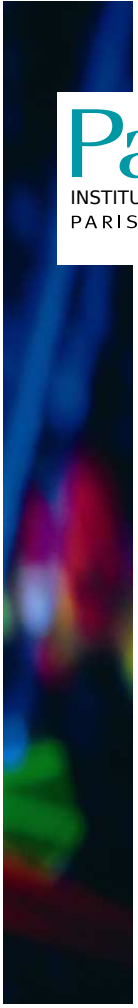
Contributions aux méthodes de localisation et cartographie simultanées par vision omnidirectionnelle

Cyril Joly



ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

MINES ParisTech

École doctorale n°84 :
Sciences et technologies de l'information et de la communication

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

l'École nationale supérieure des mines de Paris

Spécialité « Informatique temps-réel, robotique et automatique »

présentée et soutenue publiquement par

Cyril JOLY

le 30 juin 2010

Contributions aux méthodes de localisation et cartographie simultanées par vision omnidirectionnelle

Directeur de thèse : **Patrick RIVES**

Jury

Jean-Paul MARMORAT, Directeur de recherche, CMA, MINES ParisTech	Président
Philippe BONNIFAIT, Professeur, Heudiasyc, UTC	Rapporteur
Luc JAULIN, Professeur, Développement des Technologies Nouvelles, ENSIETA	Rapporteur
Jacques BLANC-TALON, Docteur, DGA	Examinateur
Roland CHAPUIS, Professeur, LASMEA, Université Blaise Pascal	Examinateur
El Mustapha MOUADDIB, Professeur, Laboratoire MIS, UPJV	Examinateur
François PEYRET, Directeur de recherche, GEOLOC, LCPC Nantes	Examinateur
Patrick RIVES, Directeur de recherche, ARobAS, INRIA Sophia Antipolis-Méditerranée	Directeur de thèse

T H È S E

MINES ParisTech
Centre de Mathématiques Appliquées
Rue Claude Daunesse B.P. 207, 06904 Sophia Antipolis Cedex, France

Remerciements

Je souhaiterais en premier lieu remercier les membre du jury de cette thèse :
- Monsieur Jean-Paul Marmorat pour avoir accepté de présider mon jury de thèse.
- Messieurs Philippe Bonnifait et Luc Jaulin pour leur travail de rapporteurs malgré la taille importante de ce mémoire.
- Messieurs El Mustapha Mouaddib, Jacques Blanc-Talon, François Peyret et Roland Chapuis pour avoir accepté d'examiner cette thèse. Je tiens en particulier à remercier François Peyret pour avoir participé activement à l'encadrement de mes stages d'école d'ingénieurs. Ce sont ces stages qui m'ont donné goût à la recherche : sans eux ce document de thèse n'aurait peut-être pas vu le jour.
- Monsieur Patrick Rives pour avoir accepté d'être mon directeur de thèse pendant ces trois années et demi et de m'avoir laissé une grande liberté pour la réalisation de mes travaux de recherche. Sa patience, son calme et ses conseils ont eu un rôle primordial dans la réussite de ces travaux.

Je remercie la DGA pour avoir accepté de financer ces travaux de thèse.

Je tiens également à remercier tous les membres de l'équipe projet ICARE/ARobAS de l'INRIA Sophia Antipolis que j'ai côtoyés pendant ma thèse, dont la bonne humeur a permis de rendre le travail toujours agréable. Merci à Christopher, Ezio, Mauro, Omar, Youssef, Benoît, Geraldo, Hicham, Erwan, Vincent, Andrew, Mélaine, Maxime, Thomas, Patrick, Gaby, Adan, Min Duc, Tiago, Glauco, Daniele, Alexandre et Mathieu. Un merci particulier à Hicham, Min Duc et Tiago avec qui j'ai régulièrement " décompressé ", que ce soit autour d'une table de ping-pong, sur un terrain de football ou sur des pistes de ski. Je tiens également à remercier Pascal Morin et Claude Samson pour leur conseils précieux et leurs remarques pertinentes sur l'ensemble de mes travaux. Merci également à Patricia, Christine et Nathalie pour leur gentillesse.

Je voudrais enfin remercier tous mes proches dont le soutien a été primordial dans la réussite de ces travaux. Ma mère, mon frère et ma sœur. Je ne pourrais pas terminer cette section sans remercier Aurélia pour tout le soutien et l'amour qu'elle m'a apporté depuis maintenant plus de trois ans et que j'aime passionnément.

Table des matières

Table des figures

Liste des algorithmes

Liste des tableaux

Glossaire

Arithmétique des intervalles Arithmétique permettant d'effectuer des opérations sur des intervalles. Elle comprend la redéfinition des opérations classiques d'addition, soustraction, multiplication et division lorsque les éléments considérés ne sont plus des variables réelles, mais des intervalles réels.

Bearing-only SLAM ensemble des algorithmes de **SLAM** pour lesquels seul un capteur de direction est utilisé pour percevoir l'environnement (cas typique : une caméra)

Bruit Erreur inconnue sur un système de mesure ou une équation de modèle. Ce type de terme est en général modélisé par une variable (ou vecteur) aléatoire dont la **densité de probabilité** est connue.

Caméra omnidirectionnelle Caméra perspective classique filmant un miroir de révolution. Ce type de montage permet de filmer une scène à 360 degrés au prix de distortions.

Cartographie En robotique, la cartographie désigne le fait de créer une carte ou représentation de l'environnement d'un robot lorsque la trajectoire de celui-ci est connue.

Covariance de deux variables aléatoires Valeur réelle égale à l'**espérance mathématique** du produit des écarts à la moyenne de deux variables aléatoires. Cette valeur permet de mesurer directement le niveau de corrélation entre les deux variables considérées. Si la covariance est nulle, alors les deux variables aléatoires sont dites décorrélées.

Densité de probabilité Fonction représentant la répartition d'une variable (resp. vecteur) aléatoire sous forme intégrale. La probabilité que la variable (resp. vecteur) appartiennent à un certain domaine est donné par l'intégration de la densité de probabilité sur le domaine donné. L'intégrale de cette densité est unitaire sur $\mathbb{R}$ (resp. $\mathbb{R}^n$, n représentant la dimension du vecteur).

EKF Acronyme de *Extended Kalman Filter* (filtre de Kalman étendu). Il s'agit de la version non-linéaire du **filtre de Kalman** classique, basée sur une linéarisation des équations.

Espérance mathématique Valeur réelle représentant la valeur moyenne prise par une variable (ou vecteur) aléatoire.

Filtre de Kalman Méthode de filtrage probabiliste faisant appel à un modèle d'état et un modèle de mesures, tous deux bruités. Sous l'hypothèse que les **bruits** sont gaussiens et que les équations définissant le modèle et les mesures sont linéaires, le filtre de Kalman délivre une estimation optimale de l'état à filtrer (au sens du minimum de variance)

Filtre particulaire Méthode de filtrage où l'état est modélisé par un ensemble de particules non déterministes. Les mouvements des particules sont tirés aléatoirement selon une **densité de probablité définie à l'avance**. Les particules sont ensuites pondérées en fonction de la densité choisie et de leur cohérence avec les mesures effectuées.

Localisation d'un robot Calcul de la position d'un robot au sein d'un environnement connu à l'avance.

Matrice de variances-covariances d'un vecteur aléatoire Matrice composée des variances et covariances des variables aléatoires qui composent le vecteur aléatoire considéré. Le $i^{\text{ème}}$ élément de la diagonale désigne la variance de la $i^{\text{ème}}$ composante du vecteur. L'élément (i, j) $(i \neq j)$ désigne la covariance associée aux composantes i et j du vecteur. La matrice de variances-covariances a pour propriété d'être symétrique positive.

Range and bearing SLAM Ensemble des algorithmes de **SLAM** pour lesquels la position relative entre le robot et l'environnement est complètement mesurée (cas typique : un laser rotatif). Cette classe d'algorithmes n'a pas été étudiée dans ces travaux.

Range only SLAM Ensemble des algorithmes de **SLAM** pour lesquels seule la distance entre le robot et l'environnement est mesurée (cas typique : un transpondeur radio communiquant avec des balises dont la position est inconnue). Cette classe d'algorithmes n'a pas été étudiée dans ces travaux.

SAM Acronyme de Simultaneous smoothing And Mapping (lissage et cartographie simultanés). Désigne une variante du **SLAM** dans laquelle toute la trajectoire est prise en compte au lieu de la dernière position traditionnellement.

SFM Acronyme de *Structure from motion*. Désigne l'ensemble des algorithmes de vision permettant de déterminer la structure d'une scène à partir du mouvement d'une caméra. Il s'agit d'une formulation du problème du **SLAM** du point de vue de la communauté *vision*. Les méthodes utilisées sont en général basées sur des ajustements de faisceaux et des algorithmes de vision et ne font pas appel aux notions de **densité de probabilité** et variables aléatoires (contrairement à la communauté *robotique* du SLAM).

SLAM Acronyme de *Simultaneous Localization and Mapping* (localisation et cartographie simultanées). Désigne l'ensemble des algorithmes permettant d'effectuer la localisation d'un robot tout en cartographiant son environnement sans *a priori*.

SLAM par intervalles Méthode de résolution du problème de SLAM dans laquelle l'ensemble des bruits sont supposés appartenir à des intervalles connus. Le résultat final est un ensemble d'intervalles permettant d'englober à 100% la trajectoire réelle du robot et la position des éléments de l'environnement. Ce type de méthode fait appel à l'**arithmétique des intervalles**.

UKF Acronyme de *Unscented Kalman Filter* (filtre de Kalman sans parfum). Il s'agit d'une variante du filtre de Kalman classique basée sur la transformée *Unscented* (approximation d'une densité de probabilité par des particules déterministes). Contrairement à l'**EKF**, l'**UKF** ne linéarise pas les équations des modèles d'état et de mesures. On montre qu'il est équivalent à un développement limité à l'ordre 2.

Variable (resp. vecteur) aléatoire gaussienne Variable (resp. vecteur) aléatoire dont la **densité de probabilité** est une gaussienne. La densité associée est entièrement définie par l'**espérance mathématique** et la variance (resp. **matrice de variances-covariances**) de la variable (resp. vecteur) aléatoire considérée.

Variance d'une variable aléatoire Valeur réelle positive représentant l'écart quadratique moyen d'une variable aléatoire par rapport à son **espérance mathématique**. La variance permet de modéliser l'incertitude associée la réalisation d'une variable aléatoire.

Introduction générale

1 Contexte

Les problèmes de localisation et de cartographie sont des problèmes majeurs en robotique. On parle de localisation lorsque l'on cherche à retrouver la position du robot au sein d'une carte connue. Lorsqu'on cherche à construire une représentation de l'environnement connaissant la trajectoire du robot, on parle de cartographie. Pris séparéments, ces deux problèmes sont aujourd'hui résolus. Néanmoins, un des objectifs de la robotique mobile est de développer des robots autonomes capables de se déplacer sans connaissance *a priori* sur l'environnement. Pour cela, les problèmes de localisation et de cartographie doivent être résolus conjointement : on parle alors de SLAM (*Simultaneous Localization and Mapping*). La résolution complète de ce problème permettra d'assurer une autonomie accrue des robots mobiles. Les applications sont multiples, pouvant aller de l'exploration de zones dangereuses (zones de combats militaires, zone de catastrophe naturelle,...) à la cartographie d'environnement totalement inaccessibles (exploration de planètes...).

Le SLAM a d'abord été formulé par Smith et Cheeseman dans les années 1980. Depuis, la littérature concernant ce sujet est devenue très abondante et de nombreux algorithmes ont été développés. Ceux-ci sont en général classés selon le type de capteur(s) utilisé(s) pour représenter l'environnement. Un capteur populaire est le laser 2D : il permet de fournir une coupe précise de l'environnement à chaque instant. Néanmoins, il n'apporte qu'une information en 2 dimensions. Les lasers 3D ne sont en général pas utilisés (ils sont d'une part beaucoup trop chers, les temps d'acquisitions des différentes coupes sont trop long). Les capteurs visuels sont quant à eux beaucoup plus informatifs, fournissant une projection texturée de l'environnement.

