Samy Bourzoufi

Projet de fin d'études FISE2024

Localisation collective d'une flotte de robots

26 août 2024

Tuteurs : Antoine Nongaillard et Philippe Mathieu

Tuteur ENSTA : Luc Jaulin

Remerciements

Je tiens tout d'abord à exprimer ma sincère gratitude à l'équipe SMAC (systèmes multi-agents) et tout particulièrement à mes tuteurs, Monsieur Antoine Nongaillard et Monsieur Philippe Mathieu, pour leurs précieux conseils, leur écoute attentive, et la liberté qu'ils m'ont accordée dans l'orientation de mes recherches. Mes remerciements s'adressent également à la plateforme PRETIL pour m'avoir fourni tout le matériel nécessaire à la réalisation de mon travail, en particulier à Maxime Dusquene pour son aide précieuse concernant le matériel hardware et à Monsieur Gérald Dherbomez pour m'avoir offert l'opportunité de présenter mes travaux en public. Enfin, je remercie Madame Cindy Cappelle pour ses critiques constructives et ses conseils avisés sur les aspects liés à la fusion de données. Je souhaite également saluer chaleureusement l'équipe de Polytech Nancy pour leur collaboration enrichissante tout au long de ce projet, ainsi que pour leur visite au laboratoire, qui a été une source précieuse d'échanges et de partage de connaissances. Mes remerciements s'adressent enfin à Nathan Hallez, qui m'a épaulé avec efficacité et disponibilité dans mes manipulations, contribuant ainsi au bon déroulement de mes travaux.

1 Résumé

Dans le cadre de ma formation d'ingénieur, j'ai effectué un projet de fin d'études de 21 semaines, traitant des comportements de groupe d'une flotte de robots hétérogène, dans un cadre d'exploration de territoire inconnu. Par hétérogène, on entend que le groupe est formé de différents types de robots, par exemple des drones aériens et des drones terrestres. L'exploration d'un territoire par des robots appelle à bons nombres de problématiques telles que la localisation, la navigation ainsi que l'optimisation de chemin. Ces problématiques deviennent d'autant plus intéressantes lorsque que l'on offre la possibilité aux robots de communiquer et s'entraider. Dans des environnements denses tels qu'une forêt, l'utilisation d'un capteur GPS devient inefficace voire impossible. Des solutions existent cependant comme le SLAM. Le projet de recherche sur lequel j'ai été amené à travailler reposant sur la collaboration entre agents, il était plus intéressant de se tourner vers une méthode de localisation collaborative. La localisation par multilatération, qui consiste à calculer une position à partir de données de distances, montre en simulation, la possibilité de localiser les robots même en présence d'erreurs de mesures importantes. Pour les capteurs, nous utiliserons d'ailleurs une technologie plutôt récente : l'Ultra Wide Band, offrant de nombreux avantages par rapport à d'autres solutions comme le bluetooth ou les capteurs ultrasons. Au travers de nombreuses simulations et propositions, nous parvenons à la conclusion que la multilatération seule peut s'avérer plutôt efficace, mais limitée pour l'exploration. Cette méthode contribue à un ensemble de méthodes de localisation dont les robots disposent. Un exemple de fusion de données sur l'utilisation d'un filtre de Kalman Étendu est présenté pour aboutir à une localisation de groupe plus prometteuse pour la suite du projet.

Abstract

As part of my engineering training, I carried out a 21-week end-of-study project, dealing with the group behaviors of a heterogeneous fleet of robots, in a context of exploring unknown territory, in order to prevent forest fire. By heterogeneous, we mean that the group is made up of different types of robot, for example aerial drones and terrestrial drones. The exploration of a territory by robots involves a number of issues such as localization, navigation and path optimization. These issues become all the more interesting when we offer robots the possibility of communicating and helping each other. In dense environments such as a forest, using a GPS sensor becomes inefficient or impossible. However, solutions exist such as SLAM. As the research project on which I was led to work was based on collaboration between agents, it was more interesting to turn to a collaborative localization method. Localization by multilateration, which calculates a position form distances data, shows in simulation the possibility of locating robots even in the presence of significant measurement errors. For the sensors, we will use a recent technology : Ultra Wide Band, offering many advantages compared to other solutions such as bluetooth or ultrasonic sensors. Through numerous simulations and proposals, we reach the conclusion that multilateration alone can be rather effective alone, but limited for exploration. This method contributes to a set of localization methods available to robots. An example of data fusion using an Extended Kalman Filter is presented and we end up with a group localization more promising for the rest of the project.

Keywords— Ultra wide band, Multilateration, Portable landmakrs, Kalman Filter

Table des matières

2 Introduction

Les agents sont des entités capables d'observations, de prises de décisions et d'actions dans un environnement donné. Ils sont souvent modélisés comme des entités computationnelles ou physiques, telles que des robots, des logiciels, ou même des humains dans certains contextes. Un agent est avant tout autonome, au sens où il décide des actions à mener en fonction de ses propres perceptions. A priori on ne lui donne pas d'ordres, ou alors de très haut niveau. Un système multi-agents est un ensemble d'agents interagissant les uns avec les autres pour atteindre des objectifs individuels et/ou collectifs. Ces systèmes tirent souvent parti des comportements de groupe observés chez les humains pour résoudre des problèmes complexes. Par exemple, une patrouille de sécurité peut être modélisée comme un système multi-agents où chaque agent est responsable d'une zone spécifique, mais collabore avec les autres pour assurer la sécurité globale. De même, les organisations hiérarchiques sont souvent imitées dans les systèmes multi-agents pour coordonner efficacement les actions individuelles vers des objectifs communs. Dans le cas des flottes de robots, les comportements de groupe peuvent permettre une meilleure exploration d'un environnement, une couverture plus efficace d'une zone ou une réponse coordonnée à des événements imprévus. Un autre intérêt du formalisme multi-agents est la décentralisation de l'information. En effet, en considérant un groupe de robots naviguant dans un environnement, une communication globale permise par une centralisation de l'information peut vite devenir impossible en raison des limitations de distance de communication.

Mon stage a eu lieu au laboratoire CRIStAL (Centre de Recherche en Informatique, Signal et Automatique de Lille), au sein de l'équipe SMAC (spécialisée dans les systèmes multi-agents et les comportements). Cette expérience s'inscrit dans le cadre d'un projet soutenu par l'agence nationale de la recherche (ANR) et traite des comportements collectifs entre robots. En particulier, ce projet ANR se concentre sur la problématique suivante : un groupe de robots explore une forêt inconnue, pour soutenir une équipe de pompiers. Leur mission est alors de localiser les potentiels incendies de la zone et aider à les éteindre. Comme beaucoup de projets, celui-ci fait collaborer différentes équipes avec des spécialités différentes, par exemple une équipe de fusion de données avec une équipe spécialisée en systèmes multi-agents. Ce projet regroupe aussi des acteurs venus d'autres entités de recherche, en particulier pour ce projet, une équipe de chercheurs de Polytech Nancy. Ce stage de fin d'études marque d'ailleurs le début de ce projet, c'est pourquoi la majeure partie des travaux effectués au cours de cette expérience se résume à de la simulation.

L'approche multi-agents permet de simuler une expérience, dans laquelle des agents répondant à des règles et des hypothèses, interagissent entre eux et avec l'environnement. On peut par exemple imaginer un système de traders (les agents) achetant ou vendant des actions sur le marché financier (l'environnement). Ces agents disposent donc d'une capacité d'observation : accès aux caractéristiques du marché, de décision (on leur définit une stratégie), et d'action (la vente, l'achat ou ne rien faire). Le tout est alors de voir quelle stratégie "marche" le mieux. Cela peut se faire par simulation dans un environnement haut niveau comme NetLogo, en faisant varier les caractéristiques des agents. Netlogo est à la fois un langage de programmation et un environnement de simulation pour le développement de systèmes multi-agents. La figure 1 montre un exemple de simulation sur l'interface de Netlogo. Cet outil s'avère extrêmement efficace pour réaliser des simulations comme le modèle proie-prédateur ou encore la propagation d'une épidémie. Cet outil est redoutable grâce au caractère très haut niveau du langage, mais aussi grâce à la facilité de créer une interface, munie de tous types de widgets.

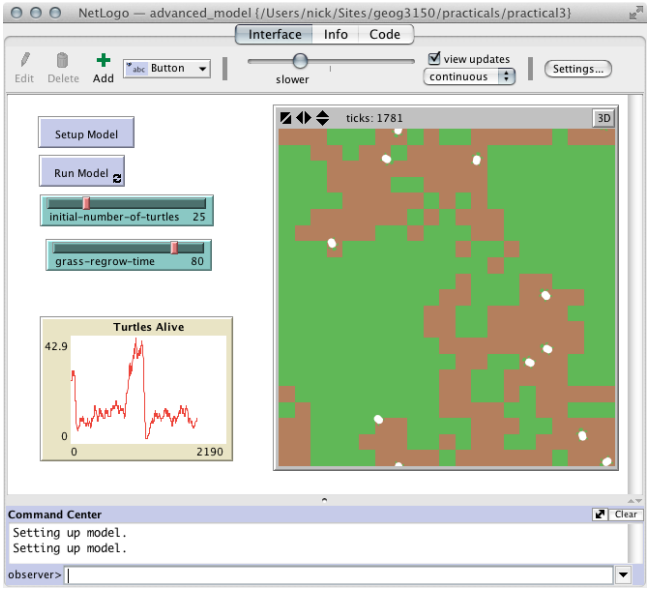

FIGURE 1 – Exemple de simulation sur Netlogo.

Cependant, il s'agit là d'une approche très haut niveau : les agents évoluent dans un monde parfait où chaque action est exécutée avec succès, ce qui n'est généralement pas le cas dans un contexte de robotique. Prenons l'exemple suivant : un robot doit se rendre d'un point A à un point B. Dans l'approche agent et surtout comme le ferait un informatitien, on aura tendance à déplacer instantanément le robot du point A à B, ou bien à le déplacer de "case en case" de A à B (la problématique de mobilité est totalement ignorée). En revanche, un roboticien considérera les actionneurs dont dispose son robot, ses moyens de se localiser ainsi que la loi de commande à fournir afin que la position du robot converge vers le point d'arrivée, tout en tenant compte de perturbation ou d'erreur de mesures etc. En particulier toutes sortes de problèmes peuvent arriver dans le monde réel, comme des pertes de signal, des collisions ou autres pannes. Il est possible de caractériser l'état d'un robot par plusieurs variables évoluant au cours du temps. Ceci aurait pour but de définir un état de santé ou d'efficacité d'un robot au cours de la mission, afin d'en déduire le comportement à adopter.

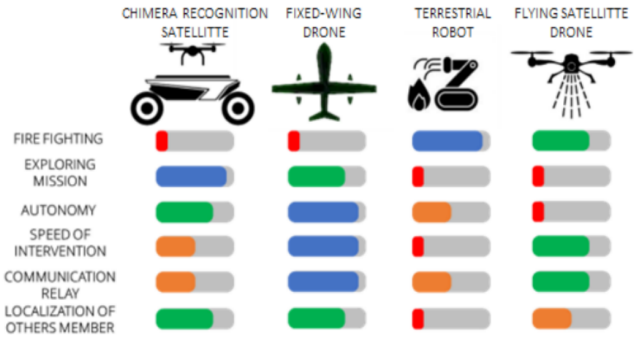

FIGURE 2 – Illustration conceptuelle des caractéristiques des agents au cours d'une mission .

Mon rôle est alors d'appliquer des stratégies de groupes dans une flotte de robots tout en tenant compte des problèmes qu'un robot mobile peut rencontrer lors d'une mission. La stratégie et la méthode de localisation fonctionneront en autonomie sur chaque robot, sans quelconque intervention extérieure. Nous nous concentrerons sur la localisation collective des robots, ainsi que sur leur navigation.

En mettant en place une méthode de localisation collective, dont nous remarquerons les avantages et les inconvénients, nous en déduirons des problèmes sous-jacents ainsi que des hypothèses à respecter quant-au comportement à adopter d'un point de vue de l'agent.

3 Etat de l'art et contexte

La localisation est un aspect incontournable de la robotique mobile, car il est obligatoire pour un robot de connaître sa position et son attitude dans un environnement donné, surtout si celui-ci se déplace de façon autonome. La localisation est une phase qui d'un point de vue de l'automatisme se trouve entre la mesure de son environnement et la prise de décision. Parmi les méthodes classiques, nous trouvons l'odométrie, le GPS, ou encore le SLAM (Simultaneous Localization and mapping, en français : cartographie et localisation simultanées). Les systèmes de localisation basés sur le GNSS se sont constamment améliorés au cours du temps grâce au RTK (real-time kinematic) pour atteindre une précision centimétrique. Cependant, la précision de ces systèmes se verra réduite quand ceux-ci se verront confrontés à des environnements denses. Dans le contexte de l'essaim de robots, le GNSS pose aussi problème lorsque l'essaim se trouve dans une configuration trop dense et que les robots se bloquent entre eux. De même, les techniques odométriques sont inutilisables dans un environnement tel qu'une forêt, et ce pour plusieurs raisons. D'abord le sol d'une forêt n'étant pas parfaitement plat, il est impossible de calculer avec précision la trajectoire d'un robot terrestre. Le SLAM, quant à lui, nécessite une puissance de calcul considérable et peut être difficile à mettre en place. Pour ces raisons, les techniques de localisation relative connaissent un essor dans la recherche depuis quelques années, car elles résolvent les problèmes de localisation en environnement complexe, dans un contexte de collaboration [2].

La localisation relative repose sur des systèmes de mesures comme le Bluetooth, la RFID (radio frequency identification), la technologie **UWB** (ultra wide band), les lidars, les caméra RGB ou encore les caméras infrarouges. Chen et al [2], dans leur article regroupent l'ensemble des méthodes que l'on retrouve dans la littérature récente et résument les caractéristiques des solutions produites. Les auteurs parviennent à la conclusion que la technologie UWB serait la plus prometteuse car cette méthode permet d'atteindre un niveau de précision centimétrique jusqu'à 100m, dans un contexte de collaboration. Toutefois, cette solution posera problème au fur et à mesure que le nombre de robots dans l'essaim augmente.

Le projet ANR regroupant déjà une équipe travaillant sur la fusion de données, il était plus bénéfique pour moi en termes de contribution de travailler sur une méthode de localisation relative. D'une part car la plateforme de robotique ne disposait pas de telles solutions, mais aussi car la localisation relative s'avère intéressante sur plusieurs points en termes de collaboration. Ce sujet se rapproche donc plutôt bien des thématiques qu'abordait l'équipe SMAC. C'est pourquoi j'ai choisi de partir dans cette direction avec l'accord de mes tuteurs.

La localisation relative dans notre contexte se résume donc à la phrase suivante : une flotte de robots évolue dans un environnement inconnu ; flotte dont chaque individu estime sa position à partir des distances entre lui et ses pairs. Mes contributions au projet en rapport avec la localisation relative sont inspirées de deux publications. La première décrit un protocole pour obtenir une localisation centimétrique à l'aide de capteurs UWB [6]. La deuxième publication traite du concept de "**portable landmarks**" et du déplacement en groupe à partir de données relatives [5]

4 Localisation collective

4.1 Objectifs et hypothèses retenues

Le but de cette partie étant d'établir une méthode de localisation collective, nous mettrons de côté les caractéristiques des robots sans rapport direct avec l'estimation de position. De ce fait, nous considérons N robots tous identiques, ainsi que leurs positions spatiales. Dans le contexte d'une exploration d'environnement ouvert comme une forêt, on pourrait munir chaque robot d'un capteur GNSS et estimer sa position à partir de cet unique capteur (dans ce cas, aucun comportement de groupe n'est nécessaire). On se propose alors de munir nos robots de capteurs de distances type UWB (ultra wide band) : un émetteur et un récepteur s'échangent un signal, en connaissant la vitesse de propagation des signaux, le capteur déduit du temps de propagation ("time of flight") la distance. Il est alors possible d'utiliser le principe de multilatération afin de se localiser dans l'espace. Par définition, la multilatération est la détermination d'un point isolé par mesure de plusieurs distances à partir de plusieurs points connus, sans mesure d'angles.

