Bearing-only SLAM: comparison between probabilistic and deterministic methods

Cyril Joly, Patrick Rives

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Bearing-only SLAM : comparaison entre une méthode probabiliste et une méthode déterministe

Cyril Joly — Patrick Rives

N° 6602

Juillet 2008

Thème NUM

Rapport de recherche

ISSN 0249-6399 ISRN INRIA/RR--6602--FR+ENG

Bearing-only SLAM : comparaison entre une méthode probabiliste et une méthode déterministe

Cyril Joly*, Patrick Rives *

Thème NUM — Systèmes numériques
Projet ARobAS

Rapport de recherche n° 6602 — Juillet 2008 — 91 pages

Résumé : Ce travail de recherche s'inscrit dans la problématique de la localisation et cartographie simultanée (SLAM en anglais). Les approches classiques pour résoudre le SLAM sont basées sur le filtre de Kalman étendu (EKF-SLAM) ou le filtre particulaire (FastSLAM). Ces deux familles d'algorithmes permettent une résolution en ligne du problème mais posent des problèmes d'inconsistance. Dans ce rapport nous ne nous intéressons pas aux approches précitées mais à des approches globales. Celles-ci nécessitent d'utiliser l'ensemble des mesures acquises pour calculer l'ensemble de la trajectoire du robot et la localisation des amers. Bien que les approches globales ne permettent pas un traitement en ligne, elle ne sont pas dénuées d'intérêt : elles sont *a priori* moins soumises aux problèmes d'inconsistance que l'EKF-SLAM ou le FastSLAM. Nous nous intéressons à deux méthodes : le GraphSLAM, basé sur une approche probabiliste gaussienne, et le SLAM par intervalles, basé sur une approche déterministe. Ces méthodes sont comparées à l'aide de simulations dans le cas du SLAM *bearing-only*. Nous avons simulé une trajectoire circulaire plane pour le robot. Les amers, quant à eux, sont des points 3D dont on mesure le gisement et l'élévation par rapport au robot. Les résultats mettent en évidence la consistance des deux algorithmes lorsque les erreurs sont centrées. Dans ce cas précis, le GraphSLAM délivre de bien meilleurs résultats que le SLAM par intervalles en terme de taille des zones de confiance. En revanche, si les données sont biaisées, l'algorithme de GraphSLAM est inconsistant. Le SLAM par intervalles, quant à lui, délivre des résultats consistants tout à fait cohérents.

Mots-clés : localisation, cartographie, analyse gaussienne, analyse par intervalles

* INRIA Sophia-Antipolis – Projet ARobAS, 2004 Route des Lucioles, BP 93, 06902 Sophia-Antipolis Cedex, France, FirstName.LastName@sophia.inria.fr

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

Bearing-only SLAM: comparison between probabilistic and deterministic methods

Abstract: This work deals with the problem of simultaneous localization and mapping (SLAM). Classical methods for solving the SLAM problem are based on the Extended Kalman Filter (EKF-SLAM) or particle filter (FastSLAM). These kinds of algorithms allow on-line solving but could be inconsistent. In this report, the above-mentioned algorithms are not studied but global ones. Global approaches need all measurements from the initial step to the final step in order to compute the trajectory of the robot and the location of the landmarks. Even if global approaches do not allow on-line solving, they can be more interesting than EKF-SLAM or FastSLAM since they are less sensitive to inconsistencies. Two algorithms are studied: the GraphSLAM, a probabilistic method based on gaussian hypothesis, and the "interval SLAM" which is a deterministic approach. A comparison of the algorithms is made in simulation, for the *bearing-only* case. A circular and planar trajectory for the robot is simulated. Landmarks are 3D points from which we measure the bearing and elevation angles with respect to the robot. The results show the consistency of both algorithms when the errors are centered. In this case, if we look the size of the belief areas provided by the algorithms, GraphSLAM delivers better results than interval SLAM. Finally, the GraphSLAM algorithm becomes inconsistent when input data are biased. In the latter case, interval SLAM gives good and consistent results.

Key-words: localization, mapping, gaussian analysis, interval analysis

Table des matières

1 Introduction

Les problèmes de localisation et de cartographie sont des problèmes majeurs en robotique mobile. On parle de localisation lorsqu'on cherche à positionner un robot par rapport à une carte connue. Lorsqu'on cherche à construire une représentation de l'environnement connaissant la trajectoire du robot, on parle au contraire de reconstruction ou de cartographie. Ces deux problèmes sont bien connus et aujourd'hui résolus. Si on cherche à retrouver la trajectoire du robot *et* à reconstruire l'environnement, on parle de localisation et cartographie simultanée, ou SLAM (*Simultaneous Localization and Mapping*). Il s'agit d'un problème central si on veut concevoir des robots complètements autonomes. Une des applications souvent citée est l'exploration autonome d'environnements inconnus et/ou dangereux ([9]).

Le problème du SLAM a été abordé dans les années 1980 avec les travaux de Smith et Cheeseman([17]). Dans cette première approche, les auteurs utilisent un *filtre de Kalman étendu* : on parle d'EKF-SLAM. L'EKF-SLAM est longtemps resté l'approche principale de résolution du problème de SLAM. Elle permet d'évaluer en ligne (dans la limite de la vitesse de calcul de la machine utilisée) l'état du robot ainsi que la carte d'amer. Cette approche est encore beaucoup utilisée aujourd'hui ([4, 13, 16]). Néanmoins, elle n'est pas exempte de reproches. En effet, plusieurs publications mettent en évidence le caractère *inconsistant* de l'EKF-SLAM : les ellipses de confiance déduites des matrices de variances-covariances sont trop petites par rapport à l'erreur réelle ([2, 8]). De plus, l'EKF-SLAM nécessite l'inversion de la matrice de variances-covaraiances de l'état.[1] Le coût de cette inversion grandit avec le carré du nombre d'amers. Pour corriger ce dernier point, Montemerlo a proposé l'algorithme de FastSLAM ([10, 11, 12]). Cet algorithme utilise un *filtre particulaire Rao-Blackwellisé* pour résoudre le problème du SLAM : un jeu de particules pondérées représente plusieurs hypothèses concernant l'état du robot. A chaque particule est ensuite associée une carte différente. L'utilisation du filtre particulaire permet de rendre l'ensemble des amers indépendants pour une particule donnée (il s'agit de la factorisation du SLAM conditionnellement à la trajectoire). Ainsi, l'algorithme de FastSLAM permet de garantir un temps d'exécution beaucoup plus faible que l'EKF-SLAM. En revanche, cet algorithme n'est pas consistant, ceci à cause du phénomène de *dégénérescence des particules* ([3]). Ces méthodes, ainsi que la description complète du problème du SLAM sont décrits dans les tutoriels de Tim Bailey ([1, 6]).

Nous proposons dans ce rapport de recherche d'étudier et de comparer deux autres algorithmes pour résoudre le problème du SLAM. Il s'agit de deux méthodes globales. Contrairement aux approches de type EKF-SLAM ou FastSLAM (qui traitent les informations à l'instant t uniquement à l'aide des mesures à l'instant t et de l'état à l'instant $t-1$), les approches globales résolvent le problème du SLAM sur l'ensemble de la trajectoire (ceci pouvant nécessiter plusieurs itérations avant convergence). Cela permet en théorie d'améliorer l'estimation de l'état à l'instant t à l'aide d'informations acquises aux instants $t' > t$. Dans ce cas, on parle en général de SAM (*Smoothing And Mapping*). De telles méthodes ne sont *a priori* pas adaptées au temps réel ou à l'évaluation en ligne de l'état. Néanmoins, elles

[1] L'état contient la position du robot à l'instant étudié ainsi que les paramètres de l'ensemble des amers.

ne sont pas dénuées d'intérêt, notamment en ce qui concerne la consistance. La première méthode étudiée est le *GraphSLAM*, proposé par Thrun et Montemerlo ([18]). La seconde méthode est une méthode originale basée sur l'*analyse par intervalles*. Ces deux méthodes sont sommairement décrites dans le paragraphe suivant :

GraphSLAM : il s'agit d'une méthode de filtrage et de lissage probabiliste basée sur l'hypothèse gaussienne. L'ensemble des mesures effectuées est utilisé pour construire une estimation de l'ensemble de la trajectoire et des amers ainsi que les matrices de variances-covariances associées. Les temps de calcul restent acceptables de part l'utilisation d'un *filtre d'information*. La description de cette méthode fait l'objet de la section 3.

SLAM par intervalles : il s'agit d'une méthode **déterministe** pour reconstruire la trajectoire du robot ainsi que l'ensemble des amers. L'idée est ici de supposer que les erreurs sont comprises dans des intervalles consistants afin de déduire des intervalles sur la position du robot et des amers. Aucune approximation n'est faite. La description générale de cette méthode fait l'objet de la section 5.

Les deux algorithmes comparés utilisent des hypothèses différentes concernant les erreurs. Le GraphSLAM utilise l'idée répandue que les erreurs respectent une distribution gaussienne (tout comme l'EKF). La seconde méthode, quant à elle, ne fait qu'une hypothèse de bornitude. La comparaison des performances sera faite grâce à des simulations. La façon de générer les erreurs ne sera pas innocente. Une discussion détaillée est effectuée dans la section 7.

Ce document s'articule en 6 parties. Tout d'abord, nous rappelons brièvement les hypothèses effectuées dans le cadre de cette étude. La section 3 présente l'algorithme général de GraphSLAM. L'application du GraphSLAM à notre cas est décrite dans la section 4. La section 5 de ce document présente quant à elle des généralités sur les méthodes de résolution par intervalles. Nous présentons ensuite la mise en œuvre de cet algorithme (section 6). Nous présentons enfin des résultats comparatifs obtenus en simulation (section 7).

2 Hypothèses générales du SLAM

Nous présentons dans cette section les hypothèses et notations employées dans la suite de ce document. La partie 2.1 présente les notations génériques utilisées tandis que la partie 2.2 présente les équations utilisées dans notre cas particulier.

2.1 Notations génériques

L'objectif du SLAM est de retrouver à la fois la trajectoire d'un robot et la position d'amers (supposés statiques dans notre étude). Il s'agit donc d'effectuer la reconstruction de l'état du robot à tous les instants ainsi que l'état de l'ensemble de la carte d'amers. Les notations employées sont présentées dans le tableau 1. On pourra noter que lorsqu'un état ou une mesure désigne un amer, l'indice de l'amer est noté entre parenthèses afin d'éviter toute ambiguïté avec les indices correspondant au temps.

Notation	Description	Valeurs dans
$\mathbf{x}_t$	Etat du robot à l'instant t	$\mathbb{R}^n$
N	Nombre d'amers	$\mathbb{N}$
$\mathbf{m}_{(i)}$	Etat de l'amer numéro (i)	$\mathbb{R}^{l_{(i)}}$
$\mathbf{m}$	Etat de l'ensemble des amers	$\mathbb{R}^{\sum_{i=1}^N l_{(i)}}$
$\mathbf{u}_t$	Entrée pour l'évolution du robot	$\mathbb{R}^p$
$\mathbf{z}_{t,(i)}$	Mesure associée à l'amer (i) et à l'instant t	$\mathbb{R}^{q_{(i)}}$
$\mathbf{z}_t$	Concaténations de toutes les mesures à l'instant t	$\mathbb{R}^{\sum_{i=1}^N q_{(i)}}$
$\mathbf{f}$	Fonction d'évolution du robot (dépend de $\mathbf{x}_t$ et de $\mathbf{u}_t$)	$\mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^n$
$\mathbf{h}_{(i)}$	Fonction de mesure de l'amer (i) (dépend de $\mathbf{x}_t$ et de $\mathbf{m}_{(i)}$)	$\mathbb{R}^n \times \mathbb{R}^{l_{(i)}} \to \mathbb{R}^{q_{(i)}}$
$\mathbf{h}$	Fonction de mesures (dépend de $\mathbf{x}_t$ et de $\mathbf{m}$)	$\mathbb{R}^n \times \mathbb{R}^{\sum_{i=1}^N l_{(i)}} \to \mathbb{R}^{\sum_{i=1}^N q_{(i)}}$
$\nu_t^{\mathbf{XX}}$	Erreur associée à $\mathbf{XX}$ à l'instant t ($\mathbf{XX}$: vecteur générique)	$\mathbb{R}^{\dim(\mathbf{XX})}$
$\mathbf{XX}_{t_1:t_2}$	Concaténation des vecteurs $\mathbf{XX}_t$ entre les instants t_1 et t_2	$\mathbb{R}^{\dim(\mathbf{XX})(t_2-t_1+1)}$

TAB. 1 – Notations utilisées

Remarque 2.1
Il est envisageable de traiter des amers de différents types et donc paramétrés différemment. Ceci explique la notation $l_{(i)}$ pour la dimension de l'état de l'amer (i) et $q_{(i)}$ pour la dimension de la mesure de l'amer (i).
Pour la suite, on posera $l = \sum_{i=1}^N l_{(i)}$ et $q = \sum_{i=1}^N q_{(i)}$.

Dans cette étude, on suppose que l'évolution de l'état du robot entre deux instants est donnée par la fonction $\mathbf{f}$ et le vecteur d'entrées $\mathbf{u}_t$ (supposé mesuré) :

$$\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t) \tag{1}$$

Lorsque la fonction $\mathbf{f}$ ne représente pas exactement la réalité et/ou que le vecteur $\mathbf{u}_t$ n'est pas exactement le vecteur attendu, on aura :

$$\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t + \nu_t^{\mathbf{u}}) + \nu_{t+1}^{\mathbf{f}} \tag{2}$$

où $\nu_t^{\mathbf{u}}$ désigne l'erreur sur l'entrée et $\nu_{t+1}^{\mathbf{f}}$ l'erreur de modèle à l'instant t.

Ensuite, on suppose que le vecteur de mesures peut se déduire de l'état grâce à la fonction $\mathbf{h}$:

$$\mathbf{z}_t = \mathbf{h}(\mathbf{x}_t, \mathbf{m}) + \nu_t^{\mathbf{z}} \tag{3}$$

où $\nu_t^{\mathbf{z}}$ désigne l'erreur sur les mesures à l'instant t.

Nous supposons dans la suite de ce document que les fonctions $\mathbf{f}$ et $\mathbf{h}$ sont connues analytiquement. Le problème du SLAM sera résolu grâce à la connaissance de l'ensemble des

mesures $\mathbf{z}_t$ et des entrées $\mathbf{u}_t$. Les mesures et entrées étant entachées d'erreur, les algorithmes de SLAM devront être robustes vis à vis d'elles.

2.2 Application au SLAM 2D/3D dans le cas d'un robot non holonome

Nous présentons dans cette partie les équations utilisées dans le cadre du SLAM 2D/3D. Nous désignons par SLAM 2D/3D la problématique du SLAM appliquée au cas d'un robot se déplaçant dans un plan et observant des amers répartis dans tout l'espace. Nous définissons d'abord le cadre de l'étude. Nous présentons ensuite la paramétrisation choisie. Enfin, nous exposons les équations (de modèle et de mesures) utilisées.

2.2.1 Cadre de l'étude

Cette étude traite du SLAM dans le cas " bearing only " : seuls les angles relatifs entre le robot et les amers sont mesurés. Typiquement, une caméra fournit ce type d'information. Cette approche est en opposition avec le cas " range and bearing ", dont le capteur typique est le laser. Nous justifions notre approche par le fait que les capteurs permettant d'obtenir l'information de distance sont coûteux et nécessitent plus de temps d'acquisition (cas du laser 3D) ou de traitement (cas d'une paire de caméras).

Par ailleurs, nous ne traitons ici que le cas 2D en ce qui concerne le robot : le robot évolue dans un plan horizontal. L'état du robot est donc paramétré par ses coordonnées dans le plan et son orientation. Les amers, quant à eux, sont des points 3D.[2] Les méthodes générales proposées sont néanmoins applicables au cas où tout est en 3D (il faut alors adapter les fonctions utilisées).

Enfin, nous supposons que le robot vérfie une contrainte de non-holonomie : **sa vitesse tranversale est nulle**. Ceci équivaut à supposer que le robot évolue le long d'un arc de cercle entre deux instants consécutifs.

2.2.2 Paramétrisation

Nous explicitons dans ce paragraphe les vecteurs $\mathbf{x}_t$, $\mathbf{u}_t$ et $\mathbf{z}_t$ utilisés (cf. figure 1) :

$\mathbf{x}_t$: nous supposons que le robot évolue dans le plan. A l'instant t, il est repéré par sa position dans le repère absolu (x_t, y_t) et son orientation par rapport à l'axe des abscisses (θ_t). L'environnement étant entièrement inconnu, le repère absolu est choisi égal au repère initial du robot.[3] On a :

$$\mathbf{x}_t = \begin{bmatrix} x_t & y_t & \theta_t \end{bmatrix}^T \tag{4}$$

$\mathbf{u}_t$: le robot est vu comme un solide en mouvement, animé par un torseur cinématique qui sera utilisé comme vecteur de commande. Etant donné que nous sommes dans le cas 2D, le vecteur vitesse à chaque instant t est dans le plan du véhicule. Il est repéré par

[2] Le cas d'un robot évoluant en intérieur illustre bien ces hypothèses.
[3] L'état du robot est donc initialisé à zéro, avec une incertitude nulle.

ses deux composantes exprimées dans le repère du robot (V_t^x,V_t^y). Le vecteur rotation est quant à lui perpendiculaire au plan du véhicule et est paramétré uniquement par la vitesse de rotation instantanée (ω_t). On a donc :

$$\mathbf{u}_t = \left[\begin{array}{ccc} V_t^x & V_t^y & \omega_t \end{array} \right]^T \tag{5}$$

Une mesure de ce vecteur de commande est fournie par l'**odométrie du robot**, qui nous donne une vitesse de translation axiale (supposée être égale à V_t^x) et une vitesse de rotation (supposée être égale à ω_t). Enfin, V_t^y **est fixée à 0 pour respecter la contrainte de non-holonomie** (mesure virtuelle).[4]

$\mathbf{m}_{(i)}$ **:** nous traitons dans cette étude des amers ponctuels placés dans l'espace. Ils sont repérés par leurs coordonnées cartésiennes ($x_{(i)}$,$y_{(i)}$,$z_{(i)}$). On a donc :

$$\mathbf{m}_{(i)} = \left[\begin{array}{ccc} x_{(i)} & y_{(i)} & z_{(i)} \end{array} \right]^T \tag{6}$$

$\mathbf{z}_t$ **:** cette étude traite du cas " bearing-only " : nous supposons que nous n'avons que des informations angulaires pour chaque amer. Nous supposons par ailleurs qu'il s'agit d'informations haut niveau (pouvant être par exemple déduite du traitement de l'image d'une caméra) : le gisement ($\alpha_{t,(i)}$) et l'élévation ($\beta_{t,(i)}$). On a donc :

$$\mathbf{z}_t = \left[\begin{array}{ccccc} \alpha_{t,(1)} & \beta_{t,(1)} & \dots & \alpha_{t,(N)} & \beta_{t,(N)} \end{array} \right]^T \tag{7}$$

2.2.3 Equations d'évolution du robot

Le robot étant vu comme un solide dont le torseur cinématique est $\mathbf{u}_t$, ses coordonnées vérifient (dans le domaine continu) le système 8 :

$$\left\{ \begin{array}{rcl} \dot{x} & = & V^x \cos\theta - V^y \sin\theta \\ \dot{y} & = & V^x \sin\theta + V^y \cos\theta \\ \dot{\theta} & = & \omega \end{array} \right. \tag{8}$$

Supposons désormais que l'entrée du système soit constante entre les instants t et $t+1$ et soit égale à $\mathbf{u}_t$. L'intégration du système 8 nous donne la fonction $\mathbf{f}$:

$$\left\{ \begin{array}{rcl} x_{t+1} & = & x_t + \mathrm{sinc}\left(\frac{\omega_t dt}{2}\right) \left[V_t^x dt \cos\left(\theta_t + \frac{\omega_t dt}{2}\right) - V_t^y dt \sin\left(\theta_t + \frac{\omega_t dt}{2}\right)\right] \\ y_{t+1} & = & y_t + \mathrm{sinc}\left(\frac{\omega_t dt}{2}\right) \left[V_t^x dt \sin\left(\theta_t + \frac{\omega_t dt}{2}\right) + V_t^y dt \cos\left(\theta_t + \frac{\omega_t dt}{2}\right)\right] \\ \theta_{t+1} & = & \theta_t + \omega_t dt \end{array} \right. \tag{9}$$

où dt désigne le temps écoulé entre les instants t et $t+1$.

Deux sources d'erreurs interviennent dans le système 9 :

[4] Néanmoins, nous en tenons compte dans les équations pour pouvoir tenir compte du cas où de légers glissements latéraux apparaissent. Ainsi, nous considérons que la mesure virtuelle $V_t^y = 0$ peut être entachée d'erreurs.

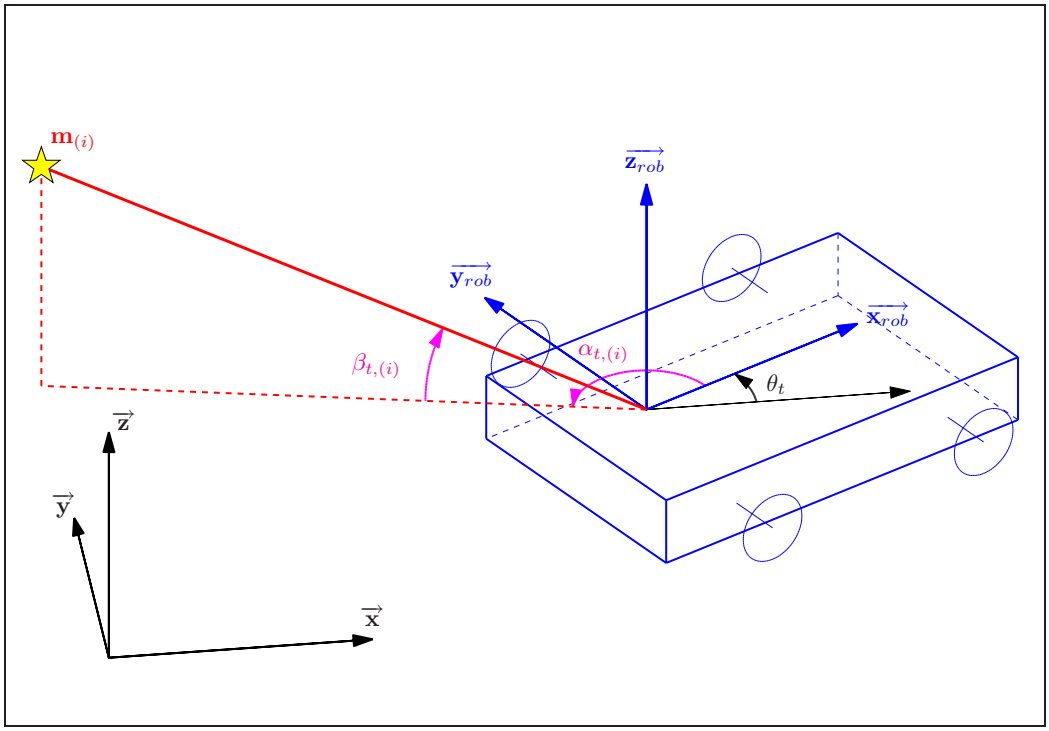

FIG. 1 – Notations du SLAM 2D

1. la commande $\mathbf{u}_t$ est mal connue. Cela peut arriver en cas d'imprécision sur l'odométrie. Des cas plus extrêmes peuvent apparaître en cas de dérapage : l'équation mécanique reste vraie mais les torseurs cinématiques utilisés pour intégrer l'équation n'ont en général plus aucun lien avec la réalité (leur mesure étant basée sur l'odométrie).
2. la commande varie continuement entre les instants t et $t+1$. Nous supposons que l'échantillonnage est suffisamment fin pour négliger ce phénomène et pouvoir le ramener à un bruit additif de " faible amplitude " ($\nu^{\mathbf{f}}_{t+1}$ dans l'équation 2).

En pratique, l'odométrie ne nous permet pas d'accéder aux vitesses. Dans le cas du robot non-holonome que nous étudions (dont l'équation de contrainte est $V^y_t = 0$), nous ne disposons que de l'incrément de rotation ($\omega_t dt$) et de la distance parcourue ($V^x_t dt$) entre les instants t et $t+1$. En notant $ds^x_t = V^x_t dt$, $ds^y_t = V^y_t dt$ et $d\omega_t = \omega_t dt$, l'équation 9 peut être écrite de la façon suivante :

$$\left\{ \begin{array}{lcl} x_{t+1} & = & x_t + \mathrm{sinc}\left(\frac{d\omega_t}{2}\right)\left[ds^x_t \cos\left(\theta_t + \frac{d\omega_t}{2}\right) - ds^y_t \sin\left(\theta_t + \frac{d\omega_t}{2}\right)\right] \\ y_{t+1} & = & y_t + \mathrm{sinc}\left(\frac{d\omega_t}{2}\right)\left[ds^x_t \sin\left(\theta_t + \frac{d\omega_t}{2}\right) + ds^y_t \cos\left(\theta_t + \frac{d\omega_t}{2}\right)\right] \\ \theta_{t+1} & = & \theta_t + d\omega_t \end{array} \right. \quad (10)$$

Remarque 2.2
Ainsi, l'équation de modèle fait appel à deux entrées mesurées et une entrée virtuelle.

2.2.4 Equations de mesures

Les équations de mesures, quant à elles, sont données par des considérations géométriques ; lorsque les mesures sont parfaites, celles-ci s'écrivent en fonction de l'état du robot et des amers (fonction **h**) :

$$\left\{ \begin{array}{rcl} \alpha_{t,(i)} & = & \text{arctan2}\left(\ y_{(i)} - y_t \quad , \quad x_{(i)} - x_t \ \right) - \theta_t \\ \beta_{t,(i)} & = & \arctan\left(\frac{z_{(i)}}{\sqrt{(x_{(i)}-x_t)^2+((y_{(i)}-y_t)^2}} \right) \end{array} \right. \tag{11}$$

Remarque 2.3 (Valeurs possibles pour les angles mesurés)
Les valeurs admissibles pour $(\alpha_{t,(i)}, \beta_{t,(i)})$ *sont prises dans* $[-\pi, \pi] \times [-\pi/2, \pi/2]$. *Ceci permet de garantir une bijection entre l'ensemble des couples angulaires possibles et l'ensemble des vecteurs à 3 composantes et de norme (euclidienne) unitaire.*
C'est pourquoi la définition de $\alpha_{t,(i)}$ *utilise la fonction* arctan2 *alors que la définition de* $\beta_{t,(i)}$ *n'utilise que* arctan.

3 Présentation du GraphSLAM

Nous présentons dans cette section l'algorithme de GraphSLAM. Il s'agit d'une méthode probabiliste basée sur l'hypothèse gaussienne. Contrairement à l'EKF-SLAM ou au FastSLAM, l'estimation de l'état du robot se fait en utilisant l'ensemble des informations depuis l'instant initial. Nous verrons que ceci est possible dans un temps raisonnable grâce à l'utilisation de la matrice d'information de la densité de probabilité. Cette section est divisée en deux paragraphes. Nous effectons dans un premier temps des rappels généraux sur le SLAM probabiliste. Le deuxième paragraphe décrit en détails l'algorithme de GraphSLAM. Cette description est essentiellement tirée de [18].

Les paragraphes suivants décrivent en détail les développements mathématiques qui permettent la construction de la matrice d'information et la résolution du problème d'inférence final.

3.1 Rappels sur le SLAM probabiliste

3.1.1 Rappels généraux

Dans la littérature, le problème du SLAM est en général résolu de manière probabiliste et est posé de façon récursive. L'objectif est alors de déterminer à chaque instant t la *densité de probabilité* de $\mathbf{x}_t$ (ou $\mathbf{x}_{0:t}$) et de la carte sachant l'ensemble des mesures, des contrôles et l'état initial. Il s'agit donc de trouver pour tout t :

$$p\left(\mathbf{x}_t, \mathbf{m} | \mathbf{z}_{0:t}, \mathbf{u}_{0:t}, \mathbf{x}_0\right) \quad \text{ou} \quad p\left(\mathbf{x}_{0:t}, \mathbf{m} | \mathbf{z}_{0:t}, \mathbf{u}_{0:t}, \mathbf{x}_0\right) \tag{12}$$

Dans le cas de l'EKF-SLAM, on cherche à calculer à chaque instant t la densité de probabilité $p(\mathbf{x}_t, \mathbf{m}|\mathbf{z}_{0:t}, \mathbf{u}_{0:t}, \mathbf{x}_0)$. Dans le cas présenté ici, à savoir le GraphSLAM, nous cherchons à calculer $p(\mathbf{x}_{0:t}, \mathbf{m}|\mathbf{z}_{0:t}, \mathbf{u}_{0:t}, \mathbf{x}_0)$.

De plus, nous effectuons les 3 hypothèses suivantes :

1. L'ensemble des états du robot forme une chaîne de Markov. Cela signifie que si $\mathbf{x}_{t-1}$ et $\mathbf{u}_t$ sont connus, la connaissance des états précédents, des commandes précédentes et des amers est inutile. Ceci se traduit par :

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t) = p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t, \mathbf{x}_{0:t-2}, \mathbf{u}_{0:t-1}, \mathbf{m}) \tag{13}$$

2. Les mesures sont conditionnellement indépendantes (au cours du temps). Ceci se traduit par :

$$p(\mathbf{z}_{0:t}|\mathbf{x}_{0:t}, \mathbf{m}) = \prod_{i=0}^{t} p(\mathbf{z}_i|\mathbf{x}_{0:t}, \mathbf{m}) \tag{14}$$

3. La mesure à l'instant i ne dépend que de l'état du robot à l'intant i et de la position des amers. On a en conséquence :

$$p(\mathbf{z}_i|\mathbf{x}_{0:t}, \mathbf{m}, \ldots) = p(\mathbf{z}_i|\mathbf{x}_i, \mathbf{m}) \tag{15}$$

Ces 3 hypothèses sont réalistes. En effet, la première suppose que l'on peut obtenir la position du véhicule en ne se servant que de la dernière position et du contrôle appliqué depuis cette position (utilisation de l'équation de modèle 10). La deuxième hypothèse, quant à elle, indique que les bruits des capteurs ne sont pas corrélés dans le temps, ce qui est également assez réaliste. Enfin, les capteurs que l'on utilise mesurent la position relative du robot et des amers à un instant bien précis, ce qui justifie la dernière hypothèse.

Une dernière hypothèse est utilisée : il s'agit de l'hypothèse **gaussienne**. Cette dernière est utilisée dans la plupart des approches probabilistes (hypothèse de base du filtre de Kalman étendu utilisé dans l'EKF-SLAM et dans la partie EKF du FastSLAM). Cette hypothèse constitue également une des briques de base des développements mathématiques du GraphSLAM.

3.2 Description du GraphSLAM

L'algorithme de GraphSLAM tire profit du caractère épars de la matrice d'information associée à l'état (trajectoire du robot et amers). Ceci provient de l'interprétation graphique du problème du SLAM, et notamment du fait que conditionnellement à toute la trajectoire, les amers sont indépendants entre eux. Le principe du GraphSLAM est de définir un état qui contient toutes les positions depuis l'instant initial ainsi que la position des amers. La matrice d'information (voir section 3.2.1 pour avoir des rappels sur cette représentation) du système est construite de manière incrémentale (cf. figure 2). Nous verrons que la matrice d'information et le vecteur d'information ne nous donnent pas directement la trajectoire du robot ainsi que la position des amers. Pour retrouver la position du robot, on marginalise

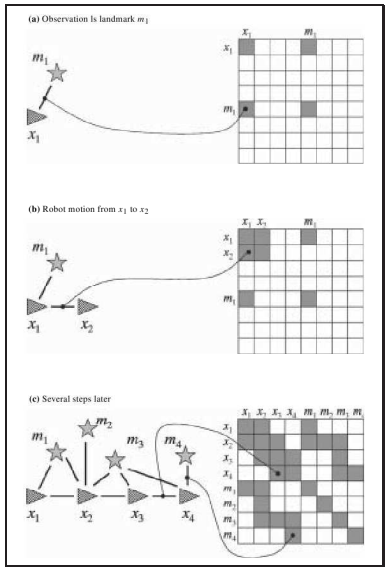

FIG. 2 – Construction incrémentale de la matrice d'information totale ([18])

incrémentalement la matrice d'information ; ceci a pour effet d'ajouter des termes dans la matrice d'information (cf. figure 3).

Les paragraphes suivants décrivent en détail les développements mathématiques qui permettent la construction de la matrice d'information et la résolution du problème d'inférence final.

3.2.1 Représentation par matrice d'information et vecteur d'information

Dans les prochaines parties, nous parlerons de la représentation de densité de probabilité gaussienne par vecteur d'information et matrice d'information. Nous proposons dans cette partie de présenter de brefs rappels en ce qui concerne cette matrice et ce vecteur, et la relation qui existe entre les " moments classiques " pour représenter une gaussienne.

Soient μ et $\mathbf{\Sigma}$ l'espérance mathématique et la matrice de variances-covariances associées à une densité de probabilité gaussienne $p(\mathbf{x})$. On a alors :

$$p(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^n \det \mathbf{\Sigma}}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T \mathbf{\Sigma}^{-1}(\mathbf{x}-\mu)\right)$$

Le principe de la représentation par vecteur d'information et matrice d'information est de trouver un vecteur ξ et une matrice $\mathbf{\Omega}$ pour que le facteur exponentiel de p soit de la forme " $\exp\left(-\frac{1}{2}\mathbf{x}^T\mathbf{\Omega}\mathbf{x} + \mathbf{x}^T\xi\right)$ ".

