |

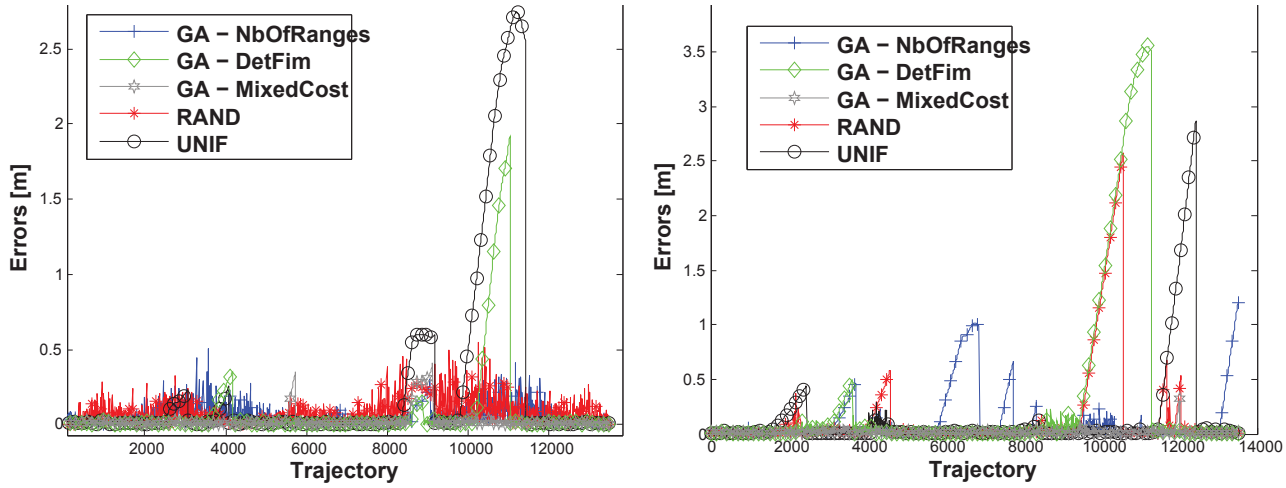
Tableau 5.3 – Erreurs de localisation par trilatération pour la *Scène* 1 avec 3 balises

### Observations

Comme attendu, on peut voir d'après les figures 5.15 (quatre coins du terrain) et 5.16 (milieu du terrain) que dans les zones mal ou non-couvertes, la position du robot n'est pas correctement estimée par l'algorithme de trilatération (trajectoires noires et rouges qui s'éloignent de la

| Cost function | Placement | Best cost | Mean [m] | Std dev. [m] | Max. [m] | Min. [m] | Time [s] |
|---|---|---|---|---|---|---|---|
| *nRanges* | GA | 2.96 | 0.140 | 0.010 | 0.173 | 0.116 | 312 |
| | RAND | 2.83 | 0.176 | 0.012 | 0.207 | 0.149 | 4 |
| *DetFim* | GA | 9.9e5 | 0.381 | 0.024 | 0.430 | 0.316 | 748 |
| | RAND | 4.3e5 | 0.787 | 0.161 | 1.253 | 0.536 | 5 |
| *MixedCosts2* | GA | 0.86 | 0.018 | 0.001 | 0.019 | 0.017 | 693 |
| | RAND | 0.78 | 0.029 | 0.001 | 0.032 | 0.025 | 5 |
| - | UNIF | - | 0.157 | 0.008 | 0.173 | 0.133 | 0 |

TABLEAU 5.4 – Erreurs de localisation par trilatération pour la *Scène* 1o avec 7 balises

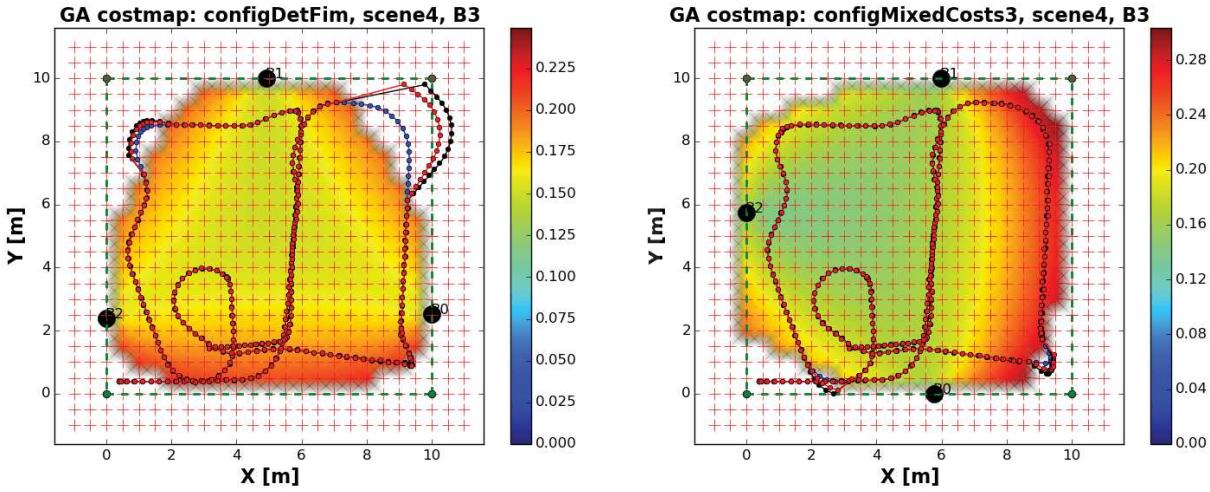



(a) *Scène* 1 (b) *Scène* 1o

FIGURE 5.14 – Évolution des erreurs de la localisation pour les *Scène* 1 (a) et *Scène* 1o (b). Les pics correspondent à des zones du terrain où l'estimation de la position par trilatération n'est pas possible à cause d'un nombre insuffisant de balises pour effectuer le calcul.

trajectoire bleue). Ces zones se retrouvent aussi à la Figure 5.14 et correspondent aux pics d'erreurs de la position estimée. En observant la Figure 5.15a, on constate que dans la zone blanche (nombre de balises insuffisant pour calculer une estimation de position) en haut à gauche, la trajectoire estimée diverge progressivement. Ceci est dû au fait que quand la trilatération ne donne pas d'estimation, la position du robot est estimée à partir de l'odométrie des roues. On remarque que généralement les zones mal couvertes sont situées sur les bords et dans les coins du terrain (Figure 5.15) ou alors derrière les obstacles (Figure 5.16). Une solution pour améliorer la situation sur les bords et dans les coins du terrain serait de reculer les balises derrière les limites extérieures si la configuration du terrain le permet.

On peut voir dans la Figure 5.14 et cela pour les deux scènes (1 et 1o), que la fonction de coût *DetFim* (courbe verte avec losange) présente les erreurs de localisation les plus importantes parmi les placements après optimisation. Une raison à cela est que cette fonction va tendre à maximiser des zones avec les plus grandes valeurs de |FIM|, et négliger d'autres zones où il est plus difficile d'obtenir une grande valeur. Ceci résulte alors en une couverture non-homogène du terrain avec des zones présentant une bonne précision et d'autres totalement non-couvertes. Ce phénomène est notamment visible dans la Figure 5.15a ou de large zones en haut à droite et à gauche se retrouvent sans le nombre requis de trois balises en vue pour calculer une estimation de position. Cependant dans la Figure 5.15b avec la fonction de coût *MixedCosts2*, on peut voir que l'aire de ces régions non-couvertes est sensiblement plus faible, ce qui améliore ainsi




(a) *DetFim* (b) *MixedCosts2*

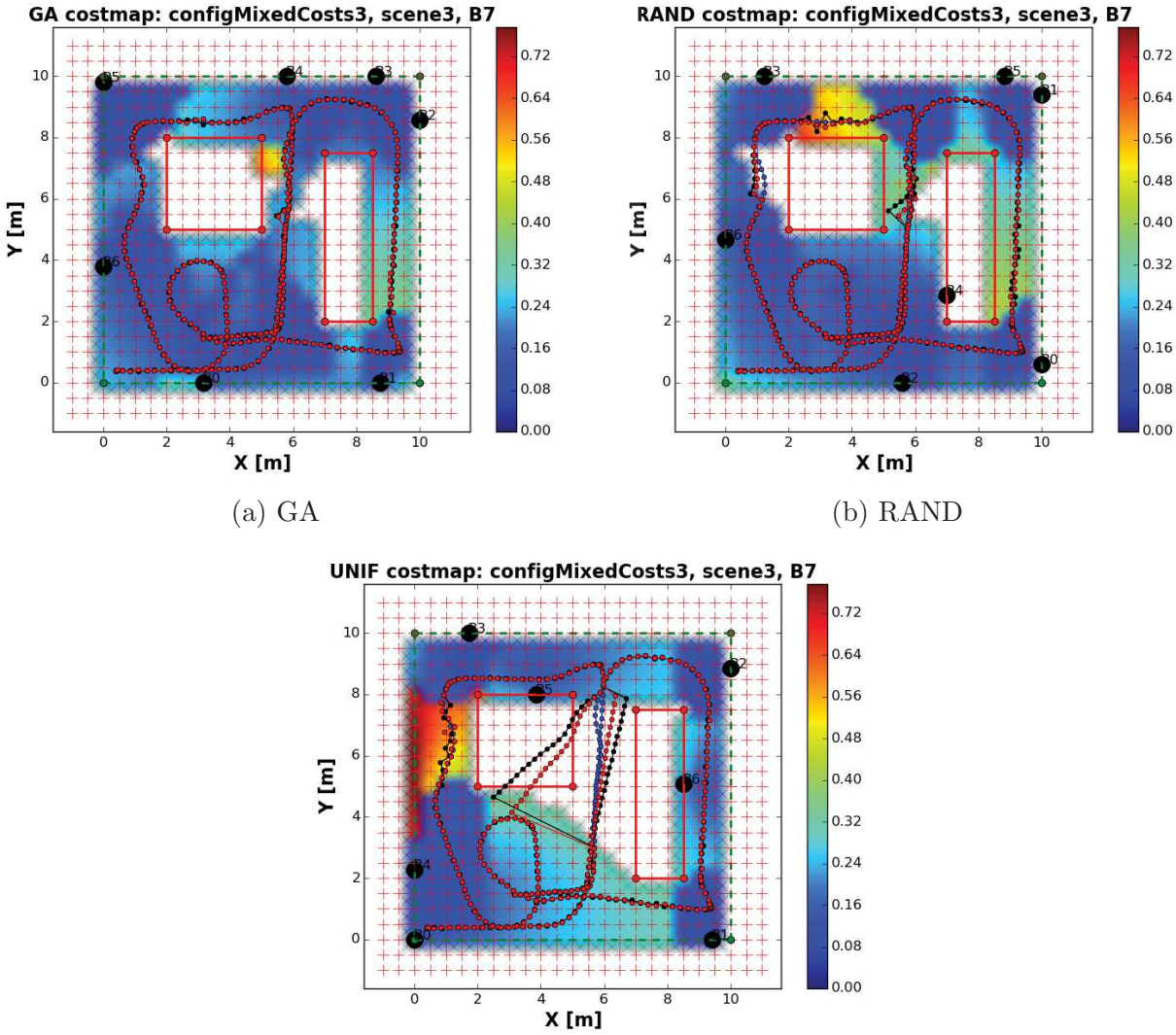
FIGURE 5.15 – *Scène* 1 : comparaison des cartes de coût et des trajectoires obtenues après placement avec les fonctions de coût (a) *DetFim* et (b) *MixedCosts2* dans le cas de 3 balises. La trajectoire bleue correspond à la vérité terrain, alors que les trajectoires rouges et noires correspondent aux meilleures et moins bonnes trajectoires estimées par trilatération.

l'homogénéité de la couverture du terrain. En effet, la fonction de coût *MixedCosts2* proposée, qui combine les critères de couverture et de précision de localisation avec une pondération, permet de favoriser en premier une couverture homogène du terrain avant un bonne précision de localisation, en donnant un poids plus élevé au premier terme. Ainsi les zones non-couvertes sont moins nombreuses et plus petites, ce qui réduit les intervalles de temps où le robot ne peut être localisé ou est mal localisé. Ceci a pour effet d'entraîner une légère amélioration de la précision de localisation sur l'ensemble de la trajectoire, comme on peut le voir à la Figure 5.14.

Finalement, en comparant les valeurs dans les tableaux 5.3 et 5.4, on s'aperçoit que les placements avec optimisation (GA) basés sur les fonctions de coût *nRanges* et *MixedCosts2* donnent dans la majorité des cas des erreurs moyennes de localisation plus faible que pour les autres placements. En regardant la Figure 5.14 on peut ainsi voir que les courbes correspondantes aux deux fonctions de coût *nRanges* (bleue avec croix) et *MixedCosts2* (rouge avec étoiles) ne présentent aucun pic notable et conservent des erreurs de localisation assez faibles.

La fonction de coût *MixedCosts2* donne des résultats légèrement meilleurs que la fonction de coût *nRanges*, mais au prix d'un temps d'exécution plus important pour trouver le placement des balises, comme on peut le voir en comparant les valeurs dans la dernière colonne des tableaux 5.3 et 5.4. Ainsi, la fonction de coût *nRanges* malgré sa simplicité et la non prise en compte du critère de précision de localisation, permet déjà d'obtenir des placements de balises qui améliorent la précision de localisation du robot. Ces résultats confirment donc qu'en sélectionnant une fonction de coût adéquate pour l'algorithme d'optimisation, il est possible d'améliorer la précision de localisation d'un robot par trilatération par rapport à un placement uniforme ou aléatoire. Le placement optimisé des balises présente donc un intérêt et un avantage pratique pour la localisation d'un robot mobile dans son espace de travail, et ceci d'autant plus que l'environnement dans lequel le robot évolue est complexe.

Malgré une amélioration de la localisation grâce à un algorithme de placement optimisé des balises, nous verrons dans la section suivante que les fonctions de coût actuelles ne sont pas aussi bien adaptées que cela à d'autres algorithmes de localisation plus évolués comme le filtre de




(a) GA

(b) RAND

(c) UNIF

FIGURE 5.16 – *Scène* 1o : comparaison des cartes de coût et des trajectoires pour les placements (a) GA, (b) RAND, (c) UNIF, avec 7 balises et la fonction de coût *MixedCosts2*. La trajectoire bleue correspond à la vérité terrain, alors que les trajectoires rouges et noires correspondent aux meilleures et moins bonnes trajectoires estimées par trilatération.

Kalman. Nous allons donc maintenant montrer la limitation des fonctions de coût basées sur la trilatération, et motiver le développement de nouvelles fonctions plus adaptées aux algorithmes de localisation par fusion de données afin d'améliorer encore les placements obtenus.

## 5.6 Limites des fonctions de coût basées sur la trilatération

Comme déjà indiqué auparavant dans le chapitre, la majorité des fonctions de coût développées et testées dans la section précédente sont basées sur un critère de localisation par trilatération, qui est un algorithme utilisant uniquement des mesures de distance. Cependant, il est possible que le dispositif à localiser, dans notre cas un robot mobile, embarque également toute une série d'autres capteurs, dont notamment des odomètres. Dans le cas de notre robot mobile, l'intégration des données fournies par les codeurs incrémentaux des roues permet d'obtenir une estimation de la position du robot. Un autre exemple serait celui d'un dispositif

combinant une balise RF avec une centrale inertielle, où cette dernière fournirait, par intégration de l'accélération et de la vitesse angulaire, une estimation de la position, de manière similaire aux codeurs incrémentaux du robot.

En général, afin de localiser un dispositif, il est préférable d'utiliser et de combiner un plus grand nombre de capteurs possible fournissant des informations complémentaires, voir redondantes. Dans ce cas-là, on a souvent recours à des algorithmes de fusion de données comme le filtre de Kalman ou le filtre particulaire pour réaliser l'algorithme de localisation. Des versions de ces algorithmes ont été présentées au chapitre 3. Il est donc intéressant de se poser la question si l'utilisation d'un algorithme de placement optimal des balises est toujours aussi utile et bénéfique quand le robot est localisé avec un algorithme de fusion de données plus élaboré qu'avec un simple algorithme de trilatération.

### 5.6.1 Présentation des simulations

Afin de réaliser la comparaison des performances de localisation entre différents algorithmes de localisation, le même principe que dans la section 5.5.2 est utilisé. À la différence qu'ici, la précision de localisation de l'algorithme de trilatération, nommé *Trilat* par la suite, est comparée à deux implémentations différentes d'un filtre de Kalman étendu (EKF).

La première version du filtre, nommée *T-EKF*, fusionne l'odométrie des roues (étape de prédiction) avec les positions obtenues par l'algorithme de trilatération (étape de correction). La seconde version, nommée *R-EKF*, fusionne l'odométrie des roues directement avec les mesures de distance disponibles à un instant donné.