4.2 Un robot et des landmarks

Pour illustrer le concept de multilatération, dans un premier temps, nous simplifions les choses en considérant un robot sur un plan 2D, de position inconnue P et 3 **landmarks** [1] de positions connues avec certitude, p_1, p_2 et p_3. Le robot, grâce aux capteurs, connaît les distances par rapport aux 3 landmarks, d_1, d_2 et d_3. Le schéma en Figure 3 illustre la situation :

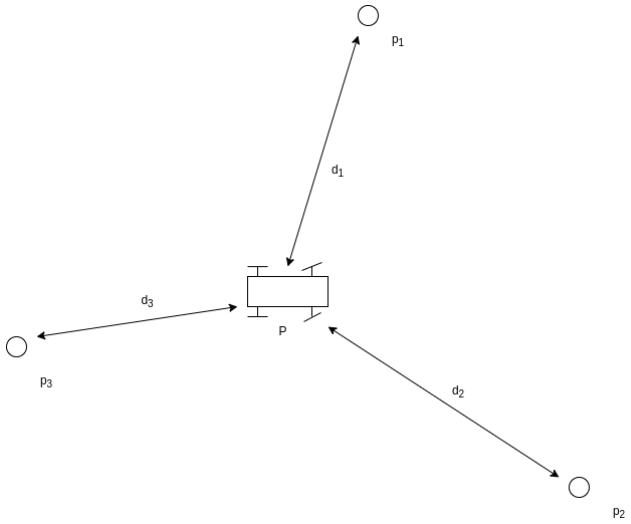

FIGURE 3 – Schéma d'un robot évaluant sa distance par rapport aux repères.

De par l'égalité théorique entre la mesure et la distance réelle, nous obtenons alors un système à 3 équations non-linéaires (pour N landmarks on a N équations) :

$$\begin{cases} \sqrt{(p_{1x} - P_x)^2 + (p_{1y} - P_y)^2} & = d_1 \\ \sqrt{(p_{2x} - P_x)^2 + (p_{2y} - P_y)^2} & = d_2 \\ \sqrt{(p_{3x} - P_x)^2 + (p_{3y} - P_y)^2} & = d_3 \end{cases} \tag{1}$$

L'ensemble des solutions de l'équation i de (1), $i \in \{1, 2, 3\}$, est l'ensemble des points appartenant au cercle de centre (p_{ix}, p_{iy}) et de rayon d_i. Si l'on utilise alors 2 landmarks nous obtenons 1 ou 2

1. En français : repère, nous garderons par la suite le terme de landmark.

solutions, issues de l'intersection des deux cercles. Utiliser 3 landmarks de positions distinctes réduit cependant l'ensemble de solutions à une solution unique. Ces deux cas sont illustrés en Figure 4. Le système étant non-linéaire, nous utilisons une méthode de calcul itérative afin d'obtenir une solution approchée de la solution exacte. Une méthode non-itérative appelée méthode "Baseline" permettant d'obtenir une estimation de la position, est exposée dans l'article [6].

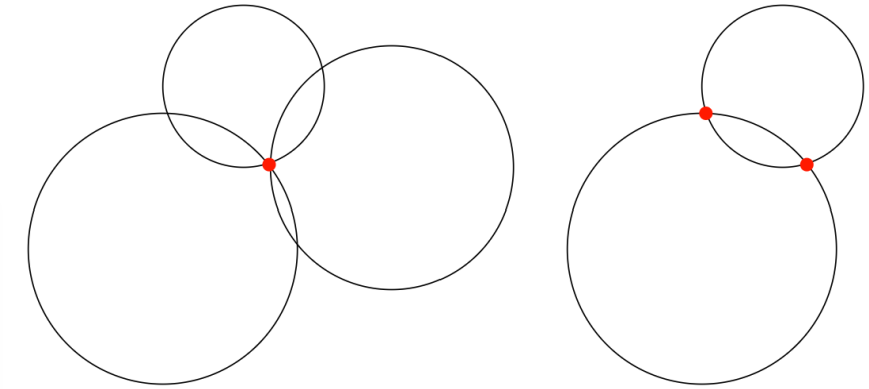

FIGURE 4 – Représentation graphique des solutions du système dans le cas de 3 et 2 landmarks.

Dans l'article [6], une disposition des capteurs dans l'espace s'appelle une **constellation**, terme que nous garderons donc pour décrire l'installation des capteurs ou landmarks. Un élément fondamental à considérer dans une constellation est la formation de symétrie. Comme expliqué précédemment, il se peut qu'une constellation produise plusieurs solutions au système d'équations. En 2 dimensions, il faudra absolument éviter qu'une constellation forme une droite (voir figure 5). Dans le cas de deux capteurs, cela est inévitable, c'est pourquoi deux solutions existent. En 3 dimensions le problème s'étend avec le fait qu'une constellation ne doit pas se trouver dans un plan. Les seules positions tolérées dans de telles constellations sont les points parfaitement au milieu de deux landmarks, ce qui réduit considérablement l'espace dans lequel le tag [2] peut naviguer. De façon générale, peu importe le nombre de dimensions avec lequel on travaille, une constellation doit à tout prix éviter de former un support de symétrie. C'est d'ailleurs ce qui est remarqué dans l'article [6] : placer un landmark au plafond améliore significativement la précision des estimations. De plus, de par la méthode de résolution que nous développons par la suite, nous ajouterons un détail (voir partie 4.2.2).

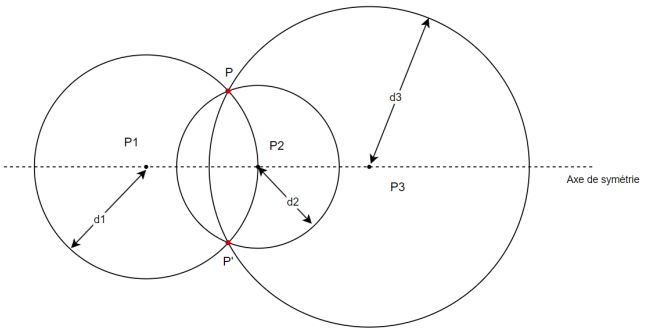

FIGURE 5 – Exemple de mauvaise constellation dûe à un alignement des landmarks dans un cas 2 dimensions, donnant deux solutions symétriques.

2. Terme utilisé pour désigné l'objet à localiser. Dans la suite du rapport, le mot "tag" sera utilisé comme objet mobile à localiser, à l'opposé de l'ancre ou landmark, immobile servant de repère.

4.2.1 Résolution de système non-linéaire avec méthode de Newton

La méthode de Newton, également connue sous le nom de méthode de Newton-Raphson [8], est une technique numérique utilisée pour des équations non-linéaires. Cette méthode est itérative et utilise des outils et hypothèses en lien avec le calcul différentiel. Construisons les outils mathématiques pour résoudre le système (1).

Posons f, la fonction calculant géométriquement le vecteur des distances entre le point P et les landmarks, ainsi que $\mathbf{d}$, le vecteur des distances mesurées par les capteurs :

$$f : \mathbb{R}^2 \to \mathbb{R}^3, \quad P \mapsto \begin{pmatrix} \|P - p_1\| \\ \|P - p_2\| \\ \|P - p_3\| \end{pmatrix} \tag{2}$$

$$\mathbf{d} = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix}$$

Posons ensuite j, la fonction associant à une position P, l'erreur entre les distances aux landmarks calculées au point P et les distances mesurées :

$$\begin{aligned} j &: \mathbb{R}^2 \to \mathbb{R}, \\ & P \mapsto \|f(P) - \mathbf{d}\|^2 . \end{aligned}$$

Par construction, résoudre le problème d'estimation revient à minimiser j, afin de trouver $\hat{P}$, l'estimation de la position :

$$\hat{P} = \underset{P \in \mathbb{R}^2}{\operatorname{argmin}}\, j(P)$$

La fonction f étant non-linéaire, il est compliqué de résoudre ce problème. Linéarisons alors f autour d'un point P_0, supposé par trop distant de la solution réelle :

$$f(P) \approx f(P_0) + J_f(P_0) \cdot (P - P_0) \tag{3}$$

Avec $J_f(P_0)$, la matrice Jacobienne de f au point P_0 :

$$J_f(P) = \begin{pmatrix} \frac{\partial f_1}{\partial P_x}(P) & \frac{\partial f_1}{\partial P_y}(P) \\ \frac{\partial f_2}{\partial P_x}(P) & \frac{\partial f_2}{\partial P_y}(P) \\ \frac{\partial f_3}{\partial P_x}(P) & \frac{\partial f_3}{\partial P_y}(P) \end{pmatrix} = \begin{pmatrix} \frac{P_x - p_{1x}}{\sqrt{(P_x - p_{1x})^2 + (P_y - p_{1y})^2}} & \frac{P_y - p_{1y}}{\sqrt{(P_x - p_{1x})^2 + (P_y - p_{1y})^2}} \\ \frac{P_x - p_{2x}}{\sqrt{(P_x - p_{2x})^2 + (P_y - p_{2y})^2}} & \frac{P_y - p_{2y}}{\sqrt{(P_x - p_{2x})^2 + (P_y - p_{2y})^2}} \\ \frac{P_x - p_{3x}}{\sqrt{(P_x - p_{3x})^2 + (P_y - p_{3y})^2}} & \frac{P_y - p_{3y}}{\sqrt{(P_x - p_{3x})^2 + (P_y - p_{3y})^2}} \end{pmatrix}$$

Remarquons que si le problème était en 3 dimensions, la jacobienne serait une matrice à 3 colonnes. De plus, il est possible de placer les landmarks hors du plan, sans que cela n'impacte la dimension de la jacobienne, il faudrait simplement ajouter les termes constants P_{iz} des landmarks dans l'expression des distances. Enfin, remarquons que la jacobienne a autant de lignes qu'il y a de landmarks dans le système.

Soit $P \in \mathbb{R}^2$. En notant $M = J_f(P_0)$, injectons l'équation (3) dans $j(P)$:

$$j(P) = \|\mathbf{d} - f(P)\|^2 \approx \|MP - \underbrace{(\mathbf{d} + MP_0 - f(P_0))}_{z}\|^2 \approx (MP - z)^\intercal (MP - z)$$

Cette nouvelle expression de j est quadratique, on en déduit alors un minimiseur P_1, plus proche de la solution que P_0. Il suffit alors de remplacer P_0 par P_1 et répeter la procédure jusqu'à respecter un critère d'arret :

$$P_1 = \overbrace{(M^\intercal M)^{-1} M^\intercal}^{K} z = P_0 + K(\mathbf{d} - f(P_0))$$

Nous utiliserons un critère d'arrêt sur la distance entre les points itérés, par exemple avec une distance minimale $\alpha = 0.01$, entre deux itérations. Ce critère arrête donc l'algorithme si les estimations successives n'évoluent plus :

Algorithm 1 Newton

while $\|P_{n+1} - P_n\|^2 > \alpha$ **do**
  $P_n \leftarrow P_{n+1}$
  $M = J_f(P_n)$
  $K = (M^\intercal M)^{-1} M^\intercal$
  $P_{n+1} = P_n + K(\mathbf{d} - f(P_n))$
end while

```
def Newton (p):
    """ Computes Newton's method. Takes in argument a 2d array and returns a 2d array aswell"""
    first_loop = True   # boolean checking if its the first loop
    px, py = 0, 0       # current coordinates estimated, initially set to 0, 0
    counter = 0         # integer counting the number of loops done
    while (sqrt((px-p[0, 0])**2 + (py-p[1, 0])**2) > 0.01 or first_loop) and counter < 10:
        first_loop = False
        px, py = p[0,0], p[1,0]
        plot(px, py, 'b+')
        M = Jf(px, py)
        K = np.linalg.inv(M.T@M)@M.T
        p = p + K@(y-f(px, py))
        counter += 1
        pause(0.01)
    return p
```

FIGURE 6 – Morceau de code mettant en pratique la méthode de Newton

4.2.2 Simulation, résultats et remarques

Par construction, avec l'image du plan par la fonction j, nous obtenons un champ scalaire (traduisant l'erreur quadratique) visualisable à l'aide de ses isolignes. Un point d'une des lignes a donc pour valeur l'image de la fonction d'erreur calculée en ce point. De plus, le vecteur de déplacement entre chaque itération de la méthode de Newton est donc censé être orthogonal à la courbe isoligne du point d'itération sur lequel on se trouve. Nous implémentons l'algorithme de localisation dans un programme Python. Il est possible d'observer graphiquement la convergence de l'estimation vers la position réelle de notre robot, en affichant l'environnement avec matplotlib. Le robot vert est à la position exacte, les points rouges représentent les landmarks et les croix bleues représentent les estimations à chaque itération de la méthode de Newton (voir figure 7).

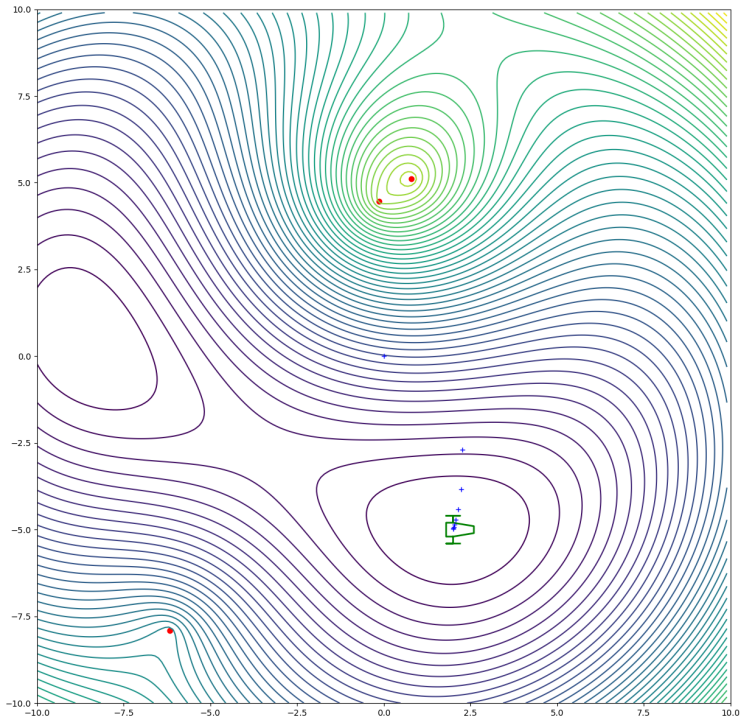

FIGURE 7 – Estimation de la position avec méthode de Newton.