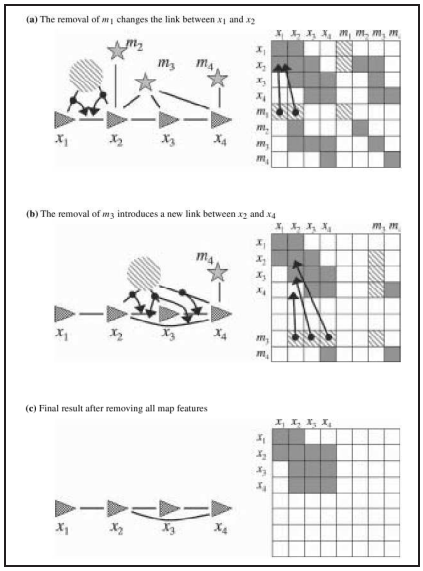

FIG. 3 – Marginalisation des amers pour retrouver la pose uniquement. Cette opération entraîne l'ajout de liens entre les positions ([18])

Pour ce faire, il suffit de ré-écrire la densité $p(\mathbf{x})$. Tous calculs faits, on obtient :

$$p(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^n \det \mathbf{\Sigma}}} \exp\left(-\frac{1}{2}\mu^T \mathbf{\Sigma}^{-1}\mu\right) \exp\left(-\frac{1}{2}\mathbf{x}^T \mathbf{\Sigma}^{-1}\mathbf{x} + \mathbf{x}^T \mathbf{\Sigma}^{-1}\mu\right)$$

Ainsi, en posant $\mathbf{\Omega} = \mathbf{\Sigma}^{-1}$ et $\xi = \mathbf{\Sigma}^{-1}\mu$, on obtient :

$$p(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^n \det \mathbf{\Omega}^{-1}}} \exp\left(-\frac{1}{2}\xi^T \mathbf{\Omega}^{-1}\xi\right) \exp\left(-\frac{1}{2}\mathbf{x}^T \mathbf{\Omega}\mathbf{x} + \mathbf{x}^T \xi\right)$$

$\mathbf{\Omega}$ est appelée *matrice d'information* et ξ est appelé *vecteur d'information*. Il est très facile d'identifier ces paramètres lorsqu'on a une expression quadratique dans une exponentielle. Les paramètres " classiques " de variance et d'espérance se retrouvent par :

$$\left\{ \begin{array}{rcl} \mathbf{\Sigma} & = & \mathbf{\Omega}^{-1} \\ \mu & = & \mathbf{\Sigma}\xi \end{array} \right.$$

3.2.2 Densité *a posteriori* de la trajectoire et de la carte

Dans ce qui suit, on cherche à exprimer la densité de probabilité *a posteriori* de l'ensemble de la trajectoire ainsi que de la carte (en supposant les associations connues), soit : $p(\mathbf{x}_{0:t}, \mathbf{m} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$.

En appliquant la règle de Bayes sur $\mathbf{z}_t$, on obtient :

$$p\left(\mathbf{x}_{0:t},\mathbf{m}|\mathbf{z}_{1:t},\mathbf{u}_{1:t}\right)=\frac{p\left(\mathbf{z}_t|\mathbf{x}_{0:t},\mathbf{m},\mathbf{z}_{1:t-1},\mathbf{u}_{1:t}\right)p\left(\mathbf{x}_{0:t},\mathbf{m}|\mathbf{z}_{1:t-1},\mathbf{u}_{1:t}\right)}{p\left(\mathbf{z}_t|\mathbf{z}_{1:t-1},\mathbf{u}_{1:t}\right)} \tag{16}$$

soit :

$$p\left(\mathbf{x}_{0:t},\mathbf{m}|\mathbf{z}_{1:t},\mathbf{u}_{1:t}\right)=\eta\, p\left(\mathbf{z}_t|\mathbf{x}_{0:t},\mathbf{m},\mathbf{z}_{1:t-1},\mathbf{u}_{1:t}\right)p\left(\mathbf{x}_{0:t},\mathbf{m}|\mathbf{z}_{1:t-1},\mathbf{u}_{1:t}\right) \tag{17}$$

où η est une constante de normalisation. En appliquant l'équation 15 sur le premier facteur de l'équation 17, on obtient :

$$p\left(\mathbf{x}_{0:t},\mathbf{m}|\mathbf{z}_{1:t},\mathbf{u}_{1:t}\right)=\eta\, p\left(\mathbf{z}_t|\mathbf{x}_t,\mathbf{m}\right)p\left(\mathbf{x}_{0:t},\mathbf{m}|\mathbf{z}_{1:t-1},\mathbf{u}_{1:t}\right) \tag{18}$$

En appliquant la règle de Bayes, le second facteur dans l'équation 18 peut s'écrire :

$$p\left(\mathbf{x}_{0:t},\mathbf{m}|\mathbf{z}_{1:t-1},\mathbf{u}_{1:t}\right)=p\left(\mathbf{x}_t|\mathbf{x}_{0:t-1},\mathbf{m},\mathbf{z}_{1:t-1},\mathbf{u}_{1:t}\right)p\left(\mathbf{x}_{0:t-1},\mathbf{m}|\mathbf{z}_{1:t-1},\mathbf{u}_{1:t}\right) \tag{19}$$

En utilisant l'équation 13, l'équation 19 devient :

$$p\left(\mathbf{x}_{0:t},\mathbf{m}|\mathbf{z}_{1:t-1},\mathbf{u}_{1:t}\right)=p\left(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{u}_t\right)p\left(\mathbf{x}_{0:t-1},\mathbf{m}|\mathbf{z}_{1:t-1},\mathbf{u}_{1:t}\right) \tag{20}$$

On peut désormais insérer le résultat de l'équation 20 dans l'équation 18 pour obtenir la formulation récursive suivante :

$$p\left(\mathbf{x}_{0:t},\mathbf{m}|\mathbf{z}_{1:t},\mathbf{u}_{1:t}\right)=\eta\, p\left(\mathbf{z}_t|\mathbf{x}_t,\mathbf{m}\right)p\left(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{u}_t\right)p\left(\mathbf{x}_{0:t-1},\mathbf{m}|\mathbf{z}_{1:t-1},\mathbf{u}_{1:t-1}\right) \tag{21}$$

On peut alors montrer par récurrence que :

$$p\left(\mathbf{x}_{0:t},\mathbf{m}|\mathbf{z}_{1:t},\mathbf{u}_{1:t}\right)=\eta'\; p\left(\mathbf{x}_0,\mathbf{m}\right)\prod_{k=1}^{t}p\left(\mathbf{x}_k|\mathbf{x}_{k-1},\mathbf{u}_k\right)p\left(\mathbf{z}_k,|\mathbf{x}_k,\mathbf{m}\right) \tag{22}$$

où η' est un facteur constant (a priori différent de η). Par ailleurs, $p\left(\mathbf{x}_0,\mathbf{m}\right)$ désigne la densité de probabilité *a priori* de la position du robot et de la carte.

A chaque instant t, le vecteur $\mathbf{z}_t$ contient la concaténation des observations effectuées sur plusieurs amers. Pour rappel, nous notons $\mathbf{z}_{t,(i)}$ l'observation de l'amer (i) à l'instant t. Soit $\mathcal{M}_t$ l'ensemble des amers observés à l'instant t. Pour simplifier la suite des développements, nous supposons qu'à un instant donné, les observations de chaque amer sont indépendantes entre elles. Ceci signifie donc que les vecteurs aléatoires $\mathbf{z}_{t,(i)}, i\in\mathcal{M}_t$ sont indépendants entre eux. En conséquence, la densité *a posteriori* de la trajectoire du robot et de la carte s'écrit :

$$p\left(\mathbf{x}_{0:t},\mathbf{m}|\mathbf{z}_{1:t},\mathbf{u}_{1:t}\right)=\eta'\; p\left(\mathbf{x}_0,\mathbf{m}\right)\prod_{k=1}^{t}\left[p\left(\mathbf{x}_k|\mathbf{x}_{k-1},\mathbf{u}_k\right)\prod_{i\in\mathcal{M}_t}p\left(\mathbf{z}_{k,(i)},|\mathbf{x}_k,\mathbf{m}_{(i)}\right)\right] \tag{23}$$

Remarque 3.1 (Elimination de m dans $p\left(\mathbf{x}_0,\mathbf{m}\right)$)
En général, la densité a priori *de la carte et de la position initiale sont indépendantes.* $p\left(\mathbf{x}_0,\mathbf{m}\right)$ *peut donc s'écrire* $p\left(\mathbf{x}_0\right)p\left(\mathbf{m}\right)$. *Si la carte est complètement inconnue, nous pouvons intégrer* $p\left(\mathbf{m}\right)$ *dans la constante en facteur. Dans ce qui suit, nous supposons que c'est le cas.*

3.2.3 Logarithme de la densité *a posteriori*

Nous allons désormais calculer le logarithme de la densité calculée dans l'équation 23. En tenant compte de la remarque 3.1, nous obtenons :

$$\log p\left(\mathbf{x}_{0:t}, \mathbf{m} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right) = \text{const.} + \log p\left(\mathbf{x}_0\right) + \sum_{k=1}^{t} \left[\log p\left(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{u}_k\right) \sum_{i \in \mathcal{M}_k} \log p\left(\mathbf{z}_{k,(i)}, | \mathbf{x}_k, \mathbf{m}_{(i)}\right) \right] \tag{24}$$

En supposant que les bruits appliqués sur le modèles d'évolution et de mesures (fonctions $\mathbf{f}$ et $\mathbf{h}$), sont additifs, centrés et gaussiens de matrices de variances-covariances $\mathbf{Q}_t$ et $\mathbf{R}_t$, on obtient :

$$\left\{ \begin{array}{rcl} p\left(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{u}_t\right) & \propto & \exp\left(-\frac{1}{2} \left(\mathbf{x}_t - \mathbf{f}_t(\mathbf{x}_{t-1}, \mathbf{u}_t)\right)^T \mathbf{Q}_t^{-1} \left(\mathbf{x}_t - \mathbf{f}_t(\mathbf{x}_{t-1}, \mathbf{u}_t)\right) \right) \\ p\left(\mathbf{z}_{t,(i)}, | \mathbf{x}_t, \mathbf{m}_{(i)}\right) & \propto & \exp\left(-\frac{1}{2} \left(\mathbf{z}_{t,(i)} - \mathbf{h}(\mathbf{x}_t, \mathbf{m}_{(i)})\right)^T \mathbf{R}_t^{-1} \left(\mathbf{z}_{t,(i)} - \mathbf{h}(\mathbf{x}_t, \mathbf{m}_{(i)})\right) \right) \end{array} \right. \tag{25}$$

Enfin, en supposant que la densité de probabilité *a priori* sur la position initiale est gaussienne, centrée, de matrice d'information $\mathbf{\Omega}_0$, $p\left(\mathbf{x}_0\right)$ est donc proportionnelle à " $\exp\left(\mathbf{x}_0^T \mathbf{\Omega}_0 \mathbf{x}_0\right)$ ". On a donc au final :

$$\begin{array}{rcl} \log p\left(\mathbf{x}_{0:t}, \mathbf{m} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right) & = & \text{const.} - \frac{1}{2} \Big[\mathbf{x}_0^T \mathbf{\Omega}_0 \mathbf{x}_0 \\ & & + \sum_{k=1}^{t} \left(\mathbf{x}_k - \mathbf{f}_k(\mathbf{x}_{k-1}, \mathbf{u}_k)\right)^T \mathbf{Q}_t^{-1} \left(\mathbf{x}_k - \mathbf{f}_k(\mathbf{x}_{k-1}, \mathbf{u}_k)\right) \\ & & + \sum_{k=1}^{t} \sum_{i \in \mathcal{M}_k} \left(\mathbf{z}_{k,(i)} - \mathbf{h}(\mathbf{x}_k, \mathbf{z}_{k,(i)})\right)^T \mathbf{R}_t^{-1} \left(\mathbf{z}_{k,(i)} - \mathbf{h}(\mathbf{x}_k, \mathbf{z}_{k,(i)})\right) \Big] \end{array} \tag{26}$$

3.2.4 Expansion de Taylor

L'équation 26 nous donne l'expression du logarithme de la densité *a posteriori* de la trajectoire du robot et de la carte. Malheureusement, celle-ci n'est pas quadratique en $\mathbf{x}$ et $\mathbf{m}$, mais seulement en $\mathbf{f}_t$ et $\mathbf{h}$. Nous allons désormais linéariser ces fonctions afin d'obtenir une expression quadratique en $\mathbf{x}$ et $\mathbf{m}$.

Soit $\mu_{0:t}^{\mathbf{x}}$ une trajectoire de linéarisation supposée proche de la trajectoire réelle et $\mu^{\mathbf{m}}$ un point de linéarisation pour la carte. Nous supposons $\mu_{0:t}^{\mathbf{x}}$ et $\mu^{\mathbf{m}}$ disponibles ; nous discuterons de la manière de les obtenir dans le paragraphe 3.2.6.

En supposant la linéarisation valide, les équations de prédiction et de mesure s'écrivent alors :

$$\left\{ \begin{array}{rcl} \mathbf{f}_t(\mathbf{x}_{t-1}, \mathbf{u}_t) & \approx & \mathbf{f}_t(\mu_{t-1}^{\mathbf{x}}, \mathbf{u}_t) + \mathbf{G}_t \left(\mathbf{x}_{t-1} - \mu_{t-1}^{\mathbf{x}}\right) \\ \mathbf{h}(\mathbf{x}_t, \mathbf{m}_{(i)}) & \approx & \mathbf{h}(\mu_t, \mu_{(i)}) + \mathbf{H}_{t,(i)}^{\mathbf{x}} \left(\mathbf{x}_t - \mu_t^{\mathbf{x}}\right) + \mathbf{H}_{t,(i)}^{\mathbf{m}} \left(\mathbf{m}_{(i)} - \mu_{(i)}^{\mathbf{m}}\right) \end{array} \right. \tag{27}$$

où :

$\mathbf{G}_t$ désigne la matrice jacobienne de $\mathbf{f}_t$ par rapport à $\mathbf{x}_{t-1}$ évalué en μ_{t-1} ;
$\mathbf{H}^{\mathbf{x}}_{t,(i)}$ désigne la matrice jacobienne de $\mathbf{h}$ par rapport à $\mathbf{x}_t$ évalué en $\mu^{\mathbf{x}}_t$ et $\mu^{\mathbf{m}}_{(i)}$;
$\mathbf{H}^{\mathbf{m}}_{t,(i)}$ désigne la matrice jacobienne de $\mathbf{h}$ par rapport à $\mathbf{m}$ évalué en $\mu^{\mathbf{x}}_t$ et $\mu^{\mathbf{m}}_{(i)}$.

Tous calculs faits, l'équation 26 peut se ré-écrire de la façon suivante :

$$
\begin{aligned}
\log p\left(\mathbf{x}_{0:t}, \mathbf{m} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right) = \text{const.}' - \tfrac{1}{2} \underbrace{\mathbf{x}_0^T \mathbf{\Omega}_0 \mathbf{x}_0}_{0} \\
+ \sum_{k=1}^{t} \left\{ \underbrace{-\tfrac{1}{2} \begin{bmatrix} \mathbf{x}_k \\ \mathbf{x}_{k-1} \end{bmatrix}^T \begin{bmatrix} \mathbf{I} \\ -\mathbf{G}_k^T \end{bmatrix} \mathbf{Q}_k^{-1} \begin{bmatrix} \mathbf{I} & -\mathbf{G}_k \end{bmatrix} \begin{bmatrix} \mathbf{x}_k \\ \mathbf{x}_{k-1} \end{bmatrix}}_{(1)} \right. \\
\left. + \begin{bmatrix} \mathbf{x}_k \\ \mathbf{x}_{k-1} \end{bmatrix}^T \underbrace{\begin{bmatrix} \mathbf{I} \\ -\mathbf{G}_k^T \end{bmatrix} \mathbf{Q}_k^{-1} \left(\mathbf{f}_k\left(\mu^{\mathbf{x}}_{k-1}, \mathbf{u}_k\right) + \mathbf{G}_k \mu^{\mathbf{x}}_{k-1} \right)}_{(2)} \right. \\
+ \sum_{i \in \mathcal{M}_k} \left\{ \underbrace{-\tfrac{1}{2} \begin{bmatrix} \mathbf{x}_k \\ \mathbf{m}_{(i)} \end{bmatrix}^T \begin{bmatrix} \left(\mathbf{H}^{\mathbf{x}}_{k,(i)}\right)^T \\ \left(\mathbf{H}^{\mathbf{m}}_{k,(i)}\right)^T \end{bmatrix} \mathbf{R}_k^{-1} \begin{bmatrix} \mathbf{H}^{\mathbf{x}}_{k,(i)} & \mathbf{H}^{\mathbf{m}}_{k,(i)} \end{bmatrix} \begin{bmatrix} \mathbf{x}_k \\ \mathbf{m}_{(i)} \end{bmatrix}}_{(3)} \right. \\
\left. \left. + \begin{bmatrix} \mathbf{x}_k \\ \mathbf{m}_{(i)} \end{bmatrix}^T \underbrace{\begin{bmatrix} \left(\mathbf{H}^{\mathbf{x}}_{k,(i)}\right)^T \\ \left(\mathbf{H}^{\mathbf{m}}_{k,(i)}\right)^T \end{bmatrix} \mathbf{R}_k^{-1} \left(\mathbf{z}_{k,(i)} - \mathbf{h}\left(\mu^{\mathbf{x}}_k, \mu^{\mathbf{m}}_{(i)}\right) + \begin{bmatrix} \mathbf{H}^{\mathbf{x}}_{k,(i)} & \mathbf{H}^{\mathbf{m}}_{k,(i)} \end{bmatrix} \begin{bmatrix} \mu^{\mathbf{x}}_k \\ \mu^{\mathbf{m}}_{(i)} \end{bmatrix} \right)}_{(4)} \right\} \right\}
\end{aligned}
\tag{28}
$$

Dans l'équation 28, nous n'avons retenu que les termes linéaires et quadratiques en $\mathbf{x}$ et $\mathbf{m}$. Le reste des termes sont des termes constants. L'équation 28 montre qu'il est ainsi possible d'écrire la densité *a posteriori* de la trajectoire et de la carte à l'aide d'une représentation par matrice d'information et vecteur d'information. On a alors :

$$
\log p\left(\mathbf{x}_{0:t}, \mathbf{m} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right) = \text{const.}' - \frac{1}{2} \begin{bmatrix} \mathbf{x}_{0:t} \\ \mathbf{m} \end{bmatrix}^T \mathbf{\Omega} \begin{bmatrix} \mathbf{x}_{0:t} \\ \mathbf{m} \end{bmatrix} + \begin{bmatrix} \mathbf{x}_{0:t} \\ \mathbf{m} \end{bmatrix}^T \xi \tag{29}
$$

La matrice $\mathbf{\Omega}$ de l'équation 29 est construite à partir du terme (0) et des termes de type (1) et (3) de l'équation 28. Le vecteur d'information ξ, quant à lui, est construit grâce aux termes de type (2) et (4) de l'équation 28.

3.2.5 Réduction de la matrice d'information et du vecteur d'information

Une fois la matrice d'information et le vecteur d'information de l'équation 29 construits, on pourrait essayer de retrouver l'équivalent en terme de matrice de variances-covariances

et d'espérance mathématique. La matrice de variances-covariances totale serait pleine. Les auteurs de [18] proposent de réduire cette forme en deux facteurs (l'un pour la trajectoire *a posteriori* et l'autre pour la carte *a posteriori* mais conditionnellement à la trajectoire). La justification de cette réduction est l'équation de factorisation de la densité *a posteriori* :

$$p\left(\mathbf{x}_{0:t}, \mathbf{m} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right) = p\left(\mathbf{x}_{0:t} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right) p\left(\mathbf{m} | \mathbf{x}_{0:t}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right) \tag{30}$$

Dans ce paragraphe, nous montrons donc comment obtenir les paramètres. Dans la suite, nous allons utiliser la décomposition canonique de l'état entre la trajectoire et la carte. Ainsi, nous pouvons adopter les notations intuitives suivantes :

$$\mathbf{\Omega} = \begin{bmatrix} \mathbf{\Omega}_{\mathbf{x}_{0:t},\mathbf{x}_{0:t}} & \mathbf{\Omega}_{\mathbf{x}_{0:t},\mathbf{m}} \\ \mathbf{\Omega}_{\mathbf{m},\mathbf{x}_{0:t}} & \mathbf{\Omega}_{\mathbf{m},\mathbf{m}} \end{bmatrix} \tag{31}$$

$$\xi = \begin{bmatrix} \xi_{\mathbf{x}_{0:t}} \\ \xi_{\mathbf{m}} \end{bmatrix} \tag{32}$$

3.2.5.1 Obtention de $p\left(\mathbf{x}_{0:t} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right)$:

Etant donné que les densités de probabilité considérées sont gaussiennes, l'estimation des paramètres de $p\left(\mathbf{x}_{0:t} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right)$ est donnée par le théorème de marginalisation. Ainsi, la matrice d'information et le vecteur d'information associés (notés $\tilde{\mathbf{\Omega}}$ et $\tilde{\xi}$) valent :

$$\tilde{\mathbf{\Omega}} = \mathbf{\Omega}_{\mathbf{x}_{0:t},\mathbf{x}_{0:t}} - \mathbf{\Omega}_{\mathbf{x}_{0:t},\mathbf{m}} \mathbf{\Omega}_{\mathbf{m},\mathbf{m}}^{-1} \mathbf{\Omega}_{\mathbf{m},\mathbf{x}_{0:t}} \tag{33}$$

$$\tilde{\xi} = \xi_{\mathbf{x}_{0:t}} - \mathbf{\Omega}_{\mathbf{x}_{0:t},\mathbf{m}} \mathbf{\Omega}_{\mathbf{m},\mathbf{m}}^{-1} \xi_{\mathbf{m}} \tag{34}$$

L'équation 28 nous montre que la matrice $\mathbf{\Omega}_{\mathbf{m},\mathbf{m}}$ est diagonale par blocs :

$$\mathbf{\Omega}_{\mathbf{m},\mathbf{m}} = \mathrm{diag}\left(\mathbf{\Omega}_{(1),(1)} , \dots , \mathbf{\Omega}_{(j),(j)} , \dots , \mathbf{\Omega}_{(N),(N)} \right) \tag{35}$$

où N représente le nombre total d'amers et :

$$\mathbf{\Omega}_{(j),(j)} = \sum_{k \in \tau(j)} \left(\mathbf{H}_{k,(i)}^{\mathbf{m}} \right)^T \mathbf{R}_k \left(\mathbf{H}_{k,(i)}^{\mathbf{m}} \right) \tag{36}$$

où $\tau(j)$ désigne l'ensemble des instants où l'amer j a été observé. De plus, la matrice $\mathbf{\Omega}_{\mathbf{x}_{0:t},\mathbf{m}}$ s'écrit :

$$\mathbf{\Omega}_{\mathbf{x}_{0:t},\mathbf{m}} = \begin{bmatrix} \mathbf{\Omega}_{\mathbf{x}_{0:t},(1)} & \cdots & \mathbf{\Omega}_{\mathbf{x}_{0:t},(j)} & \cdots & \mathbf{\Omega}_{\mathbf{x}_{0:t},(N)} \end{bmatrix} \tag{37}$$

avec :

$$\boldsymbol{\Omega}_{\mathbf{x}_{0:t},(j)} = \begin{bmatrix} \boldsymbol{\Omega}_{\mathbf{x}_1,(j)} \\ \vdots \\ \boldsymbol{\Omega}_{\mathbf{x}_k,(j)} \\ \vdots \\ \boldsymbol{\Omega}_{\mathbf{x}_t,(j)} \end{bmatrix} \quad \text{et} \quad \begin{cases} \boldsymbol{\Omega}_{\mathbf{x}_k,(j)} = \left(\mathbf{H}^{\mathbf{x}}_{k,(i)}\right)^T \mathbf{R}_k^{-1} \left(\mathbf{H}^{\mathbf{m}}_{k,(i)}\right) & \textbf{si } k \in \tau(j) \\ \boldsymbol{\Omega}_{\mathbf{x}_k,(j)} = \mathbf{0} & \textbf{sinon} \end{cases} \tag{38}$$

Au final, on a :

$$\boxed{\begin{array}{rcl} \tilde{\boldsymbol{\Omega}} & = & \boldsymbol{\Omega}_{\mathbf{x}_{0:t},\mathbf{x}_{0:t}} - \sum_{j=1}^{N} \boldsymbol{\Omega}_{\mathbf{x}_{0:t},(j)} \left(\boldsymbol{\Omega}_{(j),(j)}\right)^{-1} \boldsymbol{\Omega}_{(j),\mathbf{x}_{0:t}} \\ \tilde{\xi} & = & \xi_{\mathbf{x}_{0:t}} - \sum_{j=1} \boldsymbol{\Omega}_{\mathbf{x}_{0:t},(j)} \left(\boldsymbol{\Omega}_{(j),(j)}\right)^{-1} \xi_{(j)} \end{array}} \tag{39}$$

avec :

$$\xi_{(j)} = \sum_{k \in \tau(j)} \left\{ \left(\mathbf{H}^{\mathbf{m}}_{k,(j)}\right)^T \mathbf{R}_k^{-1} \left(\mathbf{z}_{k,(i)} - \mathbf{h}\left(\mu^{\mathbf{x}}_k, \mu^{\mathbf{m}}_{(j)}\right) + \begin{bmatrix} \mathbf{H}^{\mathbf{x}}_{k,(j)} & \mathbf{H}^{\mathbf{m}}_{k,(j)} \end{bmatrix} \begin{bmatrix} \mu^{\mathbf{x}}_k \\ \mu^{\mathbf{m}}_{(j)} \end{bmatrix} \right) \right\} \tag{40}$$

3.2.5.2 Obtention de $p(\mathbf{m}|\mathbf{x}_{0:t}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$:

Les paramètres de $p(\mathbf{m}|\mathbf{x}_{0:t}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$, quant à eux, s'obtiennent grâce au théorème sur le conditionnement par une variable aléatoire gaussienne. En notant $\tilde{\boldsymbol{\Omega}}^{\mathbf{m}|\mathbf{x}}$ et $\tilde{\xi}^{\mathbf{m}|\mathbf{x}}$ les paramètres cherchés, le théorème de conditionnement nous donne :

$$\boxed{\begin{array}{rcl} \tilde{\boldsymbol{\Omega}}^{\mathbf{m}|\mathbf{x}} & = & \boldsymbol{\Omega}_{\mathbf{m},\mathbf{m}} \\ \tilde{\xi}^{\mathbf{m}|\mathbf{x}} & = & \xi_{\mathbf{m}} + \boldsymbol{\Omega}_{\mathbf{m},\mathbf{x}_{0:t}} \mathbf{x}_{0:t} \end{array}} \tag{41}$$

3.2.5.3 Reconstruction de la trajectoire et de la carte

On peut désormais effectuer une estimation de la trajectoire *a posteriori* en transformant les paramètres d'information en espérance et matrice de variances-covariances. Soit $\hat{\mathbf{x}}_{0:t}$ et $\boldsymbol{\Sigma}^{\mathbf{x}}$ les paramètres cherchés, on a :

$$\boxed{\begin{array}{rcl} \boldsymbol{\Sigma}^{\mathbf{x}} & = & \tilde{\boldsymbol{\Omega}}^{-1} \\ \hat{\mathbf{x}}_{0:t} & = & \hat{\boldsymbol{\Sigma}}^{\mathbf{x}} \tilde{\xi} \end{array}} \tag{42}$$

L'estimation de la carte et de sa matrice de variances-covariances s'effectue de façon similaire :

$$\begin{array}{rcl} \hat{\boldsymbol{\Sigma}}^{\mathbf{m}} & = & \boldsymbol{\Omega}^{-1}_{\mathbf{m},\mathbf{m}} \\ \hat{\mathbf{m}} & = & \boldsymbol{\Sigma}^{\mathbf{m}} \left(\xi_{\mathbf{m}} + \boldsymbol{\Omega}_{\mathbf{m},\mathbf{x}_{0:t}} \hat{\mathbf{x}}_{0:t}\right) \end{array} \tag{43}$$

Sachant que la matrice $\mathbf{\Omega_{m,m}}$ est diagonale par bloc (un bloc pour chaque amer), il en résulte une décorrélation des amers conditionnellement à la trajectoire. On peut ainsi calculer une matrice de variances-covariances et une espérance pour chaque amer (j) :

$$\begin{array}{rcl} \mathbf{\Sigma}_{(j)} & = & \mathbf{\Omega}^{-1}_{(j),(j)} \\ \hat{\mathbf{m}}_{(j)} & = & \mathbf{\Sigma}_{(j)} \left(\xi_{(j)} + \mathbf{\Omega}_{(j),\mathbf{x}_{0:t}} \hat{\mathbf{x}}_{0:t}\right) \quad = \quad \mathbf{\Sigma}_{(j)} \left(\xi_{(j)} + \mathbf{\Omega}_{(j),\tau(j)} \hat{\mathbf{x}}_{\tau(j)}\right) \end{array} \tag{44}$$

Les auteurs de [18] ne suggèrent que de calculer les paramètres de la carte conditionnellement à la trajectoire. Le calcul des véritables paramètres *a posteriori* pour chaque amer est jugé trop lourd étant donné qu'il n'y a pas de factorisation possible comme c'est le cas conditionnellement à la trajectoire. Néanmoins, nous effectuerons le calcul complet des matrices de variances-covariances afin d'avoir une estimation de l'incertitude associée aux amers.

3.2.6 Obtenir les points de fonctionnement

Les développements précédents sont basés sur le fait qu'il existe un développement de Taylor permettant de rendre linéaire les équations de modèle et de mesure. Pour cela, le point de linéarisation doit être " suffisamment proche " de la vraie solution.

3.2.6.1 Première itération :

Dans un premier temps, nous pouvons initialiser la trajectoire en appliquant simplement la fonction de modèle avec les entrées acquises. Ensuite, nous pouvons utiliser cette trajectoire et les mesures pour effectuer un placement approximatif des amers (dans le cas *bearing-only*, on peut par exemple effectuer une initialisation aux moindres carrés des amers étant donné que l'on dispose de toute la trajectoire[5]).

Il devient alors possible d'appliquer l'algorithme de GraphSLAM grâce aux points de fonctionnement trouvés pour la trajectoire et la carte.

Remarque 3.2
Aucune incertitude n'est associée à ces points de fonctionnement. Ils ne doivent pas être considérés comme une information a priori *par exemple. Il s'agit juste de se placer suffisamment près de la solution pour que l'algorithme de GraphSLAM soit valide.*

3.2.6.2 Itérations suivantes :

Il est possible que les points de fonctionnement utilisés à l'origine ne soient pas parfaits (notamment dans le cas de mesures fortement bruitées). Ainsi, le résultat du GraphSLAM est peut être légèrement biaisé, tout en étant plus juste que l'initialisation initiale. On va alors relancer l'algorithme de GraphSLAM en utilisant les précédents résultats comme points de fonctionnement.

Cette stratégie itérative est alors maintenue jusqu'à convergence du résultat.

[5] Ce type d'initialisation est impossible dans le cas de l'EKF-SLAM.

3.2.7 Conclusion

Les étapes présentées mettent en évidence la philosophie du GraphSLAM, à savoir l'utilisation de la trajectoire totale du robot par le biais d'un filtre d'information qui fait ressortir la structure graphique du SLAM. Il existe d'autres algorithmes basés sur ce principe, comme le square root SAM ([5]). Dans ce dernier cas, des optimisations sont effectuées afin de factoriser la matrice d'information et de réduire les temps de calcul.

4 Mise en application du GraphSLAM

Nous présentons dans cette section l'application du GraphSLAM au cas 2D/3D étudié. Nous donnons dans un premier temps le détail des matrices jacobiennes utilisées. Nous explicitons ensuite les matrices de variances-covariances utilisées. Enfin, une discussion est faite concernant l'initialisation de la trajectoire du robot et des amers.

4.1 Matrices jacobiennes des fonctions f et h

L'application de l'algorithme de GraphSLAM nécessite de linéariser les fonctions $\mathbf{f}$ et $\mathbf{h}$ autour de points de fonctionnement supposés proches de la solution réelle. Cette linéarisation implique le calcul des matrices jacobiennes $\mathbf{f}$ et $\mathbf{h}$. Ces matrices peuvent dépendre à la fois de l'état et des entrées.

4.1.1 Matrice jacobienne de la fonction d'évolution

Cette matrice représente les variations de $\mathbf{x}_{t+1}$ par rapport à de petites variations sur $\mathbf{x}_t$. Elle est notée $\mathbf{F}_{t+1}$ et vaut :

$$\mathbf{F}_{t+1} = \begin{bmatrix} 1 & 0 & -ds_t^x \mathrm{sinc}\left(\frac{d\omega_t}{2}\right) \sin\left(\theta_t + \frac{d\omega_t}{2}\right) \\ 0 & 1 & ds_t^x \mathrm{sinc}\left(\frac{d\omega_t}{2}\right) \cos\left(\theta_t + \frac{d\omega_t}{2}\right) \\ 0 & 0 & 1 \end{bmatrix} \tag{45}$$

Remarque 4.1 (Absence de ds_t^y)
Le terme ds_t^y n'apparaît pas dans l'équation 45 car il a été préalablement remplacé par zéro.