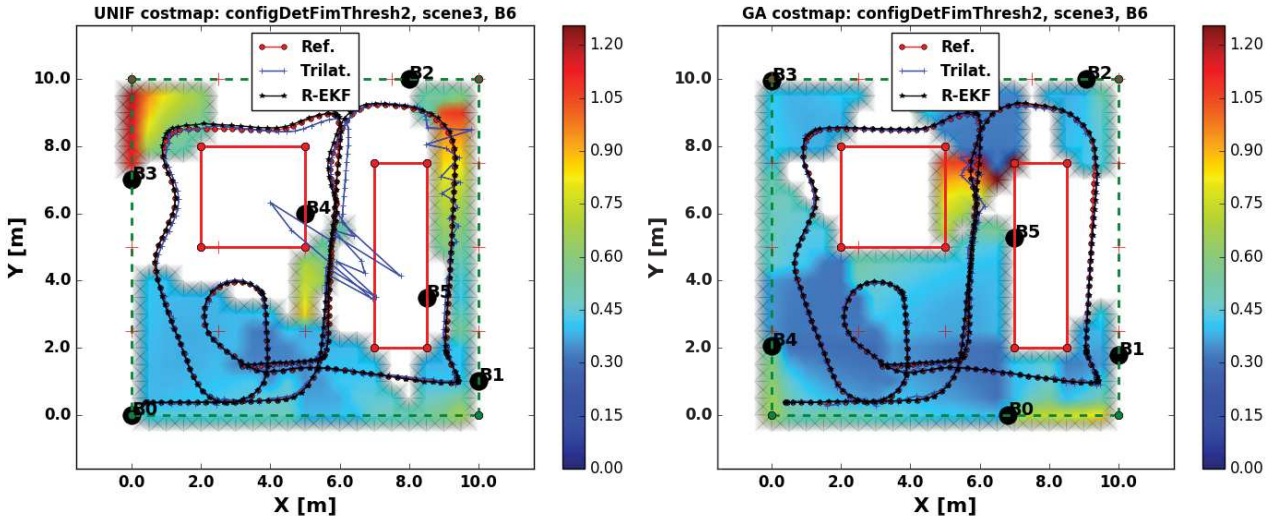
Pour référence, les résultats obtenus sont aussi comparés avec un placement uniforme (UNIF) des balises. La métrique utilisée pour comparer les performances de localisation est l'erreur RMS, donnée par l'équation (5.8). Pour chaque placement de balises, $S = 200$ simulations sont réalisées, et la moyenne, la déviation standard, le minimum et le maximum des erreurs sont calculés. Les différents paramètres utilisés pour réaliser les simulations sont :

- Variance de l'erreur des mesures de distances : $\sigma_r^2 = 1.1\text{e}^{-3}$ m$^2$,
- Paramètre de dépendance à la distance : $\eta = 0.05$,
- Portée maximale des balises : $r_{max} = 10$ m,
- Seuil sur le $\left|\text{FIM}\right|$ : $t_{\text{FIM}} = 1296$ correspondant à une précision de $\pm 0.5$ m,
- Variance du bruit Gaussien de moyenne nulle appliquée sur les vitesses des roues du robot : $\sigma_w^2 = 1.0\text{e}^{-6}$ m$^2$s$^{-2}$,
- Biais modélisant une différence de rayon entre la roue droite et la gauche : $\delta_r = 0.5$ mm.

Avant de comparer l'impact des placements optimisés sur les trois algorithmes de localisation, nous allons d'abord comparer le placement uniforme avec le placement optimisé sur les performances des algorithmes.

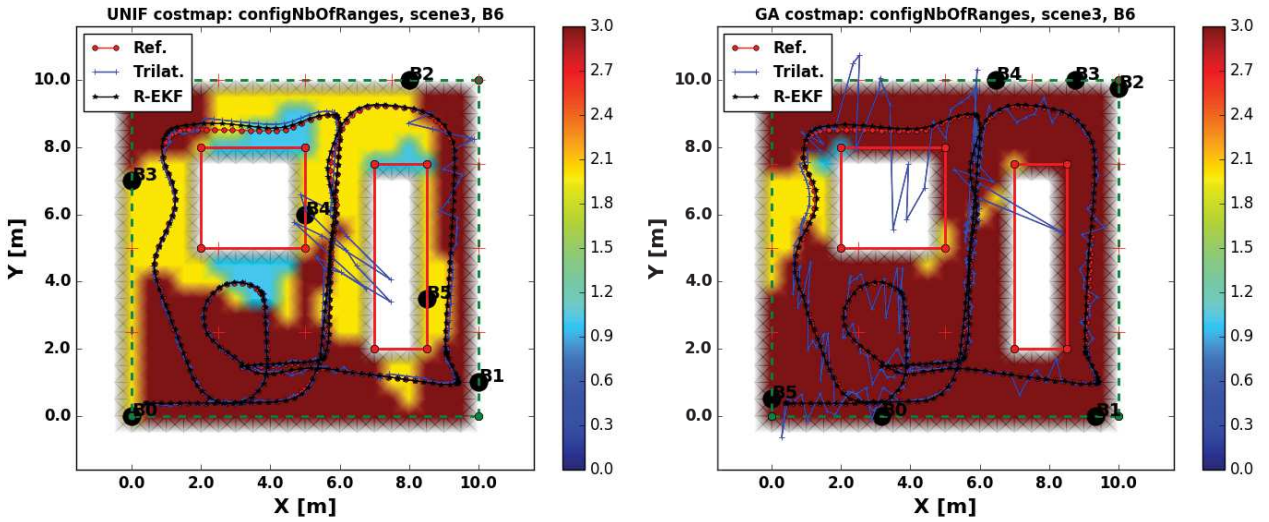
### 5.6.2 Comparaison des placements uniforme et optimisé

Les cartes de coût ainsi que les meilleures trajectoires estimées par les trois algorithmes de localisation, obtenues pour la *Scène* 1o, sont présentées à la Figure 5.17 pour la fonction de coût *DetFimThresh* et à la Figure 5.18 pour la fonction *nRanges*. Les figures 5.17a et 5.17b correspondent respectivement à un placement uniforme (UNIF) et à un placement après optimisation (GA). Dans ces deux figures, les couleurs utilisées pour la carte de coût correspondent à la précision de localisation, donnée par l'échelle en regard. Ainsi une zone en bleu correspond à une zone avec une bonne précision de localisation. Comme l'objectif de la fonction de coût *DetFimThresh* est de couvrir le maximum de terrain avec une précision de localisation inférieure à $d = \pm 0.5$ m, cela revient, en termes de couleur, à vouloir couvrir au maximum le




(a) UNIF (b) GA

FIGURE 5.17 – Cartes des coûts obtenues avec la fonction de coût *DetFimThresh* pour les placements UNIF (a) et GA (b) pour la *Scène* 1o avec 6 balises




(a) UNIF (b) GA

FIGURE 5.18 – Cartes des coûts obtenues avec la fonction de coût *nRanges* pour les placements UNIF (a) et GA (b) pour la *Scène* 1o avec 6 balises

terrain en bleu. Les lignes rouges correspondent aux trajectoires de référence alors que les lignes bleues et noires correspondent respectivement aux trajectoires estimées par l'algorithme *Trilat* et *R-EKF*.

Avec le placement uniforme, on s'aperçoit à la Figure 5.17a qu'une grande partie du terrain est remplie en blanc. Ces zones blanches correspondent à des positions où il y a moins de trois mesures de distance disponibles. Au contraire, pour le placement GA à la Figure 5.17b, la majorité du terrain est remplie de bleu clair ce qui correspond à une précision de localisation en-dessous de $\pm 0.5$ m en accord avec le seuil $t_{\text{FIM}}$ imposé. De plus, le nombre et la taille des zones blanches est plus faible, ce qui se traduit par une réduction des zones non-couvertes. Ainsi, en comparant les deux figures, on constate que l'algorithme d'optimisation permet bien d'augmenter la couverture du terrain ainsi que les zones avec une bonne précision de localisation.

Une zone notable permettant de bien visualiser le phénomène au niveau de la *Scène* 1o se situe au milieu du terrain entre les deux obstacles. En effet, à cet endroit on voit bien comment, avec le placement uniforme, la zone est mal couverte (couleur vert/jaune/blanc) et que la trajectoire bleue estimée par l'algorithme *Trilat* présente de fortes oscillations. Alors qu'au contraire pour le placement GA, la zone centrale est plutôt bien couverte et ne présente aucune zone blanche, ce qui permet ainsi de réduire le nombre et la taille des oscillations précédemment observées pour la trajectoire estimée par l'algorithme *Trilat*. Une explication possible pour les oscillations de la trajectoire bleue est que la configuration géométrique des balises dans cette zone est très mauvaise. En effet, les balises fournissant les mesures de distance (B0, B4, B2) se retrouvent presque sur la même ligne, ce qui entraîne un mauvais conditionnement du problème de trilatération.

Les même observations peuvent être faites avec la fonction de coût *nRanges* de la Figure 5.18, en comparant les figures 5.18a et 5.18b. Pour ces deux figures, les couleurs correspondent au nombre de balises disponibles à une position donnée (bleu clair=1, jaune=2 et rouge=3 et plus). Après un placement GA, qui correspond à la Figure 5.18b, la plupart du terrain possède au moins trois mesures de distance disponibles (zones en rouge) à l'exception de quelques zones sur la gauche où seulement une ou deux mesures sont disponibles. Alors que pour le placement uniforme, visible à la Figure 5.18a, on remarque que seule la moitié du terrain possède trois mesures de distance.

À la Figure 5.19 qui compare les erreurs de trajectoire, on observe que le placement uniforme présente dans presque tous les cas la plus grande erreur de trajectoire moyenne par rapport aux placements GA, et ce quel que soit la méthode de localisation utilisée. Ce constat est d'autant plus vrai que le nombre de balises utilisé est petit, comme cela est visible à la Figure 5.20 qui montre les erreurs de trajectoires pour différents nombres de balises. Ainsi, quand le nombre de balises augmente, l'écart entre les placements UNIF et GA diminue logiquement, ce qui limite l'intérêt d'un algorithme de placement des balises.

Enfin, parmi les fonctions de coûts utilisées pour les placement GA, la fonction de coût *DetFimThresh* semble être celle donnant les meilleurs résultats devant *MixedCosts2* puis *nRanges* respectivement. Cette tendance s'observe sur la Figure 5.19 où les erreurs moyennes diminuent de la gauche vers la droite et ceci indépendamment de la méthode de localisation utilisée.

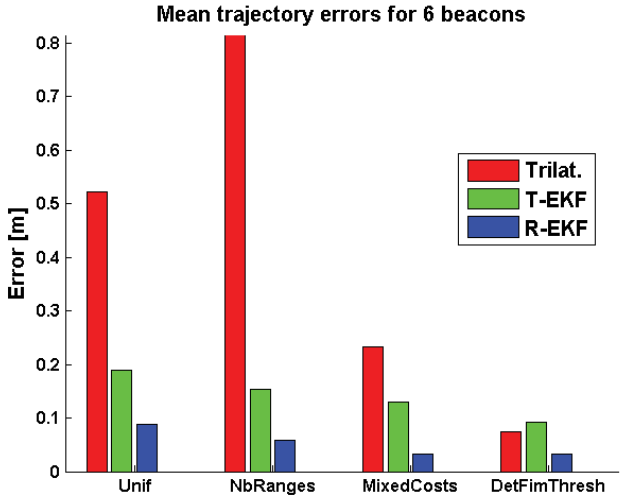



FIGURE 5.19 – Comparaison des erreurs de trajectoire moyennes des 3 algorithmes de localisation : *Trilat*, *T-EKF*, *R-EKF* pour les placements UNIF et GA (fonctions de coût : *nRanges*, *MixedCosts2* et *DetFimThresh*) avec 6 balises

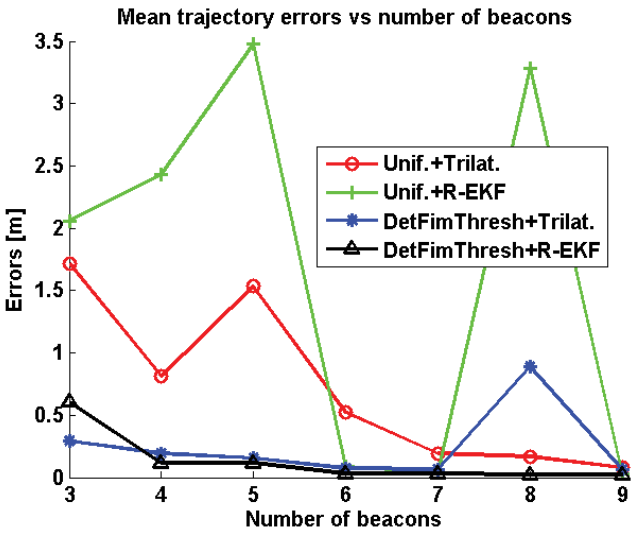



FIGURE 5.20 – Évolution des erreurs de trajectoire moyennes des algorithmes de localisation *Trilat* et *R-EKF*, pour les placements UNIF et GA (fonction de coût *DetFimThresh*)

### 5.6.3 Performances de localisation

| Erreurs<br>Méthodes | Moyenne<br>[m] | Std<br>[m] | Max<br>[m] | Min<br>[m] |
|---|---|---|---|---|
| *Trilat* | 0.549 | 0.210 | 1.603 | 0.271 |
| *T-EKF* | 0.194 | 0.056 | 0.465 | 0.118 |
| *R-EKF* | 0.081 | 0.014 | 0.112 | 0.040 |

TABLEAU 5.5 – Erreurs de localisation obtenues avec un placement uniforme de 6 balises pour les 3 algorithmes de localisation

| Erreurs<br>Méthodes | Moyenne<br>[m] | Std<br>[m] | Max<br>[m] | Min<br>[m] |
|---|---|---|---|---|
| *Trilat* | 0.766 | 0.028 | 0.858 | 0.704 |
| *T-EKF* | 0.219 | 0.050 | 0.391 | 0.147 |
| *R-EKF* | 0.046 | 0.007 | 0.064 | 0.029 |

TABLEAU 5.6 – Erreurs de localisation obtenues avec le placement GA-*nRanges* de 6 balises pour les 3 algorithmes de localisation

| Erreurs<br>Méthodes | Moyenne<br>[m] | Std<br>[m] | Max<br>[m] | Min<br>[m] |
|---|---|---|---|---|
| *Trilat.* | 0.47 | 0.017 | 0.515 | 0.437 |
| *T-EKF* | 0.154 | 0.050 | 0.410 | 0.091 |
| *R-EKF* | 0.041 | 0.006 | 0.061 | 0.028 |

TABLEAU 5.7 – Erreurs de localisation obtenues avec placement GA-*MixedCosts2* de 6 balises pour les 3 algorithmes de localisation

Les tableaux 5.5, 5.6, 5.7 et 5.8 regroupent les erreurs de localisation obtenues pour les trois algorithmes de localisation, pour les placements UNIF, GA-*nRanges*, GA-*MixedCosts2* et GA-*DetFimThresh* respectivement, pour la *Scène* 1o avec 6 balises. En comparant les résultats, on

| Erreurs<br>Méthodes | Moyenne<br>[m] | Std<br>[m] | Max<br>[m] | Min<br>[m] |
|---|---|---|---|---|
| *Trilat.* | 0.071 | 0.004 | 0.087 | 0.063 |
| *T-EKF* | 0.094 | 0.009 | 0.117 | 0.074 |
| *R-EKF* | 0.030 | 0.004 | 0.042 | 0.022 |

Tableau 5.8 – Erreurs de localisation obtenues avec placement GA-*DetFimThresh* de 6 balises pour les 3 algorithmes de localisation

s'aperçoit que la moyenne des erreurs est toujours plus faible pour l'algorithme de localisation *R-EKF*. Aussi, la plupart du temps, l'algorithme *T-EKF* donne de meilleurs résultats que l'algorithme *Trilat*. Ces résultats peuvent paraître évident, quand l'on sait que l'algorithme EKF ne s'appuie pas seulement sur les mesures de distance comme les autres méthodes, mais tire aussi avantage de l'odométrie des roues. Mais ils permettent aussi de montrer que les critères basés uniquement sur la trilatération utilisés pour l'optimisation ne sont pas aussi bien adaptés à d'autres algorithmes de localisation.

L'utilisation d'une fonction de coût basée sur la trilatération pour obtenir le placement des balises améliore principalement la localisation par trilatération, mais a beaucoup moins d'impact sur les performances de localisation avec un filtre de Kalman. Ceci peut se voir sur la Figure 5.19, où les erreurs de localisation obtenues avec l'algorithme *Trilat* (barres rouges) sont largement réduites (les hauteurs sont divisées par environ 2 et 5) entre un placement uniforme et les placements GA-*MixedCosts2* et GA-*DetFimThresh*. Au contraire, dans le cas des algorithmes EKF (barres vertes et bleues), bien que les erreurs diminuent légèrement entre le placement uniforme et optimisé comme attendu, l'ampleur de la réduction reste beaucoup plus faible. Ainsi, la localisation par l'algorithme *Trilat* bénéficie beaucoup plus du placement optimisé des balises que les algorithmes *T-EKF* et *R-EKF*.

Sur la Figure 5.20, qui montre l'évolution de la moyenne des erreurs de trajectoire pour un nombre croissant de balises, on observe qu'avec le placement GA-*DetFimThresh*, les erreurs obtenues avec l'algorithme *Trilat* (ligne bleue) sont plus importantes que celles pour l'algorithme *R-EKF* (ligne noire) exceptées dans le cas de 3 balises. On s'aperçoit aussi que pour une erreur de trajectoire donnée, il est souvent nécessaire d'utiliser une ou deux balises supplémentaires avec l'algorithme *Trilat* pour obtenir les mêmes performances qu'avec l'algorithme *R-EKF*. Cette observation suggère qu'en utilisant une fonction de coût plus adaptée à un algorithme EKF pour résoudre le problème du placement des balises, on pourrait réduire davantage le nombre de balises utilisées pour localiser le robot.