En termes de convergence, nous remarquons deux choses. Premièrement, peu d'itérations sont nécessaires afin de respecter le critère d'arrêt : moins d'une dizaine. Deuxièmement, comme nous l'avons expliqué précédemment, une constellation produit 2 solutions lorsque celle-ci crée un support de symétrie. Dans le cas 2d, une droite. Une constellation présentant une symétrie parfaite présente plusieurs minimums à une valeur de zéro. En revanche, une constellation s'approchant plus ou moins d'une forme de symétrie produira des minimums locaux non nuls. La figure 7, montre un exemple d'apparition de minimum local car les 3 landmarks sont presque alignés. Or le point de départ $(0, 0)$ étant plus proche de la vraie solution, cela n'entache pas la bonne convergence. Plus une constellation s'approche d'une symétrie, plus les minimums locaux engendrés s'approchent de zéro créant une attraction plus forte pour la méthode de Newton. En effet, la méthode de Newton adaptée à notre contexte cherche un minimum de la fonction d'erreur, le cas parfait étant de trouver un minimum global dont la valeur vaut zéro. Il se peut alors qu'en choisissant un point de départ de l'algorithme trop proche d'un "mauvais" minimum local, la méthode n'aboutisse pas à la solution recherchée. Selon mes expérimentations, j'ai constaté un dernier élément à relever : augmenter le nombre de landmarks n'accélère pas la convergence.

4.2.3 Prise en compte des bruits de mesure et du mouvement

Évidemment, en réalité un capteur ne donnera jamais de mesure exacte. En considérant que nos capteurs fournissent des mesures avec justesse mais une faible fidélité, nous modélisons les mesures de la façon suivante :

$$\tilde{\mathbf{d}} = \mathbf{d} + \beta \tag{4}$$

Avec β un bruit Gaussien : $\beta : \mathcal{N}(0, \sigma^2)$, de sorte que σ décrit l'erreur de mesure du capteur, les valeurs sont fidèles, avec une certaine justesse caractérisée par σ. Appliquer la méthode précédente en prenant en compte le bruit de mesures fait que l'algorithme ne converge plus vers la solution exacte L'estimation calculée dans de telles circonstances s'écarte spatialement de la position réelle, et la fonction d'erreur en ce point est non nulle. Cela est tout à fait normal, car géométriquement il n'y a que très peu de chances que les cercles de contraintes se croisent tous en un seul point. L'écart entre l'estimation et la position réelle semble d'ailleurs être lié à l'écart type σ. Cela peut sembler logique sur le principe, mais complexe à démontrer rigoureusement. Pour illustrer ce propos, si les capteurs sont précis au mètre près, la position estimée est aussi précise au mètre près. L'idée est alors d'effectuer n mesures et pour chaque mesure i calculer une estimation $\hat{P}_i$ puis de moyenner ces estimations.

$$\hat{P} = \frac{1}{n} \sum_{i=1}^{n} \hat{P}_i \tag{5}$$

Statistiquement, plus on prend de mesures, plus l'estimation moyennée se rapproche de la position exacte. En prenant $\sigma = 1m$, une dizaine de mesures diminuent l'erreur d'estimation à quelques centimètres. Globalement, plus σ est grand, plus le nombre de mesures à moyenner doit être important. Toutefois, devoir moyenner les estimations afin d'obtenir un résultat satisfaisant peut poser problème lorsque le robot est en mouvement. En effet, en supposant notre robot en mouvement, l'estimation moyennée $\hat{P}$ faite sur les n dernières estimations, aura un "temps de retard" sur la position exacte (illustration en figure 8), phénomène fréquemment observé lorsque l'on calcule la moyenne glissante sur une grande fenêtre de valeurs. L'expression "temps de retard" est particulièrement vraie lorsque notre robot effectue une trajectoire dont la courbure est assez faible. En revanche, lors d'un virage important, l'estimation n'épousera pas cette courbure. L'estimation suit la position exacte comme une masse tirée par une voiture avec une corde.

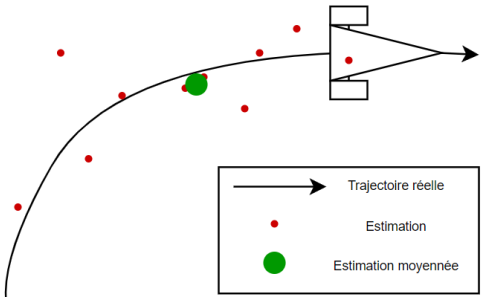

FIGURE 8 – Illustration du retard introduit par le moyennage de positions estimées.

La localisation en mouvement a été testée sur une trajectoire circulaire, avec une vitesse variable allant jusqu'à 1 m/s. Pour évaluer la précision de la méthode de localisation adaptée au mouvement, nous calculons l'erreur entre la position estimée et la position réelle à chaque instant, c'est-à-dire la

distance entre ces deux points. La figure 9 montre les courbes d'erreurs selon les deux dimensions du plan, et l'erreur sur la distance euclidienne. Nous remarquons particulièrement cette "erreur de retard", de l'ordre de quelques secondes, dûe à la taille importante de la fenêtre de moyennage (centaine de points). Cette erreur est d'autant plus importante que la vitesse est grande car l'étalement des points d'estimations augmente avec.

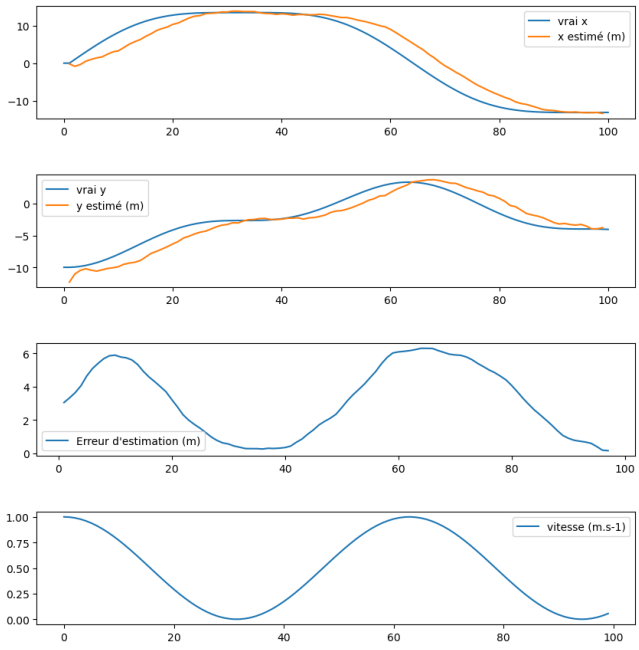

FIGURE 9 – Erreur d'estimation sans correction

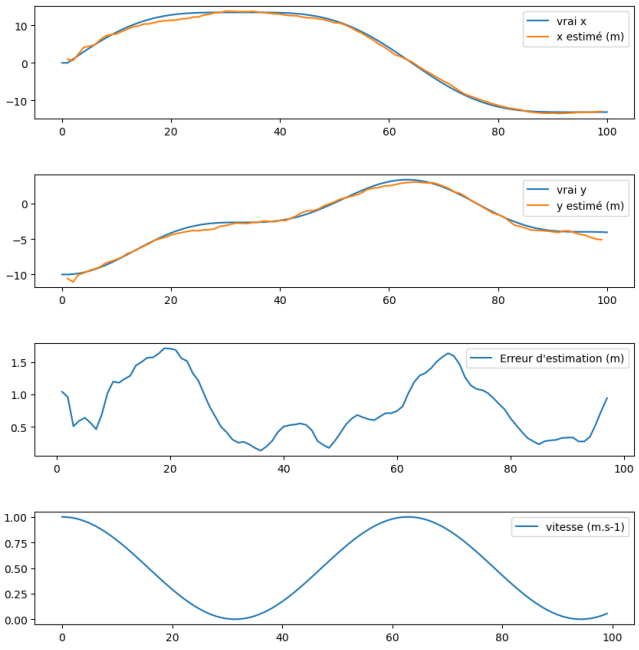

FIGURE 10 – Erreur d'estimation avec correction

Jusqu'à maintenant, nous avons basé nos estimations uniquement sur des positions. Cependant, en utilisant des données inertielles ou odométriques, il est possible de corriger l'estimation sans que celle-ci ne dérive au cours du temps. L'idée est la suivante : considérons qu'à chacune des n estimations est associée une mesure de vitesse. En sommant les vitesses des $n/2$ derniers points, nous obtenons un vecteur. En sommant l'estimation moyennée à ce vecteur, nous obtenons une estimation en théorie plus proche de la position réelle. L'instant suivant, une nouvelle estimation provenant de l'algorithme de Newton remplace la plus ancienne des n estimations, les estimations moyennées et corrigées sont alors calculées, ce qui empêche la dérive (intégration sur un temps court). Le principe de la correction est illustré en figure 11. Cette méthode réduit l'erreur mais pas totalement, ce que l'on observe en simulation en figure 10.

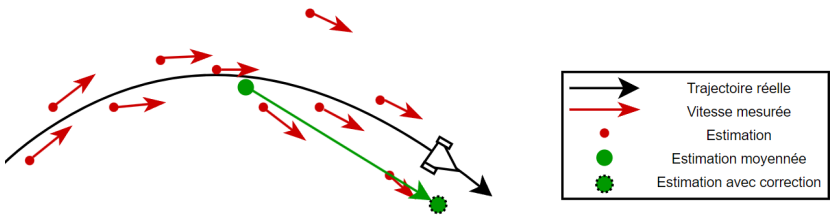

FIGURE 11 – Illustration du principe de correction par intégration limitée.

L'estimation d'abord vue en (5), devient alors :

$$\hat{P} = \frac{1}{n}\sum_{i=1}^{n} \hat{P}_i + \alpha \sum_{i=n/2}^{n} v_i \Delta t_i$$

Avec alpha un coefficient positif pour régler l'influence de la correction, v_i la vitesse mesurée au moment de l'estimation de $\hat{P}_i$ et Δt_i l'écart de temps entre les estimations de position i et $i+1$.

4.3 Possibilité d'utilisation du filtre de Kalman étendu

Un autre avantage de l'utilisation de capteur UWB, est la possibilité de mettre en place un filtre de Kalman [4]. Le filtre de Kalman est un algorithme utilisé pour estimer l'état d'un système dynamique stochastique à partir de mesures bruitées. Il fonctionne en deux étapes : la prédiction, où l'état futur est estimé en utilisant, par exemple, le modèle dynamique du système, et la correction, où cette estimation est ajustée en fonction des nouvelles mesures observées. Pour les systèmes linéaires, le filtre de Kalman standard suffit, mais pour les systèmes non linéaires, des variantes comme le filtre de Kalman étendu (EKF), qui linéarise le modèle autour de l'estimation actuelle, et le filtre de Kalman unscented (UKF), qui utilise des points d'échantillonnage pour mieux gérer les non-linéarités, sont utilisées. Ces filtres permettent de traiter efficacement les incertitudes et les bruits présents dans les systèmes réels, améliorant ainsi la précision des estimations d'état. Dans le cas d'utilisation d'un filtre de Kalman classique, nous étudions un système sous la forme :

$$\begin{cases} \mathbf{x}_{k+1} = \mathbf{A}_k \mathbf{x}_k + \mathbf{u}_k + \boldsymbol{\alpha}_k \\ \mathbf{y}_k = \mathbf{C}_k \mathbf{x}_k + \boldsymbol{\beta}_k \end{cases}$$

Ce système est linéaire et $\boldsymbol{\alpha}_k$ ainsi que $\boldsymbol{\beta}_k$ sont des vecteurs aléatoires Gaussiens indépendants dans le temps (bruit blanc Gaussien). Considérons dans notre contexte que le robot soit représenté par sa position et sa vitesse, donc avec le vecteur d'état suivant :

$$\mathbf{x}_k = \big(P_x, P_y, V_x, V_y\big)^T$$

En discrétisant avec un schéma d'Euler, nous avons la relation suivante :

$$\mathbf{x}_{k+1} = \mathbf{x}_k + dt\, \dot{\mathbf{x}}_k \tag{6}$$

Avec $\dot{\mathbf{x}}_k$, la dérivé temporelle du vecteur d'état :

$$\dot{\mathbf{x}}_k = \underbrace{\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{M}} \mathbf{x}_k + \mathbf{a}_k \tag{7}$$

et $\mathbf{a}_k$ le vecteur d'accéleration fournit au système :

$$\mathbf{a}_k = \begin{pmatrix} 0 \\ 0 \\ a_{x_k} \\ a_{y_k} \end{pmatrix}$$

En injectant l'équation (7) dans l'équation (6), nous obtenons alors l'équation d'évolution suivante :

$$\mathbf{x}_{k+1} = (\underbrace{\mathbf{I} + dt\,\mathbf{M}}_{\mathbf{A}})\mathbf{x}_k + \underbrace{dt\,\mathbf{a}_k}_{\mathbf{u}_k} \tag{8}$$

Nous sommes alors bien dans le cas linéaire de l'équation d'évolution. Etudions à présent l'équation d'observation. L'observation dans notre contexte se base sur les mesures de l'ensemble des capteurs UWB. Le calcul des distances se faisant à partir des positions des landmarks, nous avons alors :

$$\mathbf{y}_k = g(\mathbf{x}_k) \tag{9}$$

où g est la fonction (2) adaptée au vecteur d'état (ici P est le vecteur des deux premiers éléments de $\mathbf{x}_k$) :

$$g : \mathbb{R}^4 \to \mathbb{R}^n, \quad \mathbf{x}_k \mapsto \begin{pmatrix} \|P - \mathbf{p}_1\| \\ \|P - \mathbf{p}_2\| \\ \vdots \\ \|P - \mathbf{p}_n\| \end{pmatrix}$$

La fonction g étant non linéaire, nous linéariserons g en un point d'estimation $\hat{\mathbf{x}}_k$, d'où l'appellation du filtre de Kalman étendu (EKF). En réalité, nous n'utiliserons qu'une partie des équations de l'EKF, car seule l'équation d'observation doit être linéarisée. Nous devons alors calculer la jacobienne de g appliquée en une estimation $\hat{\mathbf{x}}_k$ dont les coordonnées de position (les deux premières) sont notées $\hat{P}_{x_k}$, et $\hat{P}_{y_k}$:

$$\mathbf{C}_k = \begin{pmatrix} \frac{\hat{P}_{x_k} - p_{1x}}{\sqrt{(\hat{P}_{x_k} - p_{1x})^2 + (\hat{P}_{y_k} - p_{1y})^2}} & \frac{\hat{P}_{y_k} - p_{1y}}{\sqrt{(\hat{P}_{x_k} - p_{1x})^2 + (\hat{P}_{y_k} - p_{1y})^2}} & 0 & 0 \\ \frac{\hat{P}_{x_k} - p_{2x}}{\sqrt{(\hat{P}_{x_k} - p_{2x})^2 + (\hat{P}_{y_k} - p_{2y})^2}} & \frac{\hat{P}_{y_k} - p_{2y}}{\sqrt{(\hat{P}_{x_k} - p_{2x})^2 + (\hat{P}_{y_k} - p_{2y})^2}} & 0 & 0 \\ \frac{\hat{P}_{x_k} - p_{3x}}{\sqrt{(\hat{P}_{x_k} - p_{3x})^2 + (\hat{P}_{y_k} - p_{3y})^2}} & \frac{\hat{P}_{y_k} - p_{3y}}{\sqrt{(\hat{P}_{x_k} - p_{3x})^2 + (\hat{P}_{y_k} - p_{3y})^2}} & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\hat{P}_{x_k} - p_{nx}}{\sqrt{(\hat{P}_{x_k} - p_{nx})^2 + (\hat{P}_{y_k} - p_{ny})^2}} & \frac{\hat{P}_{y_k} - p_{ny}}{\sqrt{(\hat{P}_{x_k} - p_{nx})^2 + (\hat{P}_{y_k} - p_{ny})^2}} & 0 & 0 \end{pmatrix}$$