4.1.2 Matrice jacobienne de la fonction de mesures par rapport à $\mathbf{x}_t$

Cette matrice représente les variations de $\mathbf{z}_t$ par rapport à de petites variations de $\mathbf{x}_t$. Etant donné que l'on mesure chaque amer indépendamment des autres, on donnera seulement la matrice jacobienne de la fonction de mesure de l'amer (i), notée $\mathbf{H}^{\mathbf{x}}_{t,(i)}$:

$$\mathbf{H}^{\mathbf{x}}_{t,(i)} = \begin{bmatrix} \frac{y_{(i)}-y_t}{(x_{(i)}-x_t)^2+(y_{(i)}-y_t)^2} & \frac{-(x_{(i)}-x_t)}{(x_{(i)}-x_t)^2+(y_{(i)}-y_t)^2} & -1 \\ \mathbf{H}^{\mathbf{x}}_{t,(i)}(2,1) & \mathbf{H}^{\mathbf{x}}_{t,(i)}(2,2) & 0 \end{bmatrix} \tag{46}$$

avec :

$$\left\{ \begin{array}{lcl} \mathbf{H}^{\mathbf{x}}_{t,(i)}(2,1) & = & \frac{z_{(i)}(x_{(i)}-x_t)}{\sqrt{(x_{(i)}-x_t)^2+(y_{(i)}-y_t)^2}\left((x_{(i)}-x_t)^2+(y_{(i)}-y_t)^2+z^2_{(i)}\right)} \\ \mathbf{H}^{\mathbf{x}}_{t,(i)}(2,2) & = & \frac{z_{(i)}(y_{(i)}-y_t)}{\sqrt{(x_{(i)}-x_t)^2+(y_{(i)}-y_t)^2}\left((x_{(i)}-x_t)^2+(y_{(i)}-y_t)^2+z^2_{(i)}\right)} \end{array} \right. \tag{47}$$

4.1.3 Matrice jacobienne de la fonction de mesures par rapport à m

De même, les variations des mesures par rapport à de petites variations de la carte d'amers sont données par $\mathbf{H}^{\mathbf{m}}_{t,(i)}$:

$$\mathbf{H}^{\mathbf{m}}_{t,(i)} = \left[\begin{array}{ccc} \frac{-(y_{(i)}-y_t)}{(x_{(i)}-x_t)^2+(y_{(i)}-y_t)^2} & \frac{x_{(i)}-x_t}{(x_{(i)}-x_t)^2+(y_{(i)}-y_t)^2} & 0 \\ \mathbf{H}^{\mathbf{m}}_{t,(i)}(2,1) & \mathbf{H}^{\mathbf{m}}_{t,(i)}(2,2) & \frac{\sqrt{(x_{(i)}-x_t)^2+(y_{(i)}-y_t)^2}}{(x_{(i)}-x_t)^2+(y_{(i)}-y_t)^2+z^2_{(i)}} \end{array} \right] \tag{48}$$

avec :

$$\left\{ \begin{array}{lcl} \mathbf{H}^{\mathbf{m}}_{t,(i)}(2,1) & = & -\mathbf{H}^{\mathbf{x}}_{t,(i)}(2,1) \\ \mathbf{H}^{\mathbf{m}}_{t,(i)}(2,2) & = & -\mathbf{H}^{\mathbf{x}}_{t,(i)}(2,2) \end{array} \right. \tag{49}$$

4.2 Matrices de variances-covariances utilisées

Dans le cadre du GraphSLAM, toutes les erreurs sont supposées être gaussiennes. Nous décrivons ici les matrices de variances-covariances utilisées. Elles sont au nombre de deux :

1. $\mathbf{Q}_t$ est la matrice de variances-covariances liée à l'erreur effectuée en appliquant la fonction $\mathbf{f}$.
2. $\mathbf{R}_t$ est la matrice de variances-covariances liée à l'erreur effectuée sur la mesure des amers. Les mesures des amers sont supposées indépendantes. $\mathbf{R}_t$ s'écrit donc :

$$\mathbf{R}_t = \mathrm{diag}\left(\mathbf{R}_{t,(1)}, \ldots, \mathbf{R}_{t,(N)}\right)$$

4.2.1 Matrice de variances-covariances liée à la mesure

La matrice de variances-covariances liée à la mesure est directement liée à l'erreur du capteur. Nous supposons pouvoir caractériser cette erreur (sa variance) et qu'elle est identique pour l'ensemble des amers observés. On pose :

$$\forall i \in [1 \ldots N] \quad \mathbf{R}_{t,(i)} = \left[\begin{array}{cc} \sigma^2_\alpha & 0 \\ 0 & \sigma^2_\beta \end{array} \right] \tag{50}$$

Remarque 4.2 (Matrice $\mathbf{R}_{t,(i)}$ diagonale)
Dans ce qui suit, la matrice de variances-covariances de $\mathbf{R}_{t,(i)}$ est prise diagonale. Ceci signifie que dans les simulations de la section 7, les erreurs sur les angles de gisement et d'élévation sont générées indépendament.

Néanmoins, ceci n'est pas nécessairement vrai pour les cas réels. Par exemple, dans le cas d'une image omnidirectionnelle, une erreur sur la localisation d'un point dans l'image (coordonnées en pixels) peut introduire des corrélations entre les angles de gisement et d'élévation (à cause de la géométrie du capteur).

4.2.2 Matrice de variances-covariances liée au modèle

La matrice de variances-covariances due à l'utilisation de la fonction $\mathbf{f}$ pour obtenir une prédiction de l'état du robot à l'instant $t+1$ (équation 2) est moins triviale. Il est nécessaire de prendre en compte deux types d'erreurs :

Erreur $\nu^{\mathbf{f}}_{t+1}$: c'est l'erreur sur le modèle, agissant de manière additive. Nous supposons connaître sa matrice de variances-covariances et qu'elle est indépendante du temps. Nous la notons $\mathbf{Q^f}$.

Erreur $\nu^{\mathbf{u}}_{t+1}$: c'est l'erreur commise sur la mesure des entrées. ds^x_t et dw_t sont obtenus par odométrie et entachés d'erreur. Par ailleurs, nous supposons qu'une vitesse V_y (" petite ") parasite (et non mesurée) peut également être présente entre les instants t et $t+1$ sous forme d'un bruit gaussien. Nous notons $\mathbf{Q^u}$ la matrice de variances-covariances associée. Nous supposons que cette matrice est diagonale (ie. les 3 erreurs dues à la commande sont indépendantes) et constante :[6]

$$\mathbf{Q^u} = \mathrm{diag}\left(\sigma^2_{ds^x}, \sigma^2_{ds^y}, \sigma^2_{d\omega}\right)$$

avec $\sigma^2_{ds^y} \ll \sigma^2_{ds^x}$

Dans l'algorithme général de GraphSLAM, il est supposé que l'erreur sur le modèle intervient de façon linéaire. Ceci est effectivement le cas pour $\nu^{\mathbf{f}}_{t+1}$, mais ça ne l'est pas pour $\nu^{\mathbf{u}}_t$. Une linéarisation est effectuée. On a au final :

$$\mathbf{Q}_{t+1} = \mathbf{J^u_t Q^u J^{u}_t}^T + \mathbf{Q^f} \tag{51}$$

où $\mathbf{J^u_t}$ est la matrice jacobienne $\mathbf{f}$ par rapport à $\mathbf{u}$ (évaluée en $(\mathbf{x}_t, \mathbf{u}_t)$) :

$$\mathbf{J^u_t} = \begin{bmatrix} \mathrm{sinc}\left(\frac{d\omega_t}{2}\right)\cos\left(\theta_t + \frac{d\omega_t}{2}\right) & -\mathrm{sinc}\left(\frac{d\omega_t}{2}\right)\sin\left(\theta_t + \frac{d\omega_t}{2}\right) & \mathbf{J^u_t}(1,3) \\ \mathrm{sinc}\left(\frac{d\omega_t}{2}\right)\sin\left(\theta_t + \frac{d\omega_t}{2}\right) & \mathrm{sinc}\left(\frac{d\omega_t}{2}\right)\cos\left(\theta_t + \frac{d\omega_t}{2}\right) & \mathbf{J^u_t}(2,3) \\ 0 & 0 & 1 \end{bmatrix} \tag{52}$$

avec :

$$\begin{cases} \mathbf{J^u_t}(1,3) &= \frac{2ds^x_t}{d\omega^2_t}\cos\left(\theta_t + \frac{d\omega_t}{2}\right)\cdot\left(\frac{d\omega_t}{2}\cos\left(\frac{d\omega_t}{2}\right) - \sin\left(\frac{d\omega_t}{2}\right)\right) - \frac{ds^x_t}{2}\mathrm{sinc}\left(\frac{d\omega_t}{2}\right)\cdot\sin\left(\theta_t + \frac{d\omega}{2}\right) \\ \mathbf{J^u_t}(2,3) &= \frac{2ds^x_t}{d\omega^2_t}\sin\left(\theta_t + \frac{d\omega_t}{2}\right)\cdot\left(\frac{d\omega_t}{2}\cos\left(\frac{d\omega_t}{2}\right) - \sin\left(\frac{d\omega_t}{2}\right)\right) + \frac{ds^x_t}{2}\mathrm{sinc}\left(\frac{d\omega_t}{2}\right)\cdot\cos\left(\theta_t + \frac{d\omega}{2}\right) \end{cases} \tag{53}$$

[6] C'est une hypothèse pour simplifier les simulations. Elle n'est pas nécessaire dans le cas réel (où il y a d'ailleurs un couplage entre la vitesse de translation mesurée et la vitesse de rotation mesurée).

Remarque 4.3 (sur $\mathbf{Q}_t(3,3)$)
Le terme $\mathbf{Q}_t(3,3)$ est indépendant du temps (car la matrice $\mathbf{Q^f}$ est supposée indépendante du temps et que le terme $(3,3)$ de $\mathbf{J_t^u Q^u J_t^{u^T}}$ l'est aussi). Par ailleurs, l'erreur du modèle concernant l'orientation du robot est uniquement due à l'erreur sur la rotation mesurée. Cette erreur est prise en compte dans la partie $\mathbf{J_t^u Q^u J_t^{u^T}}$. On prendra donc $\mathbf{Q^f}(3,3) = 0$.

4.3 Initialisation

L'algorithme de GraphSLAM nécessite enfin d'effectuer une initialisation. Celle-ci doit permettre d'obtenir un premier point de fonctionnement correct pour permettre la linéarisation des équations de modèle et de mesures. On distingue ici l'initialisation des amers de celle de la trajectoire du robot.

4.3.1 Initialisation de la trajectoire

L'initialisation de la trajectoire peut se faire grâce au modèle. L'intégration de l'odométrie (application de l'équation 10) permet d'obtenir une trajectoire suffisamment précise, au moins sur le court terme. Néanmoins, nous n'initialisons pas l'ensemble de la trajectoire en une seule fois (la variance de l'erreur n'étant pas bornée dans le temps). Plusieurs publications font état du caractère critique de l'angle de cap du robot lors de la résolution du problème de SLAM ([2, 7]). Ainsi, nous choisissons un critère basé sur la variance de l'angle de cap pour choisir le temps d'intégration.

Soit $\sigma^2_{\theta,max}$ la variance maximale tolérée sur l'angle de cap. Le caractère additif des matrices de variances-covariances fait que la variance de l'angle de cap à l'instant t (notée $\sigma^2_{\theta,t}$) s'écrit :

$$\sigma^2_{\theta,t} = \sum_{i=1}^{t} \mathbf{Q}_i(3,3) = t \cdot \mathbf{Q}_1(3,3) \quad (\mathbf{Q}_i(3,3) \text{ est invariant dans le temps}) \tag{54}$$

Au final, on choisit $t = E\left(\frac{\sigma^2_{\theta,max}}{\mathbf{Q}_1(3,3)}\right)$ (où $E(x)$ désigne la partie entière de x).

On peut alors appliquer l'algorithme de GraphSLAM entre les instants 0 et t. Ensuite, la variance du cap à l'instant t diminue grâce à l'application de l'algorithme. Soit $\sigma^2_{\theta-GSLAM,t}$ cette variance. On a nécessairement $\sigma^2_{\theta-GSLAM,t} < \sigma^2_{\theta,t}$, ce qui nous permet d'initialiser un nouveau morceau de la trajectoire entre les instants t et t' où t' est défini par :

$$t' = E\left(\frac{\left(\sigma^2_{\theta,max} - \sigma^2_{\theta-GSLAM,t}\right)}{\mathbf{Q}_1(3,3)}\right) \tag{55}$$

La nouvelle initialisation pour l'application du GraphSLAM entre les instants 0 et t' sera la concaténation du résultat du GraphSLAM entre les instants 0 et t et l'incrément de trajectoire obtenu en intégrant les équations de modèle entre les instants t et t'.

Ainsi, il est possible d'appliquer l'algorithme de GraphSLAM jusqu'à l'instant final.

4.3.2 Initialisation de la position des amers

L'initialisation des amers dans le cas " bearing-only " est assez délicate. Il est en effet impossible d'initialiser un amer avec une seule mesure. Visualiser l'amer depuis au moins deux positions est indispensable. Nous présentons dans un premier temps les équations de base pour initialiser les amers. Nous verrons que ces équations peuvent être mal conditionnées. Ceci nous a amené à utiliser un test pour accepter ou non l'initialisation d'un amer.

4.3.2.1 Equations de base

Supposons qu'un amer (i) soit observé aux instants t_1 et t_2. La première équation du système 11 (page 12) permet d'écrire :

$$\frac{y_{(i)} - y_k}{x_{(i)} - x_k} = \tan(\alpha_{k,(i)} + \theta_k) \qquad (k \in \{t_1, t_2\}) \tag{56}$$

soit :

$$y_{(i)} \cos(\alpha_{k,(i)}+\theta_k) - x_{(i)} \sin(\alpha_{k,(i)}+\theta_k) = y_k \cos(\alpha_{k,(i)}+\theta_k) - x_k \sin(\alpha_{k,(i)}+\theta_k) \qquad (k \in \{t_1, t_2\}) \tag{57}$$

La seconde équation du système 11 permet quant à elle d'écrire :

$$\frac{z_{(i)}}{\sqrt{(x_{(i)} - x_k)^2 + (y_{(i)} - y_k)^2}} = \tan \beta_{k,(i)} \qquad (k \in \{t_1, t_2\}) \tag{58}$$

soit :

$$z_{(i)} = \tan \beta_{k,(i)} \sqrt{(x_{(i)} - x_k)^2 + (y_{(i)} - y_k)^2} \qquad (k \in \{t_1, t_2\}) \tag{59}$$

L'utilisation de l'équation 57 aux instants t_1 et t_2 nous permet de déduire directement les valeurs $x_{(i)}$ et $y_{(i)}$: elles apparaissent de manière linéaire. Ceci n'est pas le cas avec l'équation 59. Pour utiliser cette dernière équation dans le but de calculer $x_{(i)}$ et $y_{(i)}$, il faudrait employer une méthode non linéaire nécessitant une initialisation (ce qui est ce que l'on cherche). Par contre, $z_{(i)}$ peut être déduit directement si $x_{(i)}$ et $y_{(i)}$ sont connus.

En conséquence, la stratégie employée pour l'initialisation des amers est la suivante :

1. Initialiser $x_{(i)}$ et $y_{(i)}$ à l'aide de l'équation 57.
2. Initialiser $z_{(i)}$ à l'aide de l'équation 59 si $x_{(i)}$ et $y_{(i)}$ ont été initialisé.

4.3.2.2 Initialisation de $x_{(i)}$ et $y_{(i)}$

Pour initialiser $x_{(i)}$ et $y_{(i)}$, l'équation 57 est utilisée aux instants t_1 et t_2, ce qui nous donne :

$$\begin{bmatrix} \sin(\alpha_{t_1,(i)} + \theta_{t_1}) & -\cos(\alpha_{t_1,(i)} + \theta_{t_1}) \\ \sin(\alpha_{t_2,(i)} + \theta_{t_2}) & -\cos(\alpha_{t_2,(i)} + \theta_{t_2}) \end{bmatrix} \cdot \begin{bmatrix} x_{(i)} \\ y_{(i)} \end{bmatrix} = \begin{bmatrix} x_{t_1} \sin(\alpha_{t_1,(i)} + \theta_{t_1}) - y_{t_1} \cos(\alpha_{t_1,(i)} + \theta_{t_1}) \\ x_{t_2} \sin(\alpha_{t_2,(i)} + \theta_{t_2}) - y_{t_2} \cos(\alpha_{t_2,(i)} + \theta_{t_2}) \end{bmatrix} \tag{60}$$

En posant $\gamma_1 = \alpha_{t_1,(i)} + \theta_{t_1}$ et $\gamma_2 = \alpha_{t_2,(i)} + \theta_{t_2}$, l'équation 60 nous permet de déduire $x_{(i)}$ et $y_{(i)}$:

$$\begin{bmatrix} x_{(i)} \\ y_{(i)} \end{bmatrix} = \frac{1}{\sin(\gamma_2 - \gamma_1)} \begin{bmatrix} -\cos\gamma_2 & \cos\gamma_1 \\ -\sin\gamma_2 & \sin\gamma_1 \end{bmatrix} \cdot \begin{bmatrix} x_{t_1}\sin\gamma_1 - y_{t_1}\cos\gamma_1 \\ x_{t_2}\sin\gamma_2 - y_{t_2}\cos\gamma_2 \end{bmatrix} \tag{61}$$

4.3.2.3 Bon conditionnement pour l'initialisation de $x_{(i)}$ et $y_{(i)}$

L'équation 61 est définie dès que $\sin(\gamma_2 - \gamma_1) \neq 0$. Ceci signifie que les positions du robot aux instants t_1 et t_2 ne sont pas alignées avec l'amer. Néanmoins, cette condition n'est pas suffisante pour que l'estimation soit bien conditionnée. En effet, les angles γ_1 et γ_2 ne sont pas connus exactement. Il est donc possible que l'intervalle de confiance associé à $\sin(\gamma_2 - \gamma_1)$ contienne zéro. Intuitivement, il faut avoir $\sin(\gamma_2 - \gamma_1)$ assez loin de zéro pour éviter ce problème. Nous proposons dans ce paragraphe de quantifier cette notion intuitive.

Dans ce qui suit, nous considérons qu'un amer est correctement initialisé si l'erreur commise dans l'estimation de ses paramètres est linéaire par rapport à l'erreur sur les données utilisées, à savoir $\gamma_1, \gamma_2, x_{t_1}, x_{t_2}, y_{t_1}, y_{t_2}$. Pour cela, nous supposons que les erreurs sur ces variables sont petites, de sorte à ce que l'approximation linéaire sur les fonctions cosinus et sinus soit valide. Nous pouvons d'abord effectuer les remarques suivantes :

- En ce qui concerne les variables $x_{t_1}, x_{t_2}, y_{t_1}, y_{t_2}$, l'erreur commise intervient naturellement de façon linéaire.
- En ce qui concerne les variables γ_1 et γ_2, l'erreur commise intervient de manière linéaire dans la multiplication des deux matrices de l'équation 61. En revanche, le terme $\frac{1}{\sin(\gamma_2-\gamma_1)}$ ne génère pas nécessairement une erreur linéaire par rapport à celle sur γ_1 et γ_2.

Nous proposons ici de discuter la condition permettant de rendre l'erreur commise sur $\frac{1}{\sin(\gamma_2-\gamma_1)}$ linéaire par rapport à celle commise sur γ_1 et γ_2. Soient γ_1^{approx} et γ_2^{approx} des approximations pour γ_1 et γ_2 ainsi que δ_{γ_1} et δ_{γ_2} les erreurs commises. Nous notons $\sigma^2_{\gamma_1}$ et $\sigma^2_{\gamma_2}$ les variances de δ_{γ_1} et δ_{γ_2} respectivement. Les erreurs commises sur γ_1 et γ_2 étant supposées petites, il en est de même pour $\gamma_2 - \gamma_1$, ce qui permet de linéariser la fonction $\sin(\gamma_2 - \gamma_1)$. Par ailleurs, nous supposons avoir $\sin(\gamma_2^{approx} - \gamma_1^{approx}) \neq 0$. Nous avons donc :

$$\frac{1}{\sin(\gamma_2 - \gamma_1)} = \frac{1}{\sin(\gamma_2^{approx} - \gamma_1^{approx} + \delta_{\gamma_2} - \delta_{\gamma_1})} \tag{62}$$

$$= \frac{1}{\sin(\gamma_2^{approx} - \gamma_1^{approx}) + (\delta_{\gamma_2} - \delta_{\gamma_1})\cos(\gamma_2^{approx} - \gamma_1^{approx})} \tag{63}$$

$$= \frac{1}{\sin(\gamma_2^{approx} - \gamma_1^{approx})} \cdot \frac{1}{1 + (\delta_{\gamma_2} - \delta_{\gamma_1}) \cdot \cot(\gamma_2^{approx} - \gamma_1^{approx})} \tag{64}$$

D'après l'équation 64, l'erreur sur $\frac{1}{\sin(\gamma_2-\gamma_1)}$ sera linéaire en δ_{γ_1} et δ_{γ_2} *si et seulement si* :

$$|(\delta_{\gamma_2} - \delta_{\gamma_1}) \cdot \cot(\gamma_2^{approx} - \gamma_1^{approx})| \ll 1 \tag{65}$$

L'équation 65 est finalement équivalente à $|(\delta_{\gamma_2} - \delta_{\gamma_1})| \ll |\tan(\gamma_2^{approx} - \gamma_1^{approx})|$. Ceci signifie que l'amer considéré doit avoir été vu avec un paralaxe suffisant (idéalement $\pi/2$). Etant donné que les variances concernant l'erreur sur les angles γ_1 et γ_2 sont connues[7] (et supposées consistantes), on acceptera d'initialiser l'amer *si et seulement si* :

$$\boxed{\sqrt{\sigma_{\gamma_1}^2 + \sigma_{\gamma_2}^2} < \tfrac{1}{5} |\tan(\gamma_2^{approx} - \gamma_1^{approx})|} \tag{66}$$

En pratique, on choisira comme points d'initialisation le premier instant où l'amer a été observé, et le premier point satisfaisant l'équation 66. Choisir systématiquement le premier point où l'amer a été observé permet en général de minimiser $\sigma_{\gamma_1}^2$ (ceci n'est pas toujours vrai mais ce choix permet d'éviter de tester tous les couples possibles).

Finalement, une fois qu'on a trouvé deux positions satisfaisant l'équation 66, l'équation 61 est appliquée. Il est même possible d'évaluer d'une façon réaliste la variance de l'erreur commise par cette initialisation.

4.3.2.4 Initialisation de $z_{(i)}$

Une fois que les paramètres $x_{(i)}$ et $y_{(i)}$ sont initialisés, l'équation 59 peut être utilisée à l'instant t_1 et/ou t_2 pour en déduire $z_{(i)}$.

4.3.2.5 Bon conditionnement pour l'initialisation de $z_{(i)}$

L'application de l'équation 59 peut poser problème lorsque $\beta_{k,(i)}$ est trop proche de $\frac{\pi}{2}$. Le facteur posant problème est le $\frac{1}{\cos\beta_{k,(i)}}$ contenu dans $\frac{1}{\tan\beta_{k,(i)}}$. Un raisonnement analogue à celui effectué sur la fonction $\frac{1}{\sin(\gamma_2-\gamma_1)}$ nous a amené à initialiser le paramètre $z_{(i)}$ *si et seulement si* :

$$\boxed{\sigma_\beta < \tfrac{1}{5} \left|\cot\left(\beta_{t_1,(i)}^{approx}\right)\right| \qquad \text{ou} \qquad \sigma_\beta < \tfrac{1}{5} \left|\cot\left(\beta_{t_2,(i)}^{approx}\right)\right|} \tag{67}$$

5 Présentation du SLAM par intervalles

Nous présentons dans cette section la méthode par intervalles implémentée. Cette section ne fait pas d'hypothèses sur les fonctions $\mathbf{f}$ et $\mathbf{h}$ (nous n'utilisons donc pas encore les équations 10 et 11). Les seules hypothèses utilisées sont celles faites dans la section 2.1.

Nous présentons dans un premier paragraphe des généralités sur l'analyse par intervalles (premiers chapitres de [14]). Le paragraphe suivant, quant à lui, traite de l'application de l'analyse par intervalle au problème du SLAM.

[7] $\sigma_{\gamma_1}^2$ et $\sigma_{\gamma_2}^2$ sont connus, soit en utilisant les équation de propagation d'incertitude dans l'intégration du cap du robot, soit après le résultat d'une étape précédente du GraphSLAM.

5.1 Redéfinition des opérateurs classiques

L'analyse par intervalles nécessite de redéfinir l'arithmétique classique. Soit $x \in \mathbb{R}$ une variable réelle. Soit $I_x = [\underline{x}, \overline{x}]$ un intervalle tel que $x \in I_x$. ($I_x \subset \mathbb{R}$ peut être conservatif). La plupart des opérateurs arithmétiques classiques changent dans le cas de l'analyse par intervalles. Ce paragraphe rappelle les principaux opérateurs.

Soient $x \in \mathbb{R}$ et $y \in \mathbb{R}$ deux variables réelles ainsi que I_x et I_y les intervalles correspondants. Les opérateurs de base sont définis comme suit :

Addition : si $z = x + y$, alors on a $z \in [\underline{z}, \overline{z}]$ avec

$$\left\{ \begin{array}{rcl} \underline{z} & = & \underline{x} + \underline{y} \\ \overline{z} & = & \overline{x} + \overline{y} \end{array} \right. \tag{68}$$

Soustraction : si $z = x - y$, alors on a $z \in [\underline{z}, \overline{z}]$ avec

$$\left\{ \begin{array}{rcl} \underline{z} & = & \underline{x} - \overline{y} \\ \overline{z} & = & \overline{x} - \underline{y} \end{array} \right. \tag{69}$$

Multiplication : si $z = x.y$, alors on a $z \in [\underline{z}, \overline{z}]$ avec

$$\left\{ \begin{array}{rcll} \underline{z} & = & \underline{x}.\underline{y} & \\ \overline{z} & = & \overline{x}.\underline{y} & \text{si} \quad \underline{x} > 0 \quad \text{et} \quad \underline{y} > 0 \\ \underline{z} & = & \overline{x}.\underline{y} & \\ \overline{z} & = & \overline{x}.\overline{y} & \text{si} \quad \underline{x} > 0 \quad \text{et} \quad \underline{y} < 0 \quad \text{et} \quad \overline{y} > 0 \\ & \ldots & & \end{array} \right. \tag{70}$$

La multiplication peut finalement se résumer par :

$$\left\{ \begin{array}{rcl} \underline{z} & = & \min\left(\underline{x}.\underline{y}, \underline{x}.\overline{y}, \overline{x}.\underline{y}, \overline{x}.\overline{y}\right) \\ \overline{z} & = & \max\left(\underline{x}.\underline{y}, \underline{x}.\overline{y}, \overline{x}.\underline{y}, \overline{x}.\overline{y}\right) \end{array} \right. \tag{71}$$

Division : si $z = \frac{x}{y}$, alors on a $z \in [\underline{z}, \overline{z}]$ avec

$$\left\{ \begin{array}{rcll} \underline{z} & = & \min\left(\underline{x}/\underline{y}, \underline{x}/\overline{y}, \overline{x}/\underline{y}, \overline{x}/\overline{y}\right) & \\ \overline{z} & = & \max\left(\underline{x}/\underline{y}, \underline{x}/\overline{y}, \overline{x}/\underline{y}, \overline{x}/\overline{y}\right) & \text{si} \quad \underline{y}.\overline{y} > 0 \\ \underline{z} & = & -\infty & \\ \overline{z} & = & +\infty & \text{sinon} \end{array} \right. \tag{72}$$

Les autres fonctions classiques (trigonométriques, exponentielle, logarithme,...) peuvent être redéfinies en terme d'intervalles. Une fonction quelconque dont l'argument est un intervalle sera écrite sous la forme $f \circ g \circ h \circ ...$ de sorte à n'appliquer que des fonctions élémentaires ou connues.

Remarque 5.1 (Importance de l'écriture formelle)
La façon d'écrire une fonction a une importance toute particulière dans le cas de l'analyse par intervalles. Ceci est dû au fait que l'analyse par intervalles cherche toujours des intervalles consistants. Ainsi, si $z = x + y$ et que l'on cherche I_z à partir de I_x et I_y, on ne pourra pas retrouver l'intervalle I_x original à partir du I_z calculé, de I_y et de $x = z - y$.

Considérons par exemple la fonction $x \mapsto x^2 - x$ que l'on va appliquer avec $x \in I_x = [0\ 1]$. Nous distinguons pour cet exemple deux cas :

1. *On écrit la fonction $x \mapsto x^2 - x$. Dans ce cas, l'intervalle résultant de $x \mapsto x^2$ est $[0\ 1]$, et celui de $x \mapsto -x$ est $[-1\ 0]$. L'intervalle final de la somme est donc $[-1\ 1]$.*
2. *On écrit la fonction $x \mapsto x(x-1)$. Dans ce cas, l'intervalle résultant de $x \mapsto x - 1$ est $[-1\ 0]$. Celui du produit de $x - 1$ par x est donc de $[-1\ 0]$.*

Cet exemple montre bien que la façon d'écrire la fonction est très importante. Ainsi, une " mauvaise écriture " de la fonction conduira à une évaluation trop conservatrice. L'analyse par intervalles nous assure que le résultat de la fonction est dans l'intervalle d'arrivée, mais il ne nous assure pas que cet intervalle est le plus petit possible.
On pourra de plus remarquer que si on prend $I_x = [-1\ 1]$, alors la seconde écriture est plus conservatrice. Ainsi, il n'existe pas forcément une meilleure façon d'écrire la fonction pour tous les intervalles possibles.[8]

Dans le cadre de l'analyse par intervalles, on sera souvent amené à manipuler la taille des intervalles ($\overline{x} - \underline{x}$ pour un intervalle I_x). Dans la suite, cette quantité sera notée $t(I_x)$.

5.2 Application au SLAM

Nous présentons ici l'approche générale utilisée pour résoudre le problème du SLAM à l'aide des outils d'analyse par intervalles. Nous présentons d'abord la méthode permettant de passer de l'instant t à l'instant $t + 1$. Ensuite, nous présentons la méthode utilisée pour traiter l'ensemble d'une trajectoire.

5.2.1 Traitement local : passage de t à $t + 1$

Nous présentons ici la méthode permettant de trouver un intervalle pour les variables d'état (robot et amer) à l'instant t lorsqu'on possède un intervalle de ces variables à l'instant $t - 1$. Nous supposons donc disposer des informations suivantes :

$I_{\mathbf{X}_t}$: une boite contenant $\mathbf{X}_t$.

$I_{\mathbf{u}_t}$: une boite contenant $\mathbf{u}_t$.

$I_{\mathbf{m}}$: une boite contenant $\mathbf{m}$.

$I_{\nu^{\mathbf{f}}_{t+1}}$: une boite contenant $\nu^{\mathbf{f}}_{t+1}$.

$I_{\nu^{\mathbf{z}}_{t+1}}$: une boite contenant $\nu^{\mathbf{z}}_{t+1}$.

[8] Dans ce cas, on peut choisir de calculer l'intersection des résultats avec les différentes écritures possibles.

Remarque 5.2 (Terminologie)
On employera dans la suite le terme " boîte " pour désigner l'ensemble des intervalles d'une variable multidimensionnelle. Néanmoins, il arrivera que l'on conserve la dénomination classique " intervalle ".

5.2.1.1 Obtention d'un *a priori* pour l'état du robot

La première étape de l'algorithme est de trouver un intervalle *a priori* concernant l'état du robot à l'instant $t+1$. Pour cela, nous appliquons la fonction $\mathbf{f}$ à $I_{\mathbf{X}_t}$, $I_{\mathbf{u}_t}$ et $I_{\nu^{\mathbf{f}}_{t+1}}$. Cette étape ne pose pas de difficulté particulière, étant donné que nous supposons connaître une forme analytique de la fonction $\mathbf{f}$.

Le résultat de cette opération est une boite $I_{\mathbf{X}_{t+1}}$ contenant $\mathbf{X}_{t+1}$.

5.2.1.2 Utilisation des mesures à l'instant $t+1$

Introduction au problème

L'utilisation du vecteur de mesures à l'instant $t+1$ est plus délicate. En effet, nous disposons ici d'une fonction $\mathbf{h} : \mathbb{R}^n \times \mathbb{R}^l \rightarrow \mathbb{R}^q$ dont les arguments sont l'état du robot et de la carte et dont la sortie est le vecteur de mesures. Il s'agit d'un problème inverse. Il n'est pas possible de déduire de la connaissance de $\mathbf{y}$ un intervalle *a posteriori* pour l'état du robot et de la carte étant donné que la fonction $\mathbf{h}$ n'est pas supposée inversible (pire, on aura en général $q < n+l$). Une connaissance *a priori* de l'état du robot et des amers est indispensable.

Dans ce qui suit, on notera $\tilde{\mathbf{x}}$ la concaténation des vecteurs $\mathbf{x}$ et $\mathbf{m}$. La dimension de ce vecteur est $n+l$.

Hypothèses effectuées

Pour résoudre ce problème, nous effectuons les hypothèses suivantes :

- Posséder un intervalle *a priori* concernant l'état du robot et des amers.
- Soit $\mathbf{z} = \mathbf{h}(\tilde{\mathbf{x}})$ avec $\mathbf{z} = [z_1, \ldots, z_q]^T$ et $\tilde{\mathbf{x}} = [\tilde{x}_1, \ldots, \tilde{x}_{n+l}]^T$
- On suppose que la fonction $\mathbf{h}$ permette d'écrire :

$$\boxed{\forall i \in [1 \ldots n+l] \; \exists \left(j \in [1 \ldots q] \text{ et } g_i^j : \mathbb{R}^{n+l} \mapsto \mathbb{R} \right) \quad \text{tq } \tilde{x}_i = g_i^j(z_j, \tilde{x}_1, \ldots, \tilde{x}_{i-1}, \tilde{x}_{i+1}, \ldots, \tilde{x}_{n+l})} \tag{73}$$

Les fonctions g_i^j ne sont pas nécessairement continues mais doivent pouvoir permettre d'être **évaluées analytiquement** pour des intervalles donnés.
Cela revient à supposer que lorsqu'on fixe toutes les variables d'entrées de chaque $\mathbf{h}$ sauf une, on est capable de retrouver la variable non fixée (ceci devant pouvoir se faire sur l'ensemble des composantes de l'entrée). Intuitivement, la condition 73 peut être vue comme une " condition d'observabilité " des variables $\tilde{x}_i$.