En effet, l'algorithme de trilatération possède deux principaux désavantages par rapport au filtre EKF. Le premier étant qu'avec moins de trois mesures de distance disponibles à une position donnée, il n'est pas possible d'obtenir une estimation de position avec la trilatération, ce qui peut laisser le robot totalement aveugle pendant un certain temps. Le fait de ne pas avoir d'information de position pendant un certain temps peut bien souvent être préjudiciable pour le robot comme par exemple lorsque celui-ci utilise un contrôleur de position. Le second défaut est que l'algorithme de trilatération est beaucoup plus sensible à la configuration géométrique des balises qui influence le critère de précision de localisation. Dans certaines configurations des balises, la trilatération peut ainsi fournir une solution présentant une très grande erreur. En comparaison, le filtre EKF est beaucoup moins sensible à ces situations. Grâce à l'odométrie des roues, le filtre fournit une estimation de position même sans mesures de distance. Ainsi, si aucune mesure de distance n'est disponible, le robot peut au moins compter sur l'odométrie pendant de courtes périodes de temps. De plus, dans le cas de l'algorithme *R-EKF*, il n'est pas nécessaire d'avoir trois mesures de distance à une position donnée pour avoir une estimation de position. Les mesures de distances sont intégrées une par une au fur et à mesure de leurs

disponibilités.

En regardant les figures 5.17a et 5.17b, on peut voir que la trajectoire estimée par l'algorithme *R-EKF* est toujours très proche de la trajectoire de référence, alors que pour l'algorithme *Trilat*, on observe des dérives et des oscillations dans les zones non ou mal couvertes. Ceci met donc en lumière les limitations des fonctions de coût développées sur la base de l'algorithme de trilatération, qui est un algorithme de localisation statique au sens où il n'intègre pas les mesures d'odométrie. Ainsi dans le cas où l'on cherche à optimiser le placement des balises pour une position donnée ou pour une cible mobile localisée par trilatération, les fonctions de coût basées sur la trilatération sont parfaitement adaptées. Par contre, dans le cas d'un robot mobile équipé de divers capteurs et localisé par un filtre EKF, l'apport de ces fonctions de coût est moindre. En effet, les informations supplémentaires fournies par les capteurs et par le filtre n'étant pas prises en compte par les fonctions de coût basées sur la trilatération, l'algorithme de placement des balises ne peut pas en profiter pour améliorer d'avantage la solution retournée. Il serait donc intéressant d'envisager le développement de nouvelles fonctions de coût capables de prendre en compte ces informations supplémentaires afin d'améliorer encore un peu plus le positionnement et surtout de réduire le nombre de balises utilisées.

### 5.6.4 Conclusion

Grâce aux simulations précédentes, il a été montré que l'amélioration de la précision de localisation n'est pas aussi importante pour un filtre EKF que pour un algorithme de trilatération quand les positions des balises sont obtenues par l'algorithme génétique avec des critères basés sur la trilatération. En effet, le filtre de Kalman ou tout autre algorithme de fusion de données, permet l'incorporation d'informations additionnelles provenant de divers capteurs. Ceci permet de ne plus dépendre uniquement des mesures des balises, et d'être moins sensible aux zones du terrain mal couvertes ou aux mauvaises configurations des balises comme c'est le cas avec la trilatération. L'utilisation de l'odométrie fournie par les codeurs incrémentaux des roues permet de localiser le robot sur une courte période de temps avec une précision suffisante dans une région du terrain avec une mauvaise couverture ou configuration des balises. Il est donc possible d'assouplir l'exigence d'avoir une bonne couverture et une bonne précision de localisation en chaque point du terrain. En ce sens, il serait plus intéressant d'imposer ces contraintes sur une zone du terrain dont la taille serait en relation avec la distance que le robot peut parcourir en comptant uniquement sur l'odométrie et avec une précision souhaitée. Un autre avantage des techniques de fusion par rapport à la trilatération est qu'il n'est pas nécessaire d'avoir au moins trois mesures de distance (dans le cas 2D) pour obtenir une information de position du robot. En effet, l'algorithme de trilatération nécessite au minimum trois mesures de distance pour calculer une estimation de la position, alors qu'un algorithme de fusion de données n'a pas de contraintes sur le nombre de mesures disponibles à un instant donné.

Une perspective intéressante à suivre est donc de développer de nouveaux critères plus adaptés à un algorithme de localisation par fusion de données, qui prendraient en compte les informations additionnelles comme les mesures de l'odométrie du robot. Ceci permettrait sans doute d'améliorer encore un peu plus le placement des balises et surtout de pouvoir réduire davantage le nombre de balises utilisées tout en gardant les mêmes performances de localisation.

L'une des pistes possibles à explorer pour la réalisation d'une nouvelle fonction de coût serait de développer un critère basé sur la matrice d'innovation ou de covariance du filtre de Kalman. Et ainsi, de façon similaire à ce qui est fait pour la matrice d'information de Fisher, de minimiser son déterminant ou de maximiser son conditionnement.

Jusqu'ici, nous avons présenté et évalué des méthodes et leurs résultats associés permettant de placer les balises. Cependant, comme énoncé dans l'introduction du chapitre, il est aussi

essentiel de minimiser le nombre de balises utilisées pour localiser le robot. La section qui suit présente donc une méthode permettant d'obtenir simultanément le nombre et la position des balises nécessaires.

## 5.7 Optimisation simultanée du nombre et de la position des balises

### 5.7.1 Fonction de coût *maximin*

Dans les deux sections précédentes, seule la position des balises était prise en compte dans le problème d'optimisation. Il a ainsi été constaté que la position des balises est importante pour la précision de localisation et la bonne couverture de la zone de travail. Cependant pour des raisons pratiques et de coût, il importe également de minimiser le nombre de balises à utiliser. Pour automatiser ce choix, on peut prendre en compte le nombre de balises comme une variable, et non plus comme une entrée fixe du problème. Le problème d'optimisation se fera alors sur le nombre ainsi que la position des balises. L'ajout en apparence simple du nombre de balises, implique en réalité une plus grande complexité au problème et rend donc sa résolution plus difficile.

Afin de ne pas avoir à changer complètement l'algorithme d'optimisation et le critère à minimiser, une solution permettant de conserver l'algorithme génétique a été retenue. Ainsi, à l'exception de quelques changements mineurs sur l'algorithme génétique, le seul changement notable concerne la fonction de coût. Pour cela, nous avons repris la fonction *maximin*, proposée par Balling [4], qui a été développée afin de résoudre des problèmes d'optimisation évolutionnaire multicritère. Cette fonction de coût est donc parfaitement bien adaptée à notre cas où l'on cherche à minimiser deux critères à partir d'un algorithme génétique. Les deux critères considérés ici étant le nombre de balises, et le critère combinant la couverture du terrain et la précision de localisation.

La fonction de coût *maximin* est basée sur le principe de dominance. Soit $m = 2$, le nombre de critères à minimiser, et $N$, le nombre d'individus de la population. On suppose pour la suite que les valeurs (scores) des critères sont normalisées. Si l'on définit la valeur normalisée du critère $k$ pour le $i^{\text{ème}}$ individu de la population par $f_{ki}$, on dit alors que le $i^{\text{ème}}$ individu est faiblement dominé par le $j^{\text{ème}}$ individu si :

$$f_{ki} \geq f_{kj},\ \forall k \in [1, m] \tag{5.10}$$

avec $f_{ki} \in [0, 1]$, $\forall k \in [1, m]$ et $\forall i \in [1, N]$.

L'individu $j$ domine l'individu $i$ si en plus de l'équation précédente, on a :

$$f_{ki} > f_{kj},\ \text{pour au moins un } k \in [1, m]$$

Deux individus $i$ et $j$ sont dits dupliqués si :

$$f_{ki} = f_{kj},\ \forall k \in [1, m]$$

Deux individus ne satisfaisant pas l'équation précédente sont dits distincts.

L'équation (5.10) peut se réécrire sous la forme :

$$\min_k (f_{ki} - f_{kj}) \geq 0$$

L'individu $i$ d'une génération est faiblement dominé par un autre individu de la population si :

$$\max_{j \neq i} \Big( \min_k (f_{ki} - f_{kj}) \Big) \geq 0$$

Ainsi la fonction *maximin* du $i^{\text{ème}}$ individu est donnée par [4] :

$$f_i = \max_{j \neq i, j \in P} \Big( \min_k (f_{ki} - f_{kj}) \Big) \tag{5.11}$$

Le problème d'optimisation à résoudre avec l'algorithme génétique correspond donc à :

$$\min f_i = \min \Big[ \max_{j \neq i, j \in P} \Big( \min_k (f_{ki} - f_{kj}) \Big) \Big] \tag{5.12}$$

$P$ étant l'ensemble des individus non dominés dans la population.

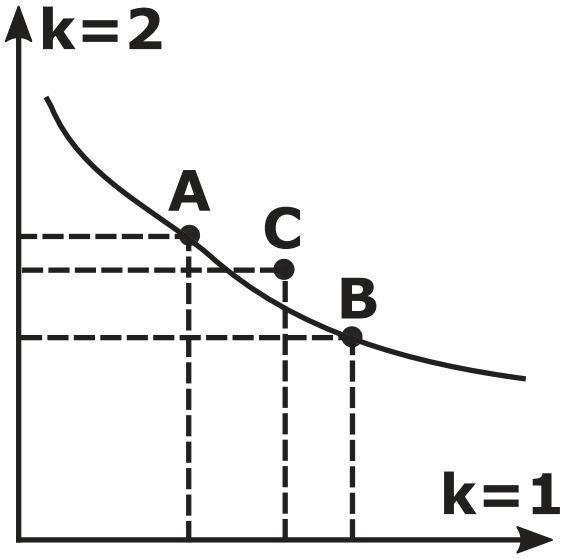



FIGURE 5.21 – Principe de fonctionnement de la fonction *maximin* sur trois individus A, B et C

Suivant la valeur de la fonction *maximin*, $f_i$, l'individu $i$ est dit :

— Dominé si $f_i > 0$,

— Non-dominé si $f_i < 0$,

— Faiblement dominé si $f_i = 0$.

La fonction de coût *maximin* favorise ainsi la diversité et pénalise le groupement des individus non-dominés. À la Figure 5.21, qui montre les valeurs des deux critères, $k = 1, 2$, pour trois individus A, B et C, on voit que l'individu C est dominé. Sa fonction *maximin* est donc positive. Au contraire, les fonctions *maximin* des individus A et B sont négatives car il s'agit d'individus non-dominés. La fonction *maximin* va donc favoriser les individus non-dominés et dans le même temps faire descendre la courbe correspondant aux individus non-dominés vers le bas. Cette courbe correspond aussi au front (diagramme) de Pareto qui représente la frontière entre les différents critères optimisés.

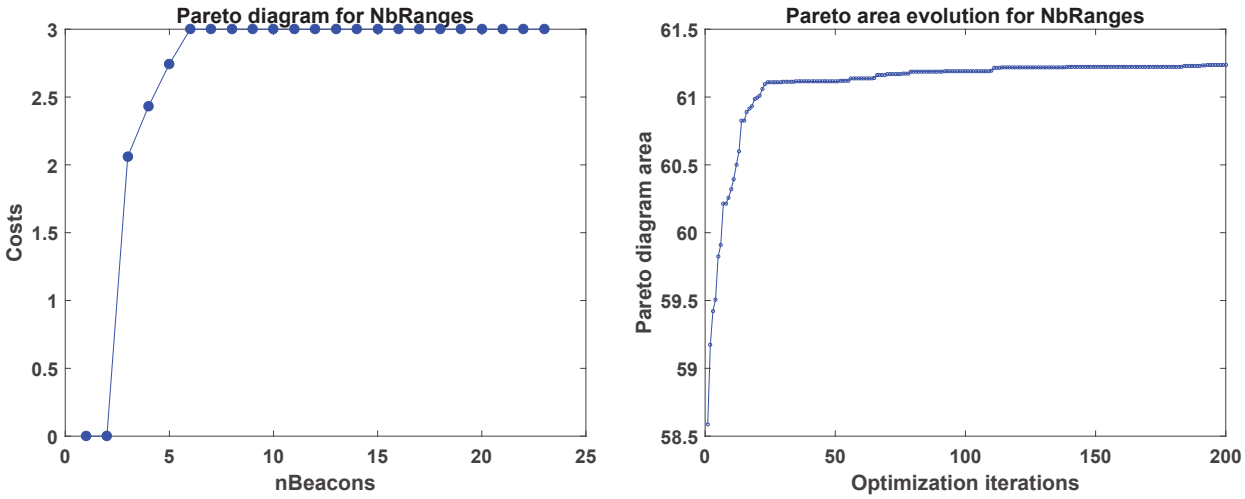
Nous allons maintenant présenter les résultats obtenus avec la fonction de coût *maximin* en utilisant notamment ces diagrammes de Pareto.

### 5.7.2 Résultats

Le temps consacré au développement de la fonction *maximin* n'ayant pas été très important, les expériences associées sont restées sommaires. Néanmoins, les premiers résultats ayant été obtenus sont présentés ci-dessous.

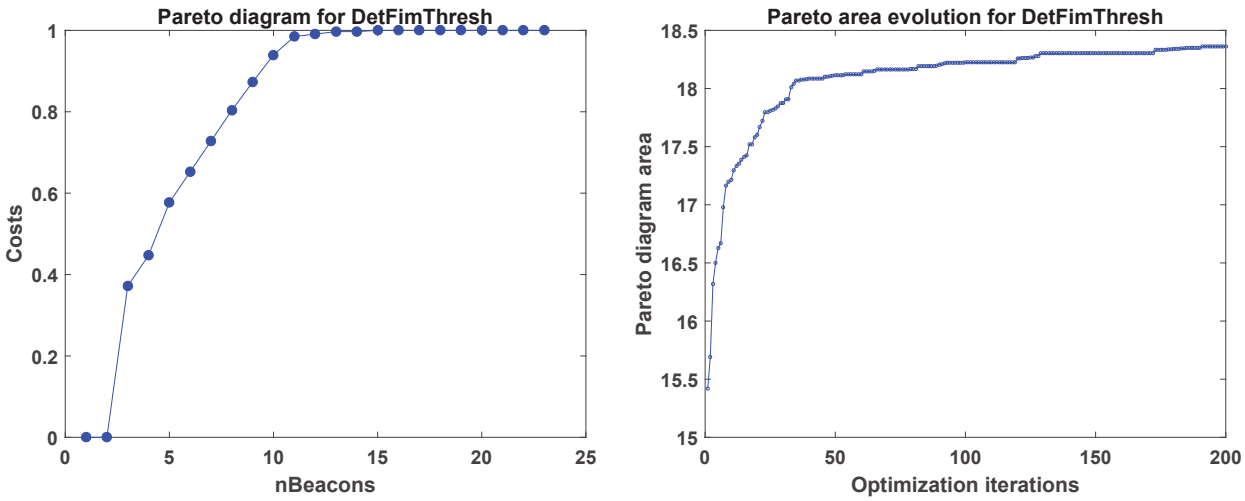
La fonction *maximin* a donc été testée sur la Scène 2 pour deux critères de couverture/localisation différents. Ces deux critères correspondent aux fonctions de coût *nRanges* et *DetFimThresh*.

Ainsi, dans la première simulation, la fonction *maximin* minimise simultanément le nombre de balises et la fonction de coût *nRanges*. Alors que dans la deuxième simulation, c'est le nombre de balises et la fonction *DetFimThresh* qui sont minimisés. Les résultats obtenus avec la fonction *nRanges* sont visibles à la Figure 5.22, alors que ceux associés à la fonction *DetFimThresh* sont donnés à la Figure 5.23. Pour chaque simulation, le diagramme de Pareto obtenu à partir des individus de la génération finale ainsi que l'évolution de l'aire du diagramme de Pareto sont affichés.




(a) Diagramme de Pareto (b) Aire du diagramme

FIGURE 5.22 – Diagramme de Pareto (a) et évolution de son aire au fil des itérations (b) avec le critère de couverture *nRanges* pour la Scène 2




(a) Diagramme de Pareto (b) Aire du diagramme

FIGURE 5.23 – Diagramme de Pareto (a) et évolution de son aire au fil des itérations (b) avec le critère de localisation *DetFimThresh* pour la Scène 2

En regardant l'évolution de l'aire du diagramme de Pareto sur les figures 5.22b et 5.23b, on constate que celle-ci augmente au fil des itérations. Ceci signifie, qu'au fil des générations, les solutions (individus) satisfont de mieux en mieux les deux critères. On note cependant, que

pour la fonction de coût *nRanges*, l'aire augmente plus rapidement et a tendance à stagner au bout de la 100ème itération. Ceci est la conséquence que le critère *nRanges* est beaucoup plus facile à remplir que le critère *DetFimThresh*. En effet, si l'on regarde la Figure 5.22a, on constate qu'à partir de 6 balises, le critère est déjà au maximum. Les seules améliorations possibles restent donc pour 3, 4 et 5 balises. Par contre, pour le critère *DetFimThresh*, à la Figure 5.23a, on voit que le critère n'est atteint complètement qu'à partir de 13 balises. Il est donc plus difficile de satisfaire le critère *DetFimThresh*, et l'aire du diagramme de Pareto évolue donc plus progressivement au fur et à mesure des itérations.

D'un point de vue pratique, et au regard des diagrammes de Pareto des figures 5.22a et 5.23a, on constate que le nombre adéquat de balises à déployer sur la Scène 2 se situe entre 8 et 12. En effet, à partir de 6 balises, on est certain de couvrir tout le terrain avec au moins 3 mesures de distance, et, entre 8 et 12 balises, le pourcentage de terrain couvert par la précision souhaitée est compris entre 80 et 100%.

Ces résultats préliminaires sur la fonction *maximin* permettent déjà de montrer l'intérêt d'un tel algorithme. En effet, avec cette nouvelle formulation, il est possible d'obtenir automatiquement une estimation ou un intervalle du nombre de balises permettant de satisfaire au mieux le critère de localisation souhaité, ainsi que la configuration des balises permettant associée au résultat. Cependant, le temps de calcul élevé d'une seule itération de l'algorithme impose d'utiliser un nombre d'itérations de l'ordre de la centaine pour obtenir un résultat dans un temps raisonnable.