Avec la linéarisation, l'équation d'observation devient alors :

$$\mathbf{y}_k \approx g(\hat{\mathbf{x}}_k) + \mathbf{C}_k \cdot (\mathbf{x}_k - \hat{\mathbf{x}}_k) \tag{10}$$

Puis, en effectuant un changement de variable, nous arrivons alors à l'équation linéaire suivante :

$$\underbrace{\mathbf{y}_k - g(\hat{\mathbf{x}}_k) + \mathbf{C}_k \cdot \hat{\mathbf{x}}_k}_{\mathbf{z}_k} = \mathbf{C}_k \cdot \mathbf{x}_k \tag{11}$$

Nous pouvons alors regrouper l'ensemble des équations de prédiction et de correction dans la fonction suivante, prenant en argument une estimation initiale $\hat{\mathbf{x}}_{\text{pred}}$ ainsi que sa matrice de covariance Γ_{pred}, une entrée $\mathbf{u}$ (l'accéleration du système), une mesure $\mathbf{y}$, les matrices de covariance des bruits blancs Γ_α, Γ_β et la matrice d'évolution $\mathbf{A}$:

Function Extended Kalman Filter ($\hat{\mathbf{x}}_{\text{pred}}$, Γ_{pred}, $\mathbf{u}$, $\mathbf{y}$, Γ_α, Γ_β, $\mathbf{A}$)
$\mathbf{C} := \frac{\partial g}{\partial \mathbf{x}}(\hat{\mathbf{x}}_{\text{pred}})$
$\mathbf{S} := \mathbf{C} \cdot \Gamma_{\text{pred}} \cdot \mathbf{C}^\top + \Gamma_\beta$
$\mathbf{K} := \Gamma_{\text{pred}} \cdot \mathbf{C}^\top \cdot S^{-1}$
$\tilde{\mathbf{z}} := \mathbf{y} - g(\hat{\mathbf{x}}_{\text{pred}})$
$\hat{\mathbf{x}}_{\text{up}} := \hat{\mathbf{x}}_{\text{pred}} + \mathbf{K} \cdot \tilde{\mathbf{z}}$
$\Gamma_{\text{up}} := (\mathbf{I} - \mathbf{K} \cdot \mathbf{C})\Gamma_{\text{pred}}$
$\hat{\mathbf{x}}_{\text{pred}} := \mathbf{A} \cdot \hat{\mathbf{x}}_{\text{up}} + \mathbf{u}$
$\Gamma_{\text{pred}} := \mathbf{A} \cdot \Gamma_{\text{up}} \cdot \mathbf{A}^\top + \Gamma_\alpha$
Return $\hat{\mathbf{x}}_{\text{pred}}$, Γ_{pred}

L'approche du Filtre de Kalman fait que la localisation ne repose plus sur la multilatération, mais plutôt sur l'intégration des paramètres cinématiques de notre robot. En revanche, l'équation d'observation repose quant-à elle bien sur la mesure des distances, et le bruit de mesure influence la caractérisation de l'innovation, sa covariance. Pour utiliser cette méthode, nous devons alors partir du postulat que notre robot dispose de capteurs proprioceptifs, comme une centrale inertielle, et dispose donc d'un moyen de connaître son accélération dans un repère absolu. Ainsi, les erreurs sur les mesures de distances et d'accélérations sont modélisées par les bruits blanc $\boldsymbol{\alpha}_k$ et $\boldsymbol{\beta}_k$ ainsi que leurs matrices de covariances. Tout comme pour la localisation par multilatération, il est nécessaire de fournir un premier point d'estimation pour faire démarrer l'algorithme. Pour la multilatération, il faudra veiller à fournir un point pas trop éloigné de la position réelle. Pour l'EKF, il est possible de donner une position sans trop s'inquiéter de son exactitude. Il suffit simplement de fournir en contrepartie une covariance importante. En simulation python, nous remarquons plusieurs choses. L'estimation de la position est plutôt correcte en comparaison avec la multilatération. L'EKF produit aussi une suite d'estimations dont l'enchaînement est peu lisse. Un avantage du filtre de Kalman est que nous avons une information témoignant de la qualité de l'estimation grâce à la matrice de covariance. Cette matrice de covariance peut être ensuite associée à une ellipse dont le centre est l'estimation calculée. Son étalement témoigne alors du niveau de précision. En particulier, si le tag commence à s'éloigner des landmarks, l'ellipse aura tendance à s'aplatir, peu importe le niveau de précision des capteurs de distance. Cela est dû au fait qu'en éloignant le tag, l'angle d'intersection des cercles a tendance à s'aplatir. Ainsi, une faible variation du rayon d'un cercle déplacera grandement l'intersection. Ce phénomène s'illustre en simulation en figure 12, avec des paramètres pessimistes pour accentuer la taille de l'ellipse.

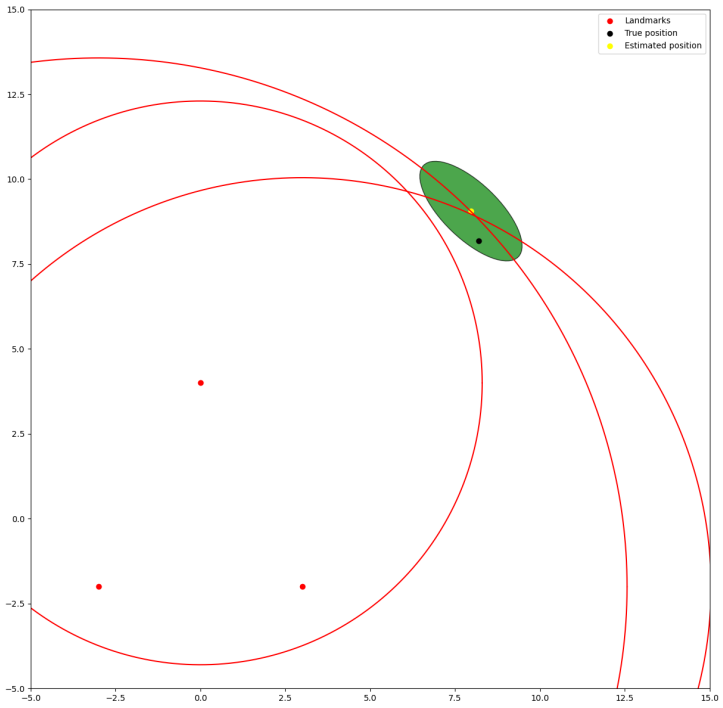

FIGURE 12 – Simulation de l'estimation de l'EKF dans un cas plan avec 3 landmarks.

4.4 Premier prototype de localisation

4.4.1 Capteurs et mise en place du système

Les résultats de la multilatération en simulation se montrant encourageants, nous avons choisi de creuser cette solution, de par ses avantages ainsi que pour les aspects collaboratifs qui en découlent. Le filtre de Kalman ne sera pas évalué plus en détail, car il s'agit du rôle de l'équipe Fusion de données. La multilatération via UWB présente plusieurs avantages. En effet, l'UWB utilise une large bande de fréquences avec des niveaux de puissance très bas, ce qui réduit les interférences avec d'autres technologies sans fil. En raison de ses transmissions à faible puissance, l'UWB peut être très économe en énergie, ce qui est bénéfique pour les appareils portables et les capteurs IoT. Enfin, cette technologie a une bonne capacité à pénétrer les obstacles comme les murs et autres structures, ce qui améliore la fiabilité de la communication dans des environnements complexes. La technologie UWB commençant à se répandre de plus en plus (notamment dans les produits Apple : dans les iPhone ou Airtag par exemple), nous trouvons sur le marché des capteurs peu chers (moins d'une centaine d'euros), avec des caractéristiques satisfaisantes vis à vis de l'avancement du projet. En effet, nous pouvons nous satisfaire de capteurs disposant d'un rayon de fonctionnement de quelques dizaines de mètres, pour une précision à l'ordre du mètre.

Nous avons alors choisi la solution de Makerfabs, UWB ESP32. Ce module comprend un système puce/antenne DWM1000 (émission/réception des signaux et calculs du time of flight), ainsi qu'un esp32. L'avantage de cette solution est d'abord sa taille et le peu de connectique nécessaire pour son fonctionnement : un simple câble micro-USB suffit pour l'alimentation et le flux de données. Makerfabs met à disposition une librairie open source pour l'utilisation des capteurs, il suffit alors simplement de téléverser les programmes dans les esp32 via l'IDE Arduino.

FIGURE 13 – Capteur Makerfabs muni de l'antenne/puce DWM1000 de Qorvo et d'un ESP32 WROOM.

Le système puce/antenne DWM1000 utilise la solution "two way ranging", qui définit deux rôles pour les capteurs : le tag et l'ancre. Ces rôles définissent la façon dont les signaux et messages sont échangés entre modules. Le two way ranging se passe de la façon suivante : le tag émet de façon périodique un signal afin de chercher la présence d'une ancre ; si ancre il y a, celle-ci répond et communique son adresse afin de s'appairer ; enfin, l'échange de signaux pour le calcul de temps de trajet a lieu.

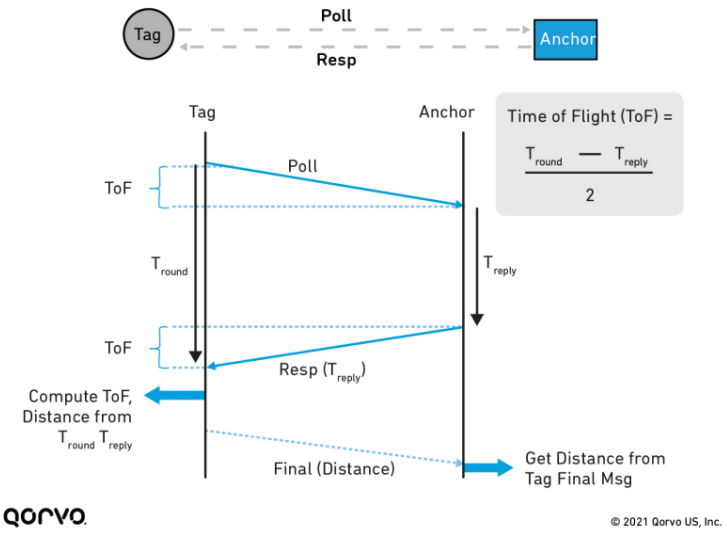

FIGURE 14 – Schéma du Two way ranging entre le tag et l'ancre.

La multilatération ayant pour but de localiser des systèmes dans un environnement en 3 dimensions, nous avons donc besoin d'un minimum de 4 ancres ainsi que d'un tag. L'estimation de position, bien que cela soit possible, ne sera pas réalisée dans l'ESP32 du tag même, mais plutôt à bord d'un ordinateur embarqué.

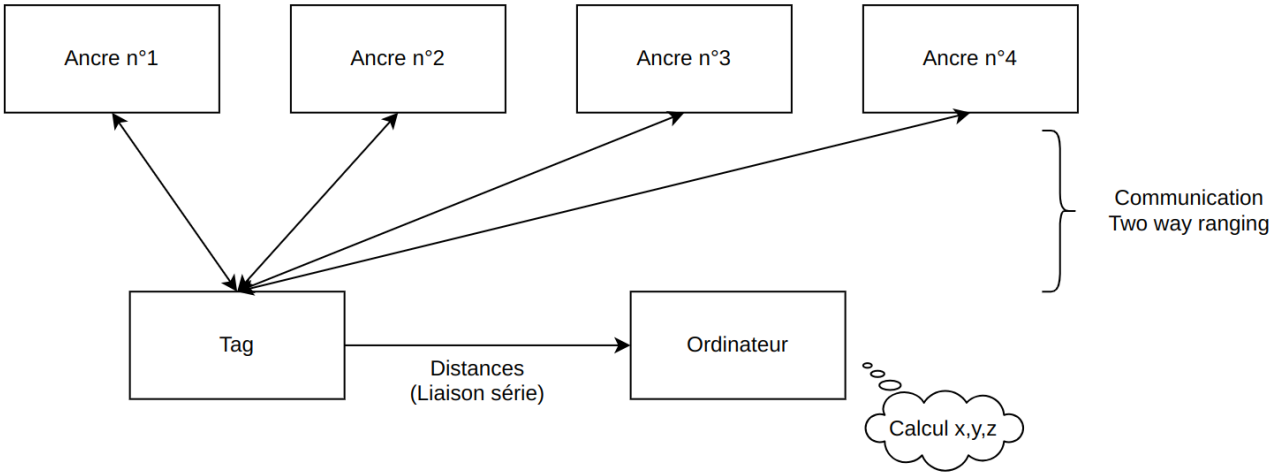

FIGURE 15 – Schéma des communications entre capteurs UWB et ordinateur.

4.4.2 Protocole d'utilisation du système et moyen d'évaluation

Afin de tester et évaluer le système, nous utilisons le système de positionnement par caméra Optitrack de la plateforme robotique Pretil. Ce système permet de localiser un objet muni de billes infrarouges avec une précision centimétrique à une fréquence d'environ 100 Hz, à condition que l'objet se trouve dans le champ de vision de l'ensemble des caméras fixées au plafond, voir figure16. De par ce niveau de précision, nous prendrons les mesures réalisées par l'Optitrack comme référence.

FIGURE 16 – Photo de la salle Optitrack

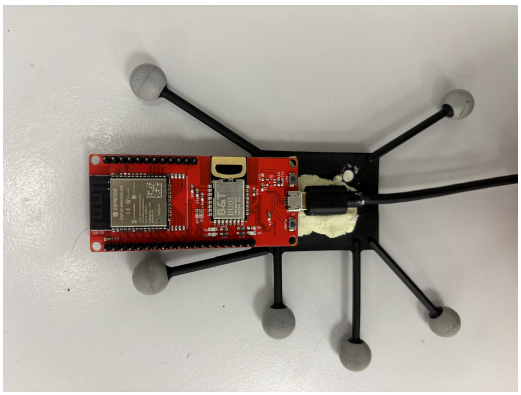

FIGURE 17 – Capteur UWB fixé à une attache de billes réfléchissantes

Le capteur tag est attaché à un objet sur lequel sont fixées plusieurs billes infrarouges (voir figure 17), puis mis en mouvement dans la salle afin de comparer la position mesurée par l'Optitrack et la position calculée via notre méthode. Afin de maximiser l'efficacité de la méthode de multilatération, il est nécessaire de connaître avec précision les coordonnées de chaque ancre, cela est fait via l'Optitrack lors de l'installation. De plus, comme nous avons pu le constater en simulation, varier les positions des ancres optimise la minimisation de la fonction d'erreurs car cela facilite l'apparition d'un unique minimum global correspondant à la solution recherchée. Pour ce faire, en plus de disposer de façon uniforme les ancres dans la salle, nous les disposons à des hauteurs différentes, la constellation formée est un tétraèdre. Les coordonnées issues de l'Optitrack et de la multilatération sont ensuite enregistrées dans un ROS bag pour être traitées et analysées par la suite. Le tag étant mis en mouvement lors d'un déplacement à pied, nous pouvons considérer une vitesse de l'ordre du mètre par seconde, ce qui est comparable au paramètre fourni en simulation (voir figure 9).

L'utilisation de ROS2 dans cette méthode de localisation est assez simple (illustrée en figure 18), elle consiste en un unique noeud réceptionnant les données, calculant la position estimée puis publie sur un topic '/rtls' un "custom message" (format de message adapté selon notre utilisation", contenant les 4 distances capteurs, ainsi que les coordonnées de la position estimée). Le nœud utilise donc 2 callbacks, un premier avec un timer de rappel très court pour mettre à jour les données capteurs et les filtrer, et un autre callback avec un timer plus long pour calculer la position estimée.