Résolution de $\mathbf{h}(\tilde{\mathbf{x}}) = \mathbf{z}$

Nous présentons ici une méthode permettant de résoudre une équation du type $\mathbf{h}(\tilde{\mathbf{x}}) = \mathbf{z}$ où $\mathbf{h}$ respecte l'hypothèse précédente. Pour cela, nous supposons posséder un intervalle $I_{\mathbf{z}}$ pour $\mathbf{z}$ et un intervalle *a priori* $I_{\tilde{\mathbf{x}}}$ pour $\tilde{\mathbf{x}}$.
Nous cherchons dans un premier temps à trouver un nouvel intervalle pour la $i^{\text{ème}}$ ($i \in \{1 \dots n+l\}$) composante de $\tilde{\mathbf{x}}$ (ie. $\tilde{x}_i$) :

1. Soient $\{g_i^1, \dots, g_i^{\gamma_i}\}$ l'ensemble des fonctions g_i^j telles que $\tilde{x}_i = g_i^j(z_j, \tilde{x}_1, \dots, \tilde{x}_{i-1}, \tilde{x}_{i+1}, \dots, \tilde{x}_{\alpha_k})$ ($j \in \{1 \dots \gamma_i\}$). γ_i désigne le nombre de fonctions permettant de retrouver $\tilde{x}_i$ en fonction des mesures et autres variables d'état.
2. Soient $I_{\tilde{x}_i}$ l'intervalle *a priori* de $\tilde{x}_i$, $I_{\tilde{x}_k}$ l'intervalle *a priori* de $\tilde{x}_k$ ($k \in \{1 \dots n+l\} \setminus \{i\}$) et I_{z_j} l'intervalle de z_j.
3. Soient $I_{\tilde{x}_i}^j$ les intervalles obtenus après évaluation de g_i^j avec les $I_{\tilde{x}_l}$ ($k \neq i$) et I_{z_j} ($j \in \{1 \dots \gamma_i\}$).
4. On peut en déduire un intervalle pour $\tilde{x}_i$:

$$I_{\tilde{x}_i} \leftarrow \left(\bigcap_{j=1}^{\gamma_i} I_{\tilde{x}_i}^j \right) \bigcap I_{\tilde{x}_i} \tag{74}$$

Nous pouvons alors déduire séquentiellement un intervalle pour toutes les composantes de $\tilde{\mathbf{x}}$. Ceci nous donnera un nouvel intervalle pour le vecteur $\tilde{\mathbf{x}}$. Néanmoins, après ce processus, la solution peut être améliorée. Par exemple, les changements obtenus pour les composantes $\tilde{x}_2, \dots \tilde{x}_{n+l}$ peuvent permettre d'améliorer la solution pour $\tilde{x}_1$. De plus, la nouvelle solution $\tilde{\mathbf{x}}$ peut également permettre d'améliorer l'intervalle associé à $\mathbf{z}$ (amélioration qui peut être exploitée pour améliorer à nouveau $\tilde{\mathbf{x}}$). En conséquence, nous proposons l'algorithme d'écrit dans le tableau (2).

Théorème 5.1 (Convergence et consistance de l'algorithme précédent)
Si les intervalles a priori *sont consistants*
Alors l'algorithme présenté pour la résolution de $\mathbf{h}(\tilde{\mathbf{x}}) = \mathbf{z}$ *termine toujours et fournit un résultat consistant.*

Démonstration du théorème 5.1 :
▲

Convergence : A chaque itération le test de convergence porte sur la taille des variables $decr_i$. Soit $t(I_{\tilde{x}}^i)[j]$ la suite des $t(I_{\tilde{x}}^i)$ et $decr_i[j]$ la suite des $decr_i$ associée, j désignant le numéro d'itération de l'algorithme.
Pour i fixé, la suite $t(I_{\tilde{x}}^i)[j]$ est (par construction de $I_{\tilde{x}}^i$) strictement décroissante. Par ailleurs, elle est par définition positive.
La suite $t(I_{\tilde{x}}^i)[j]$ est donc convergente. En conséquence, elle vérifie la propriété suivante :

$$\forall \epsilon > 0 \ \exists N_0 \text{ tq } \forall (m,n) \in \mathbb{N}^2, \left((m \geq N_0 \wedge n \geq N_0) \Rightarrow \left| t(I_{\tilde{x}}^i)[m] - t(I_{\tilde{x}}^i)[n] \right| < \epsilon \right)$$

```
Fixer ε_1, ..., ε_{n+l}                          # Seuils pour tester la convergence des composantes
conv_1, ..., conv_{n+l} ← 0                      # Variables pour tester si une composante a convergé
conv ← 0
Tant que conv = 0 faire
|    I_z ← I_z ∩ h(I_x̃)                          # Réduction de z dès que possible grâce à l'a priori de x̃
|    Pour i = 1...n+l faire
|    |    I_{x̃_i}^{ancien} ← I_{x̃_i}
|    |    I_{x̃_i} ← mise à jour par équation 74
|    |    decr_i ← t(I_{x̃_i}^{ancien}) − t(I_{x̃_i})   # decr_i < 0 désigne la différence entre la longueur
|    |                                             # du nouvel intervalle et la longueur de l'ancien
|    |    Si decr_i < ε_i alors
|    |    |    conv_i ← 1
|    |    Sinon
|    |    |    conv_i ← 0
|    |    Fin si
|    Fin pour
|    conv ← ∏_{i=1}^{n+l} conv_i
Fin tant que
Retourner I_{x̃_1}, ..., I_{x̃_{n+l}}
```

TAB. 2 – Algorithme de résolution de $\mathbf{h}(\tilde{\mathbf{x}}) = \mathbf{z}$

Ainsi, pour ϵ_i fixé, il existe un N_i tel que $\left|t(I^i_{\tilde{x}})[N_i+1] - t(I^i_{\tilde{x}})[N_i]\right| < \epsilon_i$, soit $decr_i[N_i+1] < \epsilon_i$, ce qui rend vrai le test sur la convergence de $decr_i$.
En appliquant le raisonnement précédent sur toutes les composantes de $\tilde{\mathbf{x}}$, on en déduit qu'il existe une itération telle que le test de fin soit vrai. Ceci démontre la convergence de l'algorithme.

Consistance : Si les intervalles *a priori* sont consistants, alors le résultat renvoyé par l'algorithme sera consistant par construction. En effet, aucune approximation n'est faite dans l'analyse par intervalle, les encadrements étant toujours effectués de façon pessimiste.

■

Exemple de résolution de $\mathbf{h}(\tilde{\mathbf{x}}) = \mathbf{z}$:
Soit la fonction $\mathbf{h}$ définie par :

$$\begin{array}{rccc} \mathbf{h}: & \mathbb{R}^3 & \longrightarrow & \mathbb{R}^2 \\ & \tilde{\mathbf{x}} = \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \\ \tilde{x}_3 \end{bmatrix} & \longmapsto & \mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} \frac{\tilde{x}_1}{\tilde{x}_2} \\ \tilde{x}_1 + \tilde{x}_3 \end{bmatrix} \end{array}$$

Les intervalles *a priori* concernant $\tilde{\mathbf{x}}$ ainsi que $\mathbf{z}$ sont :

- $I_{\tilde{x}_1} = [-10; 10]$
- $I_{\tilde{x}_2} = [0; 15]$
- $I_{\tilde{x}_3} = [1; 2]$
- $I_{z_1} = [-15; 1]$
- $I_{z_2} = [3; 5]$

On peut définir les fonctions g_1^1, g_1^2, g_2^1 et g_3^2 permettant de vérifier l'hypothèse 73 :

$$g_1^1 : \begin{array}{ccc} \mathbb{R}^2 & \longrightarrow & \mathbb{R} \\ (z_1, \tilde{x}_2) & \longmapsto & \tilde{x}_1 = z_1 \cdot \tilde{x}_2 \end{array} \quad , \quad g_1^2 : \begin{array}{ccc} \mathbb{R}^2 & \longrightarrow & \mathbb{R} \\ (z_2, \tilde{x}_3) & \longmapsto & \tilde{x}_1 = z_2 - \tilde{x}_3 \end{array}$$

$$g_2^1 : \begin{array}{ccc} \mathbb{R}^2 & \longrightarrow & \mathbb{R} \\ (z_1, \tilde{x}_1) & \longmapsto & \tilde{x}_2 = \frac{\tilde{x}_1}{z_1} \end{array} \quad , \quad g_3^2 : \begin{array}{ccc} \mathbb{R}^2 & \longrightarrow & \mathbb{R} \\ (z_2, \tilde{x}_1) & \longmapsto & \tilde{x}_3 = z_2 - \tilde{x}_1 \end{array}$$

L'exécution de l'algorithme est présentée dans le tableau 3. Il permet d'améliorer les variables $\tilde{x}_1$ et $\tilde{x}_2$, mais pas $\tilde{x}_3$. On peut remarquer l'importance de la réduction des intervalles de $\mathbf{z}$: supprimer l'incertitude sur le signe de $\tilde{x}_1$ a permis d'améliorer $\tilde{x}_2$ par la suite. On obtient au final :

- $I_{\tilde{x}_1} = [1; 4]$
- $I_{\tilde{x}_2} = [1; 15]$
- $I_{\tilde{x}_3} = [1; 2]$

5.2.2 Algorithme global

Nous avons présenté dans la partie précédente comment mettre à jour les intervalles des variables d'état entre les instants t et $t+1$. Ceci permet d'obtenir des intervalles consistants à chaque instant. Néanmoins, certaines variables sont observées à divers instants (les amers). Ainsi, si un amer (i) est observé aux instants t_1 et t_2 ($t_2 > t_1$) et que la boîte contenant l'amer (i) est plus petite en t_2, alors la boite trouvée en t_2 peut être utilisée pour améliorer la position du robot en t_1 (et peut être donc aussi la localisation des autres amers vus en t_1).

En conséquence, l'algorithme global revient à appliquer l'algorithme local de l'instant initial jusqu'à l'instant final. Ensuite, l'opération est répétée avec les nouveaux intervalles *a priori*, et ce jusqu'à la convergence des intervalles.

6 Mise en application du SLAM par intervalles

Nous présentons dans cette section la mise en application du SLAM par intervalles. Nous présentons dans un premier temps les fonctions utilisées. Ensuite, le paragraphe 6.2 traite de l'initialisation des amers. Nous présentons dans le paragraphe 6.3 le problème du choix de l'orientation des axes ainsi que la solution proposée.

Etape	**Action effectuée**	$I_{\tilde{x}_1}$	$I_{\tilde{x}_2}$	$I_{\tilde{x}_3}$	I_{z_1}	I_{z_2}
0	Initialisation	$[-10;10]$	$[0;15]$	$[1;2]$	$[-15;1]$	$[3;5]$
1	Réduction des intervalles de $\mathbf{z}$					
1a	Réduction de $z_1 = \tilde{x}_1/\tilde{x}_2$: pas d'amélioration	$[-10;10]$	$[0;15]$	$[1;2]$	$[-15;1]$	$[3;5]$
1b	Réduction de $z_2 = \tilde{x}_1 + \tilde{x}_3$: pas d'amélioration	$[-10;10]$	$[0;15]$	$[1;2]$	$[-15;1]$	$[3;5]$
2	Réduction des intervalles de $\tilde{\mathbf{x}}$					
2a	Réduction de $\tilde{x}_1 = z_1 \cdot \tilde{x}_2$: pas d'amélioration	$[-10;10]$	$[0;15]$	$[1;2]$	$[-15;1]$	$[3;5]$
2b	Réduction de $\tilde{x}_1 = z_2 - \tilde{x}_3$: $I_{\tilde{x}_1} = [-10;10] \cap [1;4]$	$[1;4]$	$[0;15]$	$[1;2]$	$[-15;1]$	$[3;5]$
2c	Réduction de $\tilde{x}_2 = \tilde{x}_1/z_1$: pas d'amélioration	$[1;4]$	$[0;15]$	$[1;2]$	$[-15;1]$	$[3;5]$
2d	Réduction de $\tilde{x}_3 = z_2 - \tilde{x}_1$: pas d'amélioration	$[1;4]$	$[0;15]$	$[1;2]$	$[-15;1]$	$[3;5]$
3	Réduction des intervalles de $\mathbf{z}$					
3a	Réduction de $z_1 = \tilde{x}_1/\tilde{x}_2$: $I_{z_1} = [-15;1] \cap [\frac{1}{15};+\infty]$	$[1;4]$	$[0;15]$	$[1;2]$	$[\frac{1}{15};1]$	$[3;5]$
3b	Réduction de $z_2 = \tilde{x}_1 + \tilde{x}_3$: pas d'amélioration	$[1;4]$	$[0;15]$	$[1;2]$	$[\frac{1}{15};1]$	$[3;5]$
4	Réduction des intervalles de $\tilde{\mathbf{x}}$					
4a	Réduction de $\tilde{x}_1 = z_1 \cdot \tilde{x}_2$: pas d'amélioration	$[1;4]$	$[0;15]$	$[1;2]$	$[\frac{1}{15};1]$	$[3;5]$
4b	Réduction de $\tilde{x}_1 = z_2 - \tilde{x}_3$: pas d'amélioration	$[1;4]$	$[0;15]$	$[1;2]$	$[\frac{1}{15};1]$	$[3;5]$
4c	Réduction de $\tilde{x}_2 = \tilde{x}_1/z_1$: $I_{\tilde{x}_2} = [0;15] \cap [1;60]$	$[1;4]$	$[1;15]$	$[1;2]$	$[\frac{1}{15};1]$	$[3;5]$
4d	Réduction de $\tilde{x}_3 = z_2 - \tilde{x}_1$: pas d'amélioration	$[1;4]$	$[1;15]$	$[1;2]$	$[\frac{1}{15};1]$	$[3;5]$
5	Réduction des intervalles de $\mathbf{z}$					
5a – b	Réduction de $\mathbf{z}$: aucune composante améliorée	$[1;4]$	$[1;15]$	$[1;2]$	$[\frac{1}{15};1]$	$[3;5]$
6	Réduction des intervalles de $\tilde{\mathbf{x}}$					
6a – d	Réduction de $\tilde{\mathbf{x}}$: aucune composante améliorée	$[1;4]$	$[1;15]$	$[1;2]$	$[\frac{1}{15};1]$	$[3;5]$
7	Convergence de l'algorithme					

TAB. 3 – Exemple de résolution de $\mathbf{h}(\tilde{\mathbf{x}}) = \mathbf{z}$

6.1 Fonctions utilisées

Nous présentons dans ce paragraphe les fonctions utilisées dans le cadre du SLAM par intervalles. Dans un premier temps, nous présentons la fonction de modèle. Ensuite, nous détaillons le traitement effectué pour la fonction de mesures.

6.1.1 Equation de modèle

A chaque itération, l'état *a priori* est obtenu en utilisant l'équation de modèle 10, les intervalles obtenus pour l'état à l'instant précédent et les intervalles concernant les 3 entrées et l'erreur possible de modèle (les intervalles concernant le modèle et les entrées sont des données, que l'on supposera consistantes). En ce qui concerne les entrées, nous supposons que :

- $ds_t^x \in ds_{t,odo}^x + I_{ds^x}$, où $ds_{t,odo}^x$ est mesuré par l'odométrie et I_{ds^x} est l'intervalle possible pour l'erreur d'odométrie.

– $ds_t^y \in I_{ds^y}$. On suppose qu'il peut exister une petite vitesse transversale due à un léger glissement transversal possible. I_{ds^y} est donc centré. Typiquement, on supposera la contrainte de non-holonomie pratiquement réalisée. On prendra donc dans les simulations $t\left(I_{ds_x}\right) \gg t\left(I_{ds_y}\right)$.
– $d\omega_t \in d\omega_{t,odo} + I_{d\omega}$, où $d\omega_{t,odo}$ est la mesure d'incrément angulaire et $I_{d\omega}$ l'intervalle possible pour l'erreur commise.

Remarque 6.1 (Intervalles des erreurs constants)
Tout comme pour le cas du GraphSLAM, nous considérons ici que les intervalles possibles concernant les erreurs de modèle et les erreurs de mesure des entrées sont constants.
Cette hypothèse n'est pas nécessaire mais permet de simplifier les conditions de simulation.

Par ailleurs, si une itération de l'algorithme sur l'ensemble de la trajectoire a déjà été effectuée, nous calculons l'intersection entre la prédiction effectuée et l'intervalle calculé à l'itération précédente.

6.1.2 Equation de mesure

En ce qui concerne l'équation de mesures, nous utilisons le système d'équations 11 rappelé ci-après :

$$\left\{ \begin{array}{rcl} \alpha_{t,(i)} & = & \text{arctan2}\left(\begin{array}{ccc} y_{(i)} - y_t & , & x_{(i)} - x_t \end{array} \right) - \theta_t \\ \beta_{t,(i)} & = & \arctan\left(\frac{z_{(i)}}{\sqrt{(x_{(i)}-x_t)^2+((y_{(i)}-y_t)^2}} \right) \end{array} \right. \tag{75}$$

6.1.2.1 Traitement de la première équation de mesure

Nous donnons dans ce paragraphe les fonctions utilisées pour retrouver l'état du robot et celui de l'amer (i) en utilisant la première équation du système 75 :

$$\left\{ \begin{array}{rcl} x_t & = & x_{(i)} - \left(y_{(i)} - y_t\right) \cdot \cot\left(\theta_t + \alpha_{t,(i)}\right) \\ y_t & = & y_{(i)} - \left(x_{(i)} - x_t\right) \cdot \tan\left(\theta_t + \alpha_{t,(i)}\right) \\ \theta_t & = & \text{atan2}\left(\begin{array}{ccc} y_{(i)} - y_t & , & x_{(i)} - x_t \end{array} \right) - \alpha_{t,(i)} \\ x_{(i)} & = & x_t + \left(y_{(i)} - y_t\right) \cdot \cot\left(\theta_t + \alpha_{t,(i)}\right) \\ y_{(i)} & = & y_t + \left(x_{(i)} - x_t\right) \cdot \tan\left(\theta_t + \alpha_{t,(i)}\right) \end{array} \right. \tag{76}$$

En appliquant le système 76 avec tous les amers (au nombre de N), on déduit $3N$ équations permettant de mettre à jour l'état du robot (N équations pour chaque composante) avec les mesures $\alpha_{t,(i)}$ ($i \in [1 \dots N]$) et 2 équations pour mettre à jour chaque amer (une équation par composante).

6.1.2.2 Traitement de la seconde équation de mesure

Le traitement de la seconde équation du système 75 permet d'écrire :

$$\left\{ \begin{array}{rcl} z_{(i)} & = & \tan\beta_{t,(i)}\sqrt{(x_{(i)}-x_t)^2+((y_{(i)}-y_t)^2} \\ \left(x_{(i)}-x_t\right)^2 & = & \left(\cot\beta_{t,(i)}z_{(i)}\right)^2-\left(y_{(i)}-y_t\right)^2 \\ \left(y_{(i)}-y_t\right)^2 & = & \left(\cot\beta_{t,(i)}z_{(i)}\right)^2-\left(x_{(i)}-x_t\right)^2 \end{array} \right. \tag{77}$$

La première équation du système 77 permet de déduire un intervalle concernant l'altitude de l'amer (i). Les deux équations suivantes, quant à elles, permettent de déduire des intervalles concernant les valeurs absolues des quantités $x_{(i)}-x_t$ et $y_{(i)}-y_t$. Ceci permet au final d'améliorer les 4 variables x_t, y_t, $x_{(i)}$ et $y_{(i)}$. On peut enfin remarquer que les expressions $\left(\cot\beta_{t,(i)}z_{(i)}\right)^2-\left(y_{(i)}-y_t\right)^2$ et $\left(\cot\beta_{t,(i)}z_{(i)}\right)^2-\left(x_{(i)}-x_t\right)^2$ peuvent se factoriser. Les versions factorisées et développées sont toutes les deux évaluées afin de prendre en compte la remarque 5.1.

6.2 Initialisation des amers

Tout comme pour l'approche probabiliste, l'initialisation des amers fait l'objet d'une attention particulière. Nous devons nous assurer que l'intervalle initial est consistant. Il est par ailleurs impossible d'initialiser les amers avec des intervalles infinis (le systèmes 76 et 77 ne permettraient ni d'améliorer l'estimation de l'état du robot, ni celle de la position des amers).

Nous utilisons une approche assez semblable à celle présentée en 4.3.2. Nous initialisons dans un premier temps les variables $x_{(i)}$ et $y_{(i)}$. Pour cela, nous utilisons deux instants où l'amer est observé (t_1 et t_2) ainsi que l'équation 61 rappelée ci-après : [9]

$$\left[\begin{array}{c} x_{(i)} \\ y_{(i)} \end{array}\right] = \frac{1}{\sin(\gamma_2-\gamma_1)} \left[\begin{array}{cc} -\cos\gamma_2 & \cos\gamma_1 \\ -\sin\gamma_2 & \sin\gamma_1 \end{array}\right] \cdot \left[\begin{array}{c} x_{t_1}\sin\gamma_1 - y_{t_1}\cos\gamma_1 \\ x_{t_2}\sin\gamma_2 - y_{t_2}\cos\gamma_2 \end{array}\right] \tag{78}$$

Tout comme pour le cas du GraphSLAM, l'instant t_1 sera celui de la première observation de l'amer. L'instant t_2, quant à lui, sera la premier instant pour lequel l'intervalle de $\sin(\gamma_2-\gamma_1)$ ne contient pas zéro. Ainsi, l'équation 78 peut être appliquée avec des intervalles et nous donnera alors une boîte consistante contenant la position réelle de l'amer (i).

Remarque 6.2 (Taille de la boîte d'initialisation d'un amer)
Lorsqu'un amer est initialisé, la boîte associée peut être très grande. En effet $\sin(\gamma_2-\gamma_1)$ *peut être très proche de zéro (le critère retenu étant "* ne contient pas zéro *"). Ceci ne pose pas de problème car l'approche par intervalles ne fait pas d'hypothèse sur la taille des intervalles. Les systèmes 76 et 77 permettront de réduire considérablement la taille de la boîte des amers.*

Enfin, une fois les composantes $x_{(i)}$ et $y_{(i)}$ initialisées, la composante verticale de l'amer (i) peut être initialisée en utilisant la première équation du système 77.

[9] Pour rappel, nous avons $\gamma_1 = \alpha_{t_1,(i)} + \theta_{t_1}$ et $\gamma_2 = \alpha_{t_2,(i)} + \theta_{t_2}$

6.3 Choix de l'orientation des axes

Nous avons présenté dans les paragraphes précédents l'application basique de l'algorithme de SLAM par intervalles. Nous proposons ici d'étudier les effets de l'orientation des axes sur la qualité du résultat.

6.3.1 Remarque générale

Idéalement, les qualités d'un algorithme ne devraient pas dépendre du choix du repère initial. Dans le cas du SLAM par intervalles, on peut facilement se rendre compte que le fait de déplacer l'origine ne changera rien au problème ; le résultat final sera simplement translaté.

Malheureusement, il n'en est pas de même pour une rotation du repère. La position du robot (en x et en y) est en effet représentée par I_x et I_y, ce qui correspond géométriquement à un rectangle parallèle aux axes. Cette représentation ne tient pas compte de certaines corrélations qui peuvent exister et qui font qu'il existe une solution optimale plus complexe. Ainsi, la solution optimale (dans le sens où **tous les points** de la surface sont susceptibles d'être la position du robot) n'est pas forcément un rectangle.

Supposons par exemple qu'après plusieurs itérations la solution optimale concernant la position du robot (ou d'un amer) soit un polytope (représenté en rouge sur la figure 4). L'implémentation basique du SLAM par intervalles va retourner un intervalle I_x ainsi qu'un intervalle I_y. Au mieux, cela correspondra au rectangle vert présenté sur la figure 4 (au mieux car les seuils d'arrêt utilisés peuvent faire que l'algorithme s'arrêtera avant la solution optimale). Néanmoins, on peut voir que si les axes avaient été tournés à 45 degrés, on aurait pu obtenir une " meilleure " solution (au sens de l'aire du rectangle englobant). Cette dernière est présentée en bleu sur la figure 4.

Ainsi, la figure 4 met en évidence le fait qu'il existe une façon optimale d'orienter les rectangles contenant la position des amers et du robot. Nous allons mettre en évidence ce phénomène à l'aide d'un exemple concret.

6.3.2 Exemple 1 : modèle d'évolution du robot

Considérons désormais un robot dont la position initiale est exactement connue et est l'origine du repère. De plus, l'angle de cap initial est pris à une valeur nulle. Supposons par ailleurs que le robot roule avec une vitesse de rotation et une vitesse de translation constantes pendant 1 seconde. Nous connaissons seulement des intervalles pour ces vitesses :

- $V_1^x \in [3, 4]$ m.s^{-1}
- $V_1^y \in [0, 0]$ m.s^{-1} (condition de non holonomie parfaitement réalisée)
- $\omega \in [-2, 2]$ deg.s^{-1}

Nous proposons dans un premier temps de visualiser graphiquement 3 lieux (figure 5) :

- Le lieu réel des positions possibles du robot (en rouge).
- Le rectangle obtenu en appliquant l'analyse par intervalles à la fonction d'évolution du robot (en vert). Celui-ci est aligné sur la grille x, y.

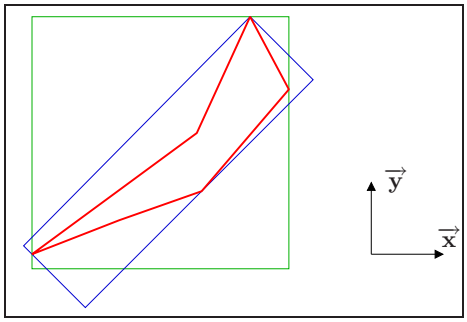

FIG. 4 – Non optimalité de la représentation par rectangles — En rouge : un polytope contenant 100% des solutions possibles et supposé optimal — En vert : le plus petit rectangle aligné selon les axes du repère et contenant le polytope — En bleu : le plus petit rectangle tourné à 45 degrés et contenant le polytope

– Le rectangle obtenu en appliquant l'analyse par intervalles à la fonction d'évolution du robot (en bleu), mais en le calculant sur une grille orientée à 45 degrés par rapport à la grille x, y. Pour cela, nous avons appliqué la théorie classique avec un angle de cap initial à 45 degrés, puis nous avons appliqué une rotation de -45 degrés pour revenir dans le repère initial.

La figure 5 met en évidence le fait que dans ce cas précis de simulation, l'utilisation du rectangle aligné sur le repère initial est plus performante que l'utilisation d'un rectangle tourné à 45 degrés. Ceci est à la fois vrai en terme d'aire du rectangle (0.044m^2 pour le rectangle vert contre 0.077m^2 pour le bleu) et de périmètre (0.84m pour le rectangle vert contre 1.11m pour le bleu).

Un tel résultat pouvait se prévoir à la lecture des équations utilisées. On pouvait en effet calculer l'aire donnée concernant l'incertitude de la position du robot en fonction de l'angle de cap initial introduit. Soit θ_0 cet angle. Dans notre cas simplifié (position initiale parfaitement connue et étude du résultat à l'instant 1), nous n'avons qu'à évaluer les équations ci-dessous avec les intervalles associés à ds_1^x et $d\omega_1$

$$\left\{ \begin{array}{rcl} x_1 & = & ds_1^x \cdot \text{sinc}\left(\frac{d\omega_1}{2}\right) \cdot \cos\left(\theta_0 + \frac{d\omega_1}{2}\right) \\ y_1 & = & ds_1^x \cdot \text{sinc}\left(\frac{d\omega_1}{2}\right) \cdot \sin\left(\theta_0 + \frac{d\omega_1}{2}\right) \end{array} \right. \tag{79}$$

Une fois tous les paramètres connus, on peut calculer les paramètres des rectangles (aires et périmètres) en fonction de l'angle θ_0. La non linéarité des fonctions sin et cos fait que les résultats vont largement varier avec θ_0. Nous avons tracé les paramètres obtenus lorsque θ_0 varie entre 0 et $\pi/2$ (figure 6). Les courbes de la figure 6 mettent en évidence le caractère optimal – pour les paramètres choisis – de la grille initiale (ie. $\theta_0 = 0$).

Nous ne donnons pas plus de détails concernant la détermination du θ_0 optimal. En effet, son calcul analytique n'est pas trivial et nécessite de distinguer différents cas suivant les intervalles considérés. Le but de ce paragraphe est surtout de mettre en évidence l'importance

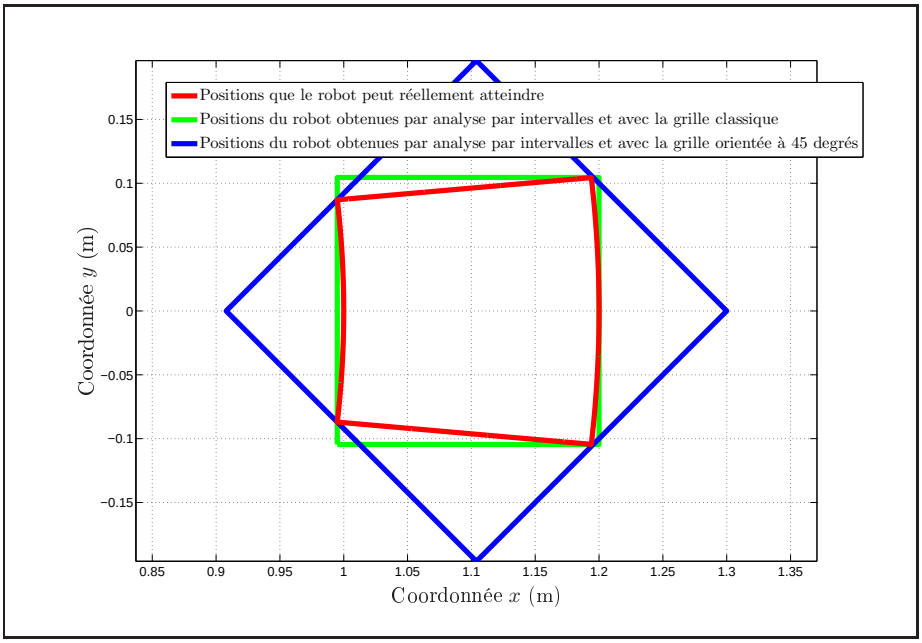

FIG. 5 – Importance de l'orientation des rectangles représentant la position du robot

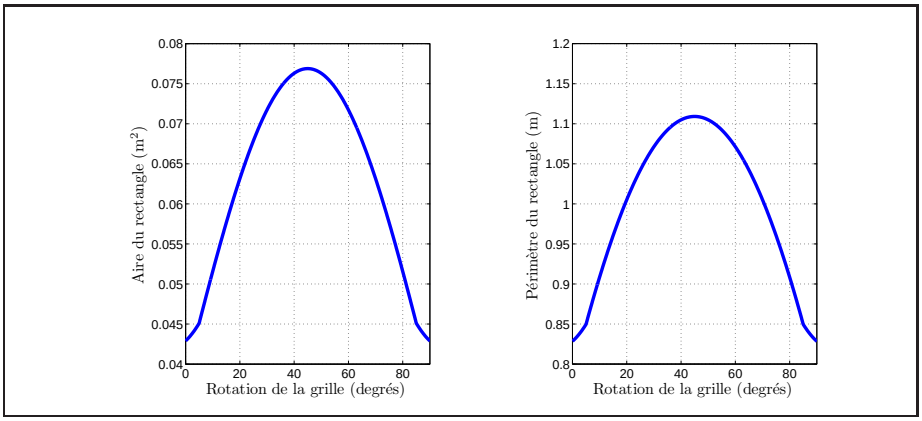

FIG. 6 – Caractéristiques des rectangles obtenus en fonction de l'angle de rotation des rectangles — A gauche : les aires — A droite : les périmètres

de l'orientation des boites ; on ne cherchera pas par la suite de solution optimale, mais une approximation permettant d'éviter les estimations beaucoup trop pessimistes.

Remarque 6.3 (Intervalle considéré pour θ_0)
Prendre $\theta_0 = \pi/2$ équivaut à intervertir les rôles des coordonnées x et y (au signe près). Ceci ne change donc pas l'orientation de la grille. Au delà de $\theta_0 = \pi/2$, on retrouvera donc des configurations déjà rencontrées. Ainsi, nous limitons l'étude pour $\theta_0 \in [0, \pi/2]$.

6.3.3 Exemple 2 : fonction de mesure

Le problème d'orientation des axes se retrouve également dans l'utilisation de la fonction de mesures. Considérons par exemple la première équation du système 76, rappelée ci-après :

$$x_t = x_{(i)} - \left(y_{(i)} - y_t\right) \cdot \cot\left(\theta_t + \alpha_{t,(i)}\right) \tag{80}$$

Nous pouvons constater que l'évaluation du terme $\left(\theta_t + \alpha_{t,(i)}\right)$ dépend de l'orientation absolue du robot θ_t. En conséquence, changer la condition initiale sur le cap du robot (ce qui revient à ajouter un angle parfaitement connu sur θ_t) peut changer la solution de l'algorithme de SLAM.

On peut remarquer avec cette équation qu'ajouter une condition initiale sur la position du robot ne changera rien en pratique car l'équation de mesure fait intervenir les termes $\left(x_{(i)} - x_t\right)$ et $\left(y_{(i)} - y_t\right)$ qui éliminent cette condition initiale. Ceci n'est malheureusement pas le cas avec l'orientation.

6.3.4 Solution proposée

Nous avons vu à travers les deux exemples précédents que l'orientation des axes a un rôle certain dans la solution de l'algorithme de SLAM par intervalles. Pour chaque position du robot, on peut trouver une orientation des boites qui minimisera l'aire de la boîte. On peut appliquer le même raisonnement sur les boîtes associées aux amers.

Trouver analytiquement la meilleure orientation pour chaque boîte ne semble pas possible. Par ailleurs, ce problème n'est pas crucial dans le sens où il ne provoque pas d'inconsistance ; l'algorithme sera uniquement trop pessimiste.