Nous nous intéressons dans cette thèse au problème du SLAM visuel monoculaire. Il s'agit d'un problème difficile car on ne dispose d'aucune mesure métrique de l'environnement. En contrepartie, il est possible d'obtenir un montage mécanique facile à calibrer et peu cher. Par ailleurs, une caméra est un capteur passif. Ceci peut être très intéressant dans le cas d'applications militaires où les interactions

avec l'environnement doivent être le plus limitées possible. Plus particulièrement, nous nous intéressons au SLAM avec une caméra omnidirectionnelle : l'intérêt de ce capteur est de fournir une vision de la scène à 360 degrés, ce qui peut être intéressant dans les cas où le robot est animé par de fortes rotations.

2 Objectifs

La plupart des méthodes de SLAM visuel s'appuient soit sur des méthodes d'ajustement de faisceaux, soit sur le filtrage de Kalman étendu. La première méthode, s'appuyant sur une modélisation géométrique, ne traite pas explicitement des incertitudes. La seconde méthode, quant à elle, intègre la notion d'incertitudes, mais est souvent inconsistante : l'erreur prédite par le filtre n'englobe pas la solution réelle. Le premier objectif de cette thèse a donc consisté à définir une méthode de SLAM visuel assurant de bonnes propriétés de consistance tout en donnant un résultat exploitable : il s'agit de fournir des zones d'incertitudes qui englobent la solution réelle tout en n'étant pas trop grandes par rapport à l'erreur réelle. Nous cherchons à la fois à obtenir d'excellentes données métriques tout en prédisant au mieux l'erreur commise.

Le second objectif réside dans l'étude d'un problème classique du SLAM qui est la gestion de l'augmentation du vecteur d'état avec la taille de la carte. Cette augmentation est en général associée à une augmentation des temps de calcul et à une augmentation des incertitudes au fur et à mesure que le robot s'éloigne de la position initiale. Nous chercherons à faire en sorte de limiter l'influence de ces deux problèmes, en définissant une méthode de segmentation de l'environnement en sous-cartes locales. Cette segmentation devra être à la fois efficace et consistante.

Enfin, un objectif transversal à toute la thèse est l'automatisation des processus. L'idée est d'éviter le plus possible l'initialisation d'opérations par un opérateur humain. Par exemple, toutes les primitives qui devront être repérées dans les images ne doivent pas être initialiséées par un opérateur (ce qui est souvent le cas dans les méthodes de suivi de plan). Nous chercherons à développer des approches nécessitant le moins de " réglages " possible : chaque réglage devra avoir une interprétation physique simple ou évidente. Il s'agit d'une contrainte indispensable dans le but de définir des algorithmes généraux et bien fondés qui peuvent être exportés d'un environnement à l'autre, d'une plate-forme robotique à une autre.

3 Contributions

Ces travaux intègrent plusieurs contributions. Nous avons dans un premier temps établi un état de l'art des méthodes de résolution de SLAM, et avons évalué leur intérêt indépendamment des modèles de robots et de mesures. Deux approches sont ressorties de cette étude : une approche basée sur un lissage probabiliste et une approche déterministe. Une comparaison rigoureuse a été effectuée en simulation dans le cas du SLAM dit " bearing-only " (les amers ne sont repérés que par des angles, ce qui est théoriquement équivalent au SLAM visuel). Nous avons montré que l'approche probabiliste présente d'excellentes propriétés de consistance. Un premier algorithme de lissage probabiliste a été implémenté en Matlab dans le cadre d'une fusion entre les données issues de l'odométrie et du capteur omnidirectionnel. Dans cette première version, la position de la caméra dans le repère du robot est supposée parfaitement connue. Cette hypothèse n'est pas toujours vérifiable, notamment si l'on monte une caméra sur un véhicule. Une seconde contribution a été de développer un algorithme prenant en compte cet aspect et estimant automatiquement ces paramètres. Une dernière version a enfin été développée afin de s'affranchir complètement de l'odométrie. Pour cela, nous avons amélioré et rendu consistant d'un point de vue théorique une paramétrisation des amers déjà proposée dans la littérature et adoptée dans la majorité des approches actuelles.

La seconde contribution de ces travaux réside dans la définition de méthodes théoriques consistantes pour la gestion de cartes locales. Nous avons dans un premier temps défini un critère numérique facilement interprétable permettant de décider pertinemment quand réinitialiser une nouvelle carte. Ensuite, nous avons mathématiquement prouvé qu'il est possible de faire en sorte qu'une nouvelle carte puisse intégrer de façon consistante les informations capitalisées dans la précédente.

La dernière contribution de ces travaux est l'obtention d'une représentation haut niveau des cartes issues du SLAM. Nous avons montré qu'il est possible de détecter et de texturer automatiquement les régions planaires issues des images, et ce sans initialisation de la part d'un opérateur et sans utiliser d'autre capteur de perception de l'environnement que la caméra. Nous proposons également une méthode simple à mettre en œuvre afin d'évaluer efficacement l'espace accessible par le robot.

4 Organisation de la thèse

Première partie

La première partie de cette thèse est consacrée à l'étude et l'évaluation des différents algorithmes

d'estimation.

Chapitre 1

Nous présentons dans ce premier chapitre les principales méthodes probabilistes utilisées dans la littérature afin de résoudre le problème du SLAM. Nous nous intéresserons surtout au filtrage de Kalman, au filtrage particulaire ainsi qu'à une méthode de lissage probabiliste.

Chapitre 2

Nous nous intéresserons ici à une méthode ensembliste pour résoudre le problème du SLAM. Il s'agit de supposer que les erreurs de capteurs sont toutes bornées et d'appliquer l'arithmétique des intervalles sur les équations du SLAM.

Chapitre 3

Le dernier chapitre de cette première partie est quant à lui destiné à évaluer en simulation l'algorithme probabiliste selectionné au chapitre 1 et l'algorithme de SLAM par intervalles présenté au chapitre précédent. Une comparaison approfondie sur la qualité de la solution et les propriétés de consistance est menée et montrera que la méthode probabiliste semble plus adaptée au SLAM visuel.

Deuxième partie

Nous présentons dans la seconde partie de cette thèse les résultats de SLAM visuel obtenus par notre algorithme de SLAM dans le cas de séquences réelles.

Chapitre 4

Avant d'appliquer notre algorithme de SLAM probabiliste, nous avons besoin d'extraire les informations des images. Pour cela, nous avons utilisé des méthodes existantes de vision par ordinateur adaptées au cas des images omnidirectionnelles. L'objectif de ce chapitre est donc de proposer des rappels des différents algorithmes utilisés.

Chapitre 5

Nous présentons ici les résultats obtenus dans le cas où le robot se déplace dans le plan avec 3 degrés de liberté (deux en translation et un en rotation), dans le cadre d'une fusion entre odométrie et images omnidirectionnelles. Les premiers résultats obtenus sont très bons ; nous verrons alors qu'il est possible

d'estimer en plus de la carte et de la trajectoire du robot la position de la caméra dans le repère du robot.

Chapitre 6

Le dernier chapitre de cette seconde partie est destiné à présenter une nouvelle version de l'algorithme de SLAM permettant d'estimer la trajectoire du robot dans l'espace (avec 6 degrés de liberté). Nous sommes alors capables d'obtenir une solution tout à fait pertinente sans utiliser l'odométrie du robot.

Troisième partie

Les résultats obtenus dans la seconde partie sont globalement bons, mais nous avons detectés certaines faiblesses lorsque l'environnement devient trop grand et que les incertitudes augmentent trop. Cette troisième partie est donc consacrée à l'étude des représentations locales dans le cadre du SLAM.

Chapitre 7

Le problème des représentations locales dans le cadre du SLAM a déjà été abordé dans la littérature. Nous proposons dans ce chapitre un état de l'art des principales méthodes existantes.

Chapitre 8

Ce chapitre est quant à lui destiné à la présentation de la solution employée. Nous avons choisi de segmenter la solution globale en différentes cartes locales. Nous introduirons un critère fondé sur les corrélations et l'augmentation des incertitudes afin de choisir judicieusement quand initialiser une nouvelle carte. Nous présenterons également une méthode permettant de créer une nouvelle carte locale capable de partager de l'information avec la précédente.

Quatrième partie

La dernière partie de ce manuscrit est consacrée à l'exploitation des résultats obtenus dans les chapitre précédents afin d'obtenir des représentation de plus niveau.

Chapitre 9

Nous nous intéressons dans un premier temps à la détection automatique des plans dans les nuages de points issus du SLAM. Nous verrons qu'il est possible de projeter efficacement des textures sur les plans détectés.

Chapitre 10

Le dernier aspect abordé dans cette thèse est la détection de l'espace accessible par le robot. Les résultats de ce chapitre pourront servir de base à des algorithmes de planification de trajectoire

Enfin, une conclusion générale synthétisera les résultats et contributions de ce travail et dégagera des perspectives pour les recherches futures.

Première partie

État de l'art des méthodes d'estimation utilisées pour le SLAM

Chapitre 1

Le SLAM probabiliste

Sommaire

Nous présentons dans ce chapitre les méthodes probabilistes couramment utilisées pour résoudre le problème du SLAM. La première méthode de résolution a été introduite dans les années 1980 avec les travaux de [Smith et Cheeseman, 1987] et est basée sur l'utilisation d'un filtre de Kalman étendu. Aujourd'hui, la littérature concernant le SLAM probabiliste est abondante ; nous ne visons donc pas l'exhaustivité. Le but de ce chapitre est de donner un aperçu général des principales méthodes d'estimation utilisées afin de pouvoir justifier les principaux choix effectués dans la thèse concernant la méthode de filtrage. Les méthodes proposées dans les sections suivantes seront décrites sans définir de modèle de robot ou de type de capteur. Néanmoins, certaines propriétés énoncées nécessiteront d'instancier un modèle particulier. Nous le ferons lorsque cela sera nécessaire et afin d'illustrer le comportement d'une méthode.

Ce chapitre s'articule autour de huit sections. Nous effectuons d'abord quelques rappels concernant les principales notions liées à l'utilisation des densités de probabilité. Nous présentons ensuite la formulation probabiliste du problème ainsi que les hypothèses utilisées (section 1.2). Nous donnons dans la section 1.3 quelques bases concernant l'observabilité du problème du SLAM. La section 1.4 présente l'application du filtre de Kalman étendu au SLAM et les problèmes liés à cet algorithme. Nous présentons dans la section 1.5 quelques variantes du filtre de Kalman étendu. Nous décrivons ensuite une méthode basée sur le filtrage particulaire : le *FastSLAM* (section 1.6). Une méthode de lissage basée sur l'estimation de toute la trajectoire est présentée dans la section 1.7. Enfin, une synthèse des différents algorithmes est proposée (section1.8).

1.1 Rappels des notions de probabilité

Nous proposons dans cette section quelques rappels concernant les variables aléatoires, densités de probabilités, ainsi que les notions de conditionnement et de marginalisation. L'application de ces notions au cas spécial des variables aléatoires gaussiennes est également présentée. Le lecteur familier de ces notions de probabilité peut directement se rendre à la section suivante (page 24).