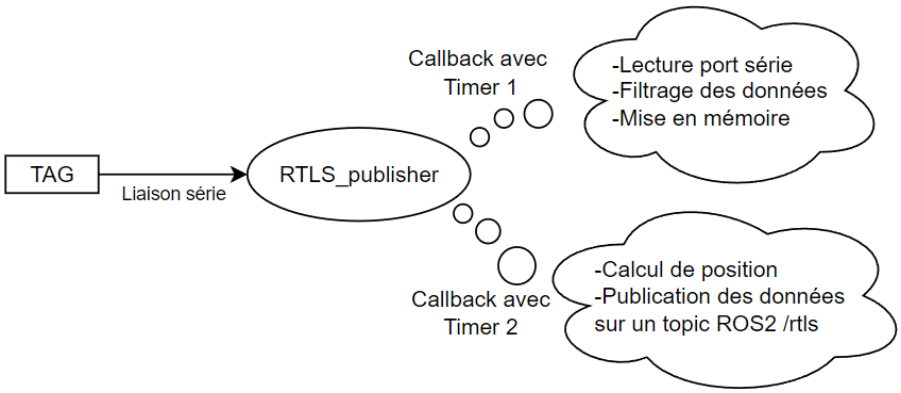

FIGURE 18 – Schéma de fonctionnement du noeud ROS2.

4.4.3 Analyse des résultats et remarques

L'optitrack pouvant détecter des objets dans un zone d'environ 3x3 mètres, nous nous sommes restreints à cet espace, tout en sachant que la solution UWB fonctionne au-delà de ces dimensions. Une trajectoire circulaire a été réalisée (en variant la hauteur du tag), à partir de laquelle nous obtenons 2 courbes : la trajectoire mesurée à partir de l'optitrack, la trajectoire calculée avec l'UWB et lissée avec une fenêtre de moyennage sur 20 points. Comme vu précédemment, le noeud ROS2 calcule une nouvelle position avec une période réglée selon un timer. Le "temps de retard" évoqué y est donc forcément lié. En supposant que le nuage de points d'estimations soit uniformément étalé et la mesure donc centrée par rapport à celui-ci, le temps de retard serait donc égal à la moitié du nombre de points multiplié par la période du timer.

$$T_{\text{retard}} = \frac{1}{2} \times N \times T_{\text{callback}} \tag{12}$$

Le noeud ROS2 calculant une nouvelle position toutes les 0.1 seconde, le retard observé devrait être d'environ 0.5 * 20 * 0.1 = 1 seconde d'après l'équation (12), décalage temporel que l'on retrouve bien en pratique. Ce retard est corrigé manuellement lors de l'analyse des courbes en figure 19. Ajoutons à cela que pour une telle configuration, la correction du retard démontrée en partie 4.2.3, n'est pas réellement utile. En effet, pour une vitesse de l'ordre du mètre par seconde, nous obtiendrons un retard spatial d'un mètre, ce qui est tout à fait négligeable. Négligeable car nos robots seront munis de capteurs extra-sensoriels, comme des sonars ou des caméras RGB, afin de gérer les déplacements et le positionnement à de telles échelles.

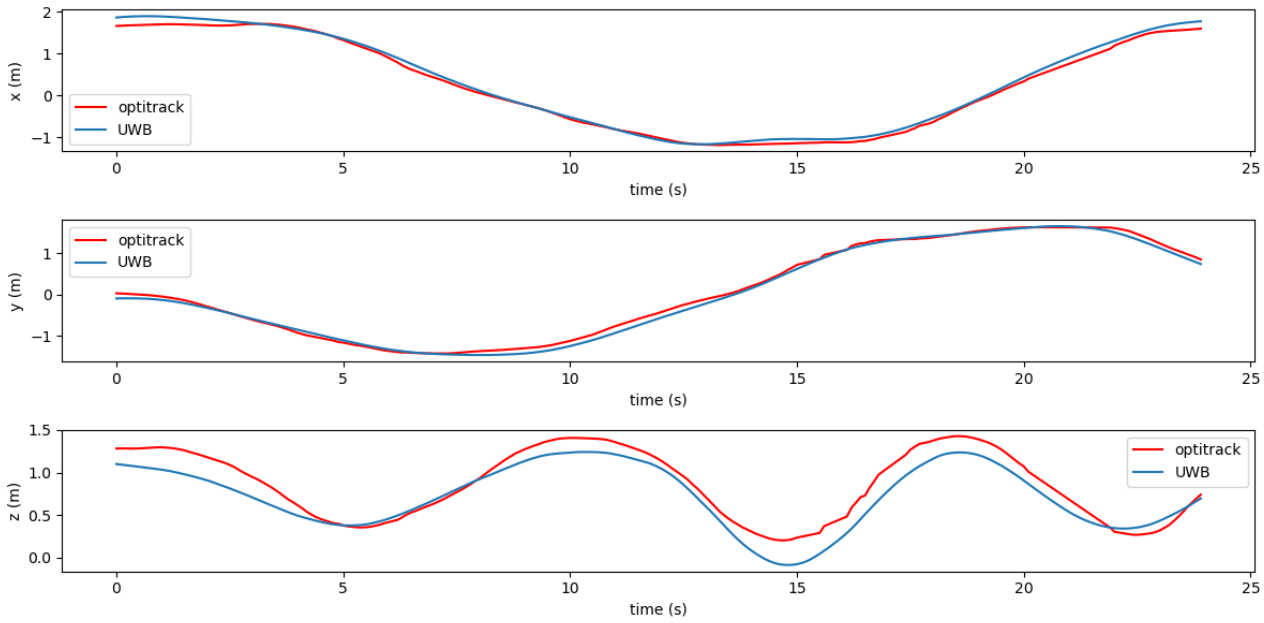

FIGURE 19 – Courbes des coordonnées au cours du temps issues de l'optitrack et de la multilatération UWB.

Comme nous pouvons le constater en figure 19, les résultats sont très encourageants. Afin de mesurer la précision de notre solution UWB, par rapport au système optitrack, nous calculons à chaque instant t, la distance entre les deux positions obtenues. Cette distance vaut en moyenne $\mu = 0.2m$ pour un écart-type $\sigma = 0.08m$. Remarquons d'ailleurs que dans ce cas présent, la majeure partie de l'erreur globale est issue de l'erreur selon l'axe z. Avec la constellation utilisée lors de cet essai, l'éloignement se faisant en majorité selon z, ce qui explique cette erreur accrue. L'erreur globale trouvée ici est un résultat que l'on retrouve dans la littérature [2].Il est aussi possible de voir la trajectoire dans le plan x et y, en figure 20 :

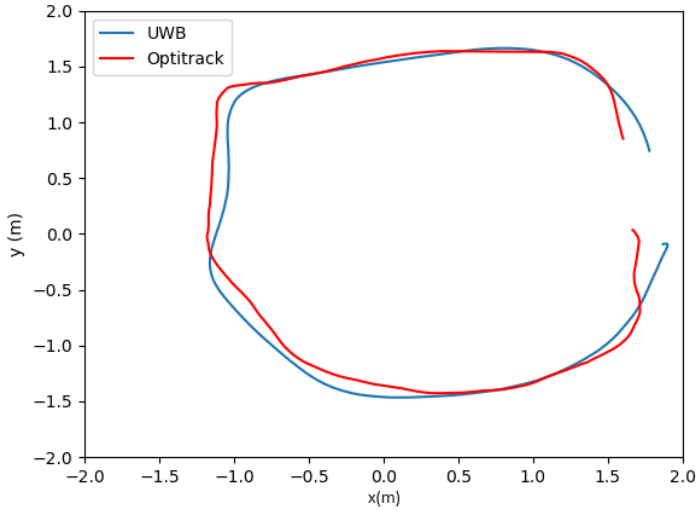

FIGURE 20 – Trajectoire parcourue dans le plan x et y.

Bien que les résultats soient encourageants en termes de précision, quelques points importants sont à relever. D'abord, bien que le lissage par moyenne glissante permet de supprimer les basses fréquences en termes de position, cela ne permet cependant pas forcément de produire une position parfaitement exacte. En d'autres termes, arrêter le robot pour un long moment et moyenner sa position sur plus de 100 points change peu de chose. La position estimée au final peut ne pas converger vers la position exacte. Voici un autre exemple de trajectoire, sur laquelle nous pouvons observer un écart de position plus important.

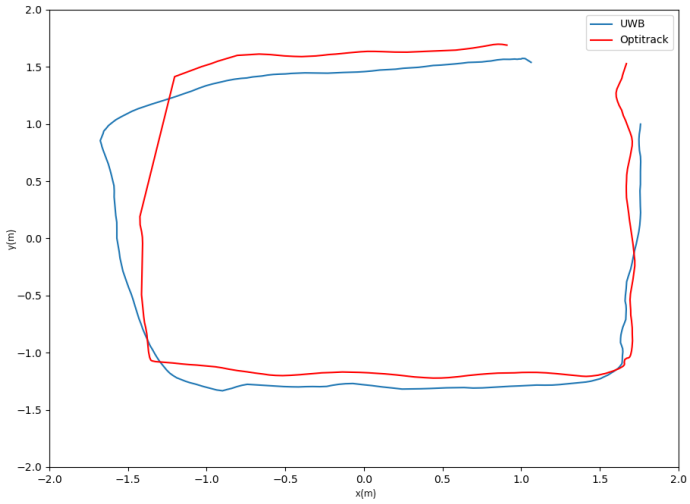

FIGURE 21 – Exemple de trajectoire avec des écarts de position plus importants.

La question est alors : comment expliquer cette différence de précision d'une expérience à l'autre ? Sachant que le même algorithme de localisation a été appliqué, cela ne peut venir du code. Il est donc sûr que le problème vient du matériel ou de son installation. En effet, entre les essais des figures 20 et 21, la position des ancres a été modifiée. Bien que les ancres lors de la deuxième expérience aient été placées afin de garantir une convergence cohérente de l'algorithme, c'est bien leur positionnement par rapport au tag qui pose problème. Mais pour comprendre cela, voyons à quel point les capteurs sont précis. Pour ce faire, comparons à un instant t, la distance mesurée par rapport à la distance recalculée via les positions observées sur l'Optitrack.

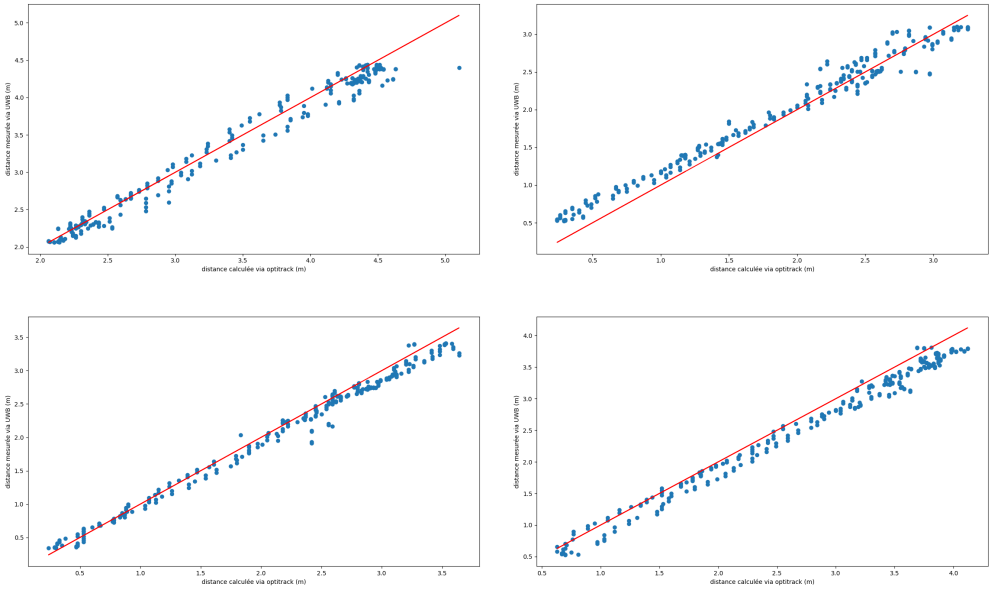

FIGURE 22 – Comparaisons de distances mesurées via UWB par rapport aux distance réelles.

Les points en bleus observés en figure 22 représentent un couple distance optitrack/distance UWB. La droite rouge, quant à elle, représenterait le capteur idéal, associant pour une distance réelle, une mesure exacte de celle-ci. Nous souhaitons alors au maximum que le nuage de points se rapproche de cette droite. En pratique, nous observons plutôt un nuage de points statistiquement assez proche de cette droite. Cependant, nous observons la raison précise pour laquelle une convergence de l'algorithme de localisation ne converge pas forcément vers la position exacte. D'abord, remarquons qu'il n'est pas possible de caractériser notre capteur par une moyenne ainsi qu'un écart-type, comme nous avons pu le faire en simulation. En effet, nous voyons bien que la distribution des points varie d'une distance réelle à une autre. Toutefois, l'étalement n'est pas la cause de ce problème, car un moyennage des données capteurs produirait une distance mesurée significativement assez proche de la réalité. Le problème est qu'un biais non nul est présent en fonction de la distance. Ce biais n'est pas constant ou quantifiable car celui-ci dépend de plusieurs facteurs comme l'orientation des antennes ou les objets environnants. C'est pourquoi, en modifiant une constellation, même en respectant un critère de non-symétrie, la précision obtenue peut changer. Des collègues de Nancy travaillant sur le projet, ayant déjà utilisé la localisation UWB témoignent avoir atteint une précision au centimètre à l'aide de 8 ancres et en évitant toute perturbation électromagnétique. De manière générale, il faudra faire en sorte d'avoir une calibration réduisant au plus les biais.

4.5 Précision lors d'un déplacement en groupe

En utilisant la méthode de localisation UWB, nous pouvons déplacer une flotte de robots à condition qu'une partie d'entre-eux reste immobile afin de jouer le rôle de portable landmark. En inversant les rôles, il est alors possible de déplacer, en plusieurs étapes, la flotte. Le tout étant de savoir si au fil du temps, la localisation continue de produire des estimations assez précises pour le bien de la mission. Dans la publication [5], Kurazume et al mettent en place une technique de positionnement coopératif. L'estimation des positions est uniquement basée sur des mesures d'angles entre les robots. Avec 3 robots et en connaissant la position de 2 d'entre-eux, il est possible d'en déduire celle du troisième, avec des calculs trigonométriques. L'existence d'expressions analytiques pour calculer la position offre donc des possibilités, comme l'utilisation de développement limité, utilisé pour calculer la variance et la position des erreurs. Ainsi en prenant pour hypothèses que les erreurs de mesures d'angles sont modélisées par des lois normales centrées, on peut montrer que l'erreur de positionnement calculée suit aussi une loi normale centrée. Les auteurs montrent alors qu'en déplaçant petit à petit les robots, leurs ellipses de position grandissent à cause de l'accumulation des erreurs. Enfin, une méthode de déplacement est trouvée avec les bons paramètres pour minimiser la variance de l'erreur de positionnement pour aller d'un point A à un point B.