Pour tenir compte de ce problème, nous choisissons d'appliquer l'algorithme initial en utilisant plusieurs conditions initiales concernant l'orientation du robot :

1. Dans un premier temps, nous appliquons l'algorithme initial avec $\theta_0 = 0$ puis nous sauvegardons les résultats.
2. Ensuite, nous projetons la solution obtenue dans un nouveau repère correspondant à l'application de l'algorithme avec $\theta_0 = \phi_1$. La projection effectuée fait augmenter la taille des boîtes initiales. Mais la nouvelle application de l'algorithme fera diminuer la taille de certaines bôites.
3. A la fin de la seconde application de l'algorithme, nous comparons les aires des nouvelles positions du robot par rapport à celle obtenues avec l'application précédente. Nous conservons les solutions les moins conservatives (au sens de l'aire des rectangles).

4. Nous pouvons ensuite appliquer les points 2 et 3 avec d'autres valeurs de θ_0.

6.3.5 Etude du gain apporté sur une simulation

Nous proposons dans ce paragraphe de présenter le gain obtenu en appliquant l'algorithme présenté en 6.3.4. Pour cela, nous comparons les résultats obtenus en simulation avec l'algorithme " classique " à ceux obtenus avec l'algorithme optimisé.

Les caractéristiques de la simultation sont :

- Vitesse constante : 1.5m.s^{-1}
- Vitesse de rotation constante : 0.15rad.s^{-1}
- Contrainte de non-honolomie respectée
- Erreur sur la vitesse mesurée :[10] uniforme dans l'intervalle $-[0.05, 0.05]\text{m.s}^{-1}$
- Erreur sur la vitesse de rotation mesurée : uniforme dans l'intervalle $[-0.03, 0.03]\text{rad.s}^{-1}$
- Pas d'erreur de modèle supplémentaire : $(\nu_t^{\mathbf{f}} = 0)$
- Erreurs sur les mesures des amers uniformes (gisement et élévation) dans l'intervalle : $[-0.0175, 0.0175]$rad ($[-1, 1]$deg)

L'algorithme de SLAM nécessite d'utiliser des intervalles consistants pour les données. Nous allons utiliser les vrais intervalles (définis ci-dessus) dans l'exécution de l'algorithme.

En ce qui concerne l'algorithme optimisé, nous utilisons 11 valeurs pour θ_0 : 0deg, 10deg, 20deg, 30deg, 40deg, 45deg, 50deg, 60deg, 70deg, 80deg.

Les résultats obtenus sont présentés sur les figures 8 à 12. Dans le cas présent, les gains obtenus en tenant compte des orientations sont très importants : l'algorithme initial converge en effet vers une solution qui est très conservatrice. Le caracère circulaire de la trajectoire permet de bien mettre en évidence l'importance du paramètre " orientation des boîtes ".

Enfin, on pourra remarquer sur la figure 8 que les amers semblent tous être orientés vers l'origine du repère. Nous n'analysons pas ce phénomène en détail, mais intuitivement, ceci est dû au fait que tous les amers sont observés pour la première fois à l'instant initial. La position du robot à cet instant étant parfaitement connue, le cône d'observation initial à de forte chances d'être plus précis que ceux obtenus aux instants suivants. Ceci conduit alors naturellement à orienter les boîtes des amers dans cette position.

[10] Les simulations sont effectuées à échantillonnage temporel constant et les vitesses sont simulées constantes entre chaque instant. Ainsi, considérer que les entrées sont les vitesses (axiales ou de rotation) ou les incréments (de distance ou de rotation) est équivalent.

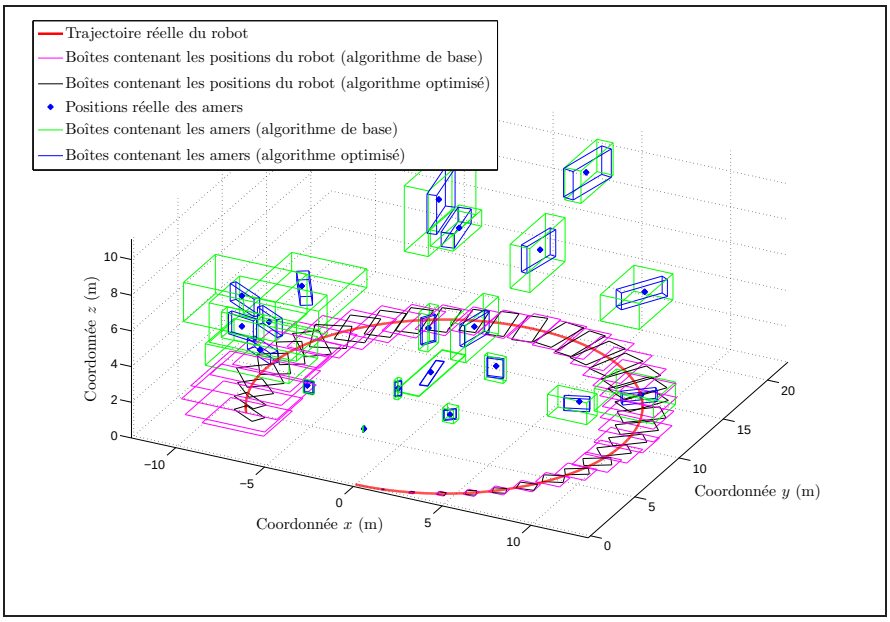

FIG. 7 – Résultats globaux des deux algorithmes

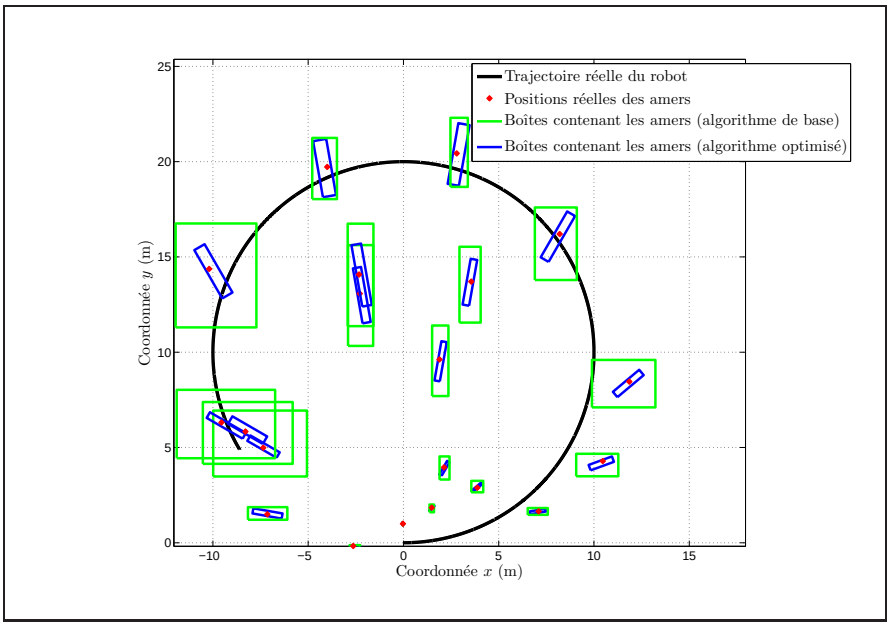

FIG. 8 – Résultats des algorithmes concernant les amers

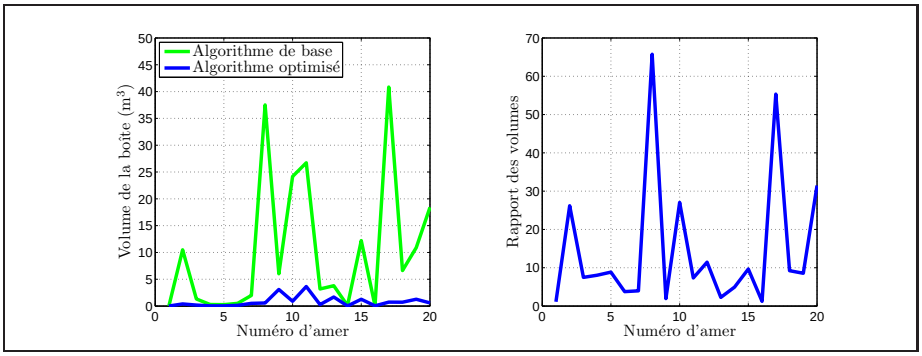

FIG. 9 – A gauche : aires des rectangles obtenus pour les amers avec les deux algorithmes — A droite : rapport entre l'aire obtenue avec l'algorithme initial et l'aire obtenue avec l'algorithme optimisé (pour les amers)

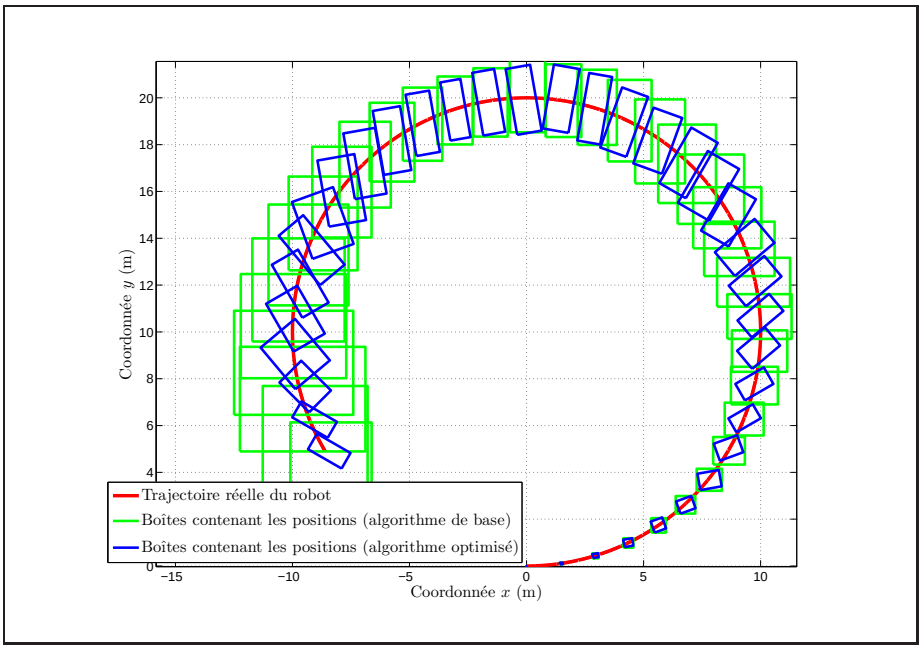

FIG. 10 – Résultats des algorithmes concernant la position du robot (coordonnées x_t et y_t uniquement)

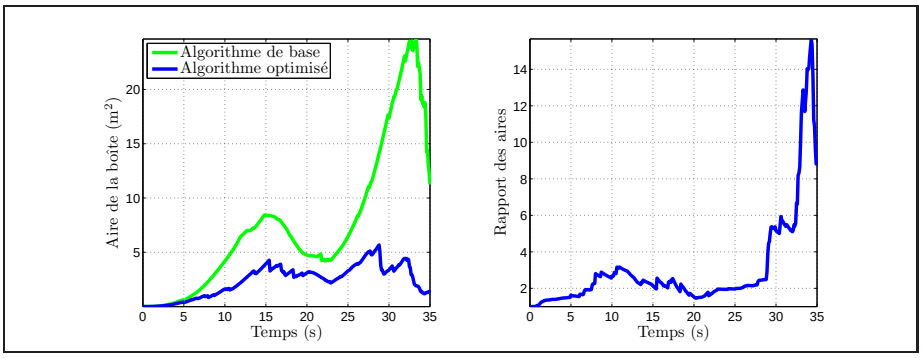

FIG. 11 – A gauche : aires des rectangles obtenus pour les positions du robot avec les deux algorithmes — A droite : rapport entre l'aire obtenue avec l'algorithme initial et l'aire obtenue avec l'algorithme optimisé (pour les positions du robot)

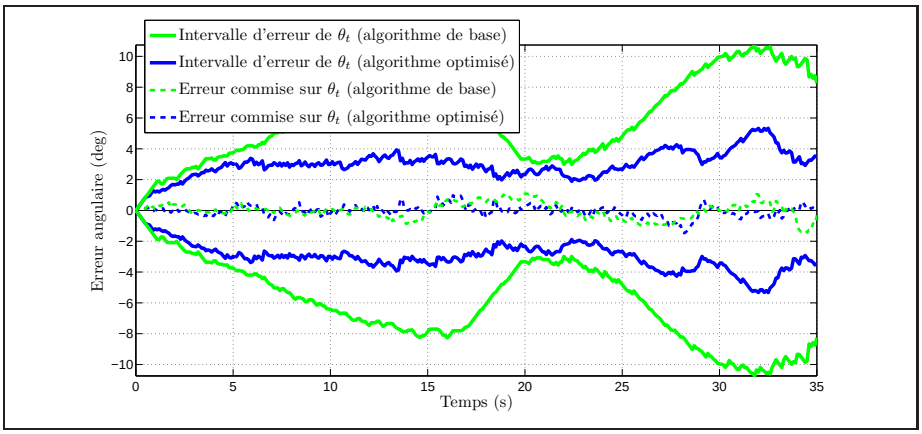

FIG. 12 – Résultats des algorithmes concernant l'angle de cap du robot

7 Résultats obtenus

Nous présentons dans cette section les résultats obtenus en simulation. Nous présentons dans un premier temps les caractéristiques communes à toutes les simulations. Nous montrons dans le paragraphe 7.2 les résultats obtenus pour le cas gaussien. Nous voyons ensuite les résultats obtenus lorsque les erreurs sont uniformes et centrées (paragraphe 7.3). Le paragraphe 7.5 présente les résultats obtenus lorsqu'on fait varier la visibilité des amers. Nous effectuons ensuite quelques remarques sur la consistance du GraphSLAM dans le paragraphe 7.6. Nous commentons enfin les temps de calcul dans le paragraphe 7.7.

7.1 Description des simulations

Nous présentons ici les caractéristiques communes à toutes les simulations.

7.1.1 Caractéristiques de la simulation

La trajectoire simulée est à vitesse de translation et de rotation constantes :
- $V = 1.5\text{m.s}^{-1}$
- $\omega = \frac{5\pi}{180}\text{rad.s}^{-1}$

Ceci correspond à un cercle de rayon 17.2m. L'essai dure 150s, ce qui permet d'effectuer environ 2.1 tours.

En ce qui concerne les amers, nous utilisons dans toutes les simulations une carte de 200 amers répartis uniformément autour de la trajectoire :
- entre -30m et 30m en ce qui concerne la coordonnée x,
- entre -10m et 50m en ce qui concerne la coordonnée y,
- entre 0m et 10m en ce qui concerne la coordonnée z.

La trajectoire ainsi que la carte d'amers sont représentés sur la figure 13.

7.1.2 Mode opératoire

A partir de la trajectoire décrite précédemment, nous allons générer plusieurs scenarii. Ceux-ci sont la plupart du temps générés en fonction des hypothèses effectuées sur les erreurs (distribution, amplitude).

En ce qui concerne les bruits, nous effectuons les hypothèses suivantes :
- Nous n'ajoutons pas de composante de vitesse transversale lors de la génération de la trajectoire,
- Nous n'ajoutons pas de bruit additif lors de la génération de la trajectoire.

Néanmoins, les filtres utiliseront des réglages laissant ouverte la possibilité d'erreur sur ces paramètres : les matrices de variances-covariances (et intervalles) associés ne seront donc pas réduits à zéro. Les valeurs utilisées pour les réglages sont :

GraphSLAM : on utilisera $\mathbf{Q_f} = \text{diag}\left(0.001^2\text{m}^2, 0.001^2\text{m}^2, 0\right)$ et $\sigma_{ds^y} = \sigma_{ds^x}/100$

SLAM par intervalles : on ajoute à chaque prédiction l'intervalle $[-0.001, 0.001]$m sur les coordonnées x et y du robot. Les bornes utilisées pour l'intervalle sur la vitesse

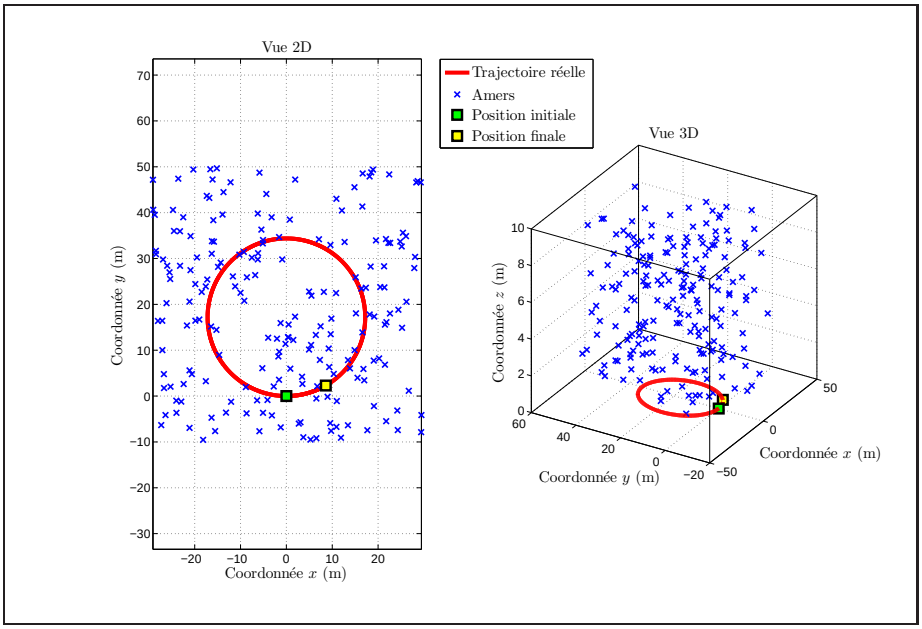

FIG. 13 – Trajectoire réelle du robot et position réelle des amers — Le sens de rotation du robot est le sens trigonométrique — La position finale (carré jaune) est obtenue après un peu plus de deux tours

transversale sont prises égales au centième des bornes utilisées pour la vitesse longitudinale.

De plus, nous supposons que l'association des données est connue. Enfin, **tous les amers sont visibles à chaque instant** (sauf les cas traités au paragraphe 7.5).

7.1.3 Résultats étudiés

Pour chaque scénario, nous observons la qualité de l'estimation ainsi que la consistance des algorithmes. Les amers et la trajectoire du robot seront traités séparément.

En ce qui concerne la consistance du GraphSLAM, on s'attachera à vérifier si chaque position x, y (robot à l'instant t et amers) est consistante en terme d'ellipse de confiance à 99% et si l'orientation l'est en terme d'enveloppe de confiance à 99%. En ce qui concerne le robot, nous séparons l'état en position et orientation car l'ellipse associée à la position a une interprétation géométrique directe. On peut en effet comparer l'aire de cette ellipse avec l'aire de la boîte du SLAM par intervalles (ce qui ne serait pas le cas si on considérait l'ellipsoïde associé à tout l'état du robot).

En ce qui concerne le SLAM par intervalles, on calculera les boîtes associées à chaque position du robot et chaque amer. On pourra ainsi vérifier si ces boîtes contiennent la position réelle et si elles sont ou non trop pessimistes.

Pour comparer les performances des deux algorithmes, on comparera l'aire (resp. volume) des régions associées à chaque position du robot (resp. amers). En ce qui concerne le SLAM par intervalles, il s'agit de l'aire (resp.volume) de chaque boîte. En ce qui concerne le GraphSLAM, on calculera l'aire de l'ellipse (resp. ellipsoïde) de confiance à 99%. Pour ce dernier cas, les aires et volumes sont proportionnels à $\sqrt{\det \mathbf{\Sigma}}$ où $\mathbf{\Sigma}$ désigne la matrice de variances-covariances associée à la position étudiée (de taille (2,2) ou (3,3) suivant le cas). Le calcul détaillé des coefficients de proportionalité est en annexe A.

Remarque 7.1 (Limitations de l'étude concernant le GraphSLAM)
On remarquera en ce qui concerne le GraphSLAM qu'on ne s'intéresse qu'aux positions marginales. On ne cherche pas à savoir si les corrélations entre les différents amers et positions sont consistantes. Seul le coefficient de corrélation pour une position donnée est pris en compte car il a une interprétation géométrique forte sur l'emplacement d'un point (il nous donne l'orientation de l'ellipse).
Ceci ne pose pas de problème en pratique. L'utilisateur du SLAM souhaite avant tout récupérer la position du robot et la position des amers avec une enveloppe de confiance pour chacune d'elles. Le maintien des corrélations est utile pour l'algorithme de SLAM mais ne constitue pas une fin en soit. D'ailleurs, la consistance des corrélations conditionne généralement la consistance de la solution finale.

7.2 Le cas gaussien centré

Nous présentons dans cette section les résultats obtenus en simulant l'hypothèse gaussienne sur l'ensemble des bruits. Tous les bruits seront donc générés avec la fonction **randn** de `Matlab`. Nous présentons d'abord les résultats obtenus avec l'algorithme de GraphSLAM, et ensuite les résultats obtenus avec l'algorithme de SLAM par intervalles. Pour les deux cas, nous considérons les 4 scenarii. Ceux-ci sont décrits dans le tableau 4.

7.2.1 GraphSLAM

Nous présentons ici les résultats obtenus avec l'algorithme de GraphSLAM. Les paramètres utilisés dans le réglage des filtres sont les paramètres réels utilisés pendant la simulation.

Les résultats obtenus sont présentés sur les figures 14 à 18.

- La figure 14 montre l'allure de la trajectoire obtenue ainsi que les ellipses de confiance pour le premier scénario. Visuellement, l'algorithme est consistant (toutes les estimations sont à l'intérieur des ellipses à 99%). De plus, celles-ci ne semblent pas pessimistes en regard de la qualité de la trajectoire odométrique. Des résultats très similaires peuvent être obtenus avec les autres scenarii.
- La figure 15, quant à elle, montre le résultat obtenu en ce qui concerne le positionnement des amers pour le premier scénario. Là aussi, les résultats sont consistants.

Scénario	Description	Commentaires
1	– Erreur sur V^x : gaussienne avec $\sigma_{V^x} = 0.1\text{m.s}^{-1}$ – Erreur sur ω : gaussienne avec $\sigma_\omega = 0.1\text{rad.s}^{-1}$ – Erreur sur α : gaussienne avec $\sigma_{amer} = \frac{\pi}{180}\text{rad}$ – Erreur sur β : gaussienne avec $\sigma_{amer} = \frac{\pi}{180}\text{rad}$	Les erreurs générées sur les entrées sont très importantes. Celles sur les mesures des amers sont assez faibles. Ceci permet de tester les algorithmes dans des conditions où il est difficile d'obtenir un bon *a priori* sur la trajectoire.
2	– Erreur sur V^x : gaussienne avec $\sigma_{V^x} = 0.1\text{m.s}^{-1}$ – Erreur sur ω : gaussienne avec $\sigma_\omega = 0.1\text{rad.s}^{-1}$ – Erreur sur α : gaussienne avec $\sigma_{amer} = \frac{0.1\pi}{180}\text{rad}$ – Erreur sur β : gaussienne avec $\sigma_{amer} = \frac{0.1\pi}{180}\text{rad}$	Les erreurs générées sur les entrées sont très importantes. Celles sur les mesures des amers sont très faibles. Ceci permet de tester la capacité des algorithmes à utiliser une information très précise concernant les amers.
3	– Erreur sur V^x : gaussienne avec $\sigma_{V^x} = 0.025\text{m.s}^{-1}$ – Erreur sur ω : gaussienne avec $\sigma_\omega = 0.005\text{rad.s}^{-1}$ – Erreur sur α : gaussienne avec $\sigma_{amer} = \frac{3\pi}{180}\text{rad}$ – Erreur sur β : gaussienne avec $\sigma_{amer} = \frac{3\pi}{180}\text{rad}$	Les erreurs générées sur les entrées sont faibles. Celles sur les mesures des amers sont fortes. Ceci permet de tester les algorithmes lorsqu'ils ont un bon *a priori* sur la trajectoire mais de mauvaises mesures d'amer.
4	– Erreur sur V^x : gaussienne avec $\sigma_{V^x} = 0.05\text{m.s}^{-1}$ – Erreur sur ω : gaussienne avec $\sigma_\omega = 0.01\text{rad.s}^{-1}$ – Erreur sur α : gaussienne avec $\sigma_{amer} = \frac{1\pi}{180}\text{rad}$ – Erreur sur β : gaussienne avec $\sigma_{amer} = \frac{1\pi}{180}\text{rad}$	Cas moyen

TAB. 4 – Scenarii pour le cas gaussien centré

– Les figures 16 et 17 présentent les aires et volumes obtenus concernant les positions du robot et des amers en fonction du scénario. On peut remarquer que ces paramètres sont très faibles : les aires sont inférieures à 1m^2 en ce qui concerne le robot et les volumes sont très inférieurs au mètre cube en ce qui concerne les amers. Dans tous les cas, on se rend compte que les aires des ellipses associées aux positions du robot ont des variations bien particulières : elles augmentent et diminuent successivement au fur et à mesure de la trajectoire. Ceci est dû au caractère circulaire de la trajectoire. Les amers proches du point de départ auront naturellement une incertitude plus faible (même si tous les amers sont vus en même temps et à toutes les positions).[11] Lorsqu'on s'éloigne du point de départ, on commence à accumuler des incertitudes. Même si l'algorithme de GraphSLAM tend à les minimiser au maximum, elles sont inévitables. Ainsi, le retour à la position initiale fait qu'on se rapproche des amers les mieux estimés.
– La figure 18 présente la qualité de l'estimation de l'orientation du robot. Dans tous les cas, elle est largement consistante. On peut juste remarquer qu'une grande erreur sur les données odométriques ne favorise pas l'algorithme (et ce même avec de bonnes mesures) et le rend pessimiste (scenarii 1 et 2).

Dans tous les cas, on peut remarquer que l'algorithme de GraphSLAM est consistant. La trajectoire finale obtenue est toujours très proche de la trajectoire réelle, et ce même lorsque la trajectoire obtenue par odométrie pure est très mauvaise (cf. scenarii 1 et 2). Le

[11] Ceci provient d'une propriété de la matrice jacobienne concernant les mesures. Les coefficients de celle-ci augmentent lorsque la distance à l'amer diminue.

comportement de cet algorithme est donc très bon lorsque les hypothèses d'erreurs centrées et gaussiennes sont respectées.

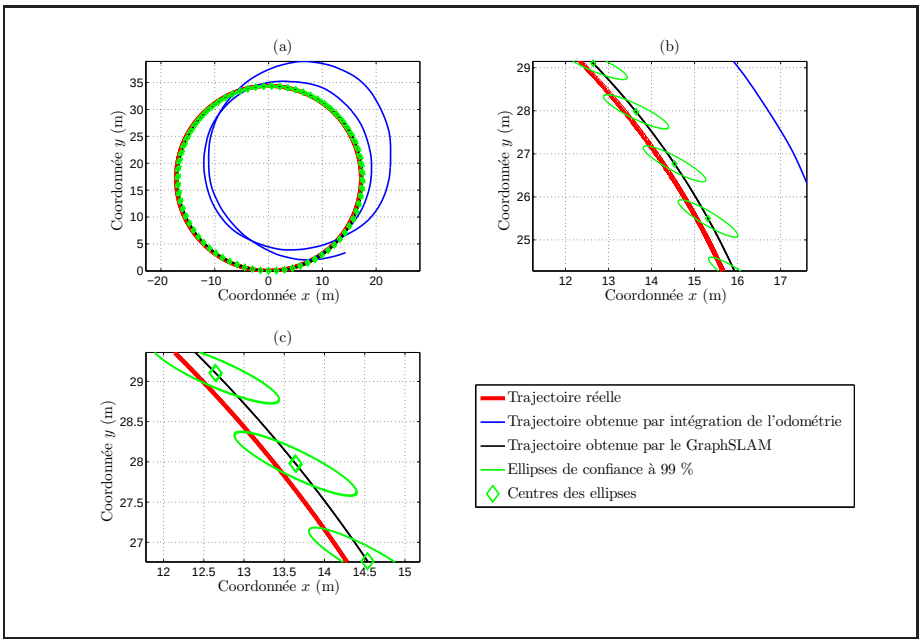

FIG. 14 – Estimation des positions du robot obtenue avec le GraphSLAM sur le scénario 1 : (a) Vue globale — (b) Leger zoom — (c) Zoom important

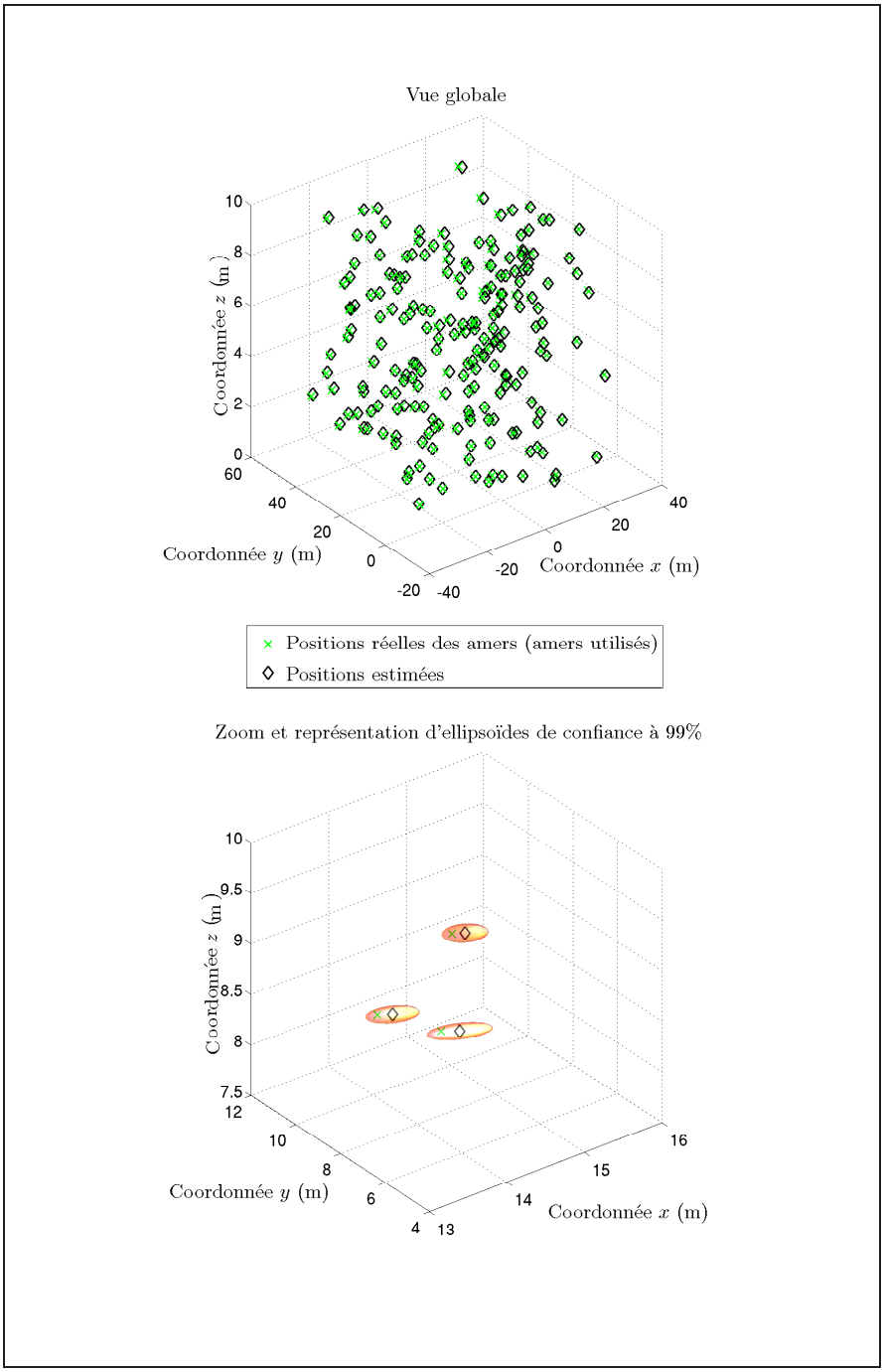

FIG. 15 – Estimation des amers obtenue avec le GraphSLAM sur le scénario 1

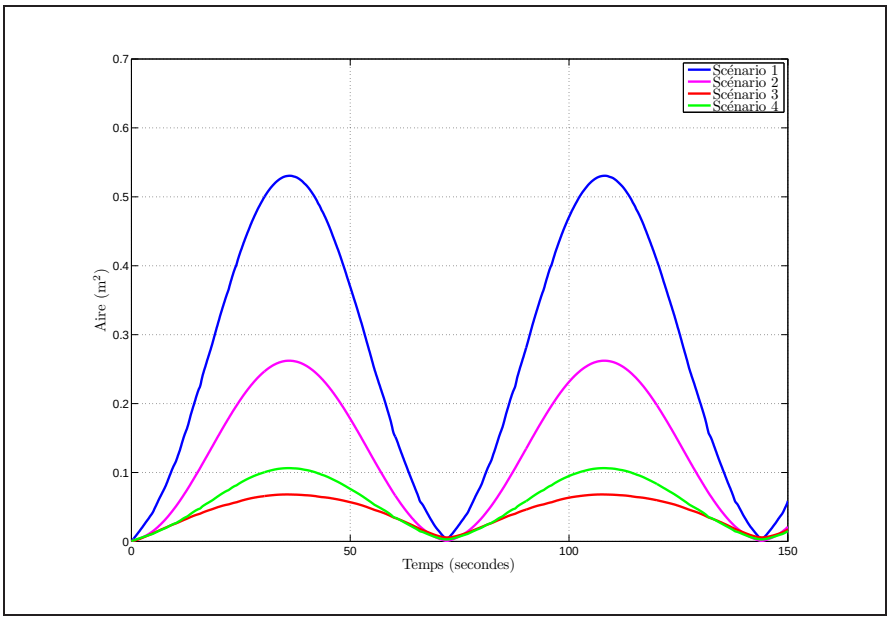

FIG. 16 – Aires des ellipses de confiance à 99% concernant les positions du robot : scenarii 1 à 4

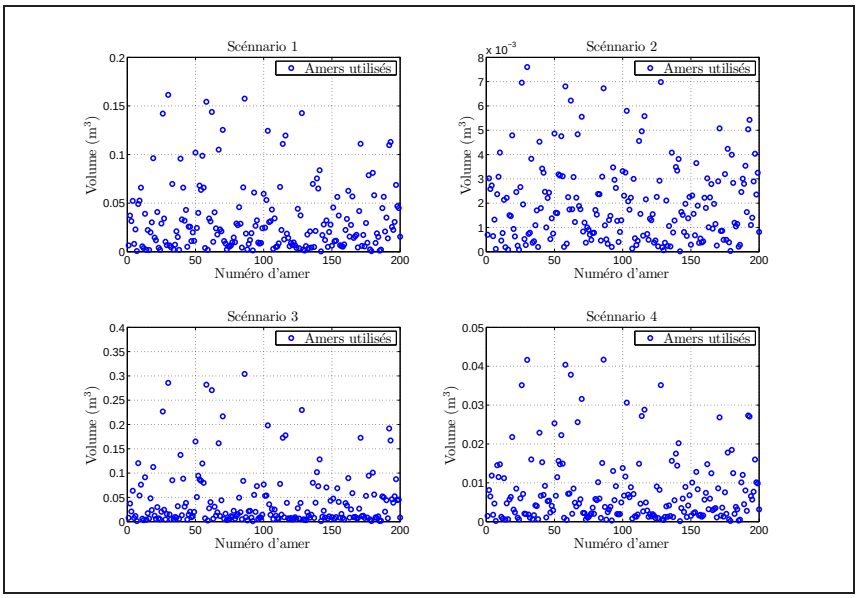

FIG. 17 – Volumes des ellipsoïdes de confiance à 99% concernant les positions des amers : scenarii 1 à 4

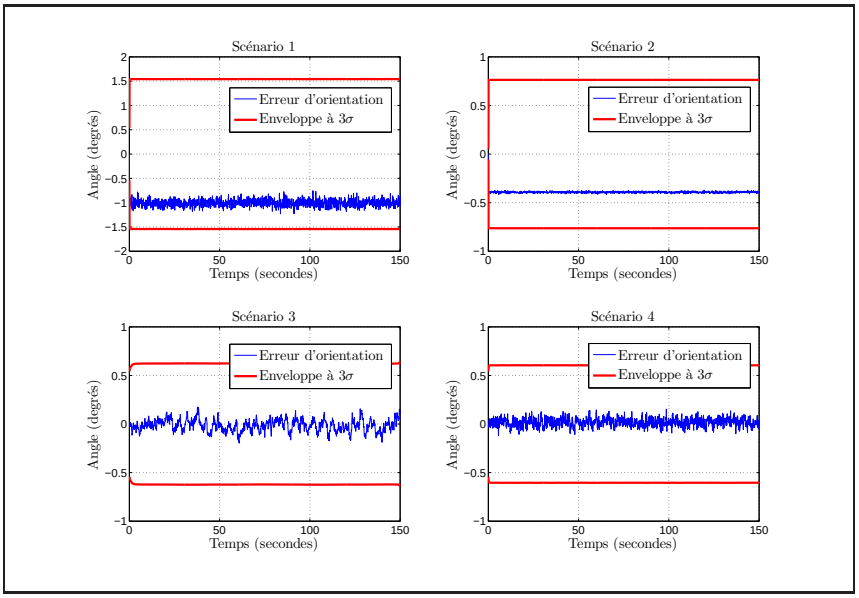

FIG. 18 – Résultats du GraphSLAM concernant l'orientation du robot : scenarii 1 à 4

7.2.2 SLAM par intervalles

Nous présentons ici les résultats obtenus avec l'analyse par intervalles. Avant de les présenter, il convient de remarquer que le cas gaussien n'est pas adapté à l'analyse par intervalles. En effet, les erreurs gaussiennes ne sont pas bornées alors que l'algorithme de SLAM par intervalles nécessite la bornitude des erreurs.