## 5.8 Conclusion

Dans le cadre de la localisation d'un robot mobile dans son environnement, il est essentiel de choisir des capteurs adaptés aux performances voulues, ainsi que de développer des techniques d'estimation de la position du robot les plus robustes et fiables possibles. Cependant, comme cela a été montré tout au long de ce chapitre, il est aussi nécessaire dans le cas de l'utilisation de balises, de les choisir en nombre adéquat et de les positionner correctement sur le terrain. Un algorithme de placement des balises permet ainsi de faciliter l'installation du système en minimisant le nombre balises, mais permet aussi d'améliorer la précision atteignable par le système de localisation sur l'ensemble du terrain. Ces deux critères étant concurrents, la solution correspondante au meilleur compromis est généralement retournée. Avoir recourt à un algorithme de positionnement des balises n'est donc pas une obligation, mais peut considérablement améliorer les performances globales du système.

En dehors de cas très simplifiés, le placement optimal des balises dans des scénarios réels reste un problème difficile. Il est en effet compliqué de trouver une solution adéquate au problème sans l'utilisation d'un certain nombre d'hypothèses restrictives. Pour cela, l'approche proposée dans ce chapitre s'appuie sur un algorithme génétique qui permet une modélisation plus réaliste et pratique du problème, mais n'offre aucune garantie d'obtenir la solution globale comme c'est le cas pour d'autres algorithmes. Cependant, l'algorithme génétique trouve le plus souvent une solution acceptable dans un temps raisonnable, tout en permettant une flexibilité dans la modélisation du problème.

Parmi les trois critères principaux à optimiser que sont la couverture, la précision de localisation et le nombre de balises utilisées, il est difficile de réunir les trois au sein d'une même fonction de coût, et donc d'obtenir une solution répondant au trois critères. Pour cela on considère généralement les sous-problèmes de minimisation du nombre de balises et de l'obtention du placement optimisant la couverture et la précision de localisation séparément. L'emploi d'une fonction de coût *maximin* avec l'algorithme génétique propose bien une solution à l'ensemble des trois critères, mais au prix d'un coût en termes de temps de calcul bien supérieur.

Dans le cas où l'on cherche uniquement le placement optimisé des balises, le choix et la formulation de la fonction de coût sont essentiels. En effet, bien que l'algorithme génétique permette d'obtenir une solution optimisée par rapport à la fonction de coût, cela ne veut pas dire la solution retournée permettra une bonne localisation. En regard des simulations menées dans ce chapitre, optimiser la fonction de coût ne fait pas tout, il faut surtout choisir celle permettant de correspondre aux attentes. Ainsi, bien que la solution obtenue soit optimale au sens du critère choisi, cette dernière peut échouer à permettre une localisation satisfaisante sur l'ensemble de l'espace de travail.

La majorité des fonctions de coût proposées dans la littérature sont basées sur un critère de localisation par trilatération. Bien que cet algorithme présente déjà de bonnes performances, il est bien plus courant d'utiliser un algorithme de filtrage de type Kalman par exemple, quand l'on possède différents capteurs équipant le robot à localiser. Les performances de localisation obtenues sont alors bien meilleures et permettent de mieux se prémunir contre une mauvaise couverture du terrain ou une mauvaise configuration des balises. Une piste intéressante à suivre afin d'améliorer encore plus la qualité du placement des balises serait donc de développer de nouvelles fonctions de coût prenant en compte ce type d'algorithmes de localisation et ainsi permettre l'utilisation d'informations provenant d'autres capteurs.

# CHAPITRE 6

# CONCLUSION GÉNÉRALE

Dans ce mémoire, nous avons proposé une méthode permettant de déployer un robot mobile dans un environnement inconnu, afin que ce dernier puisse effectuer des tâches en totale autonomie. La solution proposée repose sur l'utilisation de balises radiofréquence (RF) à très larges bandes (UWB) pour localiser le robot dans son espace de travail. Le déploiement du système s'effectue en deux phases. La première phase consiste à apprendre les limites du terrain et la position des balises. Quant à la seconde, elle correspond à la phase d'opération du robot où celui-ci remplit sa mission.

La réalisation d'un tel système fait appel à différentes thématiques de la robotique, mais celles principalement traitées dans ce travail sont liées au problème de la localisation et de la localisation et cartographie simultanées (SLAM). De par l'utilisation des balises RF, une autre thématique est abordée au cours de ces travaux. Il s'agit du problème du placement optimal de capteurs. Cette thématique, bien que moins spécifique à la robotique, entraîne dans notre cas un impact direct sur les performances du système. Nous allons maintenant passer en revue les différentes contributions dans les domaines associés, ainsi que les principaux résultats obtenus.

## 6.1 Contributions

Le premier défi à résoudre en appliquant la méthode de déploiement proposée, correspond à l'apprentissage des limites de la zone d'opération et de la position des balises. Ce problème est en fait une instance du SLAM, qui consiste à estimer la trajectoire du robot ou de l'utilisateur, tout en construisant la carte de l'environnement, représentée par la position des balises. Dans le cadre de ce travail, nous avons uniquement développé le cas où il s'agissait de reconstruire la trajectoire du robot et la position des balises dans un environnement 2D. Afin de résoudre ce problème du SLAM, un filtre de Kalman étendu (EKF) a été implémenté. Ce dernier permet de fusionner les informations fournies par les codeurs incrémentaux des roues du robot avec les mesures de distance fournies par les balises. Cependant, le fait d'employer des balises RF fournissant uniquement des mesures de distance impose d'adopter une procédure spécifique afin d'initialiser la position des balises dans le filtre. Après avoir présenté les différentes techniques déjà existantes dans la littérature, une nouvelle formulation est proposée. Cette nouvelle technique d'initialisation est basée sur le calcul de l'intersection de deux cercles à partir de deux mesures de distance prises à deux positions différentes du robot. Les deux solutions sont alors insérées dans le filtre sous la forme d'une mixture de gaussienne. La technique proposée est alors comparée à d'autres méthodes existantes. Les résultats obtenus montrent que l'approche proposée permet d'obtenir une précision similaire comparé aux méthodes les plus performantes

de la littérature, et ce, tant au niveau de la trajectoire du robot que de la position des balises. Le principal avantage de la méthode proposée se trouve plutôt dans le compromis entre le court délai d'insertion des balises dans le filtre et la complexité de calcul qui impacte le temps d'exécution de l'algorithme complet. Cependant, comme la majorité des méthodes de la littérature, le processus d'initialisation est sensible à la qualité des mesures de distance et aux positions du robot utilisées au cours de la procédure.

En supposant que la phase d'apprentissage se déroule correctement, avec une bonne estimation des limites du terrain, et surtout de la position des balises, seul un algorithme de localisation est nécessaire au robot pour qu'il puisse opérer dans sa zone de travail. Même s'il paraît évident qu'il est préférable de se localiser en fusionnant différentes sources d'informations, l'utilisation des mesures de distance permet également de se localiser en utilisant uniquement les mesures provenant des balises. Dans ce contexte-là, différentes méthodes de trilatération sont présentées. Au vu de la faible robustesse aux mesures aberrantes de la plupart des méthodes présentées, une nouvelle méthode de localisation est proposée. Cette méthode repose sur l'utilisation d'un test de faisabilité d'une inégalité linéaire matricielle (LMI), afin de déterminer si une mesure de distance est cohérente avec un intervalle donné. Ainsi, pour une position que l'on cherche à estimer, l'algorithme part d'un grand intervalle et teste pour chaque mesure de distance si elle est cohérente ou non. Si une ou un pourcentage des mesures n'est pas cohérent avec l'intervalle, ce dernier est écarté de la recherche. Dans le cas contraire, l'intervalle est subdivisé et la recherche se poursuit jusqu'à atteindre le plus petit intervalle cohérent possible. En comparant la méthode proposée avec d'autres méthodes de localisation par mesures de distance uniquement, il ressort que la méthode remplit très bien son rôle même avec de forts pourcentages de mesures aberrantes. Par contre, lorsque le bruit sur les mesures de distance est très faible, la méthode proposée n'atteint pas une aussi bonne précision que les autres méthodes. Le principal inconvénient de la méthode est son temps de calcul, d'un ordre de grandeur plus élevé que les autres méthodes. Cependant, la méthode proposée garantit, quand les mesures ne sont pas biaisées, que l'intervalle retourné contient bien la vraie solution, ce qui n'est pas le cas avec une méthode telle que RANSAC qui ne fournit aucune garantie sur le résultat.

Afin de localiser le robot en temps réel, tout en utilisant l'ensemble des informations fournies par les capteurs, un filtre EKF a été développé. Tout comme pour le SLAM, ce dernier fusionne les mesures des codeurs incrémentaux avec les mesures de distance provenant des balises. Mais à la différence du SLAM, seule la pose du robot est estimée ici. L'état de l'art consacré au chapitre correspondant, montre qu'il existe de nombreuses alternatives au filtre EKF pour localiser un robot. Cependant, parmi les différents algorithmes existants, le filtre EKF reste encore l'une des solutions les plus populaire pour sa formulation et ses bonnes performances notamment. Afin de confirmer cela, l'algorithme développé est comparé à un algorithme de trilatération robuste et à un algorithme d'optimisation incrémental faisant référence pour sa précision. Les résultats montrent que même si le filtre EKF n'atteint pas la précision de l'algorithme d'optimisation, il reste néanmoins compétitif. En plus de surpasser logiquement l'algorithme de trilatération, le test de Mahalanobis, réalisé lors de la fusion des mesures de distance, permet de rejeter efficacement la plupart des mesures erronées. En obtenant une précision de localisation de l'ordre d'un mètre, sur un jeu de données réelles dont l'écart-type des mesures de distance était de l'ordre de moins d'un mètre, il reste à confirmer que le système développé dans le cadre de ce projet, dont l'écart-type des mesures est de l'ordre 10 cm, parvient également à une précision de localisation du même ordre de grandeur.

Dans l'optique d'un déploiement du système sur tout type de terrain, il est nécessaire de prendre en compte la 3$^{\text{ème}}$ dimension dans les algorithmes. Pour cela, le filtre EKF de localisation en 2D est étendu à la 3D. Dans cette version améliorée de l'algorithme, les possibilités de fusionner les mesures d'un accéléromètre, d'un gyroscope, d'un magnétomètre et d'un GPS

sont aussi ajoutées. Cependant, par manque de jeux de données adéquats et de temps, aucune expérience avec une vérité terrain n'a pu être menée afin de valider le fonctionnement du filtre développé.

Dans le cas de l'utilisation de balises RF, le choix de l'algorithme de localisation n'est pas le seul facteur pouvant impacter la précision obtenue sur la position du robot. En effet, le positionnement ainsi que le nombre des balises utilisées pour localiser le robot influencent aussi les performances du système. Afin de traiter ce problème, une première formulation permettant d'obtenir le positionnement des balises est implémentée. Cette dernière utilise un algorithme génétique en combinaison avec différentes fonctions de coût basées sur la maximisation de la couverture du terrain et la maximisation de la précision de localisation. Parmi ces fonctions de coût, une nouvelle formulation est proposée. Cette dernière est basée sur le calcul du déterminant de la matrice de Fisher, et consiste à maximiser le pourcentage du terrain couvert par une précision de localisation souhaitée. Cette nouvelle formulation possède l'avantage de pouvoir spécifier la précision de localisation voulue sur le terrain ou même en chaque position du terrain. Les simulations menées confirment bien l'apport d'un algorithme de placement par rapport à des stratégies de placement uniforme et aléatoire, et ce quel que soit la fonction de coût. Cependant, l'impact des fonctions de coût sur la précision de localisation n'est pas le même quand on localise un robot par trilatération en utilisant les placements des balises associés. En effet, les placements de balises obtenus avec les fonctions basées uniquement sur la couverture ou maximisant la précision de localisation sur l'ensemble du terrain, bien que possédant de meilleurs coûts que les placements uniforme et aléatoire, ne garantissent pas forcément une bonne localisation du robot. Au contraire, les fonctions de coût basées sur la maximisation du pourcentage de terrain couvert par une précision donnée, permettent d'obtenir une couverture plus homogène et une meilleure configuration des balises, et par conséquent, présentent de meilleurs résultats de localisation.

L'inconvénient majeur des fonctions de coût développées provient du fait que le critère de précision de localisation est basé sur une localisation par trilatération. En pratique, le robot est généralement localisé à l'aide d'un algorithme de fusion de données tel que le filtre EKF. En fusionnant les informations de différents capteurs en plus des mesures de distance, le robot est alors capable de se déplacer et de se localiser pendant de courts instants sans avoir recourt aux mesures des balises. Les simulations comparant les résultats de localisation de différents placements optimisés des balises pour un algorithme de trilatération et deux implémentations d'un filtre EKF, montrent que l'apport de l'algorithme de placement n'est plus aussi évident avec un filtre EKF qu'il l'était avec un algorithme de trilatération. Ces constations ouvrent donc de nouvelles perspectives pour le développement de fonctions de coût mieux adaptées aux algorithmes de localisation utilisés en pratique.

En plus, d'obtenir le meilleur positionnement, la minimisation du nombre de balises a aussi son importance, en permettant de réduire les coûts du système notamment. Afin de prendre en compte cette variable supplémentaire dans le problème d'optimisation, une nouvelle fonction de coût introduite par Balling, appelée maximin, est utilisée au sein de l'algorithme génétique. Cette fonction de coût a été développée spécifiquement pour répondre au problème de l'optimisation multicritère. Les résultats préliminaires obtenus avec cette nouvelle formulation, montre la possibilité d'obtenir de façon automatique un front de Pareto entre le nombre de balises et le critère de précision de localisation qui permet de choisir plus facilement le nombre de balises à utiliser pour un pourcentage de couverture désiré.

De l'ensemble des résultats sur le placement des balises, on peut conclure que, même si un algorithme de placement ne va pas améliorer significativement la précision de localisation, son intérêt réside plutôt dans le fait qu'il permet d'éviter certaines configurations des balises pouvant être préjudiciable au système et d'assurer une bonne couverture de la zone, tout en

réduisant le nombre de balises à déployer.

## 6.2 Perspectives

Malgré le développement des principales fonctionnalités de la méthode de déploiement du robot, il reste encore un bon nombre de questions à élucider et de perspectives d'amélioration.

Premièrement, en ce qui concerne le placement optimal des balises, les résultats encourageants sur l'intérêt d'utiliser cette méthode au préalable du déploiement des balises, ne doivent pas occulter le fait que les fonctions de coût actuelles sont encore perfectibles. Afin de prendre en compte que le robot est localisé par un algorithme d'estimation permettant de fusionner les informations de différents capteurs, de nouvelles fonctions de coût doivent être développées afin d'améliorer encore un peu plus le placement des balises, et surtout de réduire d'avantage leur nombre.

L'utilisation de la fonction maximin est un premier pas vers une méthode "clé en main", qui permettrait à l'utilisateur d'obtenir de façon automatique le nombre et la position des balises à déployer sur le terrain. Il serait donc intéressant de poursuivre les recherches sur l'optimisation multicritère, mais aussi sur le moyen de rendre l'algorithme plus rapide et d'améliorer sa convergence vers une solution acceptable.

Comme cela a été entrepris pour la localisation, il est aussi nécessaire pour le SLAM d'estimer la trajectoire et la position des balises en 3D, et non plus seulement en 2D.

De plus, le fait de se baser sur le modèle d'odométrie du robot pour l'étape de prédiction, supposait implicitement que l'étape d'apprentissage se faisait par l'utilisateur en téléguidant le robot sur les limites du terrain. Cette opération peut s'avérer délicate dans certaines conditions et la qualité de la trajectoire du robot repose sur l'agilité de l'utilisateur à guider le robot correctement. Afin de faciliter cette opération, il serait préférable que l'utilisateur se déplace lui-même, tout en utilisant un dispositif portatif embarquant le calculateur et tous les capteurs nécessaires à la reconstruction de la trajectoire et de la position des balises. Une fois l'opération terminée, le dispositif portatif pourrait alors être simplement connecté au robot afin de lui transmettre les données, et même, resservir de capteur au robot pour la phase d'opération.

Un autre point capital à améliorer concerne la robustesse des algorithmes. Notamment au moment de la phase critique d'initialisation de la position des balises. Si cette étape est mal réalisée, même pour une seule balise, cela peut fausser complètement l'estimation de toutes les variables. Il serait donc très profitable d'implémenter des méthodes robustes capables de détecter les mesures erronées et de sélectionner celles pouvant servir à l'initialisation de la balise. Dans le cas où l'initialisation échoue quand même, il serait intéressant de développer un mécanisme de récupération permettant de détecter la mauvaise initialisation et de relancer le processus.

La position estimée des balises pendant la phase d'apprentissage ayant un impact direct sur la localisation du robot, il est nécessaire d'étudier la propagation des erreurs tout au long du déploiement du système, afin de savoir s'il est possible de respecter la précision de localisation du robot, estimée à une dizaine de centimètres.

Enfin, un élément essentiel dans la réalisation d'un tel système, concerne la validation des performances sur de longues périodes d'utilisation et pour un nombre important de situations. Ainsi, que ce soit pour l'étape d'apprentissage avec le SLAM ou pour la localisation du robot, il est fondamental de valider expérimentalement les performances du système. Les premiers jeux de données réalisés avec le prototype ne bénéficiant pas de vérité terrain, la première étape à poursuivre sera d'en obtenir une, et de vérifier les précisions sur la localisation du robot et sur le positionnement des balises. Une fois cette étape validée, il sera alors nécessaire d'étudier la

robustesse du système face aux nombreuses situations rencontrées lors de tests sous différentes conditions.

# ANNEXE A

# CALIBRATION DES BALISES RF

Dans la section 2.3.1 sur les balises radiofréquences (RF), nous avons présenté une équation, (2.9), permettant de modéliser le processus de mesure. Ce modèle faisant appel à différents paramètres, il est nécessaire de les déterminer au préalable par une procédure de calibration ou de les estimer en temps réel avec les autres variables d'intérêt comme la pose du robot.