Sachant que la multilatération se base sur les positions connues des landmarks, et que ces positions sont estimées à partir de la première inversion de rôle, il se peut que la flotte soit incapable de fournir de bonnes estimations sur le long terme. En revanche, dans notre cas, le calcul des positions étant basé sur la résolution d'un système non linéaire, nous ne pouvons calculer la variance de l'erreur de positionnement. Toutefois, nous avons une preuve de l'accumulation des erreurs lors du déplacement en groupe, ce que l'on peut vérifier en simulation. Simulons une flotte de 10 robots sur le plan 2d. La moitié d'entre-eux seront mobiles et l'autre moitié immobiles. Nous inverserons les rôles à des périodes de temps égales tout en laissant un temps de transition. Les robots se déplacent à une vitesse d'un mètre par seconde. Leurs capteurs produisent des mesures avec une précision de plus ou moins 20 centimètres. Ainsi, nous calculons au cours du temps l'erreur de position, afin de vérifier la stabilité de notre méthode de localisation en groupe. Dans un premier temps, nous disposons la flotte selon deux rangées parallèles, avançant en ligne droite pour réaliser une fauchée de l'environnement.

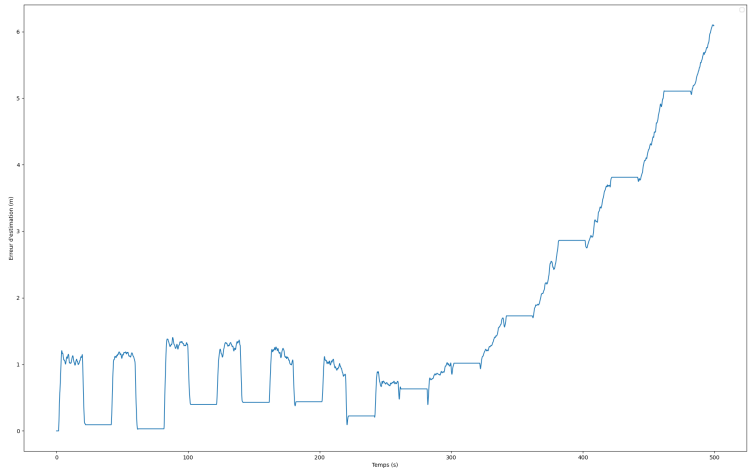

FIGURE 23 – Erreur d'estimation d'un des robots au cours du déplacement en groupe.

Plusieurs choses sont à remarquer en figure 23. De par la technique de déplacement adoptée et la technique de localisation, nous observons deux phases, s'enchainant de façon cyclique au cours du temps. En effet, un landmarks se trouvant immobile verra évidemment son erreur d'estimation fixe. Ensuite lors de l'inversion de rôle, celui se met en mouvement et voit donc son erreur d'estimation croître, avec deux courtes phases de transitoire en raison du lissage de position. Pour simplifier la chose, ce qui nous intéresse ici, c'est l'erreur atteinte lors du retour à l'état landmark, car bien-sûr, une mauvaise estimation de la position du landmark produira de mauvaises estimations pour les robots mobiles. Nous devons donc porter une attention particulière aux valeurs des plateaux atteintes au cours du temps. Comme nous pouvions nous y attendre, l'estimation des positions après plusieurs inversions de rôle, finit par se dégrader de par l'accumulation des erreurs. Néanmoins, nous disposons d'une marge de manœuvre afin de couvrir une certaine zone avant que la localisation ne se dégrade trop. Ce constat nous amène donc à la deuxième partie des travaux effectués. Nous devrons alors trouver un moyen de réduire la divergence de la localisation. En plus, nous devrons trouver une méthode de navigation maximisant la surface d'exploration suivant les contraintes évoquées jusqu'à maintenant.

5 Navigation en groupe

5.1 Hypothèses et contraintes de navigation

Notre but étant pour notre flotte de robots d'explorer une forêt à la recherche d'incendies, celle-ci devra naviguer de sorte à maximiser sa surface couverte lors de ses déplacements. Nous devons d'abord définir une façon de couvrir une zone. Par exemple, nous pouvons supposer que nos robots sont munis d'une caméra thermique couvrant une certaine zone. Bien-sûr, il est difficile de poser un cadre général, car beaucoup de paramètres inconnus entrent en jeu. Un drone par exemple, peut pointer sa caméra vers le sol lors du vol et couvrir une zone circulaire, alors qu'un robot terrestre avec une caméra pointant vers l'horizon couvrira une zone en cône. Nous pouvons supposer que notre flotte dispose d'un protocole de communication, leur permettant de diffuser des informations dans un rayon supérieur à la distance maximum des capteurs UWB (sinon, il n'y aurait aucun intérêt pour un robot de naviguer au-delà de sa zone de communication).

Ensuite, il est évident que de par notre façon de nous localiser, nos robots ne peuvent naviguer n'importe comment. Aller n'importe où risque de nuire au groupe, car une mauvaise position peut dégrader l'efficacité de la localisation par multilatération. D'abord, si nos robots mobiles vont trop loin du groupe de landmarks, leur position sera d'une part moins bien estimée, puis dégradera l'estimation des autres, une fois converti en landmark. Il faut donc caractériser la précision de l'estimation en fonction de la distance au groupe de landmarks, et du groupe de landmarks lui-même. En plus, pour garantir une multilatération robuste, nous devons nous assurer que les landmarks ne forment pas un support de symétrie. C'est d'ailleurs le premier point que nous aborderons.

5.2 Garantie d'une constellation valide

Comme nous l'avons mentionné en partie 4.2, une constellation de landmarks doit à tout prix s'éloigner de former un support de symétrie. En 3 dimensions, nos landmarks doivent donc éviter de former un plan, ou pire, une droite. La formation d'une droite ayant très peu de chances d'arriver, nous abordons d'abord le cas d'un plan. En premier, nous devons caractériser à quel point une constellation forme un plan. Considérons alors N landmarks ainsi que X la matrice de taille $N \times 3$, contenant les positions des N robots :

$$X = \begin{bmatrix} P_1 & P_2 & \cdots & P_N \end{bmatrix}^T$$

Ensuite, nous considérons $\bar{X}$, le centroïde de X, que nous soustrayons à X pour centrer le nuage de points :

$$\tilde{X} = X - \bar{X}$$

Pour trouver le plan s'approchant au mieux de ces points, nous pouvons utiliser la décomposition en valeur singulière [1] :

$$\tilde{X} = USV^T \tag{13}$$

Dans l'équation 13, U (de taille $N \times N$) et V (de taille 3×3) sont les matrices contenant les vecteur propres orthonormaux de $\tilde{X}\tilde{X}^T$ et $\tilde{X}^T\tilde{X}$ respectivement. S est une matrice contenant dans sa diagonale les valeurs propres de $\tilde{X}\tilde{X}^T$ ou $\tilde{X}^T\tilde{X}$ (les deux matrices ont les mêmes valeurs propres), aussi appelées valeurs singulières. En arrangeant par ordre décroissant les valeurs et vecteurs propres, nous savons alors que la dernière ligne de V traduit la direction minimale du nuage de points, soit un vecteur normal au plan recherché.

Définissons alors le plan $\mathcal{P}$:

$$\mathcal{P} : ax + by + cz + d = 0$$

Nous trouvons alors les paramètres du plan de la façon suivante :

$$(a, b, c) = V_3$$
$$d = -V_3 \cdot \bar{X}$$

Maintenant que nous avons le plan le plus proche de notre nuage de points, nous devons définir une métrique de ressemblance à ce plan, une distance. Nous pouvons par exemple prendre la somme des distances entre nos landmarks et leur projeté orthogonal au plan (voir figure 24) :

$$E = \sum_{i=1}^{N} d(P_i, \mathcal{P})$$

Avec $d(P_i, P)$ la distance entre le point P_i et le plan $\mathcal{P}$:

$$d(P_i, P) = \frac{|ax_i + by_i + cz_i + d|}{\sqrt{a^2 + b^2 + c^2}}$$

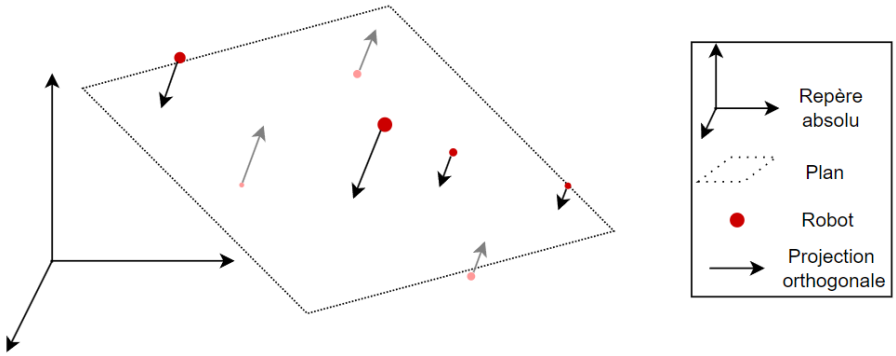

FIGURE 24 – Schéma du plan le plus proche trouvé et des projections orthogonales

De ce fait, nous pouvons définir un seuil de ressemblance η de sorte que si l'on a $E > \eta$, nous ne pouvons inverser les rôles. Dans un monde parfait, le plus optimal serait alors de déplacer le robot le plus proche du plan, selon le vecteur orthogonal au plan afin de l'éloigner. Cependant, cela n'est pas toujours possible. D'abord, si le robot le plus proche est un robot terrestre et qu'il doit se déplacer en hauteur, il s'agit d'une solution bien délicate. Nous en arrivons donc à un sujet très important dans le domaine des systèmes multi-agents : celui du choix. Comment choisir quel robot bouger pour rendre la constellation viable ? Une méthode assez courante peut s'apparenter à une courtepaille : chaque robot tire un nombre aléatoire et le communique aux autres, celui qui a le plus petit est choisi pour exécuter l'action à faire. Cette méthode n'est pas viable car tous les robots ne pourront pas s'éloigner aussi facilement du plan. Si un drone est présent, il sera plus évident pour lui de s'éloigner du plan, à condition qu'il ne croise pas d'obstacle. Chaque robot devra donc évaluer à l'aide de ses capteurs extrasensoriels sa capacité à s'éloigner du plan. Pour ce faire, nous pourrons créer une fonction d'utilité. Par exemple, un robot terrestre devant prendre de la hauteur pour s'éloigner du plan, a vu une petite colline à une vingtaine de mètres plus loin du groupe : cela vaut-il le coup de s'éloigner, alors qu'un drone peut juste s'élever de quelques mètres pour résoudre le problème ? Plusieurs paramètres seraient donc à prendre en compte, en voici quelques exemples : les paramètres du plan, les degrés de liberté du robot, le temps estimé pour se déplacer, la position des autres robots. Une fois les robots déplacés, le calcul du plan doit être refait ainsi que sa métrique de ressemblance. Une fois que $E < \eta$, la constellation est valide et les rôles peuvent être inversés.

L'idée reste la même pour vérifier la ressemblance à une droite. La droite la plus proche du nuage de points a pour vecteur support la première ligne de V, qui donne la direction principale du nuage. On peut ensuite trouver l'équation de droite associée et construire une métrique grâce aux projections orthogonales des landmarks sur la droite. Une autre façon de s'y prendre est d'évaluer les rapports entre les valeurs singulières pour évaluer l'étalement du nuage.

Enfin, si les landmarks sont trop proches les uns des autres, le nuage s'approche d'un support de symétrie sphérique, ce qui rend tous les points de l'espace plus ou moins solutions. Il va donc sans dire que nous devons faire en sorte que nos landmarks soient assez distants des uns et des autres. Toutefois, au fur et à mesure que l'on s'éloigne du nuage, celui-ci se contracte petit à petit de notre point de vue. N'importe quelle constellation devient donc inutilisable. Nous devons donc prendre en compte un élément supplémentaire : jusqu' où pouvons-nous naviguer en fonction de notre constellation ?

5.3 Distance limite par rapport aux landmarks

En théorie, il n'existe pas vraiment de limite à ne pas dépasser, tout dépend des conséquences que l'on est prêt à accepter en termes de précision. Nous devrons faire un choix en fonction de la précision obtenable et de la limite de détection des capteurs. Encore une fois, l'erreur est difficilement calculable directement, nous devrons donc expérimentalement l'évaluer en simulation. Pour cela, prenons N landmarks avec une précision de plus ou moins 20 cm. L'étalement du nuage de points sera défini par une matrice de covariance, nous permettant ainsi de gérer les directions de son étalement. Ensuite, nous calculerons pour chaque point de l'espace l'erreur associée à son estimation (sans moyennage, temps de calcul trop long sinon). Pour simplifier les choses, la matrice de covariance est diagonale pour privilégier un étalement selon x et/ou y. Prenons d'abord pour cas où la constellation est également répartie selon x et y :

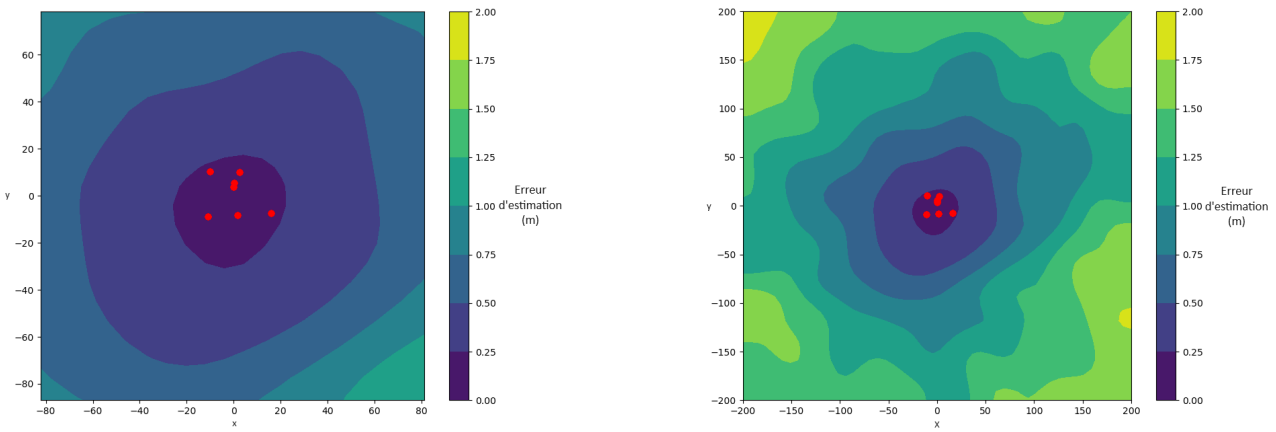

FIGURE 25 – Carte des erreurs de localisation

Comme nous pouvons l'observer en figure 25, plus on s'éloigne du centroïde de la constellation, plus l'estimation produite s'éloigne de la vérité. De plus, augmenter le nombre de landmarks à la distribution ne change rien en termes de précision : les niveaux de précision atteints sont les mêmes, mais là, les courbes iso sont plus circulaires, du moment que l'on reste proche du centroïde. Entre-autres, beaucoup d'efforts pour peu d'amélioration. Pour revenir à la partie précédente, lorsque notre constellation s'approche d'un support de symétrie, il se peut qu'elle crée des solutions non souhaitées symétriques. Mais quand bien-même cela n'arrive pas, les erreurs d'estimations croissent très vite, même en restant plus ou moins proche du centroïde de la constellation, ce que l'on observe en figure 26. Dans ce cas, la variance selon x est bien supérieure à la variance selon y. Sachant que nos robots vont alterner entre les rôles mobiles explorateurs et immobiles landmarks, nous avons donc deux solutions : vaut-il mieux parcourir la plus grande distance possible, quitte devenir un landmark de position relativement mal connue et faire peu d'inversion ou alors parcourir de petites distances pour devenir un landmarks de position bien connue et faire beaucoup d'inversions ? De plus, il se peut que les capteurs de distance utilisés imposent une distance limite et même une précision en fonction de la distance (peu de fabricants donnent une telle information). Cela impacte aussi le choix à prendre sur la distance limite d'éloignement par rapport à la constellation. Le choix du capteur sur la distance limite ainsi que la précision est donc primordial. En effet, si les capteurs fonctionnent à une distance d'un kilomètre mais avec une précision d'une dizaine de mètres, la localisation d'un objet à 800m de distance sera catastrophique.