Néanmoins, nous allons utiliser notre algorithme en définissant des bornes qui seront des fonctions des variances utilisées pour les erreurs. Pour chaque scénario, nous testons 2 possibilités :

1. Nous prenons comme intervalles possibles pour les erreurs $[-4\sigma, +4\sigma]$ (où σ est l'écart type de l'erreur considérée). Dans le cas gaussien, cet intervalle permet d'assurer que plus de 99% des réalisations sont dans l'intervalle.
2. Il est possible que les intervalles précédents conduisent à une solution beaucoup trop pessimiste. Dans ce cas, nous le réutilisons avec les intervalles à 3 écarts types : $[-3\sigma, +3\sigma]$.

Ces bornes ne contiennent pas forcemment l'erreur réelle des capteurs. Ainsi, l'hypothèse de consistance des intervalles utilisés n'est pas vérifiée. Ceci est d'autant plus vrai pour le troisième cas testé. Il n'est donc plus garanti que la solution obtenue soit consistante. Par ailleurs, il est possible que l'algorithme ait à faire l'intersection d'intervalles disjoints : ceci conduit à l'arrêt du programme.

Les résultats obtenus concernant l'état du robot et des amers sont présentés sur les figures 19 à 23.[12] Seul le cas à 4σ a permis d'obtenir des résultats. Le cas à 3σ a systématiquement conduit à effectuer des intersections vides (ce qui n'est pas surprenant). Nous n'avons obtenu des résultats que pour les scenarii 2 à 4. Dans le cas du premier scénario, l'estimation est devenue trop pessimiste et il a été impossible de calculer une solution jusqu'à l'instant final.[13]

Les figures 19 à 21 mettent en évidence le caractère pessimiste du SLAM par intervalles. En effet, les intervalles obtenus sont en général très grands alors que les erreurs d'estimation (si on considère le centre des boîtes) sont bien plus faibles. Par ailleurs, les tailles des boîtes concernant les positions du robot sont visuellement très grandes en comparaison avec l'erreur de l'odométrie. Enfin, les figures 22 et 23 viennent appuyer le fait que le SLAM par intervalles est très pessimiste. Les aires et volumes associés aux positions du robot et des amers sont en effet très grandes ; les résultats obtenus par le SLAM par intervalles ne semblent pas exploitables.

7.2.3 Conclusion quant au cas gaussien centré

Nous avons présenté dans ce paragraphe les résultats concernant le cas gaussien centré. Dans tous les cas, les algorithmes se sont avérés consistants. Néanmoins, l'algorithme de SLAM par intervalles fait preuve d'un pessimisme exagéré.

[12] En ce qui concerne les amers, seuls 20 d'entre eux sont visualisés. Ceci permet de garder la figure lisible.

[13] L'incertitude sur l'orientation atteint $+/-85$ degrés, et il est impossible ensuite d'estimer la fin de la trajectoire.

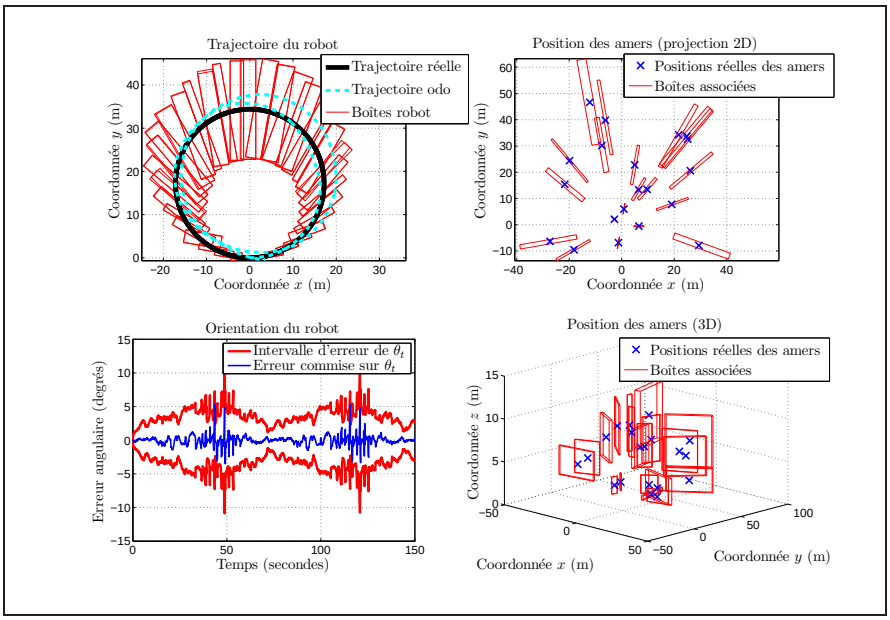

FIG. 19 – Résultats du SLAM par intervalles : scénario 2 — Réglage à 4σ

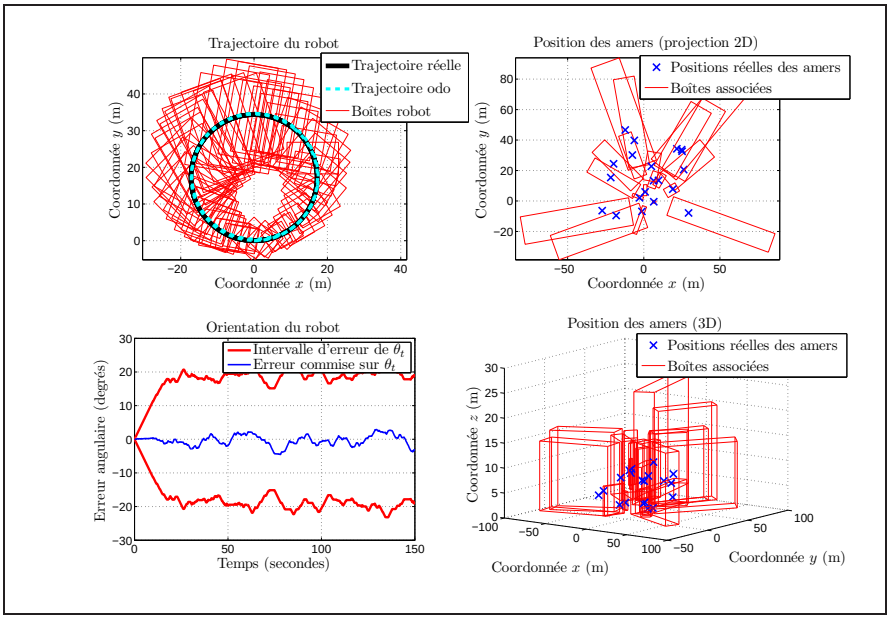

FIG. 20 – Résultats du SLAM par intervalles : scénario 3 — Réglage à 4σ

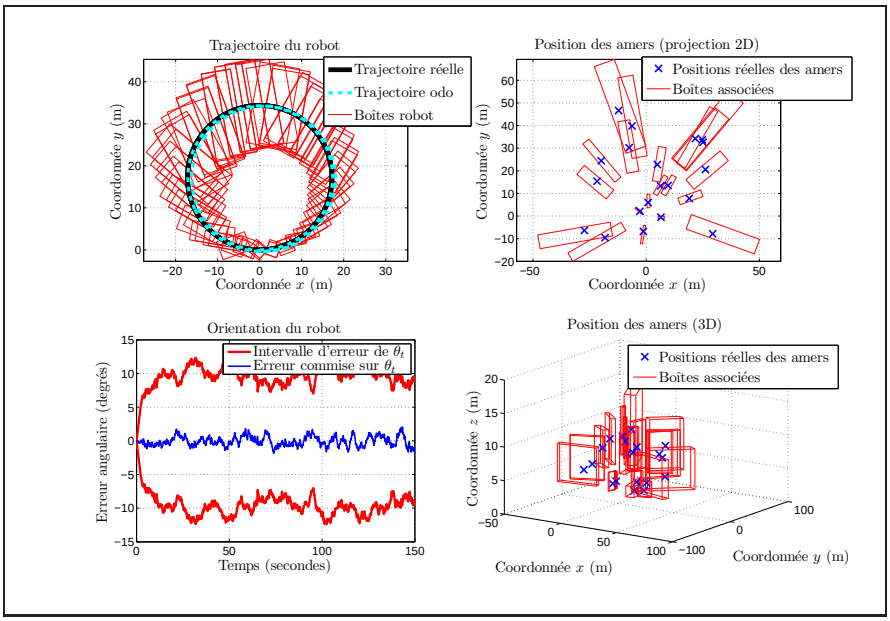

FIG. 21 – Résultats du SLAM par intervalles : scénario 4 — Réglage à 4σ

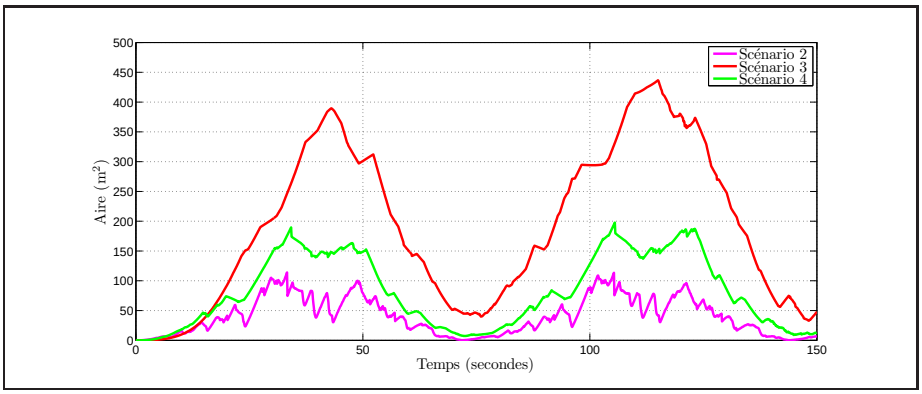

FIG. 22 – Aires des boîtes concernant les positions du robot : scenarii 1 à 4 — Réglage à 4σ

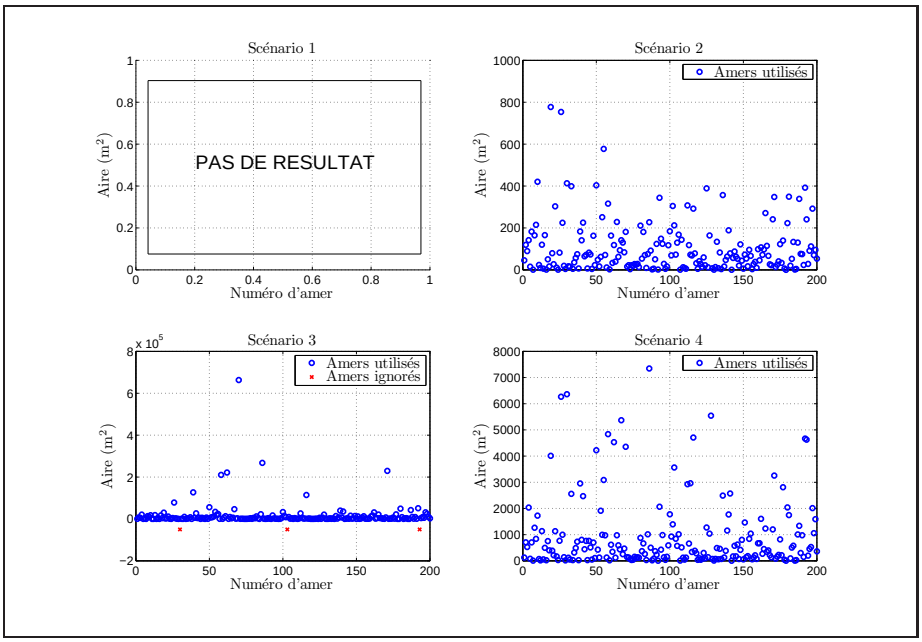

FIG. 23 – Volumes des boîtes concernant les positions des amers : scenarii 1 à 4 — Réglage à 2σ

Dans tous les cas, l'algorithme de GraphSLAM est supérieur à celui de SLAM par intervalles. Ainsi, lorsque les distributions des erreurs de nos capteurs sont gaussiennes et centrées, l'algorithme de GraphSLAM fournira d'excellents résultats pourvu que les paramètres soient connus (si ceux-ci sont trop optimistes, le GraphSLAM sera inconsistant.[14])

7.3 Cas uniforme centré

Nous présentons dans cette section les résultats obtenus en simulant l'hypothèse d'une erreur uniforme et centrée sur l'ensemble des bruits. Tous les bruits seront donc générés avec la fonction **rand** de `Matlab`. Tout comme pour le cas gaussien, nous testons les algorithmes avec 4 scenarii. Ceux-ci sont donnés dans le tableau 5.

Scénario	Description	Commentaires
5	– Erreur sur V^x : uniforme dans $[-0.2, +0.2]$ m.s^{-1} – Erreur sur ω : uniforme dans $[-0.2, +0.2]$ rad.s^{-1} – Erreur sur α : uniforme dans $[-\frac{1\pi}{180}, +\frac{1\pi}{180}]$ rad – Erreur sur β : uniforme dans $[-\frac{1\pi}{180}, +\frac{1\pi}{180}]$ rad	Les erreurs générées sur les entrées sont très importantes. Celles sur les mesures des amers sont assez faibles. Ceci permet de tester les algorithmes dans des conditions où il est difficile d'obtenir un bon *a priori* sur la trajectoire.
6	– Erreur sur V^x : uniforme dans $[-0.2, +0.2]$ m.s^{-1} – Erreur sur ω : uniforme dans $[-0.2, +0.2]$ rad.s^{-1} – Erreur sur α : uniforme dans $[-\frac{0.1\pi}{180}, +\frac{0.1\pi}{180}]$ rad – Erreur sur β : uniforme dans $[-\frac{0.1\pi}{180}, +\frac{0.1\pi}{180}]$ rad	Les erreurs générées sur les entrées sont très importantes. Celles sur les mesures des amers sont très faibles. Ceci permet de tester la capacité des algorithmes à utiliser une information très précise concernant les amers.
7	– Erreur sur V^x : uniforme dans $[-0.05, +0.05]$ m.s^{-1} – Erreur sur ω : uniforme dans $[-0.01, +0.01]$ rad.s^{-1} – Erreur sur α : uniforme dans $[-\frac{9\pi}{180}, +\frac{9\pi}{180}]$ rad – Erreur sur β : uniforme dans $[-\frac{9\pi}{180}, +\frac{9\pi}{180}]$ rad	Les erreurs générées sur les entrées sont faibles. Celles sur les mesures des amers sont fortes. Ceci permet de tester les algorithmes lorsqu'ils ont un bon *a priori* sur la trajectoire mais de mauvaises mesures d'amer.
8	– Erreur sur V^x : uniforme dans $[-0.05, +0.05]$ m.s^{-1} – Erreur sur ω : uniforme dans $[-0.05, +0.05]$ rad.s^{-1} – Erreur sur α : uniforme dans $[-\frac{1\pi}{180}, +\frac{1\pi}{180}]$ rad – Erreur sur β : uniforme dans $[-\frac{1\pi}{180}, +\frac{1\pi}{180}]$ rad	Cas moyen

TAB. 5 – Scenarii pour le cas uniforme centré

7.3.1 GraphSLAM

Nous présentons ici les résultats obtenus avec le GraphSLAM lorsque les erreurs sont centrées et uniformes. Dans ce cas, nous ne respectons plus l'hypothèse gaussienne sous-jacente à l'algorithme et il y a un risque d'inconsistance.

De plus, la nouvelle hypothèse concernant les bruits nécessite de revoir le réglage des filtres. Soit V la variance d'un bruit réparti uniformément entre les valeurs $-a$ et a. On a

[14] Ceci est logique et n'est pas étudié ici. Nous souhaitons valider les algorithmes dans des hypothèses " nominales " de fonctionnement.

alors :

$$V = \frac{a^2}{3} \tag{81}$$

En conséquence, les variances des filtres ont été réglées en utilisant l'équation 81 et les vraies valeurs concernant les bornes des intervalles.

En pratique, on constate que le filtre est consistant dans les 4 scenarii. Toutes les positions sont bien dans les ellipses à 99%. Il en est de même pour l'orientation du robot qui est systématiquement comprise dans l'enveloppe à 3σ. Les figures que l'on obtiendrait concernant l'estimation de la position du robot et des amers sont très semblables à la figure 14, et ne sont pas représentées.

Les aires (resp. volumes) des ellipses (resp. ellipsoïdes) à 99% obtenues sont données sur les figures 24 et 25. On retrouve des résultats assez similaires à ceux obtenus dans le cas gaussien, notamment pour l'évolution de l'aire des ellipses de la position du robot en fonction du temps.

Au final, on a constaté un bon comportement de l'algorithme de GraphSLAM lorsque les erreurs sont uniformes et centrées alors que l'on ne respecte plus l'hypothèse classique du GraphSLAM. Ceci est en partie dû au fait que le bruit généré est blanc et que l'on a utilisé la véritable variance du nouveau bruit pour le réglage. Une approche naïve aurait consisté à prendre pour chaque valeur de σ le tiers de la borne supérieure de la boîte, de sorte à ce que l'intervalle à 3σ contienne environ 99% des réalisations. Ceci aurait conduit à diviser par 3 la variance définie dans l'équation 81. En pratique, ce type de réglage conduit à un résultat trop optimiste et inconsistant.

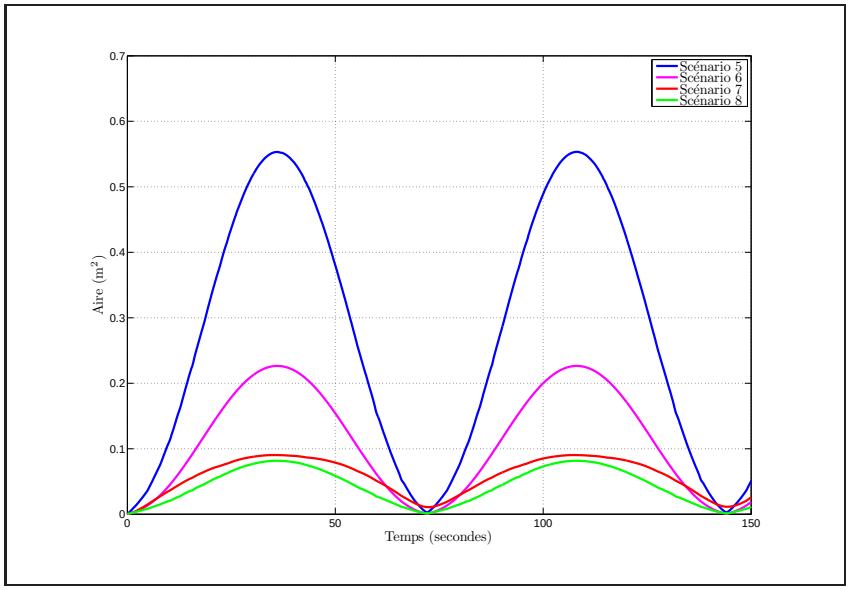

FIG. 24 – Aires des ellipses de confiance à 99% concernant les positions du robot : scenarii 5 à 8

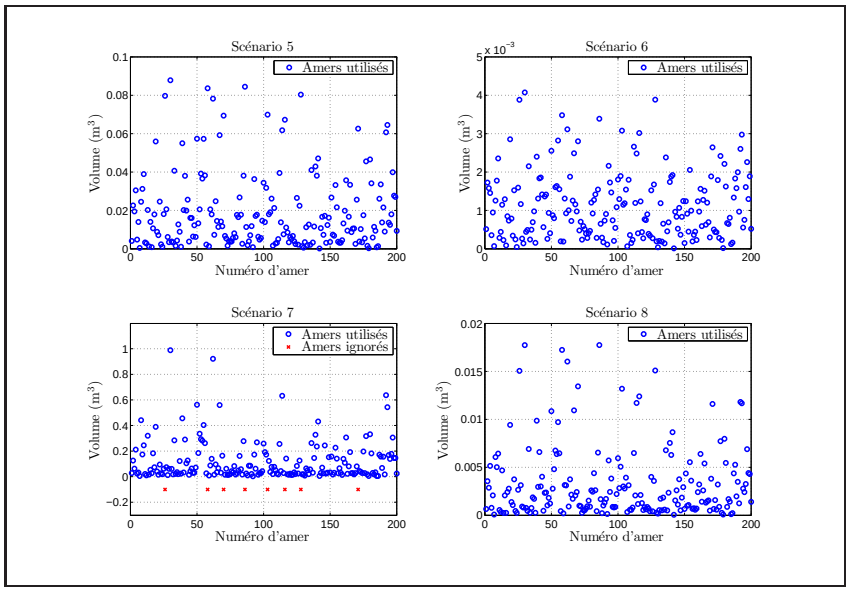

FIG. 25 – Volumes des ellipsoïdes de confiance à 99% concernant les positions des amers : scenarii 5 à 8

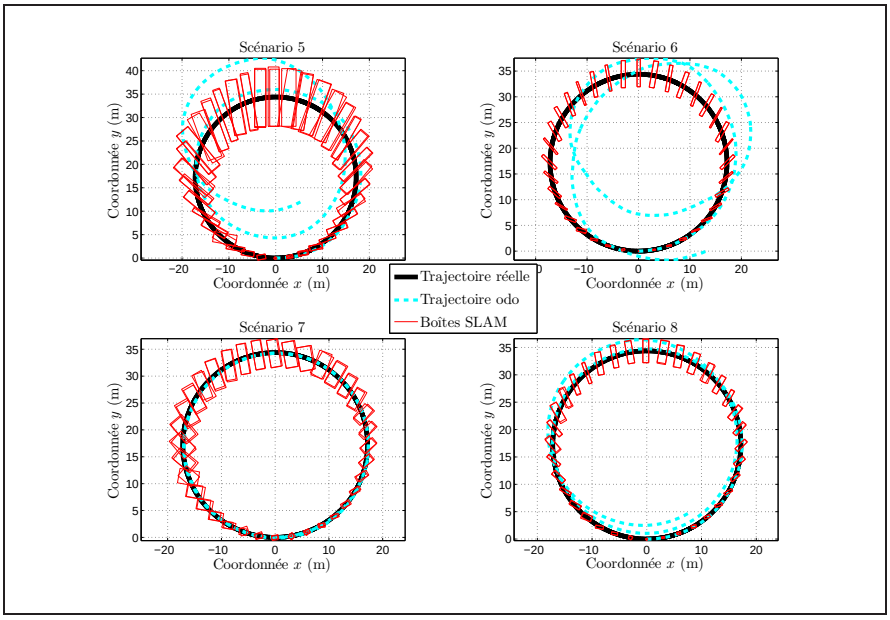

FIG. 26 – Résultats du SLAM par intervalles en ce qui concerne la position du robot : scenarii 5 à 8

7.3.2 SLAM par intervalles

Nous présentons ici les résultats du SLAM par intervalles lorsque les erreurs sont centrées et réparties uniformément. Les boîtes obtenues en ce qui concerne les positions du robot et des amers sont présentées sur les figures 26 et 27 . Les aires et volumes obtenus sont quant à eux visibles sur les figures 28 et 29.

On observe un meilleur comportement de l'algorithme lorsque l'erreur est répartie uniformément que lorsqu'elle est gaussienne. En effet, on constate que les boîtes obtenues sont visuellement plus petites. Ceci est confirmé avec la visualisation des aires et volumes des boîtes. En ce qui concerne la position du robot, on retrouve toujours les variations dues à la trajectoire circulaire.

La comparaison des résultats entre scenarii montre qu'il est nécessaire d'avoir au moins une bonne odométrie ou des bonnes mesures. Ainsi, le scénario 5 ne fournit pas de bons résultats (en terme de taille des boîtes). Diviser la borne sur l'erreur de mesure par 10 (sixième scénario) a permis d'améliorer nettement les résultats par rapport au scénario 5.

7.3.3 Conclusion quant au cas uniforme centré

Nous avons présenté ici les résultats obtenus par les deux algorithmes lorsque les erreurs sont réparties uniformément mais centrées. Les deux algorithmes se sont avérés consistants.

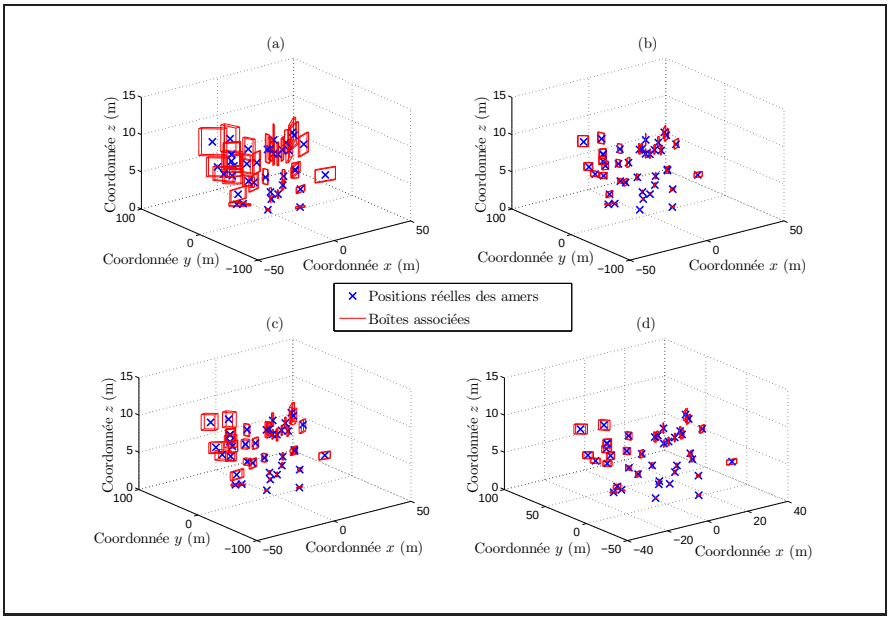

FIG. 27 – Résultats du SLAM par intervalles en ce qui concerne la position des amers : scenarii 5 à 8

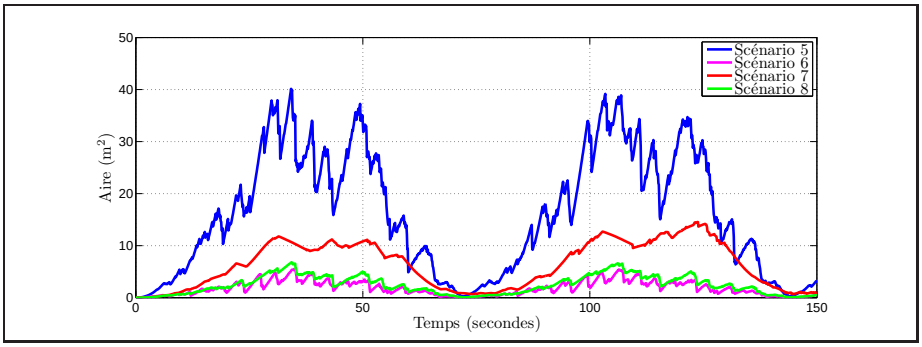

FIG. 28 – Aires des boîtes concernant les positions du robot : scenarii 5 à 8

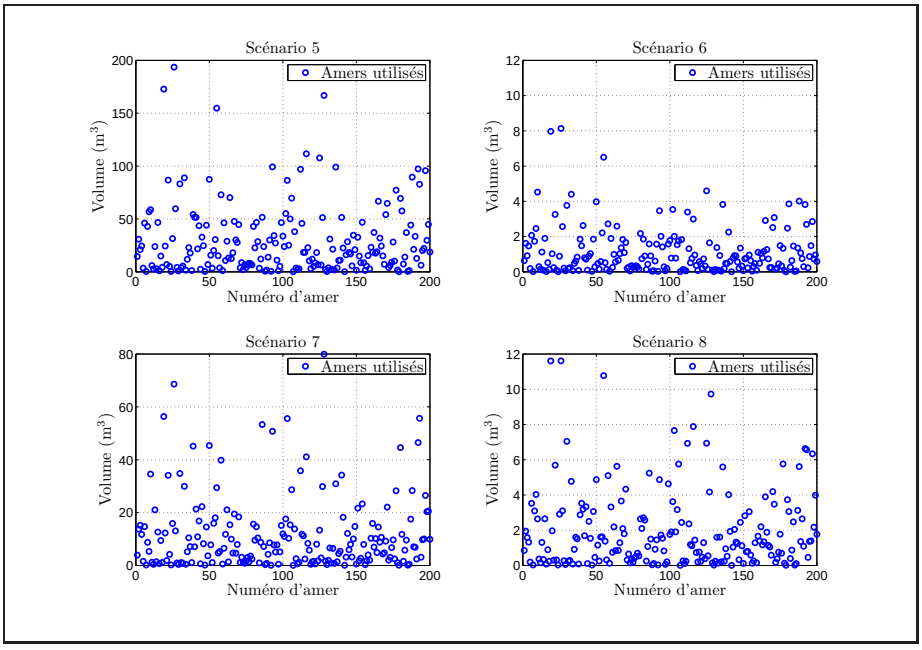

FIG. 29 – Volumes des boîtes concernant les positions des amers : scenarii 5 à 8

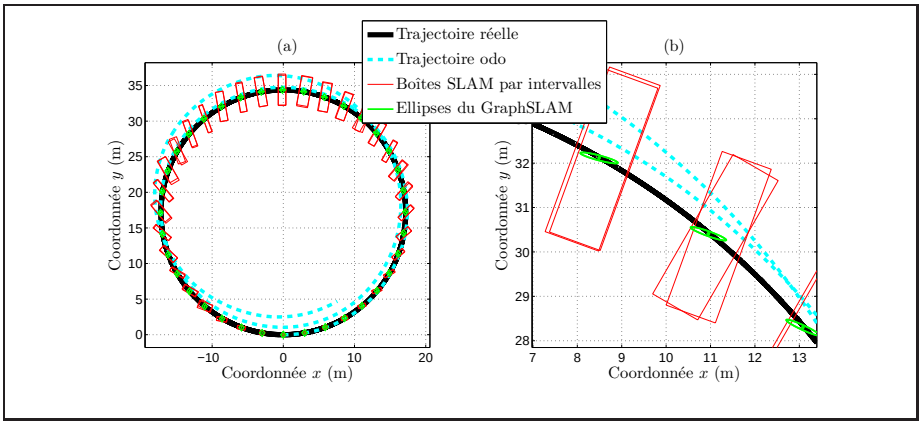

FIG. 30 – Comparaison des boîtes du SLAM par intervalles et des ellipses à 99% du GraphSLAM : scénario 8 — (a) Vue globale — (b) Zoom

Ce résultat était attendu en ce qui concerne l'algorithme de SLAM par intervalles alors qu'il était moins évident en ce qui concerne le GraphSLAM. Pour ce dernier algorithme, la bonne " conversion des bornes des intervalles en variances " a beaucoup contribué à ce résultat.