1.1.1 Rappels généraux

Soit X une variable aléatoire scalaire. La densité de probabilité associée est notée p_X. Il s'agit d'une fonction de $\mathbb{R}$ dans $\mathbb{R}^+$ et d'intégrale unitaire. Par ailleurs, la densité de probabilité p_X est la dérivée de la fonction F_X de répartition définie par :

$$\begin{array}{rccl} F_X : & \mathbb{R} & \rightarrow & \mathbb{R}^+ \\ & x & \mapsto & \mathrm{P}(X \leq x) \end{array} \tag{1.1}$$

où $P(E) \in [0,1]$ désigne la probabilité de l'évènement E. La probabilité de $X \in [a,b]$ est donnée par :

$$P(a \leq X \leq B) = \int_a^b p_X(x)\mathrm{d}x \tag{1.2}$$

Plus généralement, la probabilité de $X \in I$ (I n'étant pas forcément un intervalle connexe [1]) est donnée par :

$$P(X \in I) = \int_I p_X(x)\mathrm{d}x \tag{1.3}$$

Pour le cas multidimensionnel, lorsque $\mathbf{X} \in \mathbb{R}^n$, la densité de probabilité $p_{\mathbf{X}}$ est une fonction de $\mathbb{R}^n$ dans $\mathbb{R}^+$. L'intégrale de cette fonction sur $\mathbb{R}^n$ est unitaire. La probabilité d'avoir $\mathbf{X} \in \mathbf{I}$ ($\mathbf{I} \subset \mathbb{R}^n$) est donnée par :

$$P(\mathbf{X} \in \mathbf{I}) = \int_{\mathbf{I}} \ldots \int p_{\mathbf{X}}(x_1, \ldots, x_n)\mathrm{d}x_1 \ldots \mathrm{d}x_n \tag{1.4}$$

1.1.2 Espérance, variance et moments

1.1.2.1 Cas scalaire

Soit X une variable aléatoire scalaire et p_X sa densité de probabilité. Nous notons $\mathbf{E}(X)$ son espérance mathématique (ie. la moyenne) de X :

$$\mathbf{E}(X) = \int_{-\infty}^{+\infty} x p_X(x)\mathrm{d}x \tag{1.5}$$

en admettant que cette intégrale existe. On peut facilement démontrer la linéarité de l'espérance à partir de l'équation 1.5. On a alors le résultat suivant :

Proposition 1.1 (Linéarité de l'espérance mathématique)
Soient X et Y deux variables aléatoires scalaires et λ un scalaire déterministe réel quelconque. On a alors :

$$\mathbf{E}(\lambda X + Y) = \lambda \mathbf{E}(X) + \mathbf{E}(Y) \tag{1.6}$$

Considérons désormais une fonction f à valeurs de $\mathbb{R}$ dans $\mathbb{R}$. L'espérance de la variable aléatoire définie par $f(x)$ est donnée par :

$$\mathbf{E}(f(X)) = \int_{-\infty}^{+\infty} f(x) p_X(x)\mathrm{d}x \tag{1.7}$$

1. Dans les cas dégénérés, I peut éventuellement contenir des points isolés. On trouve en général ce cas dans le cas des variables aléatoires discrètes pour lesquelles p_X est constituée de Diracs.

On définit les moments d'ordre k ainsi que les moments centrés d'ordre k de X par :

$$\begin{cases} \mathbf{m}^k[X] = \mathbf{E}\big(X^k\big) & \text{(moment d'ordre } k\text{)} \\ \mathbf{m}_c^k[X] = \mathbf{E}\big((X-\mu_X)^k\big) & \text{(moment centré d'ordre } k\text{)} \\ \text{avec } \mu_X = \mathbf{E}(X) \end{cases} \tag{1.8}$$

Pour l'utilisation des densités de probabilités, on retiendra essentiellement le **moment d'ordre 1** ainsi que le **moment centré d'ordre 2**, ie. la variance. Cette dernière valeur permet de donner une indication de l'écartement de la variable aléatoire par rapport à la moyenne. Dans le cas gaussien notamment, elle permet de définir un intervalle de confiance à $x\%$. On note $\mathbf{var}(X)$ la variance de X. Par ailleurs, il est commode de considéder la racine carrée de la variance, à s'avoir l'écart-type noté σ (celui-ci a la même dimension [2] que la variable aléatoire associée).

Par ailleurs, on remarquera facilement que la variance ne possède pas les mêmes propriétés de linéarité que l'espérance. En effet, soient X et Y deux variables aléatoires scalaires ainsi que λ un réel quelconque. Les deux propriétés suivantes sont faciles à démontrer :

1. $\mathbf{var}(\lambda X) = \lambda^2 \mathbf{var}(X)$ (ce qui donne $\sigma_{\lambda X} = |\lambda|\sigma_X$ en termes d'écart type)
2. $\mathbf{var}(X+Y) = \mathbf{var}(X) + \mathbf{var}(Y) + 2\mathbf{E}((X-\mathbf{E}(X))(Y-\mathbf{E}(Y)))$

Le terme $\mathbf{E}((X-\mathbf{E}(X))(Y-\mathbf{E}(Y)))$ est également appelé *covariance* de X et de Y (noté $\mathbf{cov}(X,Y)$). Il traduit la " force " du lien entre les variables aléatoires X et Y. Dans le cas où les deux variables aléatoires sont indépendantes, on a $\mathbf{cov}(X,Y) = 0$. [3] Par ailleurs, on définit le coefficient de corrélation par :

$$\rho = \frac{\mathbf{cov}(X,Y)}{\sigma_X \sigma_Y} \tag{1.9}$$

On peut facilement montrer que $\rho \in [-1,1]$. Ce coefficient permet de normaliser la notion de covariance. Si $|\rho| = 1$, on est dans le cas extrême où il existe une relation déterministe entre les variables aléatoires X et Y. Si $\rho = 0$, elles sont complétement décorrélées.

1.1.2.2 Cas vectoriel

Soit $\mathbf{X} = [X_1, \ldots, X_n]^T$ une variable vectorielle de dimension n et $p_{\mathbf{X}}$ sa densité de probabilité. Son espérance mathématique est donnée par :

$$\mathbf{E}(\mathbf{X}) = \int_{\mathbb{R}^n} \!\!\ldots\!\! \int [x_1, \ldots, x_n]^T \cdot p_{\mathbf{X}}(x_1, \ldots, x_n) \mathrm{d}x_1 \ldots \mathrm{d}x_n \tag{1.10}$$

2. Au sens physique du terme (ie. savoir si on parle de distance, vitesse, vitesse au carré...)

3. On peut montrer que la réciproque est fausse. Si $\mathbf{cov}(X,Y) = 0$, les variables aléatoires X et Y ne sont pas indépendantes (mais simplement décorrélées). Pour montre l'indépendance, il faudrait montrer que $\mathbf{cov}(\varphi(X), \psi(Y)) = 0$ pour tous les couples de fonctions (φ, ψ) pour lesquels $\mathbf{cov}(\varphi(X), \psi(Y)) = 0$ existe. X et Y sont indépendantes *si et seulement si* la fonction de densité jointe se factorise, ie. $p_{X,Y} \equiv p_X . p_Y$.

Considérons désormais la fonction $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$ ($m \in \mathbb{N}^*$). L'espérance du vecteur aléatoire défini par $\mathbf{f}(\mathbf{X})$ est donnée par :

$$\mathbf{E}(\mathbf{f}(\mathbf{X})) = \int_{\mathbb{R}^n} \dots \int \mathbf{f}(x_1, \dots x_n) \cdot p_{\mathbf{X}}(x_1, \dots, x_n) \mathrm{d}x_1 \dots \mathrm{d}x_n \tag{1.11}$$

Comme pour le cas scalaire, l'espérance mathématique possède des propriétés de linéarité dans le cas vectoriel. Soient $\mathbf{X}$ et $\mathbf{Y}$ deux vecteurs aléatoires de dimension n et λ un scalaire réel déterministe. Considérons également la matrice $\mathbf{F}$ de dimension $m \times n$ (représentant une fonction linéaire de $\mathbb{R}^n$ dans $\mathbb{R}^m$).[4] L'équation 1.11 permet de montrer que :

$$\mathbf{E}(\lambda\mathbf{X} + \mathbf{Y}) = \lambda\mathbf{E}(\mathbf{X}) + \mathbf{E}(\mathbf{Y}) \tag{1.12}$$

$$\mathbf{E}(\mathbf{F}\mathbf{X}) = \mathbf{F}\mathbf{E}(\mathbf{X}) \tag{1.13}$$

Dans le cas vectoriel, nous ne définissions que le moment centré d'ordre 2. Celui-ci est donnée par :

$$\mathbf{\Sigma_X} = \mathbf{E}\big((\mathbf{X} - \mu_{\mathbf{X}})(\mathbf{X} - \mu_{\mathbf{X}})^T\big) \tag{1.14}$$

où $\mu_{\mathbf{X}}$ est l'espérance mathématique du vecteur $\mathbf{X}$. $\mathbf{\Sigma_X}$ désigne la matrice de variances-covariances du vecteur aléatoire $\mathbf{X}$. On peut montrer que cette matrice est toujours symétrique et semi-définie positive. Il est par ailleurs facile de donner l'écriture de chaque terme de la matrice. On a :

- $\mathbf{\Sigma_X}[i,i] = \mathbf{var}(X_i)$: chaque élément de la diagonale correspond à la variance de la composante correspondante,
- $\mathbf{\Sigma_X}[i,j]_{i \neq j} = \mathbf{cov}(X_i, X_j)$: les éléments hors diagonaux correspondent à la covariance entre les composantes scalaires correspondantes.

On a également une relation similaire au cas scalaire pour la matrice de variances-covariances associée à la somme de vecteurs aléatoires. Considérons les vecteurs aléatoires $\mathbf{X}$ et $\mathbf{Y}$ ainsi que la matrice $\mathbf{F}$ et le scalaire λ définis précédemment. On a alors :

$$\Sigma_{\lambda\mathbf{X}} = \lambda^2 \Sigma_{\mathbf{X}} \tag{1.15}$$

$$\Sigma_{\mathbf{F}\mathbf{X}} = \mathbf{F}\Sigma_{\mathbf{X}}\mathbf{F}^T \tag{1.16}$$

$$\Sigma_{\mathbf{X}+\mathbf{Y}} = \Sigma_{\mathbf{X}} + \Sigma_{\mathbf{Y}} + \mathbf{E}\Big((\mathbf{X} - \mu_{\mathbf{X}})\,(\mathbf{Y} - \mu_{\mathbf{Y}})^T\Big) + \mathbf{E}\Big((\mathbf{Y} - \mu_{\mathbf{Y}})\,(\mathbf{X} - \mu_{\mathbf{X}})^T\Big) \tag{1.17}$$

où $\mu_{\mathbf{X}}$ et $\mu_{\mathbf{Y}}$ désignent les espérances mathématiques des vecteurs aléatoires X et Y. Comme pour le cas scalaire, lorsque les vecteurs $\mathbf{X}$ et $\mathbf{Y}$ sont indépendants, la matrice $\mathbf{E}\Big((\mathbf{X} - \mu_{\mathbf{X}})\,(\mathbf{Y} - \mu_{\mathbf{Y}})^T\Big)$ est nulle (et donc $\mathbf{E}\Big((\mathbf{Y} - \mu_{\mathbf{Y}})\,(\mathbf{X} - \mu_{\mathbf{X}})^T\Big)$ aussi). On a alors $\Sigma_{\mathbf{X}+\mathbf{Y}} = \Sigma_{\mathbf{X}} + \Sigma_{\mathbf{Y}}$.

4. Eventuellement, on peut avoir $n = m$.

Remarque 1.1 (Propagation de la matrice de variances-covariances)
L'équation 1.16 permet d'obtenir la matrice de variances-covariances d'un vecteur aléatoire après application d'une fonction linéaire. Dans le cas non linéaire, on pourra approcher la matrice de variances-covariances en effectuant un développement limité de la fonction à l'ordre 1. La matrice jacobienne de la fonction remplace alors la matrice $\mathbf{F}$*.*

1.1.3 Marginalisation et conditionnement

Nous rappelons dans ce paragraphe les notions de marginalisation et de conditionnement d'une variable aléatoire. Il s'agit de notions essentielles qui seront utilisées dans la suite de ce manuscrit. Dans la suite de ce paragraphe, nous considérons $\mathbf{X} = [\mathbf{Y}^T, \mathbf{Z}^T]^T$ un vecteur aléatoire, décomposé en deux parties $\mathbf{Y} = [Y_1, \ldots, Y_n]^T$ et $\mathbf{Z} = [Z_1, \ldots, Z_m]^T$ (de dimensions respectives n et m). Nous notons $p_{\mathbf{X}}$ la densité de probabilitié associée à ce vecteur.