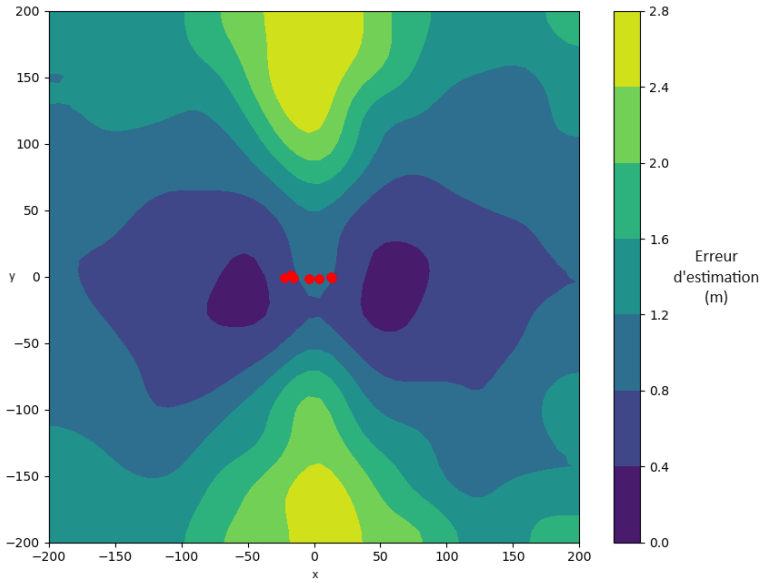

FIGURE 26 – Carte des erreurs de localisation dans un cas où la constellation se rapproche d'une droite

5.4 Accumulation d'erreurs

Le but de cette partie est d'évaluer la meilleure solution entre s'éloigner significativement des landmarks à chaque phase mobile et faire moins d'inversion de rôles ou alors ne parcourir que de petites distances et intervertir un grand nombre de fois les rôles. La réponse à cette question est difficile à évaluer : vaut-il mieux sommer beaucoup de fois un terme faiblement croissant ou sommer peu de fois un terme fortement croissant. En nous basant sur les précédentes parties, nous considérons une flotte de robots toujours divisée en deux groupes, à la différence près que nous nous assurons de leur bonne répartition. "Isotrope" semble être le bon terme, car nous souhaitons que la constellation soit faite pour produire une estimation aussi précise dans toutes les directions. Il s'agit de la même simulation qu'en partie 4.5. Les constellations seront donc systématiquement un cercle pour assurer leur validité, un cercle de 40 mètres de diamètres. Les deux groupes sont donc déplacés durant leur phase mobile dans la même direction et vitesse pour garder la constellation lors du retour à l'état landmark. Les deux cercles se "chevauchant" à tour de rôle, les distances maximales mesurées seront donc faites entre le robot mobile de front du cercle et le robot landmark à l'arrière du cercle, ce qui est illustré en figure 27. Nous choisissons d'inverser les rôles lorsque cette distance dépasse un certain seuil. Les robots sont mis en mouvement à une vitesse de 1 mètre par seconde pendant deux heures. Nous nous intéressons alors à la distance parcourue ainsi qu'à l'erreur d'estimation produite au cours du temps. Par construction, à la fin de chaque inversion, la flotte voit son front progresser d'une fois la distance imposée.

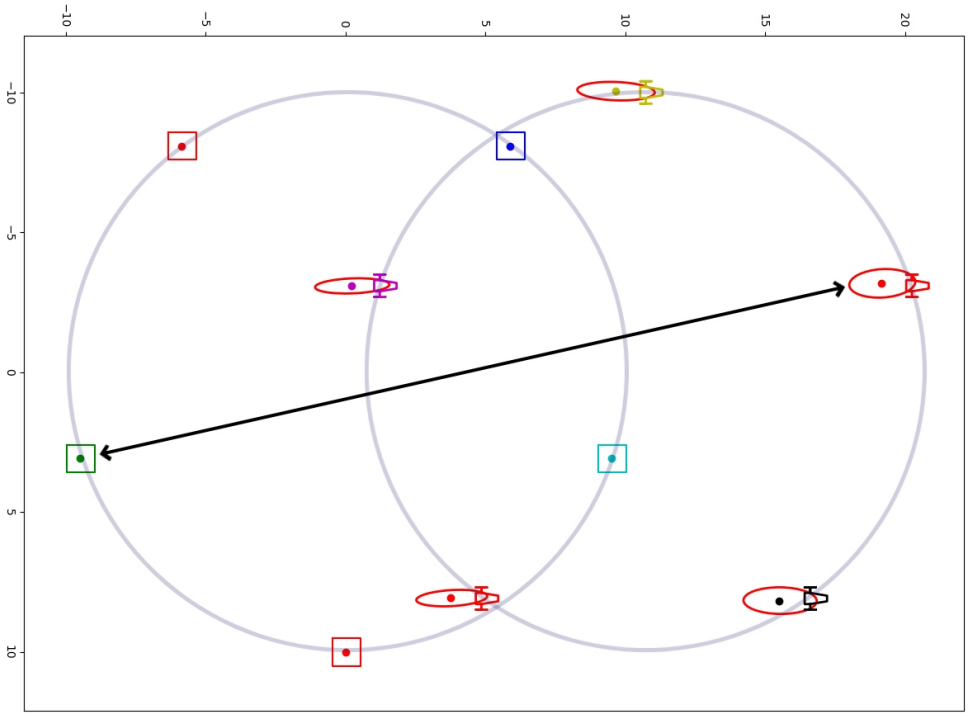

FIGURE 27 – Formation adoptée par les robots lors du déplacement en groupe

D'abord, regardons ce que donne un déplacement en groupe avec une distance limitée à 50 mètres. Les constellations ayant un diamètre de 40 mètres, à chaque inversion le front de la flotte avancera donc de 10 mètres. Nous sommes donc dans un contexte de petits déplacements avec beaucoup d'inversions.

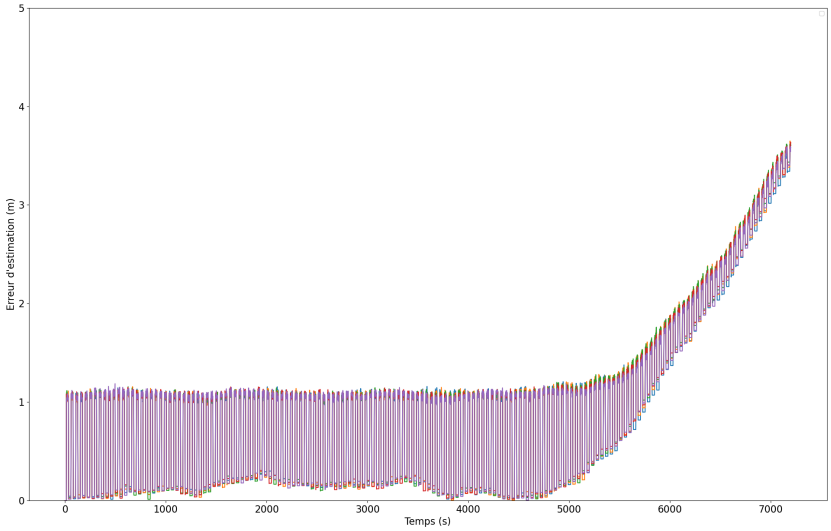

FIGURE 28 – Courbes d'erreurs au cours du temps lors d'une navigation par petits déplacements

La figure 28 montre le même type de courbe que pour la figure 23 : Une succession de pics d'erreurs issus du retard de localisation et de plateau lors du la phase immobile. Nous nous intéressons donc aux plateaux inférieurs des courbes (l'erreur de tous les robots d'un même groupe sont représentés). L'erreur finit bien par diverger. Mais le moment à partir duquel la localisation globale finit par diverger est très incertain. Dans ce cas-ci, à 5000 secondes, mais cela aurait très bien pu arriver à 2000 secondes. En simulation, les résultats sont très variables ce qui rend cette approche donc trop imprévisible. De plus, nous comptons environ 161 inversions en 2 heures, soit une distance totale parcourue de 1610 mètres. Voyons à présent la même navigation en groupe avec une distance maximale de 200 mètres (limite souvent trouvée dans les meilleurs capteurs UWB).

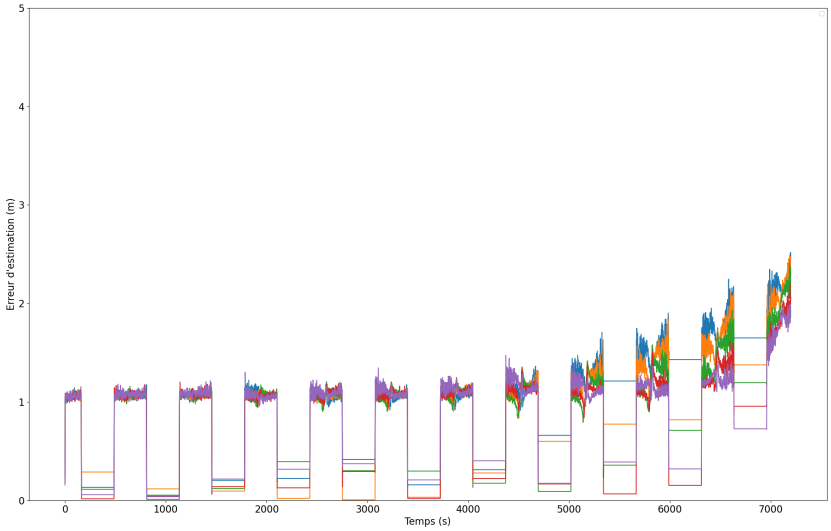

FIGURE 29 – Courbes d'erreurs au cours du temps lors d'une navigation par grands déplacements

Comme nous pouvons le voir en figure 29, bien plus lisible d'ailleurs, une divergence est aussi présente mais plus lente. L'accumulation des erreurs est cependant beaucoup plus variable d'un robot à un autre. En revanche, l'aspect le plus frappant par rapport à la simulation précédente est sa reproductibilité. L'estimation varie toujours de la même manière, ce qui rend la qualité de l'estimation beaucoup plus prédictible. De plus, nous comptons onze inversions sur 2 heures de temps soit une distance parcourue totale de 2.2 km, avec une erreur de localisation finale d'un mètre en moyenne, contre une distance de 1.6km pour une erreur finale de 3.5 mètres. Il va donc sans dire que la meilleure option est donc de parcourir la plus grande distance possible permise par les capteurs UWB, avant d'inverser les rôles.

5.5 Surface d'exploration et contraintes trop importantes

La formation de groupe évoquée précédemment est capable de naviguer sur un trajet d'environ deux kilomètres. Cependant, la façon dont les groupes naviguent réduit la surface couverte par le groupe. En effet, les deux groupes se chevauchant à tour de rôle, la moitié de la distance parcourue par chacun des groupes ne sert à rien. Or, d'après ce qui précède, les deux groupes peuvent s'éloigner de plusieurs dizaines de mètres. On peut donc imaginer une navigation au cours de laquelle les deux groupes coulissent l'un par rapport à l'autre. Nous aurions alors deux groupes d'une largeur totale de 80 mètres naviguant sur une distance de 2 km soit une surface couverte de 0.16 km^2. Ce résultat est assez décevant, mais peu étonnant en même temps. Les contraintes de positionnement des landmarks sur le terrain, ainsi que les contraintes de distances sur les capteurs posent problème. D'une part, nous devons étaler au mieux nos landmarks pour produire une localisation satisfaisante, mais en même temps, nous ne pouvons pas trop nous éloigner d'un coup si nous souhaitons utiliser l'ensemble des landmarks. Il existe plusieurs alternatives pour réduire l'impact de ces problèmes, par exemple en formant de petits sous- groupes avec 4 landmarks et un mobile, pour explorer des zones différentes. Cependant, dans ce cas, nous ralentissons la vitesse de déplacement globale du groupe. En conséquence, nous parvenons au fait que la multilatération avec des portables landmarks impose trop de contraintes dans un contexte d'exploration, surtout si le temps est compté. Sur le principe, cette méthode est utilisable, mais trop inefficace utilisée seule.

6 Propositions et améliorations possibles

Bien que la multilatération comme démontré dans ces travaux soit trop peu efficace dans notre contexte, tout n'est pas à jeter. En effet, le principe de portable landmark réduit drastiquement la capacité d'exploration de notre flotte, car elle contraint un trop grand nombre de robots à rester immobiles. En effet, il est impossible que la multilatération se base sur des landmarks mobiles : en plus de fournir une mesure de distance, la position absolue du landmarks à l'instant exact de la mesure devrait être communiquée, ce qui rend le problème récursif. Le landmark aurait aussi besoin de la position de son vis à vis, et se renverrait la question l'un à l'autre indéfiniment. Ce problème se résoudrait si nos robots étaient munis d'une méthode de localisation alternative comme une centrale inertielle, mais la méthode commence à ressembler très fortement à de la fusion de données et en particulier sur l'utilisation d'un filtre de Kalman. Chaque robot prédirait sa position individuellement par double intégration de la centrale et corrigerait la position obtenue à l'aide des positions des autres robots ainsi que des mesures de distances. Le bruit de mesures évoluerait donc en fonction des erreurs de mesures des capteurs mais aussi des erreurs de localisation des autres robots. Une fusion de données avec un filtre de Kalman semble offrir plusieurs possibilités. D'abord, il serait possible de faire en sorte que l'ensemble des robots restent mobiles. La correction pourrait être dynamique dans le sens où un robot en particulier se corrigerait en fonction du nombre de robots à proximité, ce qui lui laisserait la possibilité de naviguer en dead reckoning [3] le temps que ses collègues le rejoignent. Remarquons que comme la correction utilisera les repères de la même manière que pour la multilatération, le placement des landmarks reste primordial et sera contraint par les mêmes conditions évoquées. Nous sommes alors particulièrement dans un problème de comportements collectifs, le problème étant alors le suivant : une flotte de N robots explore un environnement inconnu, de sorte que chaque robot doit au maximum se trouver à proximité de 3 ou 4 robots repères, et que ces repères ne forment pas une droite ou un plan.

D'autres idées sont envisageables en s'inspirant de comportements collectifs comme l'exploration par stigmergie. L'exploration par stigmergie est retrouvée chez les colonies de fourmis formant des routes en déposant leurs phéromones, indiquant ensuite aux ouvrières quelles routes suivre. Dans notre cas, nous pourrions imaginer de petites balises contenant un capteur UWB, et une petite batterie, faisant office d'un dépôt de phéromone. Un drone navigant en hauteur et disposant de sa position

3. En français : navigation à l'estime

(à l'aide d'un GPS par exemple) déciderait de lâcher la balise au sol, dont la position serait alors connue. En couvrant la zone à explorer de ces balises, nos robots pourraient naviguer en se basant sur la multilatération. Cette idée se rapproche du principe de dust computing. La plateforme PRETIL travaillant sur des robots mous avec des bras en trompe, nous pourrions imaginer nos robots capables de ramasser les balises lors du retour à la base.