De plus, on constate que l'algorithme de SLAM par intervalles reste plus conservatif que le GraphSLAM. Ceci est visible en comparant les figures 24 et 28 (aires associées aux positions du robot) ainsi que les figures 25 et 29 (volumes associés aux positions des amers). La figure 30 illustre également l'aspect nettement plus conservatif du SLAM par intervalles.

Néanmoins, l'écart de performances est moins important que dans le cas gaussien. La distribution uniforme des erreurs est plus favorable à l'algorithme de SLAM par intervalles.

7.4 Cas uniforme biaisé

Nous présentons dans ce paragraphe les résultats obtenus lorsque les erreurs sur les capteurs contiennent des biais. Simuler de telles erreurs devrait permettre de mettre en avant les bonnes performances du SLAM par intervalles en ce qui concerne la consistance. Par contre, on s'attend à avoir des problèmes en ce qui concerne le GraphSLAM. Les deux algorithmes sont testés avec les 4 scenarii présentés dans le tableau 6.

Les réglages effectués pour les filtres ne tiennent pas compte de la présence de biais. **Nous les effectuons en supposant que toutes les erreurs sont uniformément réparties et centrées**. Les bornes utilisées pour les scenarii 9 à 12 sont :

Erreur sur V^x : dans $[-0.1, +0.1]$ m.s^{-1}

Erreur sur ω : dans $[-0.02, +0.02]$ rad.s^{-1}

Erreur sur z : dans $[-\frac{1\pi}{180}, +\frac{1\pi}{180}]$ rad pour les deux composantes

Ces bornes permettent d'assurer la consistance des intervalles. Pour chaque intervalle, au moins une des bornes correspond à la borne réelle utilisée pour générer l'erreur (cf. tableau 6).

Scénario	Description	Commentaires
9	– Erreur sur V^x : uniforme dans $[-0.1, 0]$ m.s^{-1} – Erreur sur ω : uniforme dans $[0, +0.05]$ rad.s^{-1} – Erreur sur α : uniforme dans $[-\frac{1\pi}{180}, +\frac{1\pi}{180}]$ rad – Erreur sur β : uniforme dans $[-\frac{1\pi}{180}, +\frac{1\pi}{180}]$ rad	Biais sur les entrées mais pas sur les mesures.
10	– Erreur sur V^x : uniforme dans $[-0.1, +0.1]$ m.s^{-1} – Erreur sur ω : uniforme dans $[-0.05, +0.05]$ rad.s^{-1} – Erreur sur α : uniforme dans $[0, +\frac{1\pi}{180}]$ rad – Erreur sur β : uniforme dans $[-\frac{1\pi}{180}, 0]$ rad	Biais sur les mesures mais pas sur les entrées.
11	– Erreur sur V^x : 1. uniforme dans $[-0.1, 0]$ m.s^{-1} pendant la première moitié de l'essai 2. uniforme dans $[0, +0.1]$ m.s^{-1} pendant la seconde moitié de l'essai – Erreur sur ω : 1. uniforme dans $[0, +0.05]$ rad.s^{-1} pendant la première moitié de l'essai 2. uniforme dans $[-0.05, 0]$ rad.s^{-1} pendant la seconde moitié de l'essai – Erreur sur α : uniforme dans $[0, +\frac{1\pi}{180}]$ rad pendant tout l'essai – Erreur sur β : uniforme dans $[-\frac{1\pi}{180}, 0]$ rad pendant tout l'essai	Erreurs à biais variables sur les entrées et à biais constants sur les mesures.
12	– Erreur sur V^x : constante à -0.1 m.s^{-1} – Erreur sur ω : constante à 0.05 rad.s^{-1} – Erreur sur α : constante à $\frac{1\pi}{180}$ rad – Erreur sur β : constante à $\frac{1\pi}{180}$ rad	Erreurs constantes égales à la limite de ce qui est admis au niveau des intervalles. Il s'agit *a priori* du scénario le plus défavorable pour le GraphSLAM.

TAB. 6 – Scenarii pour le cas biaisé

7.4.1 GraphSLAM

Les résultats concernant le GraphSLAM sont mauvais. En effet, toutes les estimations sont inconsistantes avec le réglage initial. De plus, l'application du douzième scénario a fait diverger l'algorithme (aucun résultat retourné). Les figures 31 et 32 montrent l'inconsistance des enveloppes à 99%, tant pour le robot que pour les amers (dans le cas du neuvième scénario).

Nous avons alors essayé de modifier les réglages du filtre. Nous avons multiplié l'ensemble des écarts types utilisés par 2, puis par 4, 8 et 16. Seule la multiplication par 16 a permis de rendre consistants les résultats obtenus sur le neuvième scénéario (cf. figures 33 et 34). Les scenarii 10 et 11 ont conduit à des résultats inconsistants (pour toutes les tentatives de réglages), alors que le douzième scénario a systématiquement conduit à une divergence de l'algorithme.

Aucune conclusion positive ne peut être tirée de ce dernier résultat. En effet, même si multiplier les réglages d'écarts types par 16 a permis d'améliorer la consistance de l'algo-

rithme pour un scénario précis, on ne peut en déduire de réglage global à appliquer pour contrer l'effet des biais.

Néanmoins, on a pu constater que les cas inconsistants ne génèrent pas des trajectoires absurdes. En effet, la dérive globale de l'odométrie est en général compensée et les amers sont disposés d'une façon cohérente. Ce qui est ici mis en cause est la **très mauvaise appréciation de l'erreur commise**.

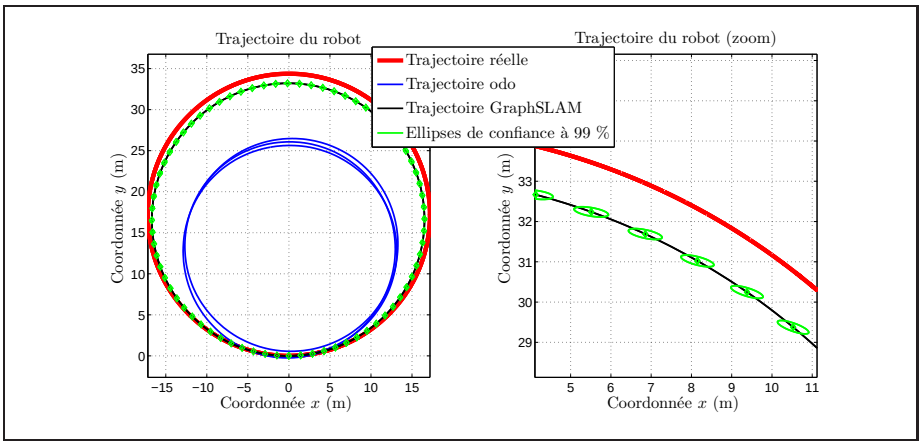

FIG. 31 – Résultats du GraphSLAM en ce qui concerne la trajectoire du robot : scénario 9 — Réglage à 1σ

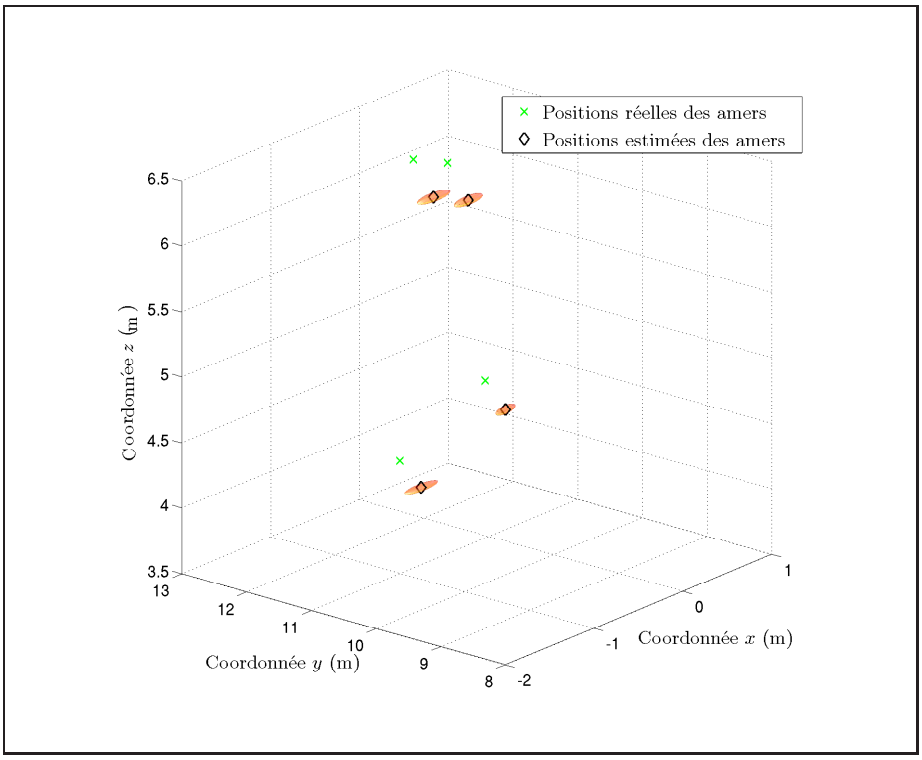

FIG. 32 – Résultats du GraphSLAM en ce qui concerne la position des amers : scénario 9 — Réglage à 1σ

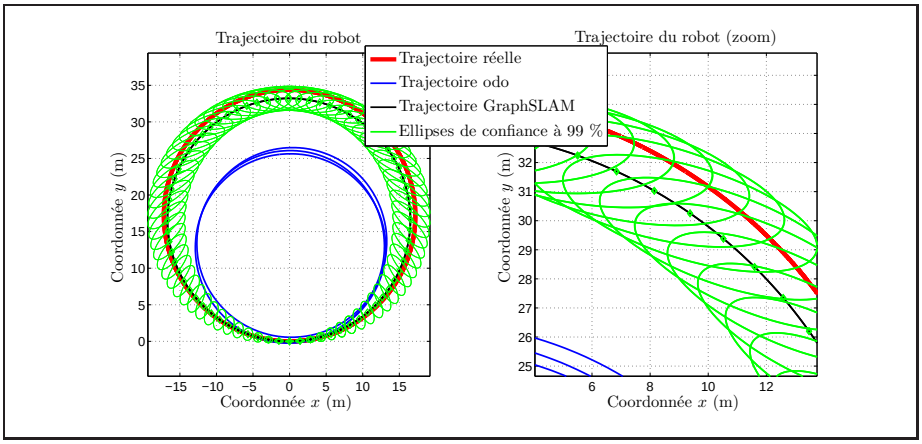

FIG. 33 – Résultats du GraphSLAM en ce qui concerne la trajectoire du robot : scénario 9 — Réglage à 16σ

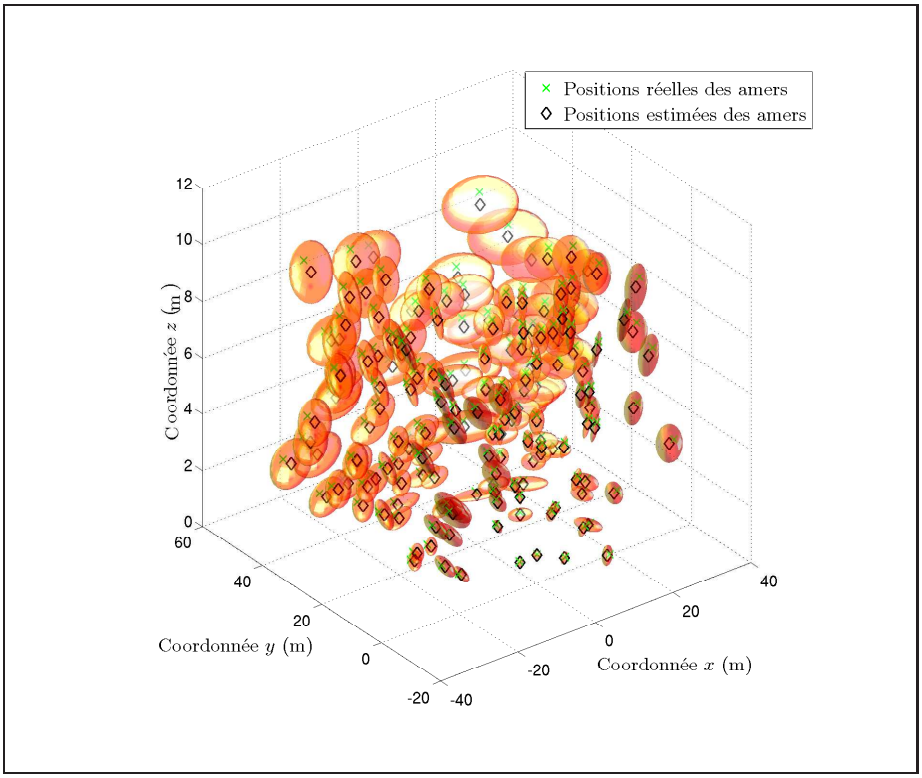

FIG. 34 – Résultats du GraphSLAM en ce qui concerne la position des amers : scénario 9 — Réglage à 16σ

7.4.2 SLAM par intervalles

Nous présentons ici les résultats concernant le SLAM par intervalles. Les figures 35 à 38 montrent les boîtes obtenues pour l'état du robot et pour 40 amers. On peut constater que les résultats sont consistants comme cela était prévu. La figure 38 montre par ailleurs que les résultats fournis ne sont pas trop pessimistes par rapport aux hypothèses. En effet, les erreurs générées dans le douzième scénario représentent le pire cas possible : elles sont à la limite des intervalles admissibles. Dans ce cas, on s'attend à ce que les boîtes obtenues avec l'algorithme de SLAM soient à la limite de l'inconsistance, ce qui est effectivement le cas. Le résultat concernant l'orientation du robot est encore plus impressionnant : l'erreur réelle est constamment égale à la limite inférieure concernant l'erreur admise.

Les figures 39 et 40 présentent les aires et volumes obtenus. On constate des variations entre les scenarii, bien que les réglages effectués soient identiques. Cela montre que la façon dont sont générées les erreurs peut avoir une influence assez conséquente sur le résultat. Néanmoins, les ordres de grandeurs obtenus sont comparables. On retrouve également les variations classiques dues au caractère circulaire de la trajectoire.

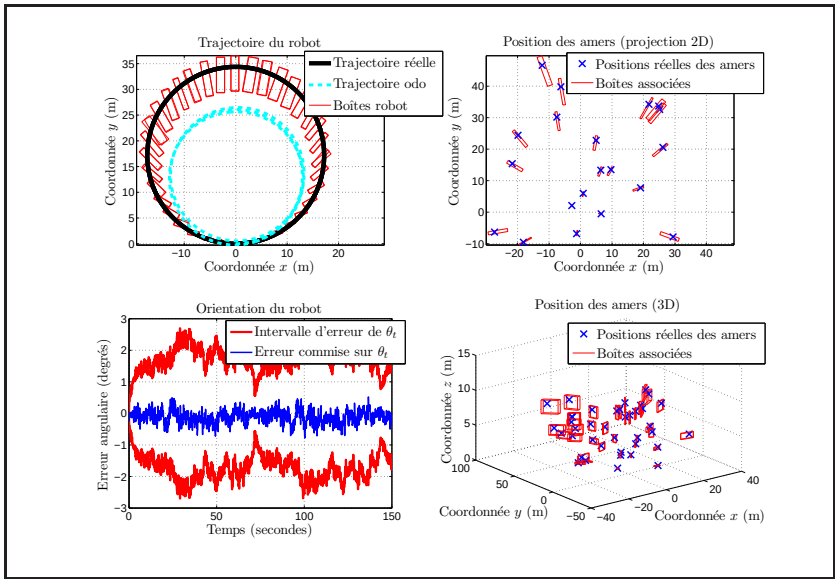

FIG. 35 – Résultats du SLAM par intervalles : scénario 9

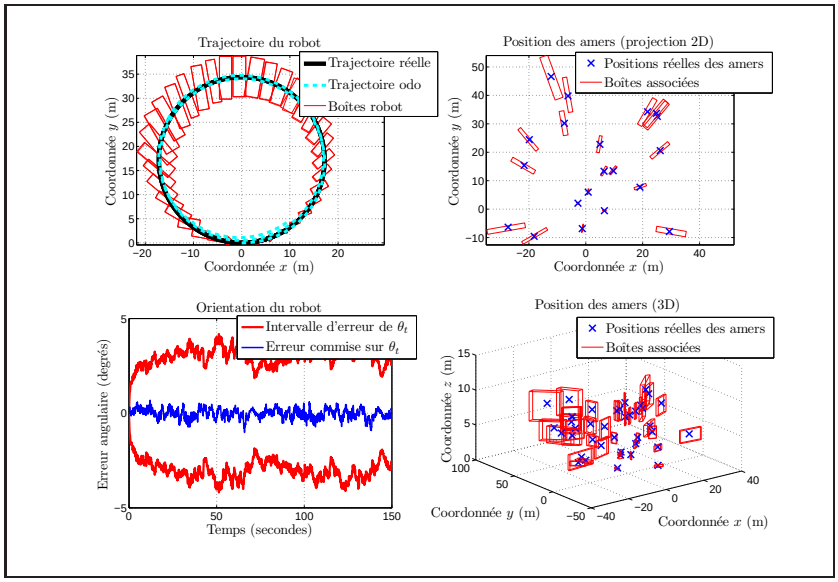

FIG. 36 – Résultats du SLAM par intervalles : scénario 10

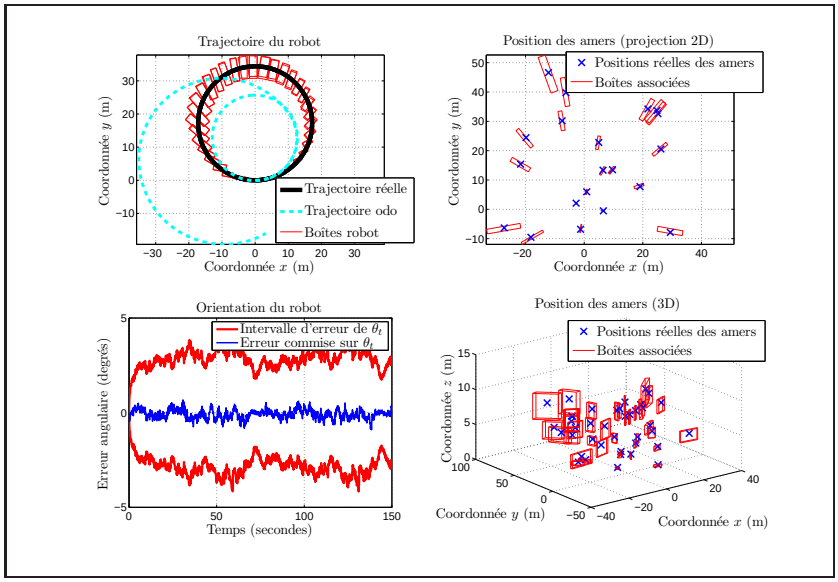

FIG. 37 – Résultats du SLAM par intervalles : scénario 11

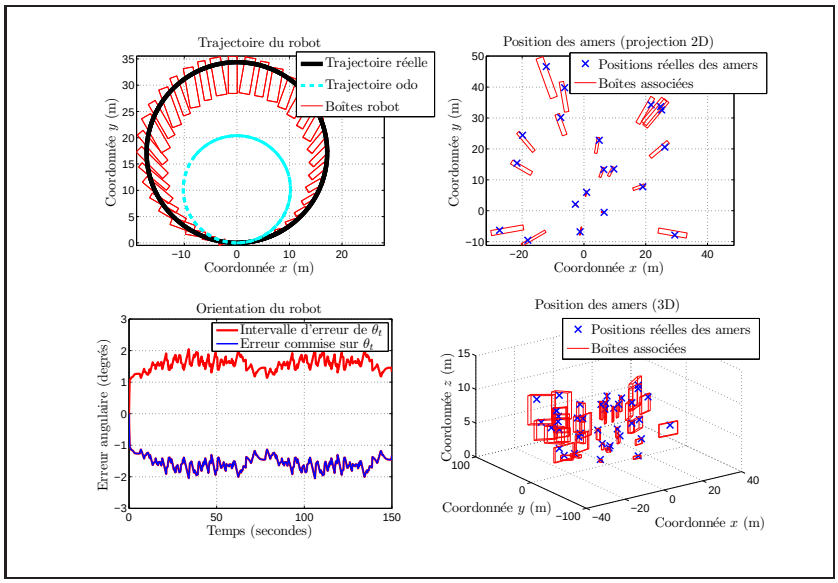

FIG. 38 – Résultats du SLAM par intervalles : scénario 12

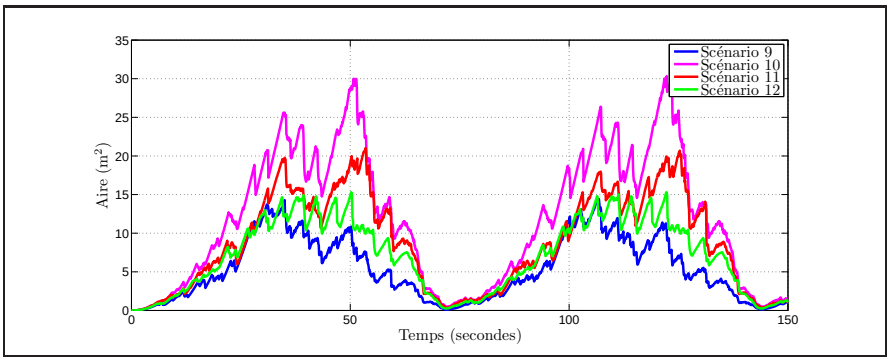

FIG. 39 – Aires des boîtes concernant les positions du robot : scenarii 9 à 12

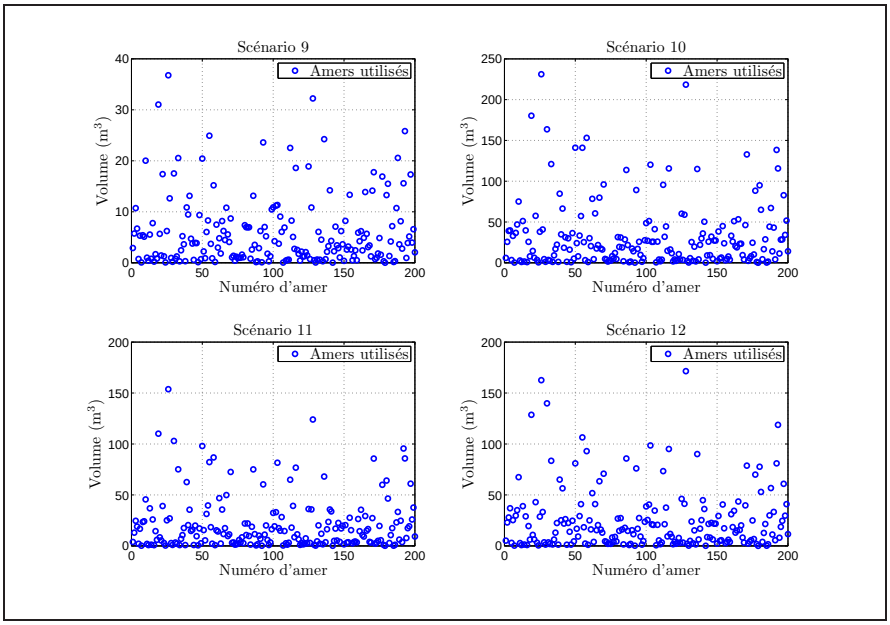

FIG. 40 – Volumes des boîtes concernant les positions des amers : scenarii 9 à 12

7.4.3 Conclusion quant au cas biaisé

Les résultats quant au cas biaisé montrent bien la supériorité du SLAM par intervalles dans ce cas. Alors qu'il est impossible de garantir la consistance du GraphSLAM, l'algorithme de SLAM par intervalles n'a posé aucun soucis en ce qui concerne le traitement des biais, qu'ils soient variables ou non.

7.5 Résultats en visibilité réduite

Dans tous les résultats présentés dans les scenarii 1 à 12, tous les amers sont visibles à chaque instant. Il s'agit de conditions très favorables pour les algorithmes. Nous présentons dans ce paragraphe quelques résultats obtenus lorsque tous les amers ne sont pas visibles simultanément. Nous avons repris le huitième scénario et avons testé quatre cas :

1. Les amers sont visibles à une distance infinie mais avec un angle de gisement compris entre -60deg et 60deg.
2. Les amers sont visibles à une distance infinie mais avec un angle de gisement compris entre -90deg et 90deg.
3. Les amers sont visibles à l'intérieur d'un cylindre centré sur le robot, de hauteur infinie et de rayon 17m.
4. Les amers sont visibles à l'intérieur d'un cylindre centré sur le robot, de hauteur infinie et de rayon 20m.

Les deux premiers cas peuvent se rapprocher d'un cas où on observe un environnement ouvert avec une caméra perspective (angle de vision réduit). Les deux autres, quant à eux, se rapprochent plus du cas d'un robot dans un environnement fermé (comme un couloir), mais avec une caméra omnidirectionnelle (pas de restriction quant à l'angle d'observation). Dans tous les cas, **l'association des données est connue**.

Les résultats concernant les aires et volumes des zones de confiance sont fournis dans les figures 41 à 44.

Les résultats concernant le GraphSLAM sont très proches de ce qui a déjà été observé dans le scénario 8 en ce qui concerne les aires associées à la position du robot. En ce qui concerne les volumes des ellipsoïdes de confiance, ils sont globalement assez proches des résultats initiaux. Certains amers ont néanmoins des enveloppes dont le volume est multiplié par 10 voire 100, mais cela reste marginal.[15] Par ailleurs, l'algorithme est toujours resté consistant.

[15] On peut noter que le volume de la majorité des amers reste très faible. Néanmoins, une multiplication du volume par 100 peut être vue comme une multiplication de la taille des axes par 4 ; la taille des axes des ellipsoïdes des amers reste donc du même ordre de grandeur.

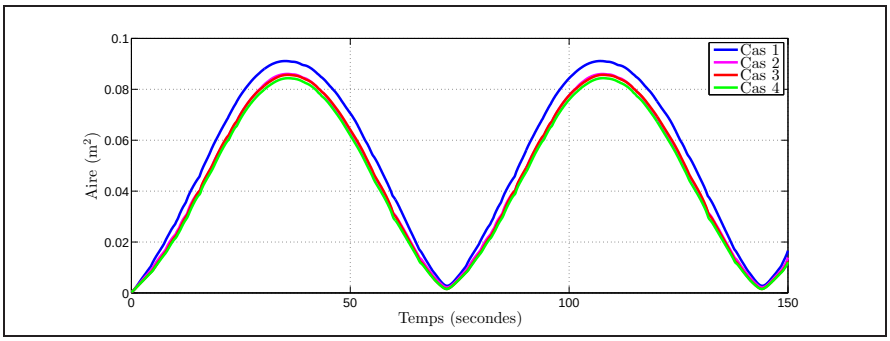

FIG. 41 – Aires des ellipses de confiance à 99% concernant les positions du robot : scénario 8 visibilité diminuée (GraphSLAM)

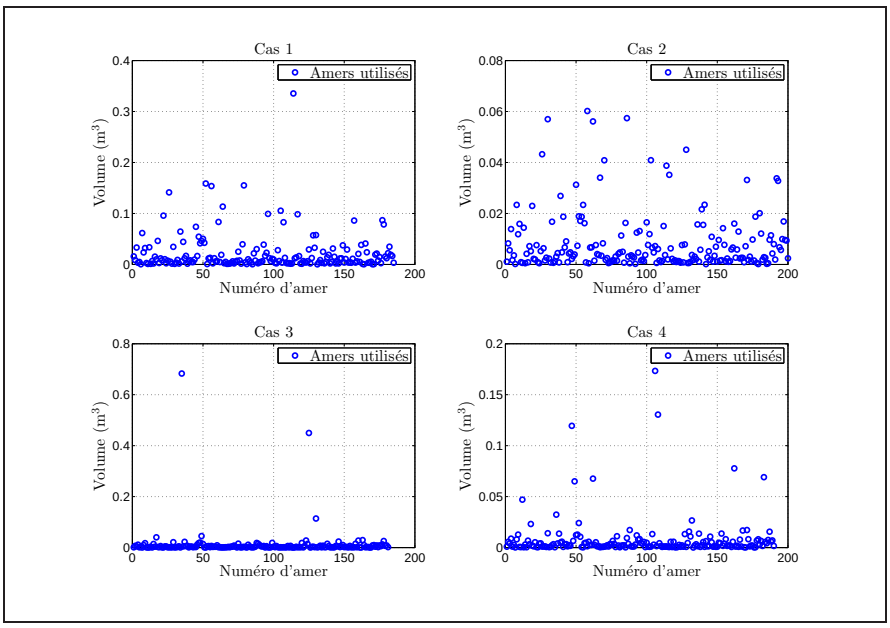

FIG. 42 – Volumes des ellipsoïdes de confiance à 99% concernant les positions des amers : scénario 8 visibilité diminuée (GraphSLAM)

En revanche, les résultats du SLAM par intervalles sont nettement dégradés. Ceci est par ailleurs illustré sur les figures 45 et 46 (à comparer avec les figures 26 et 27 pages 61-62). Les dégradations sont nettement visibles sur la qualité du positionnement des amers. L'estimation de la trajectoire du robot reste acceptable (notamment pour le cas 2). On remarque enfin l'apport de la fermeture de boucle sur la figure 45 : la partie gauche du cercle de la trajectoire (pourtant observée en dernier) possède des boîtes plus petites que la partie haute (pourtant observée avant), ceci grâce à la réobservation des amers initiaux.

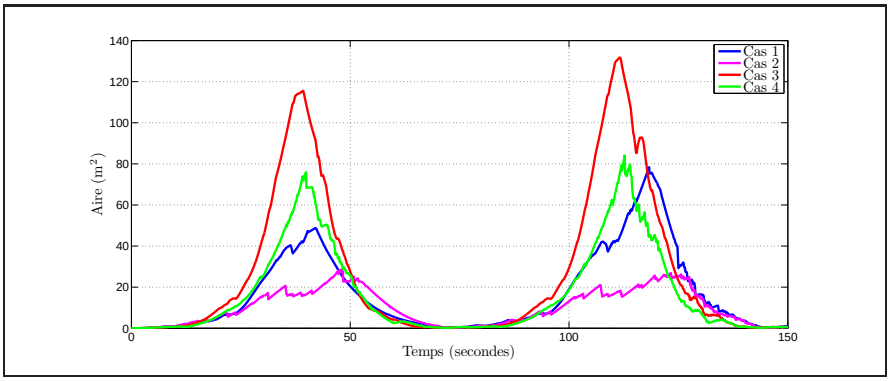

FIG. 43 – Aires des boîtes concernant les positions du robot : scénario 8 visibilité diminuée (SLAM par intervalles)

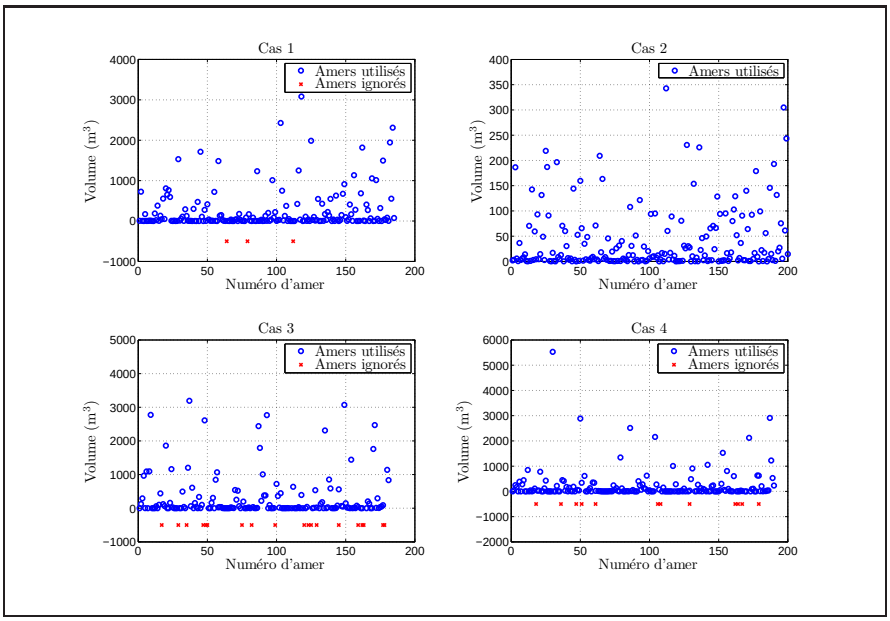

FIG. 44 – Volumes des boîtes concernant les positions des amers : scénario 8 visibilité diminuée (SLAM par intervalles)

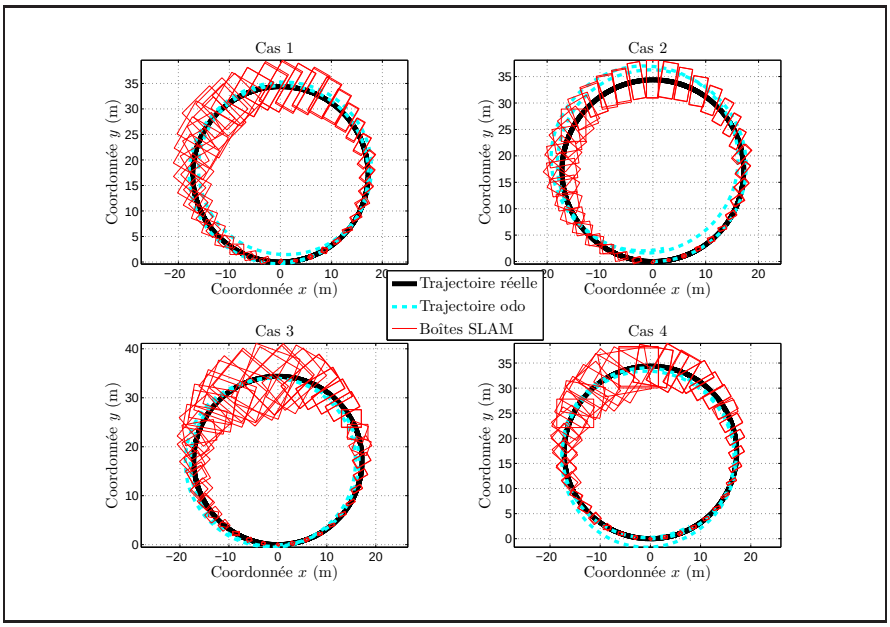

FIG. 45 – Résultats du SLAM par intervalles concernant la trajectoire du robot : scénario 8 visibilité diminuée — Les trajectoires odométriques sont différentes car les échantillons de bruit le sont. En revanche, les paramètres sont identiques et les bruits toujours centrés. Le résultat n'est donc pas impacté par l'utilisation d'échantillons différents.