1.1.3.1 Marginalisation

Supposons que l'on cherche à calculer la densité de probabilité $p_{\mathbf{Y}}$ à partir de la fonction $p_{\mathbf{X}}$. Ce processus s'appelle la marginalisation : on cherche à retrouver la fonction de répartion marginale de $\mathbf{Y}$. Autrement dit, on veut connaître la densité de probabilité *a priori* de $\mathbf{Y}$, ie sans aucune connaissance de la valeur des autres composantes de $\mathbf{X}$. Il s'agit d'intégrer la fonction $p_{\mathbf{X}}$ sur toutes les valeurs que peut prendre $\mathbf{Z}$. Ainsi, la fonction $p_{\mathbf{Y}}$ est donnée par :

$$\begin{array}{rccl} p_{\mathbf{Y}} : & \mathbb{R}^n & \rightarrow & \mathbb{R}^+ \\ & (y_1, \ldots, y_n) & \mapsto & \underbrace{\int \ldots \int}_{\mathbb{R}^m} p_{\mathbf{X}}(y_1, \ldots, y_n, z_1, \ldots, z_m) \mathrm{d}z_1 \ldots \mathrm{d}z_m \end{array} \tag{1.18}$$

1.1.3.2 Conditionnement

Supposons désormais que l'on cherche à calculer la densité de probabilité de $\mathbf{Y}$ lorsque le vecteur $\mathbf{Z}$ est connu (et fixé à $\mathbf{z} = [z_1, \ldots, z_m]^T$). Il s'agit d'une fonction de $\mathbb{R}^n$ dans $\mathbb{R}$, notée $p_{\mathbf{Y}|\mathbf{Z}=\mathbf{z}}$.

Théorème 1.1 (Expression de la densité conditionnelle)
La fonction $p_{\mathbf{Y}|\mathbf{Z}=\mathbf{z}}$ *vaut :*

$$\begin{array}{rccl} p_{\mathbf{Y}|\mathbf{Z}=\mathbf{z}} : & \mathbb{R}^n & \rightarrow & \mathbb{R}^+ \\ & (y_1, \ldots\ y_n) & \mapsto & \alpha p_{\mathbf{x}}(y_1, \ldots\ y_n, z_1, \ldots, z_m) \end{array} \tag{1.19}$$

où α est une constante multiplicative permettant d'assurer que l'intégrale de $p_{\mathbf{Y}|\mathbf{Z}=\mathbf{z}}$ est unitaire. On a donc :

$$\alpha = \frac{1}{\underset{\mathbb{R}^n}{\int\cdots\int} p_{\mathbf{X}}(y_1, \ldots, y_n, z_1, \ldots, z_m)\mathrm{d}y_1, \ldots, \mathrm{d}y_n} \tag{1.20}$$

La démonstration de ce théorème est faite dans le paragraphe suivant (après introduction de la règle de Bayes).

Remarque 1.2 (Rôle de z)

Dans la définition de la fonction $p_{\mathbf{Y}|\mathbf{Z}=\mathbf{z}}$, le vecteur $\mathbf{z}$ joue le rôle de paramètre connu. Néanmoins, si on considère ce vecteur comme variable et qu'on fixe le vecteur $\mathbf{y}$, l'expression de $p_{\mathbf{Y}|\mathbf{Z}=\mathbf{z}}(\mathbf{y})$ permet de définir la fonction de vraisemblance *(qui est dans ce cas à valeur de $\mathbb{R}^n$ dans $\mathbb{R}$). On pourra remarquer que contrairement à une densité de probabilité, l'intégrale de la fonction de vraisemblance n'a aucune raison d'être unitaire.*

1.1.4 Règle de Bayes

En probabilité, le théorème de Bayes permet d'exprimer la probabilité jointe de deux évènements à l'aide de la probabilité conjointe des deux éléments et des probabilités marginales. Soient A et B deux évènements. On a alors :

$$P(A|B) = \frac{P(A,B)}{P(B)} \tag{1.21}$$

ou encore :

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{1.22}$$

Les équations 1.21 et 1.22 peuvent être transposées au cas des densités de probabilités continues, conduisant au théorème de Bayes pour le cas des densités de probabilités continues.

Théorème 1.2 (Règle de Bayes pour les densités de probabilités)

$$\forall(\mathbf{y},\mathbf{z}) \in \mathbb{R}^n \times \mathbb{R}^m \qquad \begin{cases} p_{\mathbf{Y}|\mathbf{Z}=\mathbf{z}}(\mathbf{y}) = \dfrac{p_{\mathbf{Y},\mathbf{Z}}(\mathbf{y},\mathbf{z})}{p_{\mathbf{Z}}(\mathbf{z})} & \textit{(Première formulation)} \\ p_{\mathbf{Y}|\mathbf{Z}=\mathbf{z}}(\mathbf{y}) = \dfrac{p_{\mathbf{Z}|\mathbf{Y}=\mathbf{y}}(\mathbf{z})p_{\mathbf{Y}}(\mathbf{y})}{p_{\mathbf{Z}}(\mathbf{z})} & \textit{(Seconde formulation)} \end{cases} \tag{1.23}$$

où nous utilisons la notation $p_{\mathbf{Y},\mathbf{Z}}$ à la place de $p_{\mathbf{X}}$ pour rendre l'équation 1.23 plus claire.

La démonstration de la règle de Bayes n'est pas donnée dans ce document. Le lecteur intéressé pourra par exemple se référer à [Papoulis, 1984] (section 7.3). Il convient de remarquer que dans le théorème 1.2, les valeurs $\mathbf{y}$ et $\mathbf{z}$ sont utilisées à la fois comme **paramètres** de fonctions et comme **argument** de fonctions. On ne peut alors plus rigoureusement parler de densité de probabilité ou de fonction de vraisemblance étant donné que " tout peut bouger ". Il s'agit simplement d'une identité entre deux expressions.

La règle de Bayes nous permet désormais de démontrer le théorème 1.1.

Démonstration du théorème 1.1

▲ Utilisons la première formulation de la règle de Bayes pour calculer la densité conditionnelle $p_{\mathbf{Y}|\mathbf{Z}=\mathbf{z}}$ telle que définie dans l'équation 1.21. On a alors :

$$p_{\mathbf{Y}|\mathbf{Z}=\mathbf{z}}(\mathbf{y}) = \frac{p_{\mathbf{Y},\mathbf{Z}}(\mathbf{y},\mathbf{z})}{p_{\mathbf{Z}}(\mathbf{z})} = \frac{p_{\mathbf{Y},\mathbf{Z}}(y_1,\ldots,y_n,z_1,\ldots,z_m)}{p_{\mathbf{Z}}(\mathbf{z})} \tag{1.24}$$

Le dénominateur se calcule à l'aide de la marginalisation de $p_{\mathbf{Y},\mathbf{Z}}$ (cf. équation 1.18) :

$$p_{\mathbf{Z}}(\mathbf{z}) = p_{\mathbf{Z}}(z_1,\ldots,z_m) = \int_{\mathbb{R}^n}\!\!\ldots\!\int p_{\mathbf{Y},\mathbf{Z}}(y_1,\ldots,y_n,z_1,\ldots,z_m)\mathrm{d}y_1\ldots\mathrm{d}y_n \tag{1.25}$$

Le résultat final est immédiat en injectant le résultat de l'équation 1.25 dans l'équation 1.24.

■

1.1.5 Notation abrégée des densités de probabilités

Dans les paragraphes précédents, nous avons utilisé une notation très " complète " pour décrire les fonctions de densité de probabilité. Nous avons systématiquement noté en **lettre capitale la variable aléatoire et en lettre minuscule lorsqu'il s'agissait d'une variable de la fonction de densité de probabilité**. Cette notation n'est en général pas utilisée dans la littérature. Dans la suite, nous désignons directement les densités de probabilités comme des fonctions. Par exemple, si X est une variable aléatoire, on notera $p(X)$ sa densité de probabilité évaluée en X (on " mélange " le nom de la variable aléatoire et le nom de la variable utilisée pour évaluer la fonction). Ce type de notation n'est en général pas ambigu et permet d'alléger les expressions. Des simplifications similaires sont effectuées pour la notation des probabilités conditionnelles. La correspondance entre les notations est décrite dans le tableau 1.1. Dans toute la suite du manuscrit, nous utilisons les notations abrégées pour décrire les densités de probabilités (avec l'utilisation indifférente de majuscule ou minuscule pour les variables et vecteurs aléatoires).

	Ancienne notation	Nouvelle notation
Densité de probabilité	$p_X(x)$	$p(X)$
Densité de probabilité conditionnelle	$p_{X\|Y=y}(x)$	$p(X\|y)$ ou $p(X\|Y=y)$
Marginalisation	$p_X(x) = \int p_{X,Y}(x,y)\mathrm{d}y$	$p_X = \int p(X,Y)\mathrm{d}Y$
Règle de Bayes	$p_{Y\|Z=z}(y) = \frac{p_{Y,Z}(y,z)}{p_Z(z)}$	$p(Y\|Z) = \frac{p(Y,Z)}{p(Z)}$

TABLE 1.1 – Résumé concernant les simplifications des notations de densités de probabilités.

1.1.6 Cas des vecteurs aléatoires gaussiens

Nous présentons dans ce paragraphe les principaux résultats concernant les densités de probabilités gaussiennes. La loi gaussienne est une densité unimodale qui permet de décrire assez fidèlement la distribution des erreurs de capteurs.

1.1.6.1 Généralités

Soit $\mathbf{x}$ un vecteur aléatoire de dimension $n \in \mathbb{N}^*$ suivant une loi gaussienne. La densité de probabilité de $\mathbf{x}$ ne dépend que de deux paramètres : son espérance mathématique (notée μ dans ce paragraphe) et sa matrice de variances-covariances (notée $\mathbf{\Sigma}$ dans ce paragraphe). On a alors :

$$p(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^n \det \mathbf{\Sigma}}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T \mathbf{\Sigma}^{-1}(\mathbf{x}-\mu)\right) \tag{1.26}$$

La fonction définie dans l'équation 1.26 est strictement positive quel que soit $\mathbf{x} \in \mathbb{R}^n$. De plus, on peut montrer que son intégrale est unitaire. Réciproquement, si on peut montrer que la densité de probabilité d'un vecteur aléatoire peut être identifiée à l'équation 1.26, alors on peut en déduire que ce vecteur aléatoire est gaussien d'espérance μ et de matrice de variances-covariances $\mathbf{\Sigma}$.

1.1.6.2 Représentation par matrice et vecteur d'information

Considérons le vecteur aléatoire $\mathbf{x}$ gaussien défini au paragraphe précédent. 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 $p(\mathbf{x})$ soit de la forme " $\exp\left(-\frac{1}{2}\mathbf{x}^T\mathbf{\Omega}\mathbf{x} + \mathbf{x}^T\xi\right)$ ". Pour ce faire, il suffit de ré-écrire la densité $p(\mathbf{x})$. Tous calculs faits, on obtient :

$$p(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^n \det \mathbf{\Sigma}}} \exp\left(-\frac{1}{2}\mu^T\mathbf{\Sigma}^{-1}\mu\right) \exp\left(-\frac{1}{2}\mathbf{x}^T\mathbf{\Sigma}^{-1}\mathbf{x} + \mathbf{x}^T\mathbf{\Sigma}^{-1}\mu\right) \tag{1.27}$$

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

$$p(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^n \det \mathbf{\Omega}^{-1}}} \exp\left(-\frac{1}{2}\xi^T\mathbf{\Omega}^{-1}\xi\right) \exp\left(-\frac{1}{2}\mathbf{x}^T\mathbf{\Omega}\mathbf{x} + \mathbf{x}^T\xi\right) \tag{1.28}$$

$\mathbf{\Omega}$ 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 :

$$\begin{cases} \mathbf{\Sigma} &= \mathbf{\Omega}^{-1} \\ \mu &= \mathbf{\Sigma}\xi \end{cases} \tag{1.29}$$

Nous verrons dans la section 1.7 que la représentation par paramètres d'information peut être plus avantageuse que la représentation classique par moyenne et matrice de variances-covariances pour rendre les calculs efficaces.