Dans cette partie, nous présentons rapidement la composition du dispositif expérimental des balises, puis nous développons les différents tests réalisés afin de déterminer certaines des caractéristiques du modèle des mesures de distance.

Dans un premier temps, afin de tester la puce RF DW1000 [1], un kit d'évaluation EVK1000 [2] a été acheté. Une première série de tests a été menée afin de valider que les performances obtenues étaient suffisantes pour l'application visée. Une fois cette étape validée, de nouvelles balises ont été développées. Celles-ci sont réalisées à partir du module DWM1000 de Decawave, combiné à un kit Nucleo de STMicroelectronics équipé d'un microcontrôleur STM32F446RE [3].

La vérité terrain des mesures de distance a été obtenue en utilisant un double décamètre et un télémètre laser Bosch DLE70, visible à la Figure A.1. Ce dernier possède une précision de mesure de ±1.5 mm. La puce RF DW1000 est configurée en mode Fast Ranging avec le mode 6, afin d'avoir la fréquence de mesure la plus rapide possible, le détail du reste de la configuration est donné dans le Tableau A.1. Deux tests ont été menés, le premier sans boîtier de protection autour des balises et le second avec.

Tableau A.1 – Configuration utilisée pour les balises avec le module DWM1000

| Config. | Mode | Channel | Center frequency (GHz) | data rate (Mbps) | PRF (MHz) | preamble length |
|---|---|---|---|---|---|---|
| Fast Ranging | 6 | 5 | 6.5 | 6.8 | 16 | 128 |

1. http://www.decawave.com/products/dw1000
2. http://www.decawave.com/products/evk1000-evaluation-kit
3. http://www.st.com/en/evaluation-tools/nucleo-f446re.html


FIGURE A.1 – Télémètre laser Bosch DLE70 utilisé pour obtenir la vérité terrain des mesures de distance

## A.1 Tests sans boitier de protection

Dans la première série de tests où les balises n'ont pas de boîtier de protection, les deux antennes sont directement en ligne de vue. De plus, les antennes sont à environ 1.1 m du sol. Pour ce test, les distances de référence vont de 1 à 20 m, par pas de 1 m environ, et puis 25 m, 30 m et 35 m. Pour chaque distance de référence, environ 300 à 500 mesures ont été faites. Il est alors possible pour chaque distance de référence de calculer la moyenne, l'écart-type et le biais des mesures. Les résultats sont présentés sur les figures A.2, A.3, A.4 et A.5.

De plus, afin de constater l'influence d'un obstacle entre les deux balises, un test avec un humain à 10 m de la première balise et une distance de 30 m entre les deux balises a été mené. Le résultat de ce test est le suivant :

— Vraie distance : 29.76 m

— Moyenne : 30.0172 m

— Écart-type : 0.0222 m

— Erreur moyenne : $-0.2572$ m

— RMSE : 0.2581 m

Par contre, en mettant un obstacle métallique entre les deux balises, voir Figure A.6, plus aucune mesure de distance n'est disponible. Les ondes radiofréquences sont donc bloquées par un obstacle métallique.

De cette première série de tests, on peut faire les observations suivantes : les mesures sont affectées d'une erreur moyenne/biais de l'ordre de $-20$ cm sur toute la plage de distance, alors que l'écart-type est de l'ordre de 2 cm. Ainsi, ni le biais, ni l'écart-type ne semblent dépendre de la distance, tout du moins jusqu'à la distance maximale testée qui était de 35 m. En résumé de ce test, et en identifiant les paramètres du modèle donnée par (2.9), on obtient : $s = 1$, $b = -0.2$ m, $\sigma = 0.02$ m.

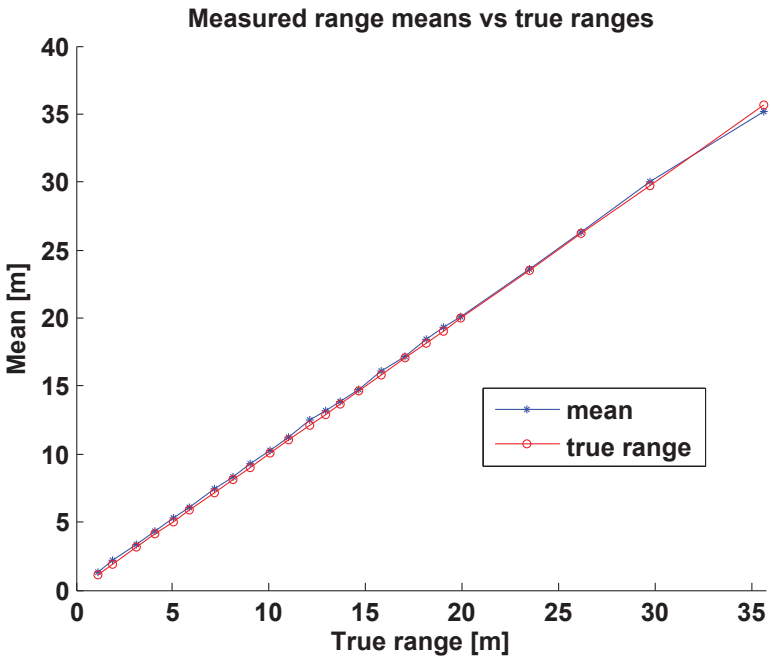



FIGURE A.2 – Moyennes des distances mesurées en fonction des vraies distances pour le module DWM1000 sans boîtier

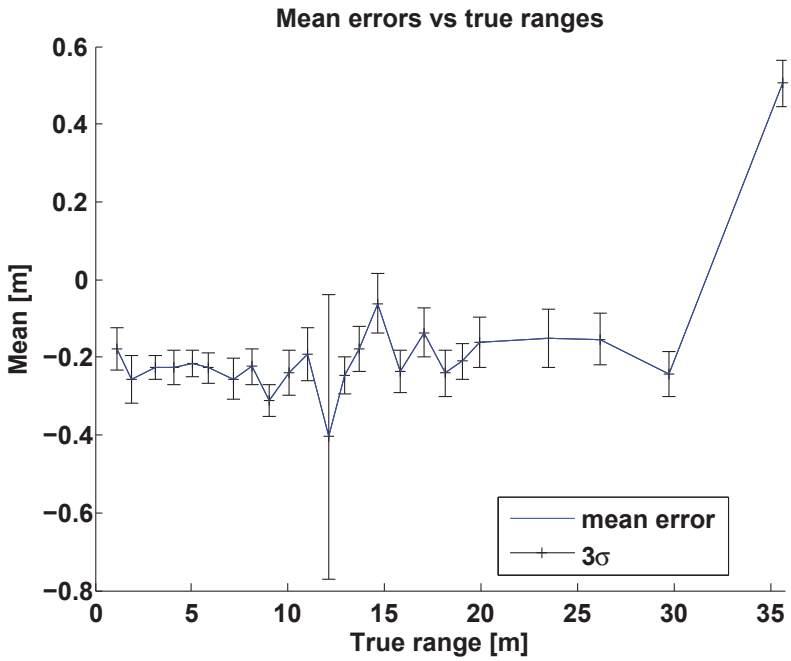



FIGURE A.3 – Erreurs moyennes des distances mesurées en fonction des vraies distances pour le module DWM1000 sans boîtier

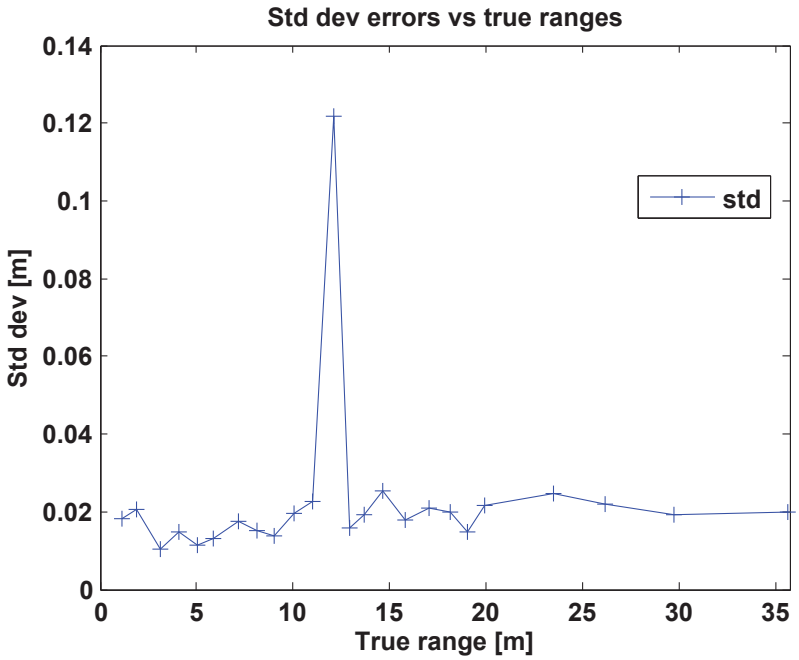



FIGURE A.4 – Écarts-types des distances mesurées en fonction des vraies distances pour le module DWM1000 sans boîtier

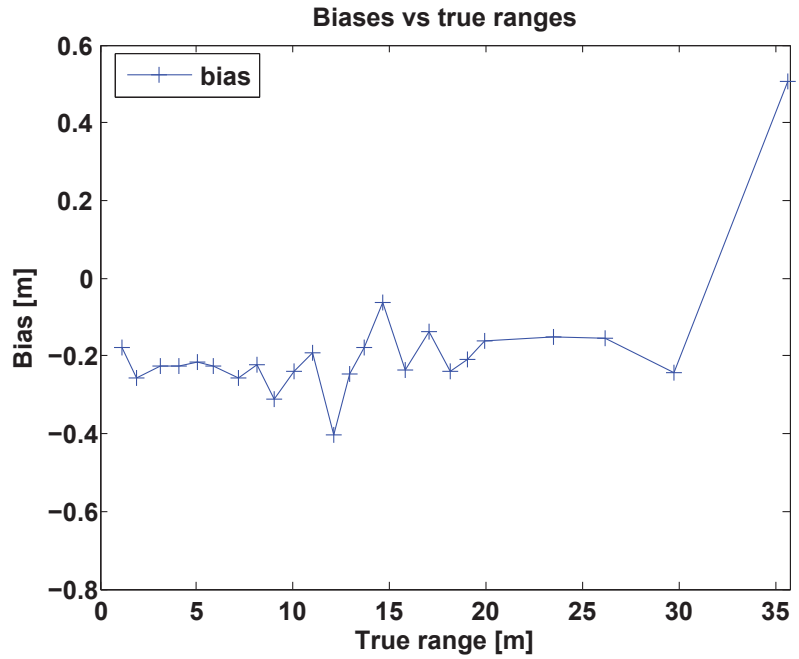



FIGURE A.5 – Biais entre les distances mesurées et les vraies distances pour le module DWM1000 sans boîtier

FIGURE A.6 – Dispositif de test du module DWM1000 sans boîtier avec une porte métallique comme obstacle au milieu des deux balises

## A.2 Tests avec boitier de protection

En intérieur, il n'est pas forcément nécessaire de protéger l'antenne et le circuit électronique de la balise. Cependant, en extérieur, il est impératif de mettre un boîtier de protection. Ainsi, pour cette deuxième série de tests, les mêmes balises que précédemment sont utilisées, mais avec des boîtiers de protection en plastique en plus. Les figures A.7 et A.8 montrent le dispositif expérimental utilisé ainsi que les balises avec leurs boîtiers. Les antennes des balises sont approximativement à 1.2 m du sol. Pour ce test, les distances de référence vont de 1 à 20 m tous les 1 m et de 20 à 30 m tous les 2 m. La vérité terrain est de nouveau obtenue grâce au télémètre laser. La mesure au télémètre laser pour la distance de 30 m n'a pas été faite, elle a été estimée avec le double décamètre. De plus, durant ce test des problèmes de réception des mesures à partir de 18 m ont été constatés. Il était parfois nécessaire de déplacer légèrement l'une des balises pour obtenir de nouveau une bonne réception des mesures de distance. Une explication possible à ce problème est que le test est réalisé le long d'un couloir, comme on peut le voir à la Figure A.7. Celui-ci pouvant créer des interférences (réflexions sur les murs) qui diminuent les performances, voir empêchent la réception d'un bon signal. Les résultats de ces tests sont présentés sur les figures A.9, A.10, A.11 et A.12. On observe ici que l'erreur moyenne/biais est comprise entre $-30$ cm et $-60$ cm. Ici aussi le biais ne semble pas dépendre de la distance mesurée. Par rapport au test précédent sans boîtier, le biais est plus important. L'ajout du boîtier engendre donc une augmentation de la valeur du biais. L'écart-type est compris entre 5 cm et 12 cm, mais on ne constate pas d'augmentation avec la distance. Au-delà de 20 m, l'écart-type augmente jusqu'à $20 - 30$ cm, mais il faut être prudent étant donné les problèmes rencontrés lors des tests pour ces distances. Par rapport aux tests sans boîtier, on constate une légère augmentation de l'écart-type. Sans boîtier, elle était d'environ 2 cm, avec celui-ci elle est comprise entre 5 et 10 cm. En résumé les paramètres identifiés avec ce dernier test sont : $s = 1$, $b = -0.5$ m, $\sigma = 0.1$ m. Ces dernières valeurs seront celles utilisées comme référence pour réaliser les simulations et régler les paramètres des algorithmes présentés dans ce mémoire.

Lors de la détermination des paramètres du modèle de mesure de distance effectuée grâce aux deux tests précédents, le facteur d'échelle a été à chaque fois fixé à un. Même si cette hypothèse se justifie en pratique, il est intéressant de bénéficier d'un outil permettant de s'en assurer. Pour ce faire, il est possible de réaliser une procédure de calibration des mesures de distance permettant d'estimer le facteur d'échelle $s$ et le biais $b$. Cette procédure de calibration, qui n'est pas développée ici, est basée sur un algorithme d'optimisation non-linéaire permettant de minimiser l'erreur entre la mesure et la vraie distance pour des mesures effectuées sur une certaine plage de distances.

FIGURE A.7 – Dispositif de test du module DWM1000 avec boîtier. La cible au bas du poteau rouge gauche permet de mieux visualiser le point du laser posé au bas du poteau droit.

FIGURE A.8 – Dispositif de test du module DWM1000 avec boîtier. La réalisation des tests le long du couloir peut être la cause des difficultés rencontrées pour l'obtention de certaines mesures. Les murs de part et d'autre du couloir peuvent provoquer des réflexions des ondes qui vont alors perturber la mesure.

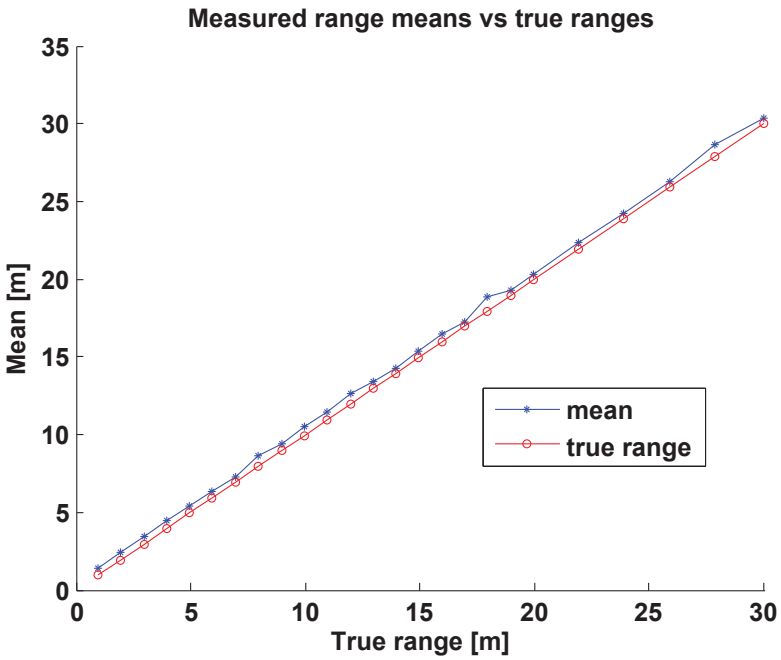



FIGURE A.9 – Moyennes des distances mesurées en fonction des vraies distances pour le module DWM1000 avec boîtier

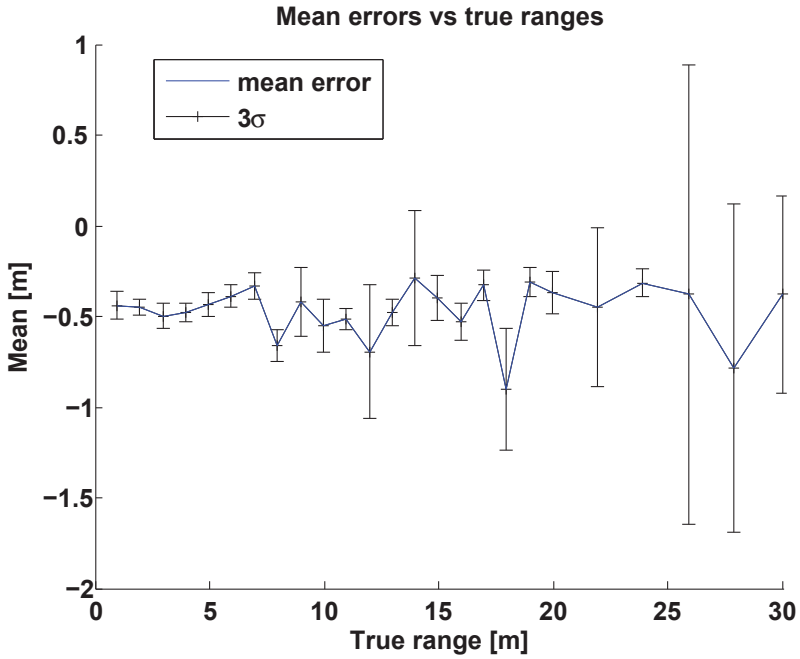



FIGURE A.10 – Erreurs moyennes des distances mesurées en fonction des vraies distances pour le module DWM1000 avec boîtier