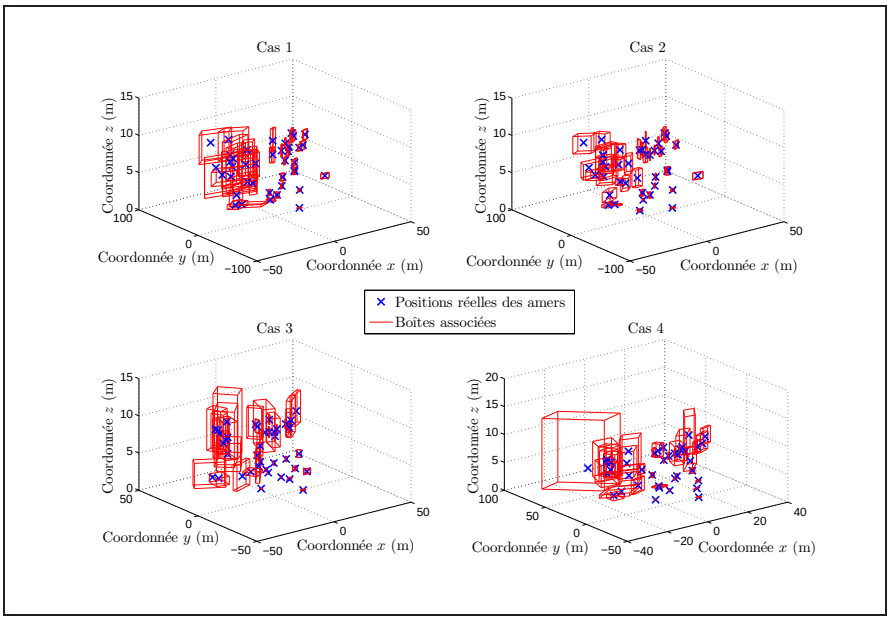

FIG. 46 – Résultats du SLAM par intervalles concernant les positions des amers : scénario 8 visibilité diminuée

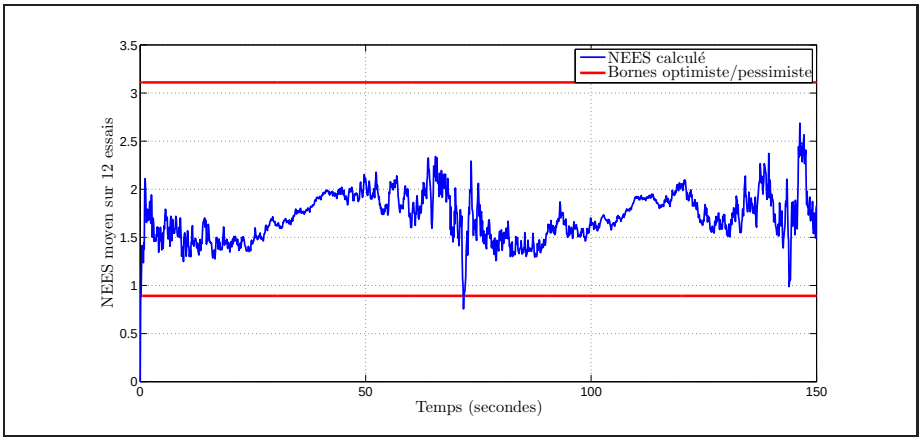

FIG. 47 – Consistance du GraphSLAM par le calcul de l'erreur quadratique normalisée

7.6 Remarque sur la consistance du GraphSLAM

Nous avons vu dans les paragraphes 7.2 et 7.3 que l'algorithme de GraphSLAM est consistant lorsque les erreurs sont centrées et le filtre bien réglé (scenarii 1 à 8 et les 4 essais à visibilité réduite). Par ailleurs, la taille des ellipses semble raisonnable. Nous proposons de confirmer ce dernier résultat en calculant l'estimation normalisée de l'erreur quadratique (NEES[16]) pour les positions du robot. Cela revient à calculer pour chaque scénario la variable ϵ_t définie ainsi :

$$\epsilon_t = \left(\mathbf{x}_{t,Pos}^{Vrai} - \mathbf{x}_{t,Pos}\right)^T \mathbf{\Sigma}_{t,Pos}^{-1} \left(\mathbf{x}_{t,Pos}^{Vrai} - \mathbf{x}_{t,Pos}\right) \tag{82}$$

où $\mathbf{x}_{t,Pos}^{Vrai}$ désigne la vraie position du robot à l'instant t (x_t et y_t), $\mathbf{x}_{t,Pos}$ la position fournie par le GraphSLAM et $\mathbf{\Sigma}_{t,Pos}$ la matrice de variances-covariances associée.

Lorsque le filtre est consistant, la variable ϵ_t suit une loi du χ^2 à $\dim(\mathbf{x}_{t,Pos})$ degrés de liberté. Pour que les résultats soient plus significatifs, nous proposons de moyenner cette variable sur l'ensemble des 12 scenarii déclarés consistants. Cette méthode est celle utilisée dans [2] pour montrer l'inconsistance de l'EKF-SLAM.

Soit ϵ_t^i la valeur de ϵ_t associé au scénario i et à l'instant t. Si le filtre est consistant, la variable $\sum_{i=1}^{12} \epsilon_t^i$ suit une loi du χ^2 à $12\dim(\mathbf{x}_{t,Pos})$ degrés de liberté (donc 24 degrés de liberté dans notre cas). Ainsi, 95% des réalisations de $\frac{1}{12}\sum_{i=1}^{12} \epsilon_t^i$ sont comprises dans l'intervalle $[0.892, 3.11]$. Si le calcul du NEES moyen est majoritairement en dessous de 0.892, le filtre sera trop pessimiste. S'il est majoritairement au dessus de 3.11, il sera trop optimiste et donc inconsistant.

La figure 47 montre que l'algorithme de GraphSLAM est consistant sans être trop pessimiste. Ceci confirme les résultats observés dans les paragraphes 7.2 et 7.3.

[16] Normalised Estimation Error Squared

7.7 Temps de calcul

Nous effectuons dans ce paragraphe quelques remarques concernant les temps de calcul des deux algorithmes. Nous n'avons pas étudié en détails la complexité des algorithmes (qui ne sont par ailleurs pas optimisés). Ce paragraphe est donc constitué de remarques permettant de comprendre les facteurs qui interviennent dans les temps de calcul.

7.7.1 GraphSLAM

L'algorithme de GraphSLAM nécessite le remplissage incrémental de la matrice d'information et du vecteur d'information. Cette opération ne nécessite que peu de temps. L'étape couteuse en temps de calcul est l'inversion du système pour retrouver les matrices de variances-covariances et les espérances associées aux différents états. Etant donné que la matrice d'information est globalement éparse, les fonctions dédiées de `Matlab` sont utilisées. Ce temps augmente avec le nombre d'amers et le nombre de fois où ils sont observés (matrice d'information plus dense). Il y a enfin une perte de temps supplémentaire liée au fait que nous ne traitons pas l'ensemble de la trajectoire dès la première itération afin d'éviter d'avoir une trajectoire initiale trop mauvaise (cf. paragraphe 4.3.1 page 25). Le nombre de pas nécessaire pour traiter l'ensemble de la trajectoire grandit donc linéairement avec la variance de l'erreur sur la mesure de vitesse de rotation du robot.

7.7.2 SLAM par intervalles

Le SLAM par intervalles ne nécessite pas l'inversion de matrices. Il s'agit ici de calculer itérativement l'intersection d'intervalles. Le temps de calcul mis augmente avec le nombre d'amers vus ainsi que le nombre de positions traitées. Mais il est aussi dépendant de la taille des intervalles associés aux mesures. Le nombre d'itérations nécessaire pour converger est en effet beaucoup plus faible lorsque les intervalles initiaux ont une taille très faible. Enfin, la finesse angulaire que l'on souhaite sur l'orientation des boîtes des amers a un impact sur les temps de calcul. Ce dernier augmente linéairement avec le nombre d'angles testés.

7.7.3 En pratique

En pratique, le SLAM par intervalles s'est montré en général plus rapide que le GraphSLAM. Par exemple, les temps d'exécution pour des conditions " normales " (scénario 8) sont de 1000s pour le GraphSLAM contre 800s pour le SLAM par intervalles.

Enfin, aucun des deux algorithmes n'est utilisable en temps réel sans modifications. Une adaptation envisageable serait de ne traiter que des **fenêtres locales**. Ce dernier aspect nécessite une étude à part entière des nouvelles propriétés des algorithmes dans ces conditions.

7.8 Conclusion quant aux simulations

Nous avons présenté dans cette section les résultats obtenus par les deux algorithmes de SLAM en simulation. Ceux-ci ont permis de mettre en évidence le **bon comportement**

du GraphSLAM lorsque les erreurs sont centrées et gaussiennes ou centrées et uniformes. Plus généralement, il semble que l'hypothèse d'erreur centrée et indépendante dans le temps suffise pour que le GraphSLAM soit consistant. Néanmoins, il faut que les paramètres de variances soient correctement choisis.

Les simulations ont également mis en évidence le comportement moyen du SLAM par intervalles dans le cas gaussien. Le réglage de l'algorithme n'étant pas évident (l'hypothèse gaussienne n'implique pas de borne sûre sur l'erreur), le SLAM par intervalles a en général conduit à des résultats trop pessimistes. Néanmoins, ceux-ci sont restés consistants. En revanche, le SLAM par intervalles a montré un bien meilleur comportement dans le cas où l'erreur est parfaitement bornée (et où les bornes sont connues), même s'il a continué a rester **trop pessimiste par rapport au GraphSLAM**.

La **supériorité du SLAM par intervalles a été mise en évidence lorsque les erreurs sur les capteurs sont biaisées**. Alors que le GraphSLAM est inconsistant, le SLAM par intervalles a retourné des résultats consistants dont le niveau de pessimisme semble cohérent avec le type d'erreur appliqué. Il convient néamoins de rappeler une des hypothèses du SLAM par intervalles (hypothèse citée dans le théorème 73 page 31) : les intervalles utilisés en entrées sont consistants. Ceci signifie qu'**il n'y a aucun outlier**. Ainsi, la présence d'une seule mesure aberrante peut suffire à générer des intersections vides empêchant la bonne exécution de l'algorithme. Ce problème est aussi présent avec le GraphSLAM, mais dans une moindre mesure; le support de la densité de probabilité gaussienne n'étant pas borné, la présence de quelques mesures aberrantes[17] peut être acceptable. Ainsi, il est recommandé voire indispensable (pour le SLAM par intervalle) de pré-traiter les mesures avec un algorithme robuste de rejet des points aberrants (cet aspect n'est pas abordé dans ce rapport).

En conclusion, ces simulations ne permettent pas de désigner un algorithme qui soit meilleur dans tous les cas de figure. Si les erreurs de nos capteurs sont centrées, le choix d'une méthode probabiliste semble plus judicieux. Mais si ce n'est pas forcément le cas, le SLAM par intervalles donnera de meilleurs résultats. Il faut néanmoins que les bornes des erreurs ne soient pas trop grandes pour que le résultat reste exploitable. Dans le cas où les erreurs ne sont pas centrées, l'algorithme de GraphSLAM donnera en général des résultats exploitables[18] mais inconsistants.

8 Conclusion et perspectives

Nous avons présenté dans ce document deux algorithmes de SLAM. Le GraphSLAM est basé sur une approche probabiliste alors que le SLAM par intervalles est basé sur une approche déterministe. Nous avons étudié en détails l'application de ces algorithmes au cas du SLAM 2D/3D. Nous avons tenté d'apporter une solution à tous les problèmes pratiques posés, les deux principaux étant l'initialisation des amers (GraphSLAM et SLAM par

[17] La fréquence des points aberrants doit être compatible avec la loi gaussienne.

[18] Dans le sens où la position retournée est cohérente avec la réalité. Par exemple, on retrouvait bien la caractère circulaire de la trajectoire dans les simulations. Même si celle-ci n'est pas exacte et que l'erreur prédite n'est pas bonne, il n'y a pas une divergence totale.

intervalles) et le problème d'orientation des boîtes (SLAM par intervalles). Une étude en simulation sur une trajectoire simple a permis de mettre en évidence les principaux avantages et inconvénients des deux algorithmes.

La première suite à donner à ces travaux est l'extension au cas 3D complet. Ce passage fait largement augmenter la complexité des algorithmes. En effet, on double directement le nombre de degrés de liberté associés à la position du robot (on passe de 3 à 9). Cela nécessite en outre redéfinir ce qui remplacera l'odométrie (si elle est remplacée) pour effectuer la prédiction des 6 degrés de liberté du robot.

La seconde extension prévue à ces travaux est l'étude de l'intérêt des cartes locales. Nous pensons actuellement que celles-ci peuvent améliorer les résultats du SLAM par intervalles. Par exemple, lorsque l'incertitude sur la position du robot est grande, l'incertitude des amers proches de cette position est grande aussi. Ainsi, l'incertitude sur certains paramètres locaux de la carte (comme la distance entre amers) risque d'être très élevée. Il serait peut-être possible de la rendre plus faible en créant une carte locale (car l'incertitude sur la position du robot serait faible, ce qui impliquerait une incertitude plus faible sur la position des amers dans la carte locale). Ceci peut être vu comme une façon " manuelle " de prendre en compte les corrélations existants entre les amers (l'équivalent des covariances entre amers différents n'existe pas encore dans notre approche de SLAM par intervalles). Nous comptons donc évaluer l'intérêt et les performances d'une telle approche. Par ailleurs, ce type d'approche commence à être étudié dans le cas du SAM ([15]).

Enfin, il est prévu d'appliquer les algorithmes sur des données réelles, afin de compléter l'évaluation faite en simulation. Pour cela, nous utiliserons des données acquises avec le robot ANIS (traversée d'un couloir). Le capteur de mesure utilisé est une caméra omnidirectionnelle.

Références

[1] Tim Bailey and Hugh Durrant-Whyte. Simultaneous localisation and mapping (slam) : Part ii - state of the art. *Robotics and Automation Magazine*, 2006.

[2] Tim Bailey, Juan Nieto, Jose Guivant, Michael Stevens, and Eduardo Nebot. Consistency of the ekf-slam algorithm. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006.

[3] Tim Bailey, Juan Nieto, and Eduardo Nebot. Consistency of the fastslam algorithm. In *IEEE International Conference on Robotics and Automation*, 2006.

[4] A. J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *International Conference on Computer Vision*, 2003.

[5] Frank Dellaert and Michael Kaess. The GraphSLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures. *International Journal of Robotics Research*, 2006.

[6] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localisation and mapping (slam) : Part i the essential algorithms. *Robotics and Automation Magazine*, 2006.

[7] Shoudong Huang and Gamini Dissanayake. Convergence Analysis for Extended Kalman filter based SLAM. In *International Conference on Robotics and Automation (ICRA)*, 2006.

[8] Simon J. Julier and Jeffrey K. Uhlmann. A counter example to the theory of simultaneous localization and map building. In *International Conference on Robotics and Automation (ICRA)*, 2001.

[9] Christopher Mei. *Couplage Vision Omnidirectionnelle et Télémétrie Laser pour la Navigation en Robotique – Laser-Augmented Omnidirectional Vision for 3D Localisation and Mapping.* PhD thesis, Ecole des Mines de Paris, INRIA Sophia Antipolis, 2007.

[10] M Montemerlo, S Thrun, D Koller, and B Wegbreit. FastSLAM : A Factored Solution to the Simultaneous Localization and Mapping Problem. In *AAAI National Conference on Artificial Intelligence*, pages 593–598, 2002.

[11] M Montemerlo, S Thrun, D Koller, and B Wegbreit. FastSLAM 2.0 : An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping that Provably Converges. In *International Joint Conference on Artificial Intelligence*, pages 1151–1156, 2003.

[12] Michael Montemerlo. *FastSLAM : A Factored Solution to the Simultaneous Localization and Mapping Problem with Unknown Data Association.* PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, July 2003.

[13] J.M.M. Montiel, J. Civera, and A.J. Davison. Unified inverse depth parametrization for monocular slam. In *Robotics Science and Systems Conference. Philadelphia.*, 2006.

[14] RE Moore. *Methods and Applications of Interval Analysis.* Sociey for Industrial and Applied Mathematics, 1979.

[15] Kai Ni, Drew Steedly, and Frank Dellaert. Tectonic SAM : Exact, Out-of-Core, Submap-Based SLAM. In *International Conference on Robotics and Automation (ICRA)*, 2007.

[16] Robert Sim. Stable exploration for bearing-only slam. In *International Conference on Robotics and Automation (ICRA)*, 2005.

[17] Randall Smith and Peter Cheeseman. On the representation and estimation of spatial uncertainty. *International Journal of Robotics Research*, 5(4) :56–68, 1987.

[18] Sebastian Thrun and Michael Montemerlo. The GraphSLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures. *The International Journal of Robotics Research*, 25(5-6) :403–429, May-June 2006.

ANNEXE

A Coefficients associés aux ellipses de confiance

Nous présentons dans cette section des coefficients associés aux ellipses et ellipsoïdes de confiance à $P\%$.

A.1 Généralités

Nous présentons dans cette section des généralités concernant les enveloppes de confiance à $P\%$ dans le cas gaussien. Nous rappelons d'abord brièvement les raisons du choix d'ellipsoïdes de confiance. Nous effectuons ensuite quelques remarques concernant la forme utilisée pour les matrices de variances-covariances.

Soit p une densité de probabilité gaussienne d'espérance μ (avec $\dim \mu = n$) et de matrice de variances-covariances $\mathbf{\Sigma}$. On a alors :

$$p(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^n \det \Sigma}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T \mathbf{\Sigma}^{-1}(\mathbf{x}-\mu)\right) \tag{83}$$

D'après l'équation 83, les zones isoprobables sont des ellipsoïdes centrés sur μ d'équation :

$$(\mathbf{x}-\mu)^T \mathbf{\Sigma}^{-1}(\mathbf{x}-\mu) = K^2 \tag{84}$$

où K est une constante positive.
Il apparaît " naturel " de choisir comme frontière de zone de confiance une zone équiprobable. En conséquence, les zones de confiance cherchées sont des ellipsoïdes vérifiant l'équation 84. Lorsqu'on connaît μ, K et $\mathbf{\Sigma}$, la probabilité que $\mathbf{x}$ soit à l'intérieur de l'ellipsoïde défini en 84 est :

$$P = \mathrm{P}\left((\mathbf{x}-\mu)^T \mathbf{\Sigma}^{-1}(\mathbf{x}-\mu) \leq K^2\right) \tag{85}$$

On a donc :

$$P = \iint_{ellipse} p(\mathbf{x})\,\mathbf{dx} \qquad \text{ou} \qquad P = \iiint_{ellipsoïde} p(\mathbf{x})\,\mathbf{dx} \tag{86}$$

La valeur de K choisie dépend de la valeur de P souhaitée. Par ailleurs, on a $\lim_{P\to 0} K = 0$ et $\lim_{P\to 1} K = +\infty$.

A.2 Cas particulier traité

Dans les sections suivantes, nous chercherons à calculer l'aire (ou le volume) des ellipses (ou ellipsoïdes) de confiance à $P\%$ en fonction de P, de la matrice de variances-covariances et de l'espérance de la densité de probabilité considérée. Il est clair que la valeur cherchée est indépendante du centre de l'ellipsoïde, et donc de μ. Dans la suite de cette section, nous effectuerons tous nos calculs avec $\mu = 0$.

De plus, $\mathbf{\Sigma}$ est une matrice symétrique réelle définie positive (propriété d'une matrice de variances-covariances non dégénérée). En conséquence, elle est diagonalisable dans une base orthonormée. On a donc :

$$\mathbf{\Sigma} = \mathbf{P}^T \mathbf{D} \mathbf{P} \tag{87}$$

où $\mathbf{D}$ est une matrice diagonale (à coefficients positifs) et $\mathbf{P}$ est une matrice vérifiant $\mathbf{P}^T\mathbf{P} = \mathbf{I}$. Une nouvelle équation de l'ellipse définie en 84 est :

$$(\mathbf{Px})^T \mathbf{D}^{-1} (\mathbf{Px}) = K^2 \tag{88}$$

Etant donné que la matrice $\mathbf{P}$ ne modifie pas la norme de $\mathbf{x}$, le volume de l'ellipsoïde paramétré par μ, $\mathbf{\Sigma}$ et K (équation 84) est identique au volume de l'ellipsoïde paramétré par $\mathbf{0}$, $\mathbf{D}$ et K.

En conséquence, les études qui suivent seront simplifiés par l'utilisation de la matrice diagonale associée à $\mathbf{\Sigma}$ et de $\mu = 0$.

A.3 Cas 2D : ellipses de confiance

Nous considérons dans ce paragraphe le cas 2D. Soient σ_1^2 et σ_2^2 les valeurs propres de la matrice de variances-covariances considérée. Nous proposons d'abord de calculer K en fonction de P, puis d'en déduire le volume associé.

A.3.1 Calcul de K

Considérons d'abord K fixé. Cherchons la probabilité P d'appartenir à l'intérieur de l'ellipse de confiance définie par $\mathbf{\Sigma}$ et K. Celle-ci est donnée par :

$$P = \iint_{ellipse} \frac{1}{\sqrt{(2\pi)^2 \det \mathbf{\Sigma}}} \exp\left(-\frac{1}{2}\mathbf{x}^T\mathbf{D}^{-1}\mathbf{x}\right) \mathbf{dx} \tag{89}$$

En notant x_1 et x_2 les composantes du vecteur $\mathbf{x}$, on obtient :

$$P = \iint_{ellipse} \frac{1}{2\pi\sigma_1\sigma_2} \exp\left(-\frac{x_1^2}{2\sigma_1^2} - \frac{x_2^2}{2\sigma_2^2}\right) dx_1 dx_2 \tag{90}$$

soit :

$$P = \frac{1}{2\pi\sigma_1\sigma_2} \iint_{ellipse} \exp\left(-\frac{x_1^2}{2\sigma_1^2} - \frac{x_2^2}{2\sigma_2^2}\right) dx_1 dx_2 \tag{91}$$

soit désormais le changement de variable $(x_1, x_2) \mapsto (\rho, \theta)$ tel que :

$$\left\{ \begin{array}{lcl} x_1 & = & \rho\sigma_1 \cos\theta \\ x_2 & = & \rho\sigma_1 \sin\theta \end{array} \right. \tag{92}$$

Ce changement de variable permet de parcourir entièrement (et uniquement) l'intérieur de l'ellipse si on fait varier ρ entre 0 et K et si on fait varier θ entre 0 et 2π. De plus, le jacobien de ce changement de variable vaut $\rho\sigma_1\sigma_2$. L'équation 92 devient donc :

$$\begin{aligned} P &= \frac{1}{2\pi\sigma_1\sigma_2}\int_0^{2\pi}\int_0^K \sigma_1\sigma_2\rho\exp\left(-\frac{1}{2}\rho^2\right)d\rho d\theta \\ \Leftrightarrow P &= \frac{2\pi\sigma_1\sigma_2}{2\pi\sigma_1\sigma_2}\int_0^K \exp\left(-\frac{1}{2}\rho^2\right)d\rho \\ \Leftrightarrow P &= \left[-e^{-\rho^2/2}\right]_0^K \\ \Leftrightarrow P &= 1-e^{-K^2/2} \end{aligned} \tag{93}$$

On peut alors déduire de l'équation 93 :

$$\boxed{K=\sqrt{-2\log\left(1-P\right)}} \tag{94}$$

A.3.2 Calcul de l'aire de l'ellipse

Nous avons vu au paragraphe précédent comment calculer le paramètre K de l'ellipse en fonction de la probabilité P souhaitée. Nous allons désormais en déduire l'aire de l'ellipse.

L'équation de l'ellipse paramétrée par $\mathbf{D}$ et K est :

$$\mathbf{x}^T\mathbf{D}^{-1}\mathbf{x}=K^2 \tag{95}$$

soit :

$$\frac{x_1^2}{\left(K\sigma_1\right)^2}+\frac{x_2^2}{\left(K\sigma_2\right)^2}=1 \tag{96}$$

Il s'agit donc d'une ellipse dont les demi-axes valent $K\sigma_1$ et $K\sigma_2$. Son aire vaut donc $\pi K^2\sigma_1\sigma_2$. Sachant que $\det\mathbf{\Sigma}=\det\mathbf{D}=\sigma_1^2\sigma_2^2$, on en déduit l'aire de l'ellipse à $P\%$ associée à la matrice de variances-covariances $\mathbf{\Sigma}$:

$$\boxed{\mathcal{A}=-2\pi\log\left(1-P\right)\sqrt{\det\mathbf{\Sigma}}} \tag{97}$$

pour $P=99\%$, on obtient :

$$\boxed{\mathcal{A}=28.9351\sqrt{\det\mathbf{\Sigma}}} \tag{98}$$

A.4 Cas 3D : ellipsoïdes de confiance

Nous présentons ici le cas 3D. Le principe est similaire au calcul effectué en A.3. Nous allons d'abord tenter de trouver K en fonction de P, puis en déduire le volume de l'ellipsoïde associé.

A.4.1 Calcul de K

Considérons d'abord K fixé. Cherchons la probabilité P d'appartenir à l'intérieur de l'ellipsoïde de confiance défini par $\mathbf{\Sigma}$ et K. Celle-ci est donnée par :

$$P = \iiint_{ellipsoïde} \frac{1}{\sqrt{(2\pi)^3 \det \mathbf{\Sigma}}} \exp\left(-\frac{1}{2}\mathbf{x}^T\mathbf{D}^{-1}\mathbf{x}\right)\mathbf{dx} \tag{99}$$

En notant x_1, x_2 et x_3 les composantes du vecteur $\mathbf{x}$, on obtient :

$$P = \frac{1}{2\pi\sqrt{2\pi}\sigma_1\sigma_2\sigma_3} \iiint_{ellipsoïde} \exp\left(-\frac{x_1^2}{2\sigma_1^2} - \frac{x_2^2}{2\sigma_2^2} - \frac{x_2^2}{2\sigma_3^2}\right) dx_1 dx_2 dx_3 \tag{100}$$

soit désormais le changement de variables $(x_1, x_2, x_3) \mapsto (\rho, \theta, \varphi)$ tel que :

$$\left\{\begin{array}{lcl} x_1 &=& \rho\sigma_1 \cos\varphi\cos\theta \\ x_2 &=& \rho\sigma_1 \cos\varphi\sin\theta \\ x_3 &=& \rho\sigma_3 \sin\varphi \end{array}\right. \tag{101}$$

Ce changement de variables permet de parcourir entièrement (et uniquement) l'intérieur de l'ellipsoïde si on fait varier ρ entre 0 et K, θ entre 0 et 2π et φ entre $-\pi/2$ et $\pi/2$. On peut montrer que le jacobien de ce changement de variables est égal à $\sigma_1\sigma_2\sigma_3\rho^2\cos\varphi$. Ainsi, l'équation 100 devient :

$$P = \frac{1}{2\pi\sqrt{2\pi}} \int_0^{2\pi}\int_{-\pi/2}^{\pi/2}\int_0^K \rho^2 \exp\left(-\frac{1}{2}\rho^2\right)\cos\varphi d\rho d\varphi d\theta \tag{102}$$

soit :

$$P = \frac{1}{2\pi\sqrt{2\pi}} \underbrace{\int_0^{2\pi} d\theta}_{2\pi} \underbrace{\int_{-\pi/2}^{\pi/2} \cos\varphi d\varphi}_{2} \int_0^K \rho^2 \exp\left(-\frac{1}{2}\rho^2\right) d\rho \tag{103}$$

on a donc :

$$P = \frac{\sqrt{2\pi}}{\pi} \int_0^K \rho^2 \exp\left(-\frac{1}{2}\rho^2\right) d\rho \tag{104}$$

L'intégrale à calculer dans l'équation 104 n'est pas triviale. On choisit d'effectuer une intégration par parties :

$$\begin{aligned} \int_0^K \rho^2 \exp\left(-\frac{1}{2}\rho^2\right) d\rho &= \int_0^K \rho \cdot \rho e^{-\rho^2/2} d\rho \\ &= \left[-\rho e^{-\rho^2/2}\right]_0^K - \int_0^K -e^{-\rho^2/2} d\rho \\ &= -Ke^{-K^2/2} + \int_0^K e^{-\rho^2/2} d\rho \end{aligned} \tag{105}$$

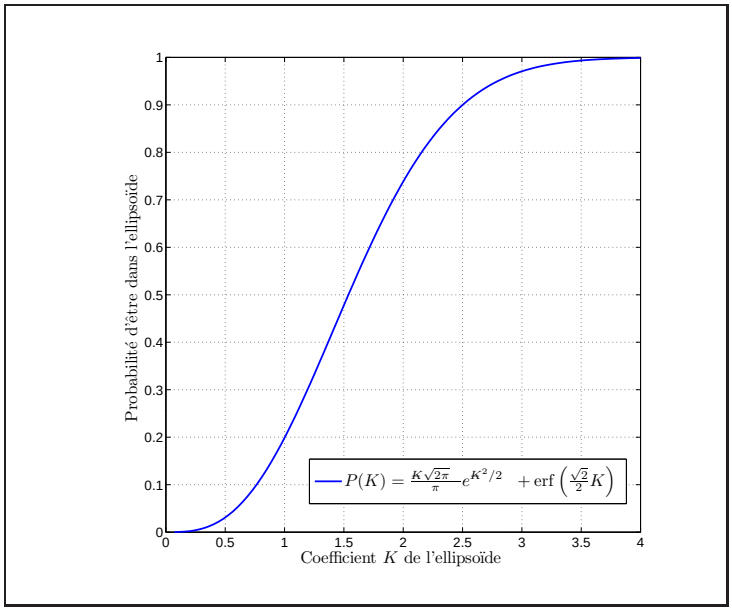

FIG. 48 – Probabilité d'être à l'intérieur de l'ellipsoïde en fonction de K

L'intégrale restante dans l'équation 105 n'est pas calculable analytiquement avec les fonctions usuelles. Ce type d'intégrale a conduit à la définition de la fonction nommée *fonction d'erreur* et notée erf. Sa forme analytique est :

$$\begin{array}{rccc} \mathrm{erf}: & \mathbb{R} & \longrightarrow & [-1\ \ 1] \\ & x & \longmapsto & \dfrac{2}{\sqrt{2\pi}} \displaystyle\int_0^x e^{-t^2}\,dt \end{array} \tag{106}$$

Un changement de variable approprié permet d'obtenir :

$$\int_0^K -e^{-\rho^2/2} d\rho = \frac{\sqrt{2\pi}}{2} \mathrm{erf}\left(\frac{\sqrt{2}}{2}K\right) \tag{107}$$

En utilisant les équations 104, 105 et 107, on obtient :

$$\boxed{P = \frac{-K\sqrt{2\pi}}{\pi} e^{-K^2/2} + \mathrm{erf}\left(\frac{\sqrt{2}}{2}K\right)} \tag{108}$$

Une étude de la fonction associant P à K permettrait de montrer que la fonction est bien définie sur $[0\ ,\ +\infty]$, croissante, continue, bijective et avec $\lim_{P\to 0} K = 0$ et $\lim_{P\to 1} K = +\infty$. Le graphique de P est donné sur la figure 48.

Mais l'expression de P donnée dans l'équation 108 ne permet pas de retrouver K analytiquement. Nous avons donc utilisé un solveur numérique (Maple) pour trouver le coefficient adéquat. Pour $P = 99\%$, on trouve :

$$\boxed{K = 3.368214175}$$

A.5 Calcul du volume de l'ellipsoïde

Comme cela a été fait pour le cas 2D, on peut montrer que le volume cherché est égal au volume d'un ellipsoïde dont les demi-axes sont $K\sigma_1$, $K\sigma_2$ et $K\sigma_3$. Son volume est donc : $\frac{4}{3}\pi K^3\sigma_1\sigma_2\sigma_3$. Sachant par ailleurs que $\det\mathbf{\Sigma} = \det\mathbf{D} = \sigma_1^2\sigma_2^2\sigma_3^2$, on en déduit l'aire de l'ellipse à $P\%$ associée à la matrice de variances-covariances $\mathbf{\Sigma}$:

$$\boxed{\mathcal{V} = \tfrac{4}{3}\pi K^3\sqrt{\det\mathbf{\Sigma}}} \tag{109}$$

pour $P = 99\%$, on obtient :

$$\boxed{\mathcal{V} = 160.0618\sqrt{\det\mathbf{\Sigma}}} \tag{110}$$

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399