1.1.6.3 Théorèmes de marginalisation et de conditionnement

Nous donnons dans ce paragraphe les formules analytiques permettant de marginaliser et conditionner des lois gaussiennes. Pour cela, nous considérerons un vecteur gaussien $\mathbf{V} = [\mathbf{x}^T, \mathbf{y}^T]^T$ ($\mathbf{x}$ et $\mathbf{y}$ sont de dimensions respectives n et m) dont les paramètres s'écrivent :

$$\begin{aligned} \text{Paramètres standards} &: \begin{cases} \mu = \begin{bmatrix} \mu_{\mathbf{x}}^T & \mu_{\mathbf{y}}^T \end{bmatrix}^T \\ \mathbf{\Sigma} = \begin{bmatrix} \mathbf{\Sigma_{xx}} & \mathbf{\Sigma_{xy}} \\ \mathbf{\Sigma_{yx}} & \mathbf{\Sigma_{yy}} \end{bmatrix} \end{cases} \\ \text{Paramètres d'information} &: \begin{cases} \xi = \begin{bmatrix} \xi_{\mathbf{x}}^T & \xi_{\mathbf{y}}^T \end{bmatrix}^T \\ \mathbf{\Omega} = \begin{bmatrix} \mathbf{\Omega_{xx}} & \mathbf{\Omega_{xy}} \\ \mathbf{\Omega_{yx}} & \mathbf{\Omega_{yy}} \end{bmatrix} \end{cases} \end{aligned} \tag{1.30}$$

où μ, $\mathbf{\Sigma}$, ξ et $\mathbf{\Omega}$ satisfont le système 1.29 (ce qui n'est pas nécessairement le cas des sous matrices et sous-vecteurs associés).

Théorème de marginalisation

Nous présentons dans ce paragraphe le théorème de marginalisation dans le cas gaussien. Celui-ci permet de déduire $p(\mathbf{x})$ à partir de $p(\mathbf{x}, \mathbf{y})$.

Théorème 1.3 (Marginalisation de vecteurs aléatoires gaussiens)

Sous les hypothèses du paragraphe précédent, $\mathbf{x}$ *est un vecteur aléatoire gaussien dont les paramètres sont donnés par :*

$$\begin{aligned} \textit{Espérance mathématique} &: \mu_{\mathbf{marg}} = \mu_x \\ \textit{Matrice de variances-covariances} &: \mathbf{\Sigma_{marg}} = \mathbf{\Sigma_{xx}} \\ \textit{Vecteur d'information} &: \xi_{\mathbf{marg}} = \xi_{\mathbf{x}} - \mathbf{\Omega_{xy}}\mathbf{\Omega_{yy}^{-1}}\xi_{\mathbf{y}} \\ \textit{Matrice d'information} &: \mathbf{\Omega_{marg}} = \mathbf{\Omega_{xx}} - \mathbf{\Omega_{xy}}\mathbf{\Omega_{yy}^{-1}}\mathbf{\Omega_{yx}} \end{aligned} \tag{1.31}$$

Le théorème 1.3 nous dit simplement que la marginalisation d'un vecteur aléatoire gaussien reste un vecteur aléatoire gaussien. Ses paramètres (espérance et matrice de variances-covariances) sont logiquement repris des paramètres standards initiaux. Néanmoins, le fait que la marginalisation reste gaussienne n'est pas complètement trivial.

Démonstration du théorème 1.3 :

▲ Après application du théorème de marginalisation et de l'équation 1.28, on a :

$$\begin{aligned}
p(\mathbf{x}) &= \int\!\!\cdots\!\!\int_{\mathbb{R}^m} p(\mathbf{x},\mathbf{y})\,\mathbf{dy} \\
&= \int\!\!\cdots\!\!\int_{\mathbb{R}^m} \frac{\exp\left(-\frac{1}{2}\xi^T\boldsymbol{\Sigma}^{-1}\xi\right)}{\sqrt{(2\pi)^{n+m}\det\boldsymbol{\Omega}^{-1}}} \exp\left(-\frac{1}{2}\begin{bmatrix}\mathbf{x}^T & \mathbf{y}^T\end{bmatrix}\begin{bmatrix}\boldsymbol{\Omega}_{\mathbf{xx}} & \boldsymbol{\Omega}_{\mathbf{xy}} \\ \boldsymbol{\Omega}_{\mathbf{yx}} & \boldsymbol{\Omega}_{\mathbf{yy}}\end{bmatrix}\begin{bmatrix}\mathbf{x} \\ \mathbf{y}\end{bmatrix} + \begin{bmatrix}\mathbf{x}^T & \mathbf{y}^T\end{bmatrix}\begin{bmatrix}\xi_{\mathbf{x}} \\ \xi_{\mathbf{y}}\end{bmatrix}\right)\mathbf{dy} \\
&= K\exp\left(-\frac{1}{2}\mathbf{x}^T\boldsymbol{\Omega}_{\mathbf{xx}}\mathbf{x} + \mathbf{x}^T\xi_{\mathbf{x}}\right)\int\!\!\cdots\!\!\int_{\mathbb{R}^m}\exp\left(-\frac{1}{2}\mathbf{y}^T\boldsymbol{\Omega}_{\mathbf{yy}}\mathbf{y} - \mathbf{y}^T\boldsymbol{\Omega}_{\mathbf{yx}}\mathbf{x} + \mathbf{y}^T\xi_{\mathbf{y}}\right)\mathbf{dy} \\
&= \exp\left(-\frac{1}{2}\mathbf{x}^T\boldsymbol{\Omega}_{\mathbf{xx}}\mathbf{x} + \mathbf{x}^T\xi_{\mathbf{x}}\right)\times\exp\left(\frac{1}{2}\mathbf{x}^T\boldsymbol{\Omega}_{\mathbf{xy}}\boldsymbol{\Omega}_{\mathbf{yy}}^{-1}\boldsymbol{\Omega}_{\mathbf{yx}}\mathbf{x} - \xi_{\mathbf{y}}^T\boldsymbol{\Omega}_{\mathbf{yy}}^{-1}\boldsymbol{\Omega}_{\mathbf{yx}}\mathbf{x} + \frac{1}{2}\xi_{\mathbf{y}}^T\boldsymbol{\Omega}_{\mathbf{yy}}^{-1}\xi_{\mathbf{y}}\right) \\
&\qquad\times\int\!\!\cdots\!\!\int_{\mathbb{R}^m}\exp\left(-\frac{1}{2}\left(\mathbf{y}-\boldsymbol{\Omega}_{\mathbf{yy}}^{-1}\left(\xi_{\mathbf{y}}-\boldsymbol{\Omega}_{\mathbf{yx}}\mathbf{x}\right)\right)^T\boldsymbol{\Omega}_{\mathbf{yy}}\left(\mathbf{y}-\boldsymbol{\Omega}_{\mathbf{yy}}^{-1}\left(\xi_{\mathbf{y}}-\boldsymbol{\Omega}_{\mathbf{yx}}\mathbf{x}\right)\right)\right)\mathbf{dy} \qquad (1.32)
\end{aligned}$$

avec $K = \frac{\exp\left(-\frac{1}{2}\xi^T\boldsymbol{\Sigma}^{-1}\xi\right)}{\sqrt{(2\pi)^{n+m}\det\boldsymbol{\Omega}^{-1}}}$. Dans l'équation 1.32, le terme sous l'intégrale correspond à la partie exponentielle de la forme canonique d'une densité de probabilité gaussienne telle que définie dans l'équation 1.26 (avec $\boldsymbol{\Omega}_{\mathbf{yy}}^{-1}\left(\xi_{\mathbf{y}}-\boldsymbol{\Omega}_{\mathbf{yx}}\mathbf{x}\right)$ pour espérance et $\boldsymbol{\Omega}_{\mathbf{yy}}^{-1}$ pour matrice de variances-covariance). Son intégrale vaut donc $\sqrt{(2\pi)^m\det\boldsymbol{\Omega}^{-1}}$ et est indépendante de $\mathbf{x}$. De plus, on peut remarquer un autre facteur indépendant de $\mathbf{x}$ dans l'équation 1.32 : il s'agit de $\exp\left(\frac{1}{2}\xi_{\mathbf{y}}^T\boldsymbol{\Omega}_{\mathbf{yy}}^{-1}\xi_{\mathbf{y}}\right)$. On a donc :

$$p(\mathbf{x}) \propto \exp\left(-\frac{1}{2}\mathbf{x}^T\left(\boldsymbol{\Omega}_{\mathbf{xx}}-\boldsymbol{\Omega}_{\mathbf{xy}}\boldsymbol{\Omega}_{\mathbf{yy}}^{-1}\boldsymbol{\Omega}_{\mathbf{yx}}\right)\mathbf{x} + \mathbf{x}^T\left(\xi_{\mathbf{x}}-\boldsymbol{\Omega}_{\mathbf{xy}}\boldsymbol{\Omega}_{\mathbf{yy}}^{-1}\xi_{\mathbf{y}}\right)\right) \qquad (1.33)$$

L'équation 1.33 montre que $p(\mathbf{x})$ est proportionnel à l'exponentielle d'une forme quadratique. D'après l'équation 1.28, $\mathbf{x}$ suit donc une loi gaussienne dont les paramètres d'information correspondent aux coefficients de la forme quadratique. La matrice d'information est donc égale à $\boldsymbol{\Omega}_{\mathbf{marg}} = \boldsymbol{\Omega}_{\mathbf{xx}} - \boldsymbol{\Omega}_{\mathbf{xy}}\boldsymbol{\Omega}_{\mathbf{yy}}^{-1}\boldsymbol{\Omega}_{\mathbf{yx}}$ et son vecteur d'information est égal à $\xi_{\mathbf{marg}} = \xi_{\mathbf{x}} - \boldsymbol{\Omega}_{\mathbf{xy}}\boldsymbol{\Omega}_{\mathbf{yy}}^{-1}\xi_{\mathbf{y}}$.

Pour passer aux paramètres standards, on peut utiliser le théorème d'inversion des matrices blocs et remarquer que :

$$\boldsymbol{\Omega}^{-1} = \begin{bmatrix}\left(\boldsymbol{\Omega}_{\mathbf{xx}}-\boldsymbol{\Omega}_{\mathbf{xy}}\boldsymbol{\Omega}_{\mathbf{yy}}^{-1}\boldsymbol{\Omega}_{\mathbf{yx}}\right)^{-1} & -\left(\boldsymbol{\Omega}_{\mathbf{xx}}-\boldsymbol{\Omega}_{\mathbf{xy}}\boldsymbol{\Omega}_{\mathbf{yy}}^{-1}\boldsymbol{\Omega}_{\mathbf{yx}}\right)^{-1}\boldsymbol{\Omega}_{\mathbf{xy}}\boldsymbol{\Omega}_{\mathbf{yy}}^{-1} \\ -\boldsymbol{\Omega}_{\mathbf{yy}}^{-1}\boldsymbol{\Omega}_{\mathbf{xy}}\left(\boldsymbol{\Omega}_{\mathbf{xx}}-\boldsymbol{\Omega}_{\mathbf{xy}}\boldsymbol{\Omega}_{\mathbf{yy}}^{-1}\boldsymbol{\Omega}_{\mathbf{yx}}\right)^{-1} & \left(\boldsymbol{\Omega}_{\mathbf{yy}}-\boldsymbol{\Omega}_{\mathbf{yx}}\boldsymbol{\Omega}_{\mathbf{xx}}^{-1}\boldsymbol{\Omega}_{\mathbf{xy}}\right)^{-1}\end{bmatrix} = \begin{bmatrix}\boldsymbol{\Sigma}_{\mathbf{xx}} & \boldsymbol{\Sigma}_{\mathbf{xy}} \\ \boldsymbol{\Sigma}_{\mathbf{yx}} & \boldsymbol{\Sigma}_{\mathbf{yy}}\end{bmatrix} \qquad (1.34)$$