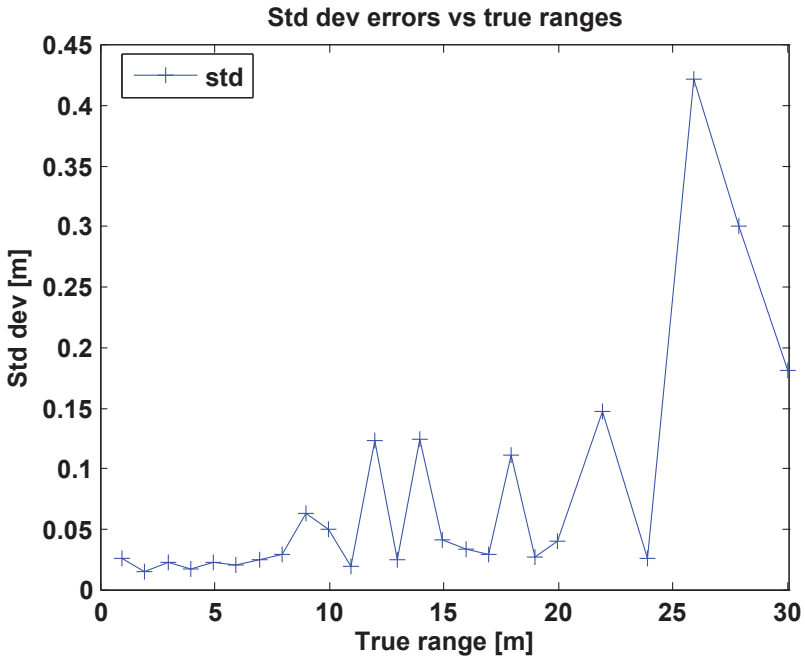



FIGURE A.11 – Écarts-types des distances mesurées en fonction des vraies distances pour le module DWM1000 avec boîtier

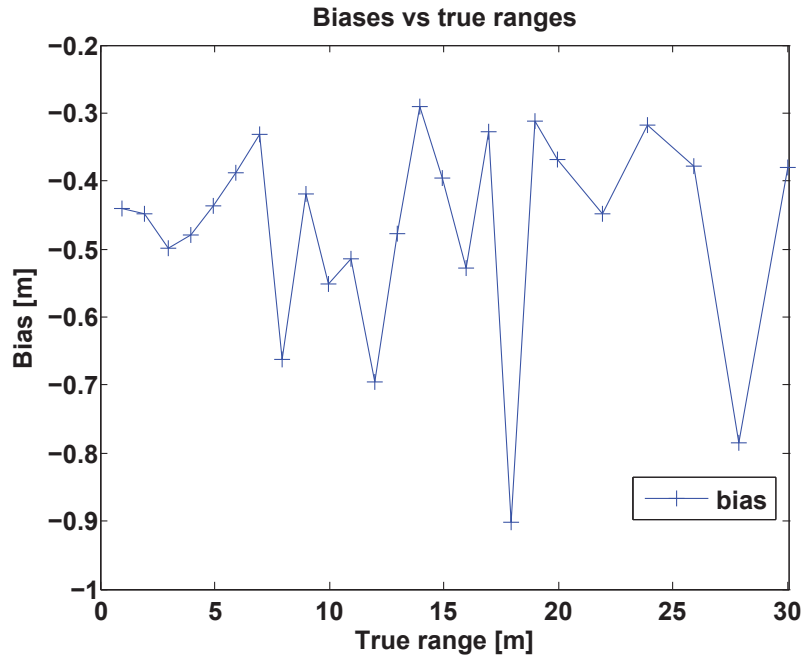



FIGURE A.12 – Biais entre les distances mesurées et les vraies distances pour le module DWM1000 avec boîtier

# ANNEXE B

# CALIBRATION DE LA CENTRALE INERTIELLE

Dans cette annexe, différentes procédures de calibration permettant de déterminer les valeurs des paramètres des modèles de mesures d'une centrale inertielle sont présentées. Nous présentons ainsi, dans un premier temps, deux méthodes de calibration de l'accéléromètre, puis une méthode de calibration du magnétomètre.

## B.1 Calibration de l'accéléromètre

Lors de la présentation du fonctionnement de l'accéléromètre, qui a été présenté à la section 2.3.2, l'importance de déterminer les paramètres du modèle a été soulignée. En effet, la mesure d'accélération étant intégrée une, voire, deux fois, pour obtenir la vitesse ou la position, il est essentiel d'obtenir une estimation la plus précise possible des paramètres, pour ne pas propager les erreurs.

Dans cette section, nous présentons deux méthodes de calibration de l'accéléromètre permettant de déterminer le biais et le facteur d'échelle du modèle de mesure.

### B.1.1 Méthode 1

Cette première méthode est tirée de [106].

Dans la section 2.3.2, l'équation de mesure de l'accélération est modélisée par (2.10) :

$$\mathbf{z_a} = \mathbf{K}_a\mathbf{R}^T(\mathbf{a} - \mathbf{g}) + \mathbf{b}_a + \epsilon_a$$

Le but de la calibration de l'accéléromètre est d'estimer le biais $\mathbf{b}_a$ et la matrice d'échelle $\mathbf{K}_a$. Pour cela on enregistre les données de l'accéléromètre au repos (pas de mouvement) dans différentes orientations pour avoir le vecteur gravité $\mathbf{g}$ qui pointe dans le plus de directions possibles dans le repère du capteur. Plus le nombre d'orientations sera important, meilleur sera le résultat de la calibration. L'idéal étant de couvrir toute la sphère des orientations. On obtient en statique :

$$\mathbf{z_a} = -\mathbf{K_a}\mathbf{R}^T\mathbf{g} + \mathbf{b_a} + \epsilon_\mathbf{a}$$

En faisant la moyenne sur un grand nombre de mesures de l'accélération on peut considérer que le bruit blanc Gaussien $\epsilon_\mathbf{a}$ va s'annuler et donc :

$$(\mathbf{z_a} - \mathbf{b_a}) = -\mathbf{K_a}\mathbf{R}^T\mathbf{g}$$

$$\Leftrightarrow \mathbf{K_a}^{-1}(\mathbf{z_a} - \mathbf{b_a}) = -\mathbf{R}^T\mathbf{g}$$

$$\Leftrightarrow \mathbf{R}^{-T}{\mathbf{K_a}}^{-1}(\mathbf{z_a} - \mathbf{b_a}) = -\mathbf{g} = \text{cste}$$

Ainsi :

$$\begin{aligned}
\mathbf{g}^T\mathbf{g} &= (\mathbf{R}^{-T}{\mathbf{K_a}}^{-1}(\mathbf{z_a} - \mathbf{b_a}))^T(\mathbf{R}^{-T}{\mathbf{K_a}}^{-1}(\mathbf{z_a} - \mathbf{b_a})) \\
&= (\mathbf{z_a} - \mathbf{b_a})^T(\mathbf{R}^{-T}{\mathbf{K_a}}^{-1})^T(\mathbf{R}^{-T}{\mathbf{K_a}}^{-1})(\mathbf{z_a} - \mathbf{b_a}) \\
&= (\mathbf{z_a} - \mathbf{b_a})^T{\mathbf{K_a}}^{-T}\mathbf{R}^{-1}\mathbf{R}^{-T}{\mathbf{K_a}}^{-1}(\mathbf{z_a} - \mathbf{b_a}) \\
&= (\mathbf{z_a} - \mathbf{b_a})^T{\mathbf{K_a}}^{-T}\mathbf{R}^{T}\mathbf{R}{\mathbf{K_a}}^{-1}(\mathbf{z_a} - \mathbf{b_a}) \\
&= (\mathbf{z_a} - \mathbf{b_a})^T{\mathbf{K_a}}^{-T}{\mathbf{K_a}}^{-1}(\mathbf{z_a} - \mathbf{b_a}) \\
&= (\mathbf{z_a} - \mathbf{b_a})^T\mathbf{A}(\mathbf{z_a} - \mathbf{b_a})
\end{aligned}$$

Avec $\mathbf{A} = {\mathbf{K_a}}^{-T}{\mathbf{K_a}}^{-1}$ une matrice symétrique telle que :

$${\mathbf{K_a}}^{-1} = \mathbf{A}^{\frac{1}{2}}$$

Et donc ${\mathbf{K_a}}^{-1}$ est aussi une matrice symétrique.

Les paramètres à déterminer pour la calibration de l'accéléromètre sont :

— Le biais : $\mathbf{b_a} = (b_{ax}, b_{ay}, b_{az})^T$, vecteur de taille $[3 \times 1]$

— Le facteur d'échelle : $\mathbf{K_a} = \begin{pmatrix} K_{axx} & K_{axy} & K_{axz} \\ K_{axy} & K_{ayy} & K_{ayz} \\ K_{axz} & K_{ayz} & K_{azz} \end{pmatrix}$, matrice symétrique de taille $[3 \times 3]$

Ce qui fait en tout $3 + 6 = 9$ paramètres à estimer.

Pour estimer ces paramètres, on peut utiliser un algorithme d'optimisation non-linéaire de type Levenberg-Marquardt (LM). La fonction de coût choisie est :

$$L(\mathbf{\Theta}_{acc}) = \sum_{k}^{M}(\mathbf{g}^T\mathbf{g} - (\mathbf{z_a}_k - \mathbf{b_a})^T\mathbf{A}(\mathbf{z_a}_k - \mathbf{b_a}))^2 \tag{B.1}$$

avec $\mathbf{\Theta}_{acc} = (K_{axx}, K_{axy}, K_{axz}, K_{ayy}, K_{ayz}, K_{azz}, b_{ax}, b_{ay}, b_{az})$.

La valeur de $g$ étant connue (9.80665 m$^2$ environ suivant la position sur Terre). L'algorithme est alors initialisé avec les paramètres suivants : $\mathbf{\Theta}^0_{acc} = (1, 0, 0, 1, 0, 1, 0, 0, 0)$

### B.1.2 Méthode 2

Cette deuxième méthode est tirée de l'article de Tedaldi et al. dans [124].

On considère que les données brutes de l'accéléromètre $\tilde{\mathbf{a}}$ sont affectées par une matrice d'échelle $\mathbf{K}_a$ et un biais $\mathbf{b}_a$ par rapport à la vraie valeur $\bar{\mathbf{a}}$ :

$$\bar{\mathbf{a}} = \mathbf{K}_a\tilde{\mathbf{a}} + \mathbf{b}_a$$

avec $\mathbf{K}_a$ la matrice symétrique d'échelle, et $\mathbf{b}_a$ le vecteur représentant le biais. On a donc un total de 9 paramètres à estimer (3 pour le biais et 6 pour la matrice d'échelle qui est symétrique) :

$$\mathbf{\Theta}_{acc} = (K_{axx}, K_{axy}, K_{axz}, K_{ayy}, K_{ayz}, K_{azz}, b_{ax}, b_{ay}, b_{az})$$

En statique, l'accélération mesurée doit être égale au vecteur gravité local :

$$\bar{\mathbf{a}} = \mathbf{g}$$

On cherche alors à minimiser la différence entre la norme de la mesure et la norme du vecteur gravité. La fonction de coût est donnée par :

$$L(\mathbf{\Theta}_{acc}) = \sum_{k=1}^{M}(||\mathbf{g}||^2 - ||h_k(\mathbf{a}_k, \mathbf{\Theta}_{acc})||^2)^2$$

$$L(\mathbf{\Theta}_{acc}) = \sum_{k=1}^{M}(1 - (\mathbf{K}_a\tilde{\mathbf{a}} + \mathbf{b}_a)^T(\mathbf{K}_a\tilde{\mathbf{a}} + \mathbf{b}_a))^2$$

avec :

$$h_k(\mathbf{a}_k, \mathbf{\Theta}_{acc}) = \mathbf{K}_a\mathbf{a}_k + \mathbf{b}_a$$

$$h_k(\mathbf{a}_k, \mathbf{\Theta}_{acc}) = \begin{pmatrix} K_{axx}a_{kx} + K_{axy}a_{ky} + K_{axz}a_{kz} + b_{ax} \\ K_{axy}a_{kx} + K_{ayy}a_{ky} + K_{ayz}a_{kz} + b_{ay} \\ K_{axz}a_{kx} + K_{ayz}a_{ky} + K_{azz}a_{kz} + b_{az} \end{pmatrix} = \begin{pmatrix} h_{kx} \\ h_{ky} \\ h_{kz} \end{pmatrix}$$

Donc :

$$L(\mathbf{\Theta}_{acc}) = \sum_{k=1}^{M}(g^2 - (h_{kx}^2 + h_{ky}^2 + h_{kz}^2))^2$$

$$L(\mathbf{\Theta}_{acc}) = \sum_{k=1}^{M}(1 - (h_{kx}^2 + h_{ky}^2 + h_{kz}^2))^2$$

$M$ étant le nombre de mesures statiques prises dans différentes orientations. Chaque mesure effectuée étant la moyenne d'un échantillon de données brutes afin de réduire les effets du bruit. En utilisant un algorithme de LM, il est possible d'estimer $\mathbf{\Theta}_{acc}$. Pour l'algorithme LM, il faut calculer le Jacobien $\mathbf{J}_{acc} = \frac{\partial L(\mathbf{\Theta}_{acc})}{\partial \mathbf{\Theta}_{acc}}$ de la fonction de coût $L(\mathbf{\Theta}_{acc})$ par rapport aux paramètres estimés. Pour initialiser l'algorithme, on considère que le biais $\mathbf{b}_a$ est proche de 0, que les termes diagonaux de $\mathbf{K}_a$ sont proches de 1 et que les termes hors-diagonale sont proches de 0. Pour obtenir de meilleurs résultats de l'algorithme d'optimisation, il est préférable de ramener les valeurs d'accélération entre $[-1, 1]$ (en g).

$$\mathbf{J}_{acc} = \frac{\partial}{\partial \mathbf{\Theta}_{acc}}(\sum_{k=1}^{M}(g^2 - (h_{kx}^2 + h_{ky}^2 + h_{kz}^2))^2$$

$$= \sum_{k=1}^{M}\frac{\partial}{\partial \mathbf{\Theta}_{acc}}(g^2 - (h_{kx}^2 + h_{ky}^2 + h_{kz}^2))^2$$

$$= \sum_{k=1}^{M}2(g^2 - (h_{kx}^2 + h_{ky}^2 + h_{kz}^2))\frac{\partial}{\partial \mathbf{\Theta}_{acc}}(g^2 - (h_{kx}^2 + h_{ky}^2 + h_{kz}^2))$$

$$= \sum_{k=1}^{M}-2(g^2 - (h_{kx}^2 + h_{ky}^2 + h_{kz}^2))\frac{\partial}{\partial \mathbf{\Theta}_{acc}}(h_{kx}^2 + h_{ky}^2 + h_{kz}^2)$$

$$= \sum_{k=1}^{M}-4(g^2 - (h_{kx}^2 + h_{ky}^2 + h_{kz}^2))(h_{kx}\frac{\partial h_{kx}}{\partial \mathbf{\Theta}_{acc}} + h_{ky}\frac{\partial h_{ky}}{\partial \mathbf{\Theta}_{acc}} + h_{kz}\frac{\partial h_{kz}}{\partial \mathbf{\Theta}_{acc}})$$

$$\frac{\partial L}{\partial K_{axx}} = -4(g^2 - h_k^2)h_{kx}a_{kx}$$

$$\frac{\partial L}{\partial K_{ayy}} = -4(g^2 - h_k^2)h_{ky}a_{ky}$$

$$\frac{\partial L}{\partial K_{azz}} = -4(g^2 - h_k^2)h_{kz}a_{kz}$$

$$\frac{\partial L}{\partial K_{axy}} = -4(g^2 - h_k^2)(h_{kx}a_{ky} + h_{ky}a_{kx})$$

$$\frac{\partial L}{\partial K_{axz}} = -4(g^2 - h_k^2)(h_{kx}a_{kz} + h_{kz}a_{kx})$$

$$\frac{\partial L}{\partial K_{ayz}} = -4(g^2 - h_k^2)(h_{ky}a_{kz} + h_{kz}a_{ky})$$

$$\frac{\partial L}{\partial b_{ax}} = -4(g^2 - h_k^2)h_{kx}$$
$$\frac{\partial L}{\partial b_{ay}} = -4(g^2 - h_k^2)h_{ky}$$
$$\frac{\partial L}{\partial b_{az}} = -4(g^2 - h_k^2)h_{kz}$$

## B.2 Calibration du magnétomètre

Dans la section 2.3.4, le principe de fonctionnement du magnétomètre, ainsi que son modèle ont été présentés. Afin d'obtenir un maximum de précision des mesures du magnétomètre, il est nécessaire de déterminer les valeurs des différents paramètres du modèle. Pour cela, il est possible d'avoir recourt à une procédure de calibration.

Le même principe présenté pour l'accéléromètre peut être appliqué pour le magnétomètre, en remplaçant le vecteur gravité par le vecteur du champ magnétique terrestre. Le vecteur du champ magnétique terrestre $\mathbf{m}$ peut-être mis sous la forme :

$$\mathbf{m} = m \begin{pmatrix} \cos\delta \\ 0 \\ \sin\delta \end{pmatrix}$$

Avec :

— $m$ : l'intensité du champ magnétique,

— $\delta$ : l'angle d'inclinaison.