7 Filtre de Kalman collectif

Comme vu en partie 4.3, il est tout à fait possible pour nos robots de se localiser à l'aide de capteurs proprioceptifs puis de corriger cette prédiction via les distances relatives entre les autres robots. Le plus gros avantage d'une telle méthode serait que l'ensemble de la flotte peut se permettre de rester mobile. La qualité de la correction dépendrait alors comme vu dans les parties précédentes, de la précision des capteurs, du placement des autres robots au moment de la mesure, mais aussi de la distance du robot par rapport au centroïde du reste des robots. L'adaptabilité en termes de dimension du filtre de Kalman autoriserait en plus à un robot de corriger sa position en fonction du nombre de robots à proximité. De ce fait, plus il y a de robots dans les alentours, mieux on se recale et si il n'y a personne, le robot navigue en dead reckoning.

Contrairement à la théorie exposée en partie 4.3, la fonction d'observation g, ne calcule plus les distances entre l'estimation du robot et la position des autres robots, mais bien uniquement entre des estimations. En effet, l'ensemble des positions de nos robots ne sont plus que des estimations. De ce fait, le caractère de localisation dans un repère absolu est ici ignoré par l'équation d'observation, cela aura des conséquences en simulation. De plus, le bruit β, dans l'équation d'observation sera amené à changer au fil du temps car il comprendra non seulement l'erreur de mesure capteurs, mais aussi l'erreur de positionnement des robots. Nous pouvons simplement prendre un β pour majorer l'addition des erreurs de mesures et de position en supposant que l'erreur des positions estimées ne diverge pas.

En simulation, les résultats sont les mêmes, que la flotte soit en mouvement ou non. La flotte est correctement localisée à un détail près : l'estimation de la flotte dérive par rapport à la flotte réelle, conséquence du caractère relatif de l'équation d'observation de l'EKF. En conséquence, la flotte est correctement estimée à une transformation rigide près (voir figure 30). Une telle transformation étant caractérisée par une rotation et une translation, il suffit de les calculer afin d'appliquer la transformée inverse et recaler la flotte. En deux dimensions, il suffit d'avoir 2 robots de positions estimées fiables, et en 3 dimensions de 3 robots de positions estimées fiables. On pourrait imaginer des drones aériens volant au-dessus des arbres pour avoir un point GPS, ou un robot terrestre se localisant utilisant un SLAM. Un point essentiel auquel il faut porter attention lors de l'utilisation du Filtre de Kalman sur un groupe de robots est la consanguinité des données, en anglais le "data incest" [7]. Quand les mêmes données sont réutilisées plusieurs fois et ces données sont traitées comme indépendantes dans la fusion des données, cela provoque du data incest. Ce qui est le cas ici, car les positions des robots sont calculées en fonction des autres positions comme si ces variables étaient indépendantes. Pour résoudre les problèmes de consanguinité des données, il convient d'utiliser une approche décentralisée comme exposée dans [3]. Notre méthode est désignée dans la littérature scientifique comme une localisation collective centralisée (CCL en anglais). Les CCL sont capables de produire des résultats optimaux mais sont en revanche sensibles aux pertes de connexions ou inefficaces sur des réseaux centralisés trop importants. Des méthodes décentralisées ont aussi été créées permettant des résultats similaires sans perte de précision significative.

Les résultats en simulation sont, sans surprise, bien meilleurs que sur une navigation par multilatération, car l'ensemble de la flotte est mobile avec une estimation au mètre constante. La méthode de localisation étant robuste vis à vis du nombre de robots à proximité, la flotte peut donc se séparer en petits groupes. Un autre avantage de l'utilisation d'un filtre de Kalman collectif est que chaque robot dispose d'une métrique témoignant de la qualité de sa localisation : sa matrice de covariance.

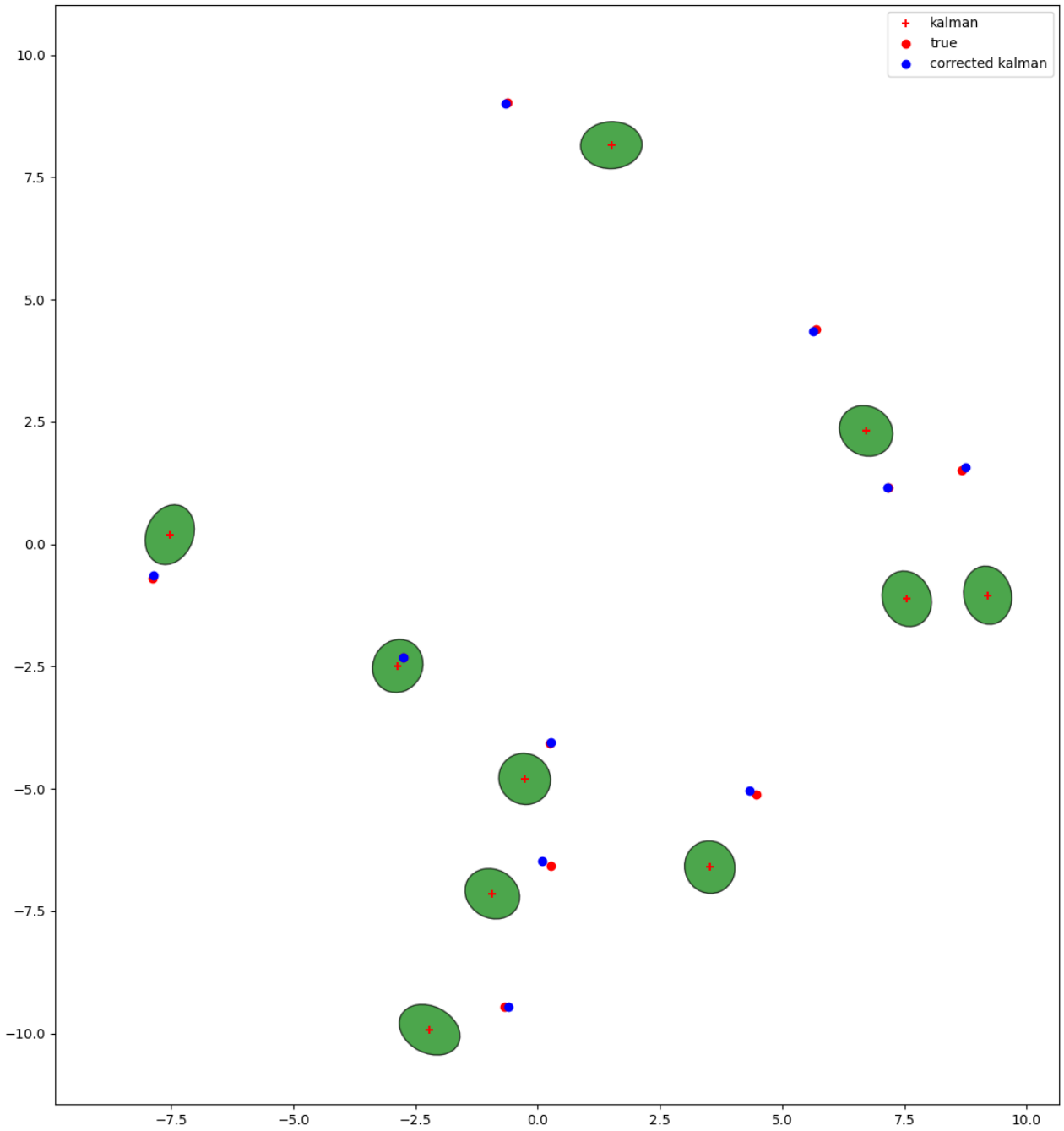

FIGURE 30 – Simulation de la flotte en mouvement avec localisation par filtre de Kalman collectif et recalage

8 Conclusion

Bien que les recherches sur la localisation en groupe via multilatération UWB ne nous aient pas menés à la conclusion espérée, nous avons obtenu plusieurs connaissances précieuses pour la direction à prendre sur la suite du projet. D'abord, la localisation par multilatération ultrawide band est parfaitement utilisable dans un contexte de déplacement en groupe, en extérieur ou en intérieur. Il s'agit d'une solution peu coûteuse et facile à mettre en place, permettant d'atteindre une précision centimétrique. En revanche, il serait impossible pour une flotte de robots de performer dans un contexte d'exploration sous contrainte de temps, en raison des contraintes de positionnement des repères et de l'inactivité d'un trop grand nombre d'agents. Les contributions de ces recherches se retrouvent dans l'optimisation des constellations de landmarks afin de garantir une localisation isotrope. Ceci amènera à une preuve de concept sur le déplacement d'une flotte de robots utilisant la multilatération et le concept de portable landmark : une flotte de 10 robots peut se déplacer sur 2 km en maintenant une localisation précise au mètre, uniquement avec leurs capteurs de distance. En ce qui concerne la suite du projet, il semble préférable que la localisation soit encadrée par de la fusion de données. L'utilisation des capteurs UWB pourra être conservée afin d'assurer par exemple la correction d'un filtre de Kalman. Cela donnera la possibilité à l'ensemble de la flotte de rester mobile à condition de respecter certaines règles établies dans ce rapport. L'aspect collaboratif est alors parfaitement conservé car les robots corrigent l'estimation de leur position en communiquant avec leurs pairs. L'équipe système multi-agents pourra donc se concentrer sur la façon de déplacer les robots de sorte à explorer au mieux la zone tout en assurant une correction collective optimale.
Avoir travaillé dans le monde de la recherche pendant ces six mois a été un réel plaisir pour moi. Si le rôle de l'ingénieur est de résoudre des problèmes complexes en s'inspirant de solutions existantes, j'ai eu la chance, en revanche, de travailler aux côtés de ceux qui créent des solutions. J'ai découvert ce qu'était réellement la recherche et les avantages qu'elle offre : l'échange de connaissances, la liberté et la créativité, le partage et l'entraide.

Références

[1] Steven Brunton and J. Kutz. *Singular Value Decomposition (SVD)*, pages 3–46. 02 2019.

[2] S. Chen, D. Yin, and Y. Niu. A survey of robot swarms' relative localization method. *Sensors*, 22(12) :4424, 2022.

[3] Jacob Hartzer and Srikanth Saripalli. Vehicular teamwork : Collaborative localization of autonomous vehicles. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. IEEE, September 2021.

[4] Rudolf Kalman. Contribution to the theory of optimal control. *Bol. Soc. Mat. Mexicana*, 5, 02 2001.

[5] Ryo Kurazume, Shigemi Nagata, and Shigeo Hirose. Cooperative positioning system with multiple robots. *J Robot Soc Japan*, 13, 06 2001.

[6] Daniel Neuhold, Aymen Fakhreddine, and Christian Bettstetter. Hipr+ : A protocol for centimeter 3d localization based on uwb. In *Proceedings of the 25th International ACM Conference on Modeling Analysis and Simulation of Wireless and Mobile Systems*, MSWiM '22, page 181–188, New York, NY, USA, 2022. Association for Computing Machinery.

[7] Xiaoyu Shan, Adnane Cabani, and Houcine Chafouk. Cooperative vehicle localization in multi-sensor multi-vehicle systems based on an interval split covariance intersection filter with fault detection and exclusion. *Vehicles*, 6 :352–373, 02 2024.

[8] E. Walter. *Numerical Methods and Optimization : A Consumer Guide*. Springer, London, 2014.

9 Annexe

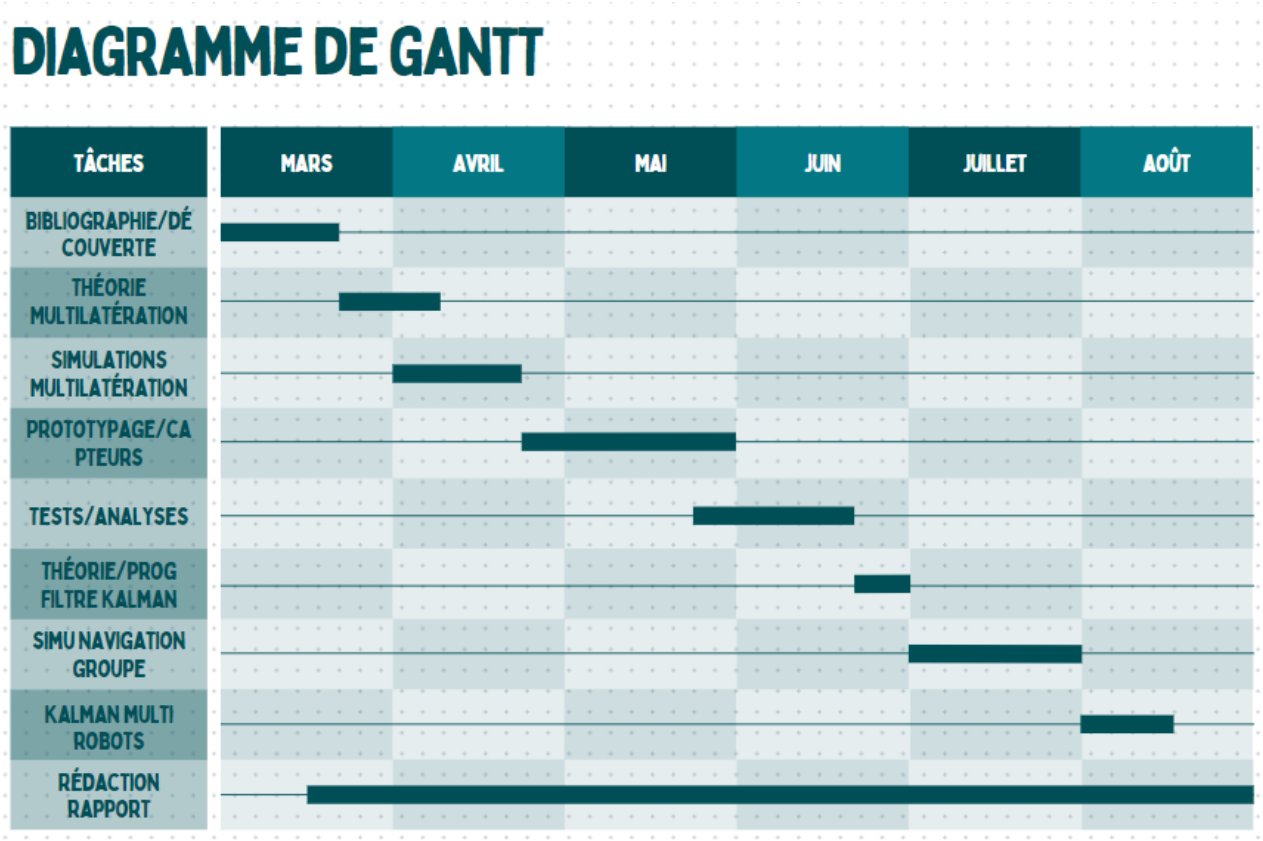

FIGURE 31 – Diagramme de Gantt du stage

Acronymes

CRIStAL Centre de Recherche en Informatique, Signal et Automatique de Lille
SMAC Système multi-agents et comportements
ANR Agence nationale de la recherche
UWB Ultra wide band
SLAM Simultaneous localization and mapping(Localisation et cartographie simultanées)
GNSS Global Navigation Satellite System (Géolocalisation et Navigation par un Système de Satellites)
RTK Real-Time Kinematic (Cinématique en Temps Réel)
EKF Extended Kalman Filter (Filtre de Kalman Étendu)
UKF Unscented Kalman Filter (Filtre de Kalman Unscented)
INS Inertial Navigation System (Système de Navigation Inertielle)
ROS Robot Operating System (Système d'Exploitation pour Robots)