Ainsi, la matrice de covariance associée à $p(\mathbf{x})$ vaut :

$$\boldsymbol{\Sigma}_{\mathbf{marg}} = \boldsymbol{\Omega}_{\mathbf{marg}}^{-1} = \underbrace{\left(\boldsymbol{\Omega}_{\mathbf{xx}} - \boldsymbol{\Omega}_{\mathbf{xy}}\boldsymbol{\Omega}_{\mathbf{yy}}^{-1}\boldsymbol{\Omega}_{\mathbf{yx}}\right)^{-1}}_{\boldsymbol{\Sigma}_{\mathbf{xx}} \text{ d'après (1.34)}} = \boldsymbol{\Sigma}_{\mathbf{xx}} \tag{1.35}$$

L'espérance mathématique, quant à elle, s'obtient par :

$$\mu_{\mathbf{marg}} = \boldsymbol{\Omega}_{\mathbf{marg}}^{-1}\xi_{\mathbf{marg}} = \boldsymbol{\Sigma}_{\mathbf{xx}}\left(\xi_{\mathbf{x}} - \boldsymbol{\Omega}_{\mathbf{xy}}\boldsymbol{\Omega}_{\mathbf{yy}}^{-1}\xi_{\mathbf{y}}\right) \tag{1.36}$$

D'après l'équation 1.34, on a $-\left(\boldsymbol{\Omega}_{\mathbf{xx}} - \boldsymbol{\Omega}_{\mathbf{xy}}\boldsymbol{\Omega}_{\mathbf{yy}}^{-1}\boldsymbol{\Omega}_{\mathbf{y}}\mathbf{x}\right)^{-1}\boldsymbol{\Omega}_{\mathbf{xy}}\boldsymbol{\Omega}_{\mathbf{yy}}^{-1} = \boldsymbol{\Sigma}_{\mathbf{xy}}$. Sachant par ailleurs que $\boldsymbol{\Sigma}_{\mathbf{xx}} = \left(\boldsymbol{\Omega}_{\mathbf{xx}} - \boldsymbol{\Omega}_{\mathbf{xy}}\boldsymbol{\Omega}_{\mathbf{yy}}^{-1}\boldsymbol{\Omega}_{\mathbf{yx}}\right)^{-1}$, on en déduit que $\boldsymbol{\Sigma}_{\mathbf{xx}}\boldsymbol{\Omega}_{\mathbf{xy}}\boldsymbol{\Omega}_{\mathbf{yy}}^{-1} = -\boldsymbol{\Sigma}_{\mathbf{xy}}$. Ainsi, l'équation 1.36 devient :

$$\mu_{\mathbf{marg}} = \boldsymbol{\Sigma}_{\mathbf{xx}}\xi_{\mathbf{x}} + \boldsymbol{\Sigma}_{\mathbf{xy}}\xi_{\mathbf{y}} \tag{1.37}$$

Finalement, en utilisant l'équation 1.30 et le fait que $\mu = \boldsymbol{\Sigma}\xi$, on en déduit $\mu_{\mathbf{x}} = \boldsymbol{\Sigma}_{\mathbf{xx}}\xi_{\mathbf{x}} + \boldsymbol{\Sigma}_{\mathbf{xy}}\xi_{\mathbf{y}}$. Au final, on a donc bien $\mu_{\mathbf{marg}} = \mu_{\mathbf{x}}$ avec l'équation 1.37. Ceci achève de démontrer le résultat.

■

Remarque 1.3 (Variante concernant la démonstration du théorème 1.3)
Une autre possibilité de démonstration aurait consisté à s'arrêter après l'équation 1.33. En effet, cette équation nous informe que $\mathbf{x}$ *est un vecteur aléatoire gaussien. On a par ailleurs directement ses paramètres d'information. En ce qui concerne les paramètres " classiques ", on aurait pu utiliser la définition formelle de l'espérance et de la matrice de variances-covariances et remarquer que ceux-ci correspondent systématiquement aux paramètres des distributions marginales. Néanmoins, le calcul effectué permet de montrer d'une autre façon la cohérence des résultats ; il sera par ailleurs utilisé pour le théorème de conditionnement.*

Théorème de conditionnement

Nous présentons dans ce paragraphe le théorème de conditionnement dans le cas gaussien. Celui-ci permet de déduire $p(\mathbf{x}|\mathbf{y})$ à partir de $p(\mathbf{x}, \mathbf{y})$.

Théorème 1.4 (Conditionnement de vecteurs aléatoires gaussiens)
Sous les mêmes hypothèses que celles du théorème de marginalisation, la densité de probabilité

$p(\mathbf{x}|\mathbf{y}=\mathbf{y_0})$ *est gaussienne dont les paramètres sont donnés par :*

$$\begin{aligned} \textit{Matrice de variances-covariances :}\ & \boldsymbol{\Sigma}_{\mathbf{cond}} = \boldsymbol{\Sigma}_{\mathbf{xx}} - \boldsymbol{\Sigma}_{\mathbf{xy}}\boldsymbol{\Sigma}_{\mathbf{yy}}^{-1}\boldsymbol{\Sigma}_{\mathbf{yx}} \\ \textit{Espérance mathématique :}\ & \mu_{\mathbf{cond}} = \boldsymbol{\Sigma}_{\mathbf{cond}}\xi_{\mathbf{x}} + \boldsymbol{\Sigma}_{\mathbf{xy}}\boldsymbol{\Sigma}_{\mathbf{yy}}^{-\mathbf{1}}\mathbf{y_0} \\ \textit{Matrice d'information :}\ & \boldsymbol{\Omega}_{\mathbf{cond}} = \boldsymbol{\Omega}_{\mathbf{xx}} \\ \textit{Vecteur d'information :}\ & \xi_{\mathbf{cond}} = \xi_{\mathbf{x}} - \boldsymbol{\Omega}_{\mathbf{xy}}\mathbf{y_0} \end{aligned} \tag{1.38}$$

Le théorème 1.4 nous dit que le conditionnement d'un vecteur aléatoire gaussien reste gaussien. Par ailleurs, les nouveaux paramètres s'obtiennent particulièrement facilement lorsqu'on travaille dans l'espace d'information.

Démonstration du théorème 1.4 :

▲ Pour démontrer ce théorème, on utilise le résultat sur le calcul des densités conditionnelles (théorème 1.1). On a donc :

$$\begin{aligned} p(\mathbf{x}|\mathbf{y}=\mathbf{y_0}) &\propto p(\mathbf{x},\mathbf{y_0}) \\ &\propto \exp\left(-\frac{1}{2}\begin{bmatrix}\mathbf{x}^T & \mathbf{y_0}^T\end{bmatrix}\begin{bmatrix}\boldsymbol{\Omega}_{\mathbf{xx}} & \boldsymbol{\Omega}_{\mathbf{xy}} \\ \boldsymbol{\Omega}_{\mathbf{yx}} & \boldsymbol{\Omega}_{\mathbf{yy}}\end{bmatrix}\begin{bmatrix}\mathbf{x} \\ \mathbf{y_0}\end{bmatrix} + \begin{bmatrix}\mathbf{x}^T & \mathbf{y_0}^T\end{bmatrix}\begin{bmatrix}\xi_{\mathbf{x}} \\ \xi_{\mathbf{y}}\end{bmatrix}\right) \\ &= \exp\left(-\frac{1}{2}\mathbf{x}^T\boldsymbol{\Omega}_{\mathbf{xx}}\mathbf{x} + \mathbf{x}^T\xi_{\mathbf{x}} - \mathbf{x}^T\boldsymbol{\Omega}_{\mathbf{xy}}\mathbf{y_0}\right) \times \underbrace{\exp\left(-\frac{1}{2}\mathbf{y_0}^T\boldsymbol{\Omega}_{\mathbf{yy}}\mathbf{y_0} + \mathbf{y_0}^T\xi_{\mathbf{y}}\right)}_{\text{Indépendant de } \mathbf{x}} \end{aligned}$$

Au final, on a donc :

$$p(\mathbf{x}|\mathbf{y}=\mathbf{y_0}) \propto \exp\left(-\frac{1}{2}\mathbf{x}^T\boldsymbol{\Omega}_{\mathbf{xx}}\mathbf{x} + \mathbf{x}^T\left(\xi_{\mathbf{x}} - \boldsymbol{\Omega}_{\mathbf{xy}}\mathbf{y_0}\right)\right) \tag{1.39}$$

L'équation 1.39 montre que $p(\mathbf{x})$ est proportionnel à l'exponentielle d'une forme quadratique. D'après l'équation 1.28, $\mathbf{x}$ suit donc une loi gaussienne dont les paramètres d'information correspondent aux coefficients de la forme quadradique. La matrice d'information est donc égale à $\boldsymbol{\Omega}_{\mathbf{cond}} = \boldsymbol{\Omega}_{\mathbf{xx}}$ et son vecteur d'information est égal à $\xi_{\mathbf{cond}} = \xi_{\mathbf{x}} - \boldsymbol{\Omega}_{\mathbf{xy}}\mathbf{y_0}$. En utilisant un raisonnement semblable à celui effectué dans la démonstration précédente, on montre que :

$$\boldsymbol{\Sigma}_{\mathbf{cond}} = \boldsymbol{\Sigma}_{\mathbf{xx}} - \boldsymbol{\Sigma}_{\mathbf{xy}}\boldsymbol{\Sigma}_{\mathbf{yy}}^{-1}\boldsymbol{\Sigma}_{\mathbf{yx}} \tag{1.40}$$

$$\mu_{\mathbf{cond}} = \underbrace{\left(\boldsymbol{\Sigma}_{\mathbf{xx}} - \boldsymbol{\Sigma}_{\mathbf{xy}}\boldsymbol{\Sigma}_{\mathbf{yy}}^{-1}\boldsymbol{\Sigma}_{\mathbf{yx}}\right)}_{(\boldsymbol{\Sigma}_{\mathbf{cond}})}\left(\xi_{\mathbf{x}} + \underbrace{\left(\boldsymbol{\Sigma}_{\mathbf{xx}} - \boldsymbol{\Sigma}_{\mathbf{xy}}\boldsymbol{\Sigma}_{\mathbf{yy}}^{-1}\boldsymbol{\Sigma}_{\mathbf{yx}}\right)^{-1}\boldsymbol{\Sigma}_{\mathbf{xy}}\boldsymbol{\Sigma}_{\mathbf{yy}}^{-\mathbf{1}}}_{(-\boldsymbol{\Omega}_{\mathbf{xy}})}\mathbf{y_0}\right) \tag{1.41}$$

Au final, l'équation 1.41 devient :

$$\mu_{\mathbf{cond}} = \left(\mathbf{\Sigma_{xx}} - \mathbf{\Sigma_{xy}}\mathbf{\Sigma_{yy}^{-1}}\mathbf{\Sigma_{yx}}\right)\xi_{\mathbf{x}} + \mathbf{\Sigma_{xy}}\mathbf{\Sigma_{yy}^{-1}}\mathbf{y_0} = \mathbf{\Sigma_{cond}}\xi_{\mathbf{x}} + \mathbf{\Sigma_{xy}}\mathbf{\Sigma_{yy}^{-1}}\mathbf{y_0} \tag{1.42}$$

Le résultat des équations 1.40 et 1.42 achèvent la démonstration.

■

Nous avons désormais effectué les principaux rappels concernant les probabilités, et plus particulièrement les densités de probabilités gaussiennes. Ces rappels étant faits, nous pouvons désormais formuler le problème du SLAM probabiliste.

1.2 Formulation du problème

1.2.1 Notations et hypothèses

Nous présentons dans cette partie les bases du problème du SLAM ainsi que les notations employées. Celles-ci sont adaptées de [Durrant-Whyte et Bailey, 2006; Mei, 2007].