Le but de la calibration est de transformer la forme ellipsoïdale des données brutes du magnétomètre en une sphère. Les paramètres permettant de passer de l'ellipsoïde à la sphère sont les paramètres de calibration.

$$\mathbf{z}_m = \mathbf{K}_m \mathbf{R}^T \mathbf{m} + \mathbf{b}_m + \epsilon_m$$
$$\mathbf{z}_m = \mathbf{K}_m \mathbf{R}^T m \begin{pmatrix} \cos\delta \\ 0 \\ \sin\delta \end{pmatrix} + \mathbf{b}_m + \epsilon_m$$

En enregistrant plusieurs mesures et en faisant la moyenne, on annule le terme lié au bruit $\epsilon_m$ :

$$\mathbf{z}_m - \mathbf{b}_m = \mathbf{K}_m \mathbf{R}^T m \begin{pmatrix} \cos\delta \\ 0 \\ \sin\delta \end{pmatrix}$$

$$\mathbf{K}_m^{-1}(\mathbf{z}_m - \mathbf{b}_m) = \mathbf{R}^T m \begin{pmatrix} \cos\delta \\ 0 \\ \sin\delta \end{pmatrix}$$

$$\mathbf{R}^{-T}\mathbf{K}_m^{-1}(\mathbf{z}_m - \mathbf{b}_m) = m \begin{pmatrix} \cos\delta \\ 0 \\ \sin\delta \end{pmatrix}$$

En faisant :

$$m \begin{pmatrix} \cos\delta \\ 0 \\ \sin\delta \end{pmatrix}^T .m \begin{pmatrix} \cos\delta \\ 0 \\ \sin\delta \end{pmatrix}$$

on obtient :

$$\left[\mathbf{K}_m^{-1}(\mathbf{z}_m - \mathbf{b}_m)\right]^T \left[\mathbf{K}_m^{-1}(\mathbf{z}_m - \mathbf{b}_m)\right] = m^2$$

$$(\mathbf{z}_m - \mathbf{b}_m)^T \mathbf{K}_m^{-T} \mathbf{K}_m^{-1}(\mathbf{z}_m - \mathbf{b}_m) = m^2$$

$$(\mathbf{z}_m - \mathbf{b}_m)^T \mathbf{A}(\mathbf{z}_m - \mathbf{b}_m) = m^2$$

Avec $\mathbf{A} = \mathbf{K}_m^{-T}\mathbf{K}_m^{-1}$ une matrice symétrique et donc :

$$\mathbf{K}_m^{-1} = \mathbf{A}^{\frac{1}{2}}$$

On a donc 9 ou 10 paramètres à estimer :

— 3 pour le biais statique $\mathbf{b}_m$ ou "hard-iron" ,

— 6 pour la matrice symétrique d'échelle $\mathbf{K}_m$ ou "soft-iron",

— plus 1 si la valeur de l'intensité du champ magnétique $m$ n'est pas connue.

# BIBLIOGRAPHIE


[1] R. Allen, N. MacMillan, D. Marinakis, R. I. Nishat, R. Rahman, and S. Whitesides, "The range beacon placement problem for robot navigation," in *Canadian Conference on Computer and Robot Vision*, May 2014, pp. 151–158. xvii, 131, 132, 140

[2] J. Andrade-Cetto and A. Sanfeliu, "The effects of partial observability in SLAM," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 1, April 2004, pp. 397–402 Vol.1. 69

[3] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (SLAM) : part ii," *IEEE Robotics Automation Magazine*, vol. 13, no. 3, pp. 108–117, Sept 2006. 5

[4] R. Balling, *The Maximin Fitness Function ; Multi-objective City and Regional Planning*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2003, pp. 1–15. 162, 163

[5] B. Barshan and H. Durrant-Whyte, "Inertial navigation systems for mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 11, no. 3, pp. 328–342, Jun 1995. 26, 29

[6] P. Batista, C. Silvestre, and P. Oliveira, "Single range aided navigation and source localization : Observability and filter design," *Systems and Control Letters*, vol. 60, no. 8, pp. 665 – 673, 2011. [Online]. Available : http://www.sciencedirect.com/science/article/pii/S0167691111001174 105

[7] M. Bayat and A. P. Aguiar, "AUV range-only localization and mapping : Observer design and experimental results," in *European Control Conference (ECC)*, July 2013, pp. 4394–4399. 107, 108

[8] ——, "Observability analysis for AUV range-only localization and mapping measures of unobservability and experimental results," *IFAC Proceedings Volumes*, vol. 45, no. 27, pp. 325–330, 2012. [Online]. Available : http://www.sciencedirect.com/science/article/pii/S1474667016312496 107

[9] A. N. Bishop, B. Fidan, B. D. Anderson, K. Doğançay, and P. N. Pathirana, "Optimality analysis of sensor-target localization geometries," *Automatica*, vol. 46, no. 3, pp. 479 – 492, 2010. [Online]. Available : http://www.sciencedirect.com/science/article/pii/S0005109809005500 134, 140

[10] J. L. Blanco, J. A. Fernandez-Madrigal, and J. González, "Efficient probabilistic range-only SLAM," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2008, pp. 1017–1022. 104, 108

[11] J. L. Blanco, J. González, and J. A. Fernandez-Madrigal, "A pure probabilistic approach to range-only SLAM," in *IEEE International Conference on Robotics and Automation*, May 2008, pp. 1436–1441. 102, 104, 108, 116

[12] B. Boots and G. Gordon, "A spectral learning approach to range-only SLAM," in *Proceedings of the 30th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, S. Dasgupta and D. McAllester, Eds., vol. 28, no. 1. Atlanta, Georgia, USA : PMLR, Jun 2013, pp. 19–26. [Online]. Available : http://proceedings.mlr.press/v28/boots13.html 105, 108

[13] J. Borenstein, H. R. Everett, and L. Feng, *Where Am I? Systems and Methods for Mobile Robot Positioning*. J. Borenstein, 1996. [Online]. Available : http://onlinebooks.library.upenn.edu/webbin/book/lookupid?key=olbp12530 17, 18

[14] W. Bosworth, S. Kim, and N. Hogan, "The mit super mini cheetah : A small, low-cost quadrupedal robot for dynamic locomotion," in *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, Oct 2015, pp. 1–8. 1

[15] M. Burke and N. Bos, "Optimal placement of range-only beacons for mobile robot localisation," in *4th Robotics and Mechatronics Conference of South Africa (RobMech)*, 2011. [Online]. Available : http://hdl.handle.net/10204/5386 45, 132

[16] S. Busanelli and G. Ferrari, *UWB-Based Tracking of Autonomous Vehicles with Multiple Receivers*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2010, pp. 188–198. 41

[17] F. Caballero, L. Merino, and A. Ollero, "A general gaussian-mixture approach for range-only mapping using multiple hypotheses," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2010, pp. 4404–4409. 104, 108, 109, 112, 116

[18] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping : Toward the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, Dec 2016. 5, 99

[19] G. Campion and W. Chung, *Wheeled Robots*. Springer Berlin Heidelberg, 2008, pp. 391–410. [Online]. Available : https://doi.org/10.1007/978-3-540-30301-5_18 14

[20] M. J. Caruso, "Applications of magnetoresistive sensors in navigation systems," in *Sensors and Actuators, SAE*. SAE International, 1997. [Online]. Available : http://dx.doi.org/10.4271/970602 30, 31

[21] J. Castellanos, R. Martinez-Cantin, J. Tardós, and J. Neira, "Robocentric map joining : Improving the consistency of ekf-slam," *Robotics and Autonomous Systems*, vol. 55, no. 1, pp. 21 – 29, 2007, simultaneous Localisation and Map Building. [Online]. Available : http://www.sciencedirect.com/science/article/pii/S0921889006001448 105

[22] J. Chaffee and J. Abel, "GDOP and the cramer-rao bound," in *IEEE Position Location and Navigation Symposium*, 1994. 132

[23] S. Cousins, "Welcome to ROS Topics [ROS Topics]," *IEEE Robotics Automation Magazine*, vol. 17, no. 1, pp. 13–14, March 2010. 13

[24] J. Dattorro, *Convex Optimization & Euclidean Distance Geometry*. Meboo Publishing, 2005. 105

[25] DecaWave, "APS013 APPLICATION NOTE : The implementation of two-way ranging with the DW1000," DecaWave, Tech. Rep., 2014. xi, 21, 22

[26] F. Dellaert and M. Kaess, "Square Root SAM : Simultaneous localization and mapping via square root information smoothing," *Intl. J. of Robotics Research (IJRR)*, vol. 25, no. 12, pp. 1181–1204, dec 2006. 41, 107

[27] F. Dellaert, "Factor Graphs and GTSAM : A Hands-on Introduction," GT RIM, Tech. Rep. GT-RIM-CP&R-2012-002, Sept 2012. [Online]. Available : https://research.cc.gatech.edu/borg/sites/edu.borg/files/downloads/gtsam.pdf 81, 105, 108

[28] B. Desrochers, S. Lacroix, and L. Jaulin, "Set-membership approach to the kidnapped robot problem," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2015, pp. 3715–3720. 108

[29] M. Di Marco, A. Garulli, A. Giannitrapani, and A. Vicino, "A set theoretic approach to dynamic robot localization and mapping," *Autonomous Robots*, vol. 16, no. 1, pp. 23–47, 2004. [Online]. Available : http://dx.doi.org/10.1023/B:AURO.0000008670.09004.ce 105, 108

[30] M. Di Marco, A. Garulli, S. Lacroix, and A. Vicino, "Set membership localization and mapping for autonomous navigation," *International Journal of Robust and Nonlinear Control*, vol. 11, no. 7, pp. 709–734, 2001. [Online]. Available : http://dx.doi.org/10.1002/rnc.619 105, 108

[31] J. Djugash, S. Singh, G. Kantor, and W. Zhang, "Range-only SLAM for robots operating cooperatively with sensor networks," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2006, pp. 2078–2084. 102, 106, 108

[32] J. Djugash, B. Hamner, and S. Roth, "Navigating with ranging radios : Five data sets with ground truth," *Journal of Field Robotics*, vol. 26, no. 9, pp. 689–695, 2009. 58, 81, 107, 116

[33] J. Djugash and S. Singh, "A robust method of localization and mapping using only range," in *Experimental Robotics*. Springer, 2009, pp. 341–351. 102, 104, 107, 108, 109, 111, 116

[34] J. Djugash, S. Singh, and P. Corke, "Further results with localization and mapping using range from radio," in *Field and Service Robotics*. Springer, 2006, pp. 231–242. 102, 108

[35] L. Doherty, K. S. J. pister, and L. E. Ghaoui, "Convex position estimation in wireless sensor networks," in *IEEE Conference on Computer Communications (INFOCOM)*, vol. 3, 2001, pp. 1655–1663 vol.3. 40, 52

[36] G. Dudek and M. Jenkin, "Inertial sensors, GPS, and odometry," *Springer Handbook of Robotics*, pp. 477–490, 2008. 26, 29, 32

[37] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping : part i," *IEEE Robotics Automation Magazine*, vol. 13, no. 2, pp. 99–110, June 2006. 5

[38] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, June 1989. 38

[39] P. K. Enge, "The global positioning system : Signals, measurements, and performance," *International Journal of Wireless Information Networks*, vol. 1, no. 2, pp. 83–105, Apr 1994. [Online]. Available : https://doi.org/10.1007/BF02106512 32, 33

[40] F. R. Fabresse, F. Caballero, I. Maza, and A. Ollero, "Undelayed 3D RO-SLAM based on Gaussian-mixture and reduced spherical parametrization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov 2013, pp. 1555–1561. 104, 108

[41] ——, "Localization and mapping for aerial manipulation based on range-only measurements and visual markers," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 2100–2106. 108

[42] ——, "Robust range-only slam for aerial vehicles," in *International Conference on Unmanned Aircraft Systems (ICUAS)*, May 2014, pp. 750–755. 24

[43] F. R. Fabresse, F. Caballero, L. Merino, and A. Ollero, "Active perception for 3d range-only simultaneous localization and mapping with uavs," in *International Conference on Unmanned Aircraft Systems (ICUAS)*, June 2016, pp. 394–398. 108

[44] F. R. Fabresse, F. Caballero, and A. Ollero, "Decentralized simultaneous localization and mapping for multiple aerial vehicles using range-only sensors," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 6408–6414. 108

[45] F. R. Fabresse, F. Caballero, I. Maza, and A. Ollero, "Robust Range-Only SLAM for Unmanned Aerial Systems," *Journal of Intelligent & Robotic Systems*, vol. 84, no. 1-4, pp. 297–310, dec 2016. [Online]. Available : https://doi.org/10.1007/s10846-015-0322-z 108

[46] M. A. Fischler and R. C. Bolles, "Random sample consensus : A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, jun 1981. [Online]. Available : http://doi.acm.org/10.1145/358669.358692 49

[47] H. T. Friis, "A note on a simple transmission formula," *Proceedings of the IRE*, vol. 34, no. 5, pp. 254–256, May 1946. 20

[48] P. Gahinet, A. Nemirovski, A. J. Laub, and M. Chilali, *LMI Control Toolbox*, The MathWorks Inc., 1995. 52

[49] L. Génevé, O. Kermorgant, and E. Laroche, "A composite beacon initialization for EKF range-only SLAM," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2015, pp. 1342–1348. 108

[50] S. Gezici, Z. Tian, G. B. Giannakis, H. Kobayashi, A. F. Molisch, H. V. Poor, and Z. Sahinoglu, "Localization via ultra-wideband radios : a look at positioning aspects for future sensor networks," *IEEE Signal Processing Magazine*, vol. 22, no. 4, pp. 70–84, July 2005. 20

[51] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st ed. Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc., 1989. 130, 135

[52] G. H. Golub and C. Reinsch, "Singular value decomposition and least squares solutions," *Numer. Math.*, vol. 14, no. 5, pp. 403–420, 1970. [Online]. Available : http://dx.doi.org/10.1007/BF02163027 47

[53] J. González, J. Blanco, C. Galindo, A. O. de Galisteo, J. Fernandez-Madrigal, F. Moreno, and J. Martinez, "Mobile robot localization based on ultra-wide-band ranging : A particle filter approach," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 496–507, 2009. [Online]. Available : http://www.sciencedirect.com/science/article/pii/S0921889008001747 41

[54] D. Gouaillier, V. Hugel, P. Blazevic, C. Kilner, J. Monceaux, P. Lafourcade, B. Marnier, J. Serre, and B. Maisonnier, "Mechatronic design of nao humanoid," in *IEEE International Conference on Robotics and Automation*, May 2009, pp. 769–774. 1

[55] J. Graefenstein and M. E. Bouzouraa, "Robust method for outdoor localization of a mobile robot using received signal strength in low power wireless networks," in *IEEE International Conference on Robotics and Automation*, May 2008, pp. 33–38. 41

[56] F. Gustafsson and F. Gunnarsson, "Mobile positioning using wireless networks : possibilities and fundamental limitations based on available wireless network measurements," *IEEE Signal Processing Magazine*, vol. 22, no. 4, pp. 41–53, July 2005. 21

[57] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004. 50

[58] R. Hermann and A. Krener, "Nonlinear controllability and observability," *IEEE Transactions on Automatic Control*, vol. 22, no. 5, pp. 728–740, Oct 1977. 69

[59] F. Herranz, A. Llamazares, E. Molinos, and M. Ocaña, "A comparison of SLAM algorithms with range only sensors," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 4606–4611. 105, 107, 108

[60] Z. Hu, R. McCauley, S. Schaeffer, and X. Deng, "Aerodynamics of dragonfly flight and robotic design," in *IEEE International Conference on Robotics and Automation*, May 2009, pp. 3061–3066. 1

[61] G. P. Huang, A. I. Mourikis, and S. I. Roumeliotis, "Observability-based Rules for Designing Consistent EKF SLAM Estimators," *The International Journal of Robotics Research*, vol. 29, no. 5, pp. 502–528, 2010. [Online]. Available : http://dx.doi.org/10.1177/0278364909353640 106

[62] J. Huang, D. Millman, M. Quigley, D. Stavens, S. Thrun, and A. Aggarwal, "Efficient, generalized indoor WiFi GraphSLAM," in *IEEE International Conference on Robotics and Automation*, May 2011, pp. 1038–1043. 105, 108

[63] L. Jaulin, "A nonlinear set membership approach for the localization and map building of underwater robots," *IEEE Transactions on Robotics*, vol. 25, no. 1, pp. 88–98, Feb 2009. 106

[64] ——, "Range-only SLAM with occupancy maps : A set-membership approach," *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 1004–1010, Oct 2011. 108

[65] L. Jaulin, M. Kieffer, E. Walter, and D. Meizel, "Guaranteed robust nonlinear estimation with application to robot localization," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 32, no. 4, pp. 374–381, Nov 2002. 42

[66] L. Jaulin, "Pure range-only slam with indistinguishable landmarks ; a constraint programming approach," *Constraints*, pp. 1–20, Oct 2015. [Online]. Available : https://hal.archives-ouvertes.fr/hal-01298354 108

[67] D. B. Jourdan, D. Dardari, and M. Z. Win, "Position error bound for uwb localization in dense cluttered environments," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 44, no. 2, pp. 613–628, April 2008. 24

[68] D. B. Jourdan, J. J. Deyst, M. Z. Win, and N. Roy, "Monte carlo localization in dense multipath environments using UWB ranging," in *IEEE International Conference on Ultra-Wideband*, Sept 2005, pp. 314–319. 41

[69] D. B. Jourdan and N. Roy, "Optimal sensor placement for agent localization," in *IEEE/ION Position, Location, And Navigation Symposium*, April 2006, pp. 128–139. 133, 134