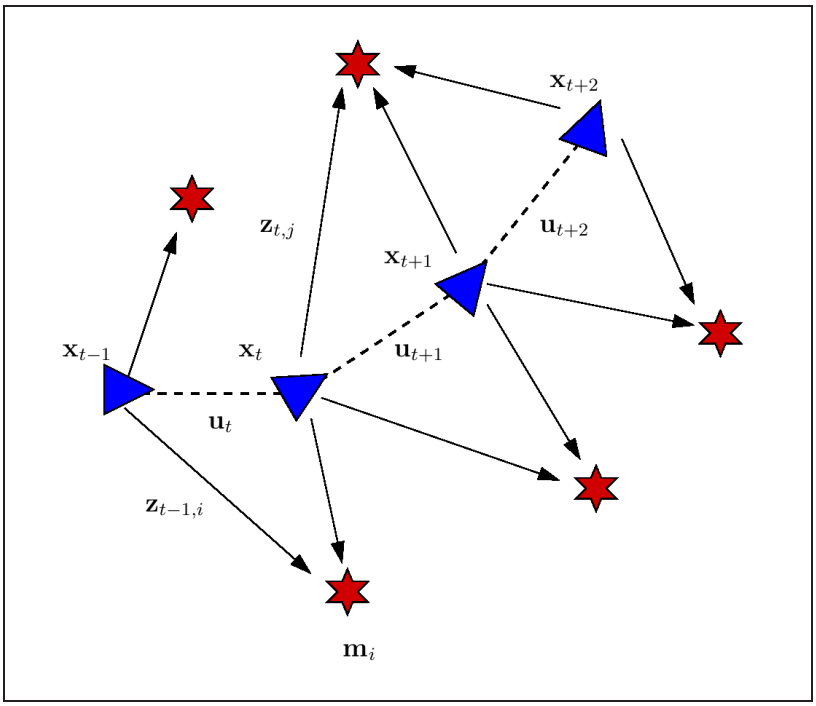


FIGURE 1.1 – Notations pour le problème du SLAM ([Mei, 2007])

Dans le cadre du SLAM, on cherche à retrouver à chaque instant l'état du robot, noté $\mathbf{x}$, ainsi que la position d'un certain nombre d'amers (supposés statiques) dans l'environnement. Le robot ne

pourra mesurer que des informations relatives par rapport aux amers. Nous nous plaçons dans le cas où nous n'avons aucune information globale ni aucun *a priori* concernant les amers. Nous définissons les notations suivantes (cf. figure 1.1) :

- $t = 0, 1, 2, \ldots$ le temps discret,
- $\mathbf{x}_t$ l'état du robot à l'instant t. Il s'agit en général de la position et de l'orientation, mais cet état peut contenir aussi les vitesses par exemple,
- $\mathbf{u}_t$ le vecteur de contrôle appliqué à l'instant $t-1$ pour conduire le robot de l'état $\mathbf{x}_{t-1}$ à l'état $\mathbf{x}_t$,
- $\mathbf{m}_{(i)}$ la position du $i^{\text{ème}}$ amer, [5]
- $\mathbf{z}_{t,(i)}$ le vecteur d'observation du $i^{\text{ème}}$ amer obtenu depuis la position $\mathbf{x}_t$ à l'instant t,
- $\mathbf{z}_t$ le vecteur d'observation de tous les amers vus à l'instant t.

Par ailleurs, nous utilisons des " notations Matlab " pour définir les ensembles suivants :

- les états entre les instants t_1 et t_2 : $\mathbf{x}_{t_1:t_2} = \{\mathbf{x}_{t_1}, \mathbf{x}_{t_1+1}, \ldots, \mathbf{x}_{t_2}\} = \{\mathbf{x}_{t_1:t_2-1}, \mathbf{x}_{t_2}\}$,
- les commandes entre les instants t_1 et t_2 : $\mathbf{u}_{t_1:t_2} = \{\mathbf{u}_{t_1}, \mathbf{u}_{t_1+1}, \ldots, \mathbf{u}_{t_2}\} = \{\mathbf{u}_{t_1:t_2-1}, \mathbf{u}_{t_2}\}$,
- les observations entre les instants t_1 et t_2 : $\mathbf{z}_{t_1:t_2} = \{\mathbf{z}_{t_1}, \mathbf{z}_{t_1+1}, \ldots, \mathbf{z}_{t_2}\} = \{\mathbf{z}_{t_1:t_2-1}, \mathbf{z}_{t_2}\}$.

Nous notons également l'ensemble de tous les amers $\mathbf{m} = \{\mathbf{m}_{(1)}, \mathbf{m}_{(2)}, \ldots, \mathbf{m}_{(M)}\}$. Enfin, nous effectuons les hypothèses suivantes :

- aucune information *a priori* n'est disponible quant à la position des amers,
- l'état initial $\mathbf{x}_0$ est connu, [6]
- la séquence de contrôles $\mathbf{u}_{0:t}$ est connue.

Pour résoudre ce problème, nous effectuons les 4 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 :

$$p\left(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t\right) = p\left(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t, \mathbf{x}_{0:t-2}, \mathbf{u}_{0:t-1}, \mathbf{m}\right) \tag{1.43}$$

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

$$p\left(\mathbf{z}_{0:t}|\mathbf{x}_{0:t}, \mathbf{m}\right) = \prod_{i=0}^{t} p\left(\mathbf{z}_i|\mathbf{x}_{0:t}, \mathbf{m}\right) \tag{1.44}$$

5. Afin d'éviter toute confusion entre les indices désignant le temps et ceux désignant les amers, nous mettrons systématiquement *entre parenthèses* les indices désignant un *numéro d'amer*.

6. Si ce n'est pas le cas, on supposera connue sa distribution *a priori* : $p\left(\mathbf{x}_0\right)$

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

$$p\left(\mathbf{z}_k|\mathbf{x}_{0:t}, \mathbf{m}, \ldots\right) = p\left(\mathbf{z}_k|\mathbf{x}_k, \mathbf{m}\right) \tag{1.45}$$

4. A chaque instant, la mesure de l'amer (i) est indépendante des mesures des autres amers :

$$p\left(\mathbf{z}_{k,(i)}|\mathbf{x}_k, \mathbf{m}\right) = p\left(\mathbf{z}_{k,(i)}|\mathbf{x}_k, \mathbf{m}_{(i)}\right) \tag{1.46}$$

Ces 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 (les modèles de robot de type tricycle ou char par exemple respectent parfaitement cette hypothèse). 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. De plus, les capteurs que l'on utilise mesurent la position relative du robot et des amers à un instant bien précis, ce qui justifie la troisième hypothèse. Enfin, nous admettons que les mesures des amers à un instant donné soit décorrolées. Il s'agit de l'hypothèse la moins évidente en pratique (surtout dans le cas où le capteur utilisé est une caméra : une unique image donne toutes les mesures), mais elle permet de simplifier largement les développement du SLAM. [7]

Finalement, ces 4 hypothèses nous permettent de représenter le SLAM sous forme d'un réseau bayesien orienté (cf. figure 1.2). Chaque flèche traduit l'influence directe d'un vecteur sur un autre (soit par le biais de la chaîne de Markov de l'état du robot, soit par le biais des mesures). Le lecteur intéressé pourra se reporter en annexe A pour interpréter précisément un tel réseau. Plusieurs propriétés de ce réseau seront par ailleurs exploitées par la suite (chapitre 8).

1.2.2 Le SLAM probabiliste et récursif

Dans le cas du SLAM probabiliste, l'objectif est de déterminer à chaque instant t la *densité de probabilité* de $\mathbf{x}_t$ et de la carte sachant l'ensemble des mesures, des contrôles et l'état initial. Il s'agit donc de trouver pour tout t :

$$p\left(\mathbf{x}_t, \mathbf{m}|\mathbf{z}_{0:t}, \mathbf{u}_{0:t}, \mathbf{x}_0\right) \tag{1.47}$$

L'hypothèse Markovienne nous permet d'obtenir la densité *a priori* de l'état du robot et de la carte à l'instant t (ie. la densité de $\mathbf{x}_t$ et $\mathbf{m}$ sachant les mesures jusqu'à l'instant $t-1$, l'historique des

7. De plus, ces corrélations supplémentaires sont quasiment impossible à déterminer en pratique.

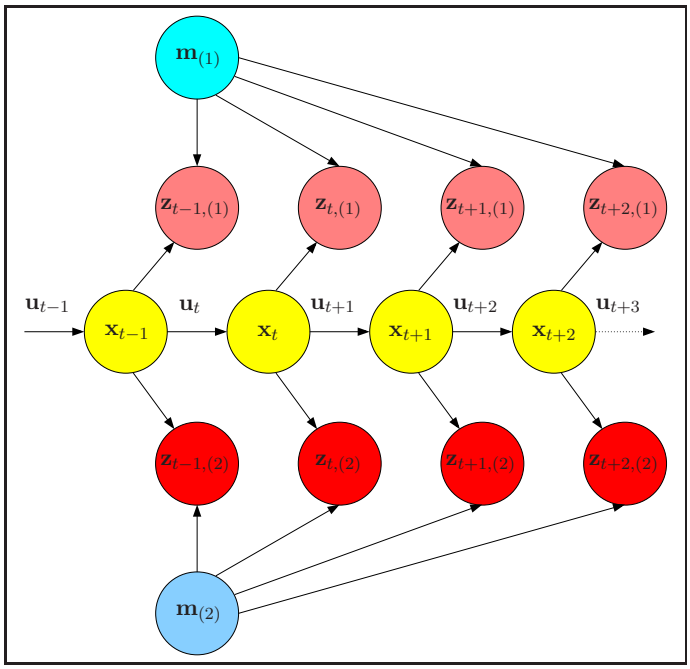


FIGURE 1.2 – Réseau bayésien associé au problème du SLAM (dans le cas de 2 amers).

commandes et l'état initial) :

$$\begin{aligned} p\left(\mathbf{x}_t, \mathbf{m} | \mathbf{z}_{0:t-1}, \mathbf{u}_{0:t}, \mathbf{x}_0\right) &= \int p\left(\mathbf{x}_t, \mathbf{x}_{t-1}, \mathbf{m} | \mathbf{z}_{0:t-1}, \mathbf{u}_{0:t}, x_0\right) d\mathbf{x}_{t-1} \\ &= \int p\left(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{m}, \mathbf{z}_{0:t-1}, \mathbf{u}_{0:t}, x_0\right) p\left(\mathbf{x}_{t-1}, \mathbf{m} | \mathbf{z}_{0:t-1}, \mathbf{u}_{0:t}, x_0\right) d\mathbf{x}_{t-1} \quad \text{(règle de Bayes)} \\ &= \int p\left(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{u}_t\right) p\left(\mathbf{x}_{t-1}, \mathbf{m} | \mathbf{z}_{0:t-1}, \mathbf{u}_{0:t-1}, \mathbf{x}_0\right) d\mathbf{x}_{t-1} \end{aligned} \tag{1.48}$$

En appliquant la règle de Bayes de deux façons différentes, on obtient :

$$p\left(\mathbf{x}_t, \mathbf{m}, \mathbf{z}_t | \mathbf{z}_{0:t-1}, \mathbf{u}_{0:t}, \mathbf{x}_0\right) = p\left(\mathbf{x}_t, \mathbf{m} | \mathbf{z}_{0:t}, \mathbf{u}_{0:t}, \mathbf{x}_0\right) p\left(\mathbf{z}_t | \mathbf{z}_{0:t-1}, \mathbf{u}_{0:t}, \mathbf{x}_0\right) \tag{1.49}$$

$$p\left(\mathbf{x}_t, \mathbf{m}, \mathbf{z}_t | \mathbf{z}_{0:t-1}, \mathbf{u}_{0:t}, \mathbf{x}_0\right) = \underbrace{p\left(\mathbf{z}_t | \mathbf{x}_t, \mathbf{m}, \mathbf{z}_{0:t-1}, \mathbf{u}_{0:t}, \mathbf{x}_0\right)}_{p(\mathbf{z}_t | \mathbf{x}_t, \mathbf{m})} p\left(\mathbf{x}_t, \mathbf{m} | \mathbf{z}_{0:t-1}, \mathbf{u}_{0:t}, \mathbf{x}_0\right) \tag{1.50}$$