[70] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM : Incremental smoothing and mapping," *IEEE Transactions on Robotics (TRO)*, vol. 24, no. 6, pp. 1365–1378, dec 2008. 41

[71] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Fluids Engineering*, vol. 82, no. 1, pp. 35–45, 1960. 39, 76

[72] G. Kantor and S. Singh, "Preliminary results in range-only localization and mapping," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2, 2002, pp. 1818–1823 vol.2. 40, 108

[73] A. Kehagias, J. A. Djugash, and S. Singh, "Range-only slam with interpolated range data," Carnegie Mellon University, Tech. Rep., 2006. 105, 108

[74] N. Kirchhof, "Optimal placement of multiple sensors for localization applications," in *International Conference on Indoor Positioning and Indoor Navigation*, Oct 2013, pp. 1–10. 133

[75] D. Kotzor and W. Utschick, "Kernel methods for ill-posed range-based localization problems," *IEEE Transactions on Signal Processing*, vol. 60, no. 8, pp. 4151–4162, Aug 2012. 40

[76] B. Kuipers, E. A. Feigenbaum, P. E. Hart, and N. J. Nilsson, "Shakey : From conception to history," *AI Magazine*, vol. 38, pp. 88–103, 2017. [Online]. Available : http://www.aaai.org/ojs/index.php/aimagazine/article/view/2716 1

[77] D. Kurth, G. Kantor, and S. Singh, "Experimental results in range-only localization with radio," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 1, Oct 2003, pp. 974–979 vol.1. 40

[78] M. Laguna, J. O. Roa, A. R. Jiménez, and F. Seco, "Diversified local search for the optimal layout of beacons in an indoor positioning system," *IIE Transactions*, vol. 41, no. 3, pp. 247–259, 2009. [Online]. Available : http://dx.doi.org/10.1080/07408170802369383 131

[79] P. Lamon and R. Siegwart, "Inertial and 3D-odometry fusion in rough terrain - towards real 3D navigation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 2, Sept 2004, pp. 1716–1721 vol.2. 27

[80] J. B. Lasserre, "A sum of squares approximation of nonnegative polynomials," *SIAM Review*, vol. 49, no. 4, pp. 651–669, 2007. 52

[81] C. Laugier, A. Martinelli, and D. A. Vasquez, "Mooc on Mobile Robots and Autonomous Vehicles," May 2015, lecture - International Mooc Course from Inria-uTOP. First edition in May 2015, second edition in February 2016. [Online]. Available : https://hal.inria.fr/cel-01256021 71

[82] C. Lee, "An algorithm for path connections and its applications," *IRE Trans. on Electronic Computers*, vol. EC-10, no. 3, pp. 346–365, Sept 1961. 58

[83] K. W. Lee, W. S. Wijesoma, and J. I. Guzman, "On the observability and observability analysis of SLAM," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2006, pp. 3569–3574. 69, 73

[84] J. J. Leonard and H. F. Durrant-Whyte, "Mobile robot localization by tracking geometric beacons," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, pp. 376–382, Jun 1991. 38

[85] J. J. Leonard, R. J. Rikoski, P. M. Newman, and M. Bosse, "Mapping partially observable features from multiple uncertain vantage points," *The International Journal of Robotics Research*, vol. 21, no. 10-11, pp. 943–975, 2002. 101, 109, 112, 114

[86] P. Lourenço, P. Batista, P. Oliveira, and C. Silvestre, "Simultaneous Localization and mapping in sensor networks : A GES sensor-based filter with moving object tracking," in *European Control Conference (ECC)*, July 2015, pp. 2354–2359. 108

[87] P. Lourenço, P. Batista, P. Oliveira, C. Silvestre, and C. L. P. Chen, "Sensor-based globally asymptotically stable range-only simultaneous localization and mapping," in *IEEE Conference on Decision and Control*, Dec 2013, pp. 5692–5697. 108

[88] P. Lourenço, P. Batista, P. Oliveira, C. Silvestre, and C. P. Chen, "Sensor-based globally exponentially stable range-only simultaneous localization and mapping," *Robotics and Autonomous Systems*, vol. 68, pp. 72 – 85, 2015. [Online]. Available : http://www.sciencedirect.com/science/article/pii/S0921889015000184 105, 107, 108

[89] S. O. H. Madgwick, A. J. L. Harrison, and R. Vaidyanathan, "Estimation of IMU and MARG orientation using a gradient descent algorithm," in *2011 IEEE International Conference on Rehabilitation Robotics*, June 2011, pp. 1–7. 30, 32

[90] P. C. Mahalanobis, "On the generalized distance in statistics," *National Institute of Sciences*, vol. 2, pp. 49–55, 1936. 80

[91] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles : Modeling, estimation, and control of quadrotor," *IEEE Robotics Automation Magazine*, vol. 19, no. 3, pp. 20–32, Sept 2012. 1

[92] A. Makni, "Inertial and magnetic data fusion for attitude estimation under energetic constraint for accelerated rigid body," Ph.D. dissertation, Université Grenoble Alpes, Mar 2016. [Online]. Available : https://tel.archives-ouvertes.fr/tel-01318310 29

[93] M. D. Marco, A. Garulli, S. Lacroix, and A. Vicino, "A set theoretic approach to the simultaneous localization and map building problem," in *IEEE Conference on Decision and Control (CDC)*, vol. 1, 2000, pp. 833–838 vol.1. 105, 108

[94] S. Martinez and F. Bullo, "Optimal sensor placement and motion coordination for target tracking," *Automatica*, vol. 42, no. 4, pp. 661 – 668, 2006. [Online]. Available : //www.sciencedirect.com/science/article/pii/S000510980600015X 133, 134, 141

[95] B. R. M.D. Choi, T.Y. Lam, "Sums of squares of real polynomials," in *Proc. of Symposia in Pure Mathematics*, vol. 58. American Mathematical Society, 1995, pp. 103–126. 52, 54

[96] E. Menegatti, M. Danieletto, M. Mina, A. Pretto, A. Bardella, S. Zanconato, P. Zanuttigh, and A. Zanella, "Autonomous discovery, localization and recognition of smart objects through wsn and image features," in *IEEE Globecom Workshops*, Dec 2010, pp. 1653–1657. 102, 108

[97] E. Menegatti, A. Zanella, S. Zilli, F. Zorzi, and E. Pagello, "Range-only SLAM with a mobile robot and a Wireless Sensor Networks," in *IEEE International Conference on Robotics and Automation*, May 2009, pp. 8–14. 101, 108, 116

[98] D. Moreno-Salinas, A. M. Pascoal, and J. Aranda, "Optimal sensor placement for underwater positioning with uncertainty in the target location," in *IEEE International Conference on Robotics and Automation*, May 2011, pp. 2308–2314. 24, 134, 136, 141

[99] P. Morin and C. Samson, *Motion Control of Wheeled Mobile Robots*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2008, pp. 799–826. [Online]. Available : https://doi.org/10.1007/978-3-540-30301-5_35 35

[100] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint kalman filter for vision-aided inertial navigation," in *IEEE International Conference on Robotics and Automation*, April 2007, pp. 3565–3572. 27, 29

[101] M. W. Mueller, M. Hamer, and R. D'Andrea, "Fusing ultra-wideband range measurements with accelerometers and rate gyroscopes for quadrocopter state estimation," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 1730–1736. 41

[102] Y. Nesterov and A. Nemirovski, *Interior-point Polynomial Methods in Convex Programming*. SIAM, 1994. 52

[103] E. Olson, J. Leonard, and S. Teller, "Robust range-only beacon localization," in *IEEE/OES Autonomous Underwater Vehicles*, June 2004, pp. 66–75. 108, 109

[104] E. Olson, J. J. Leonard, and S. Teller, "Robust range-only beacon localization," *IEEE Journal of Oceanic Engineering*, vol. 31, no. 4, pp. 949–958, Oct 2006. 102, 108, 116

[105] G. Oriolo, A. D. Luca, and M. Vendittelli, "WMR control via dynamic feedback linearization : design, implementation, and experimental validation," *IEEE Transactions on Control Systems Technology*, vol. 10, no. 6, pp. 835–852, Nov 2002. 14, 18

[106] T. Ozyagcilar, "Calibrating an eCompass in the Presence of Hard- and Soft-Iron Interference (Application Note AN4246)," Freescale Semiconductor, Tech. Rep., 2015. 183

[107] N. Patwari, J. N. Ash, S. Kyperountas, A. O. Hero, R. L. Moses, and N. S. Correal, "Locating the nodes : cooperative localization in wireless sensor networks," *IEEE Signal Processing Magazine*, vol. 22, no. 4, pp. 54–69, July 2005. 21

[108] D. P. Paudel, A. Habed, C. Demonceaux, and P. Vasseur, "Robust and optimal sum-of-squares-based point-to-plane registration of image sets and structured scenes," in *IEEE International Conference on Computer Vision (ICCV)*, Dec 2015, pp. 2048–2056. 52

[109] J. Perez-Ramirez, D. K. Borah, and D. G. Voelz, "Optimal 3-D Landmark Placement for Vehicle Localization Using Heterogeneous Sensors," *IEEE Transactions on Vehicular Technology*, vol. 62, no. 7, pp. 2987–2999, Sept 2013. 24, 134, 141

[110] V. Powers and T. Wormann, "An algorithm for sums of squares of real polynomials," *Journal of Pure and Applied Algebra*, vol. 127, no. 1, pp. 99–104, 1998. 54

[111] V. Pyati, "Computation of the circular error probability (CEP) integral," *IEEE Trans. on Aerospace and Electronic Systems*, vol. 29, no. 3, pp. 1023–1024, Jul 1993. 134

[112] N. Rajagopal, S. Chayapathy, B. Sinopoli, and A. Rowe, "Beacon placement for range-based indoor localization," in *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, Oct 2016, pp. 1–8. 133

[113] J. Roa, A. Jimenez, F. Seco, J. Prieto, and J. Ealo, "Optimal placement of sensors for trilateration : Regular lattices vs meta-heuristic solutions," in *Computer Aided Systems Theory EUROCAST*, ser. Lecture Notes in Computer Science, R. Moreno Diaz, F. Pichler, and A. Quesada Arencibia, Eds. Springer Berlin Heidelberg, 2007, vol. 4739, pp. 780–787. 133, 140, 141, 144

[114] S. I. Roumeliotis, G. S. Sukhatme, and G. A. Bekey, "Circumventing dynamic modeling : evaluation of the error-state kalman filter applied to mobile robot localization," in *IEEE International Conference on Robotics and Automation*, vol. 2, 1999, pp. 1656–1663 vol.2. 87

[115] Z. Sahinoglu, S. Gezici, and I. Güvenc, *Ultra-wideband Positioning Systems : Theoretical Limits, Ranging Algorithms, and Protocols*. Cambridge University Press, 2008. [Online]. Available : https://books.google.fr/books?id=ygXzSEzQy4oC 21

[116] A. H. Sayed, A. Tarighat, and N. Khajehnouri, "Network-based wireless location : challenges faced in developing techniques for accurate wireless location information," *IEEE Signal Processing Magazine*, vol. 22, no. 4, pp. 24–40, July 2005. 21, 40, 43

[117] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer Berlin Heidelberg, 2008. 5

[118] R. Siegwart and I. R. Nourbakhsh, *Introduction to Autonomous Mobile Robots*. Bradford Company, 2004. xi, 4, 5, 14, 17, 18, 39

[119] J. Sliwka, F. L. Bars, O. Reynet, and L. Jaulin, "Using interval methods in the context of robust localization of underwater robots," in *Annual Meeting of the North American Fuzzy Information Processing Society*, March 2011, pp. 1–6. 42

[120] J. Solà, "Quaternion kinematics for the error-state KF," Mar 2015, working paper or preprint. [Online]. Available : https://hal.archives-ouvertes.fr/hal-01122406 87, 88

[121] T. L. Song, "Observability of target tracking with range-only measurements," *IEEE Journal of Oceanic Engineering*, vol. 24, no. 3, pp. 383–387, Jul 1999. 69

[122] M. J. Spenko, G. C. Haynes, J. A. Saunders, M. R. Cutkosky, A. A. Rizzi, R. J. Full, and D. E. Koditschek, "Biologically inspired climbing with a hexapedal robot," *Journal of Field Robotics*, vol. 25, no. 4-5, pp. 223–242, 2008. [Online]. Available : http://dx.doi.org/10.1002/rob.20238 1

[123] J. R. Spletzer, "A new approach to range-only slam for wireless sensor networks," *Lehigh University, Bethlehem, PA, Tech. Rep*, vol. 8, 2003. 106, 108

[124] D. Tedaldi, A. Pretto, and E. Menegatti, "A robust and easy to implement method for imu calibration without external equipments," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 3042–3049. 184

[125] S. Thrun, "Particle filters in robotics," in *Eighteenth Conference on Uncertainty in Artificial Intelligence*, ser. UAI'02. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 2002, pp. 511–518. [Online]. Available : http://dl.acm.org/citation.cfm?id=2073876.2073937 39

[126] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005. 5, 38, 39, 76

[127] A. Torres-González, J. R. M. d. Dios, and A. Ollero, "Efficient robot-sensor network distributed SEIF range-only SLAM," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 1319–1326. 102, 104, 106, 108

[128] A. Torres-González, J. R. M. de Dios, and A. Ollero, "Accurate fast-mapping Range-Only SLAM for UAS applications," in *International Conference on Unmanned Aircraft Systems (ICUAS)*, June 2015, pp. 543–550. 106, 108

[129] A. Torres-González, J. R. Martinez-de Dios, and A. Ollero, *Integrating Internode Measurements in Sum of Gaussians Range Only SLAM*. Springer International Publishing, 2014, pp. 473–487. [Online]. Available : https://doi.org/10.1007/978-3-319-03653-3_35 106

[130] ——, "An adaptive scheme for robot localization and mapping with dynamically configurable inter-beacon range measurements," *Sensors*, vol. 14, no. 5, pp. 7684–7710, 2014. [Online]. Available : http://www.mdpi.com/1424-8220/14/5/7684 108

[131] A. Torres-González, J. R. Martínez-de Dios, and A. Ollero, "Robot-Beacon Distributed Range-Only SLAM for Resource-Constrained Operation," *Sensors*, vol. 17, no. 4, 2017. [Online]. Available : http://www.mdpi.com/1424-8220/17/4/903 104, 105, 106, 108

[132] C. D. Wagter, S. Tijmons, B. D. W. Remes, and G. C. H. E. de Croon, "Autonomous flight of a 20-gram flapping wing mav with a 4-gram onboard stereo vision system," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 4982–4987. 1

[133] Z. Wang and G. Dissanayake, "Observability analysis of SLAM using fisher information matrix," in *10th International Conference on Control, Automation, Robotics and Vision*, Dec 2008, pp. 1242–1247. 69

[134] O. Wijk and H. Christensen, "Triangulation-based fusion of sonar data with application in robot pose tracking," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 6, pp. 740–752, Dec 2000. 101, 109, 114

[135] P. Yang, "Efficient particle filter algorithm for ultrasonic sensor-based 2D range-only simultaneous localisation and mapping application," *IET Wireless Sensor Systems*, vol. 2, no. 4, pp. 394–401, December 2012. 102, 108

[136] R. Yarlagadda, I. Ali, N. Al-Dhahir, and J. Hershey, "GPS GDOP metric," *IEE Proceedings Radar, Sonar and Navigation*, vol. 147, no. 5, pp. 259–264, Oct 2000. 132, 140

[137] S. Yu, S. Ma, B. Li, and Y. Wang, "An amphibious snake-like robot with terrestrial and aquatic gaits," in *IEEE International Conference on Robotics and Automation*, May 2011, pp. 2960–2961. 1

[138] Y. Zhou, "An efficient least-squares trilateration algorithm for mobile robot localization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2009, pp. 3474–3479. 40, 101

UNIVERSITÉ DE STRASBOURG

Lionel GÉNEVÉ

msii ÉCOLE DOCTORALE

# Système de déploiement d'un robot mobile autonome basé sur des balises

## Résumé

Cette thèse s'inscrit dans le cadre d'un projet visant à développer un robot mobile autonome capable de réaliser des tâches spécifiques dans une zone préalablement définie par l'utilisateur. Afin de faciliter la mise en œuvre du système, des balises radiofréquences fournissant des mesures de distance par rapport au robot sont disposées au préalable autour du terrain. Le déploiement du robot s'effectue en deux phases, une première d'apprentissage du terrain, puis une seconde, où le robot effectue ses tâches de façon autonome. Ces deux étapes nécessitent de résoudre les problèmes de localisation et de localisation et cartographie simultanées pour lesquels différentes solutions sont proposées et testées en simulation et sur des jeux de données réelles. De plus, afin de faciliter l'installation et d'améliorer les performances du système, un algorithme de placement des balises est présenté puis testé en simulation afin de valider notamment l'amélioration des performances de localisation.

**Mots clés :** Robotique mobile, localisation, SLAM, placement optimal de capteurs, balises radiofréquence, mesures de distance.

## Abstract

This thesis is part of a project which aims at developing an autonomous mobile robot able to perform specific tasks in a preset area. To ease the setup of the system, radio-frequency beacons providing range measurements with respect to the robot are set up beforehand on the borders of the robot's workspace. The system deployment consists in two steps, one for learning the environment, then a second, where the robot executes its tasks autonomously. These two steps require to solve the localization and simultaneous localization and mapping problems for which several solutions are proposed and tested in simulation and on real datasets. Moreover, to ease the setup and improve the system performances, a beacon placement algorithm is presented and tested in simulation in order to validate in particular the improvement of the localization performances.

**Keywords :** Mobile robotics, localization, SLAM, optimal sensor placement, radiofrequency beacons, range-only measurements.