En combinant les équations 1.49 et 1.50, il vient :

$$p\left(\mathbf{x}_t, \mathbf{m} | \mathbf{z}_{0:t}, \mathbf{u}_{0:t}, \mathbf{x}_0\right) = \frac{p\left(\mathbf{z}_t | \mathbf{x}_t, \mathbf{m}\right) p\left(\mathbf{x}_t, \mathbf{m} | \mathbf{z}_{0:t-1}, \mathbf{u}_{0:t}, \mathbf{x}_0\right)}{p\left(\mathbf{z}_t | \mathbf{z}_{0:t-1}, \mathbf{u}_{0:t}\right)} \tag{1.51}$$

Le dénominateur de l'équation 1.51 ne dépend ni de l'état du robot, ni de la carte. Il peut être vu comme un facteur de normalisation. En combinant les équations 1.48 et 1.51, on aboutit finalement à :

$$\boxed{p\left(\mathbf{x}_t, \mathbf{m} | \mathbf{z}_{0:t}, \mathbf{u}_{0:t}, \mathbf{x}_0\right) = \eta p\left(\mathbf{z}_t | \mathbf{x}_t, \mathbf{m}\right) \int p\left(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{u}_t\right) p\left(\mathbf{x}_{t-1}, \mathbf{m} | \mathbf{z}_{0:t-1}, \mathbf{u}_{0:t-1}, \mathbf{x}_0\right) d\mathbf{x}_{t-1}} \tag{1.52}$$

où η est une constante de normalisation. Dans l'équation 1.52, on trouve les facteurs connus suivants :

- $p(\mathbf{x}_t, \mathbf{m}|\mathbf{z}_{0:t}, \mathbf{u}_{0:t}, \mathbf{x}_0)$: densité de probabilité *a posteriori* de l'état du robot et de la carte à l'instant t,
- $p(\mathbf{x}_{t-1}, \mathbf{m}|\mathbf{z}_{0:t-1}, \mathbf{u}_{0:t-1}, \mathbf{x}_0)$: densité de probabilité *a posteriori* de l'état du robot et de la carte à l'instant $t-1$,
- $p(\mathbf{z}_t|\mathbf{x}_t, \mathbf{m})$: la densité associée aux observations sachant la position du robot et la carte,
- $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t)$: la densité associée à la chaîne de Markov décrivant l'évolution de la position du robot en fonction de la commande et de sa dernière position.

Nous avons désormais une formulation probabiliste et récursive du problème du SLAM. Néanmoins, celle-ci n'est pas utilisable en l'état : le calcul de l'intégrale n'est pas possible sans effectuer d'hypothèses supplémentaires.

1.2.3 Hypothèses supplémentaires concernant les densités de probabilités

Dans toute la suite de ce chapitre, nous supposons que toutes les densités de probabilités considérées sont gaussiennes. Ainsi, les densités associées à la chaîne de Markov et au modèle de mesure ne dépendent que de deux moments : l'espérance et la matrice de variances-covariances. Nous posons :

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t) = \mathcal{N}(\mathbf{f}(\mathbf{x}_{t-1}, \mathbf{u}_t), \mathbf{Q}_t) \quad (1.53)$$

$$p(\mathbf{z}_t|\mathbf{x}_t, \mathbf{m}) = \mathcal{N}(\mathbf{h}(\mathbf{x}_t, \mathbf{m}), \mathbf{R}_t) \quad (1.54)$$

Dans l'équation 1.53, $\mathbf{f}$ modélise la cinématique du véhicule. La fonction $\mathbf{h}$ permet quant à elle de prédire la sortie du capteur lorsque l'on connait les états du robot et des amers.

Enfin, on peut montrer que dans le cas où les fonctions $\mathbf{f}$ et $\mathbf{h}$ sont linéaires (ie. $\mathbf{f}(\mathbf{x_{t-1}}, \mathbf{u_t}) = \mathbf{F}\mathbf{x}_{t-1} + \mathbf{B}\mathbf{u}_t$ et $\mathbf{h}(\mathbf{x}_t, \mathbf{m}) = \mathbf{H_x}\mathbf{x}_t + \mathbf{H_m}\mathbf{m}$), l'équation 1.52 peut être résolue analytiquement. Le résultat est alors identique à celui obtenu par un filtre de Kalman appliqué au système suivant :

$$\begin{cases} \mathbf{x}_t = \mathbf{F}\mathbf{x}_{t-1} + \mathbf{B}\mathbf{u}_t + \mathbf{w}_t \\ \mathbf{m}_t = \mathbf{m}_{t-1} = \mathbf{m} \end{cases} \quad \text{et} \quad \mathbf{z}_t = \mathbf{H_x}\mathbf{x}_t + \mathbf{H_m}\mathbf{m} + \mathbf{v}_t$$

où $\mathbf{w}_t$ et $\mathbf{v}_t$ sont des bruits centrés gaussiens de variances respectives $\mathbf{Q}_t$ et $\mathbf{R}_t$.[8] Néanmoins, l'intérêt de ce dernier résultat n'est que théorique car les modèles utilisés pour prédire les déplacements du robot et les mesures des capteurs sont non linéaires. Dans la suite de ce chapitre, nous présentons les méthodes les plus utilisées pour prendre en compte les non linéarités.

8. Les vecteurs $\mathbf{w}_t$ et $\mathbf{v}_t$ sont en général désignés par les expressions *bruit de modèle* et *bruit de mesures.*

1.3 Résultats préliminaires d'observabilité et importance de l'initialisation

1.3.1 Observabilité

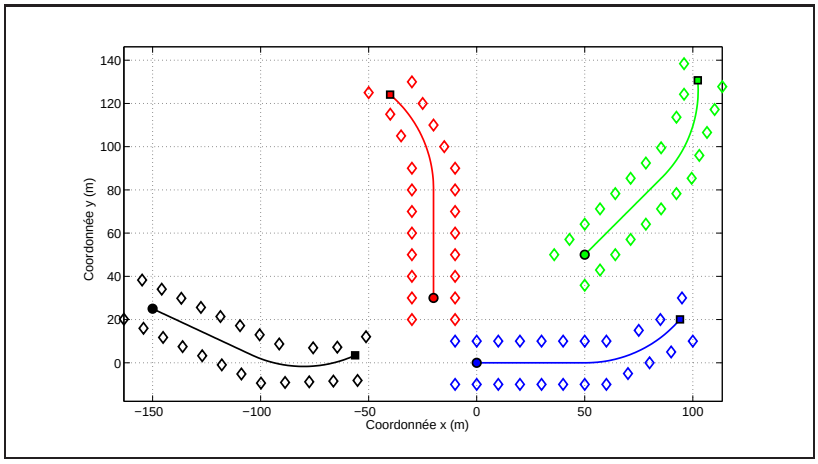


FIGURE 1.3 – Caractère non observable du SLAM – Différentes trajectoires possibles pour un jeu de données fixé sont représentées en traits pleins (un cercle marque le début et un carré marque la fin). Différents amers ponctuels sont représentés avec des losanges. Lorsque nous ne possédons aucune information absolue sur la trajectoire du robot ou sur les amers, il existe une infinité de solutions possibles. On passe de l'une à l'autre par une transformation rigide.

Nous avons vu dans l'introduction générale que nous souhaitons résoudre le problème du SLAM sans utiliser de capteur donnant une information absolue par rapport au repère global (tel que le GPS ou la boussole). Les seuls capteurs dont nous disposons sont constitués de la vision qui peut éventuellement être augmentée par des données odométriques du robot. Aucun des capteurs " autorisés " ne nous permet de donner la position dans un repère absolu :

1. La vision nous donne des informations relatives entre l'environnement et le robot. Celles-ci permettent de trouver la direction de vue d'amers dans le repère du robot. Si la position du robot est inconnue, la position des amers ne peut être déduite.
2. L'odométrie nous donne une information relative de déplacement entre deux instants. Ainsi, l'intégration des données odométriques ne peut se faire qu'avec une condition initiale.

Ainsi, il est assez facile de démontrer que le système d'état associé au SLAM est inobservable d'après le critère d'Hermann et Krenner ([Hermann et Krener, 1977]). Cela a été fait dans plusieurs publications pour le cas où l'environnement est perçu avec un capteur de type *range and bearing* (de type laser). Nous proposons une démonstration analogue pour le cas *bearing-only* en annexe B.

Cette propriété d'inobservabilité est assez intuitive. En effet, si le robot ne mesure que des informations relatives entre les positions successives et des informations relatives avec l'environnement, alors il existe une infinité de solutions équivalentes (cf. figure 1.3). Il est possible de passer d'une solution à l'autre à l'aide d'une transformation rigide.[9] De ce fait, le problème du SLAM tel qu'on le défini (trouver la position du robot et des amers) parait intrinsèquement mal posé, et ceci quelle que soit la méthode de résolution que l'on va utiliser. Il paraît donc nécessaire de fixer une hypothèse. Dans la suite, de cette section, nous discutons l'intérêt de définir la condition initiale du robot.

1.3.2 Condition initiale

Pour pallier au problème d'observabilité, nous choisissons de fixer la condition initiale du robot. Il s'agit de la solution la plus " naturelle ", et c'est celle qui est en général retenue (implicitement ou non) dans la plupart des publications de SLAM. Ainsi, si on suppose que le repère initial du robot est confondu avec le repère absolu, alors il n'existe qu'une seule solution valide parmi l'ensemble des solutions définies dans le paragraphe précédent (il s'agit de la solution bleue sur la figure 1.3). De ce fait, lorsque le but sera de créer une carte qui n'est liée à aucun repère absolu, nous supposerons que la **première position du robot est fixée à zéro** et sa **matrice de rotation initiale est fixée à l'identité**. Si nous devons fixer un repère absolu et que la position initiale n'est pas certaine, nous supposerons connaître la densité *a priori* de l'état initial du robot.

Sauf mention contraire, les densités de probabilités calculées seront systématiquement conditionnées au vecteur d'état du robot à l'instant initial.[10]

1.4 Résolution par filtre de Kalman : l'EKF-SLAM

Nous présentons dans cette section l'algorithme d'estimation le plus couramment utilisé pour résoudre le problème du SLAM ([Thrun, 2002; Castellanos *et al.*, 1999; Durrant-Whyte *et al.*, 2001; Dissanayake *et al.*, 2001; Newman, 2000]). Il s'agit du filtre de Kalman étendu : un algorithme dérivé du filtre de Kalman pour prendre en compte les non linéarités des fonctions $\mathbf{f}$ et $\mathbf{h}$.

Nous rappelons dans un premier temps le principe général de ce filtre ainsi que les équations. Ensuite, nous explicitons les principaux avantages et inconvénients de cet algorithme.

9. Dans le cas 2D présenté sur la figure 1.3, on a 3 degrés de liberté : 2 en translation et un en rotation.
10. Ce qui est équivalent à affirmer que la densité de probabilité *a priori* de la position initiale du robot est un dirac centré à l'origine.

1.4.1 Principe

Dans l'EKF-SLAM, les modèles utilisés sont définis par :

$$\left\{ \begin{array}{l} \mathbf{x}_t = \mathbf{f}_t\left(\mathbf{x}_{t-1}, \mathbf{u}_t\right) + \mathbf{w}_t \\ \mathbf{m}_t = \mathbf{m}_{t-1} = \mathbf{m} \end{array} \right. \quad \text{et} \quad \mathbf{z}_t = \mathbf{h}\left(\mathbf{x}_t, \mathbf{m}\right) + \mathbf{v}_t \tag{1.55}$$

où $\mathbf{w}_t$ et $\mathbf{v}_t$ sont des bruits centrés gaussiens de variances respectives $\mathbf{Q}_t$ et $\mathbf{R}_t$. L'objectif de l'EKF-SLAM est d'estimer au cours du temps l'**état courant du robot** et la **carte**.

En raison de la présence de modèles non linéaires, la densité de probabilité définie à l'équation 1.47 n'est pas gaussienne. Néanmoins, le filtre de Kalman étendu linéarise les fonctions $\mathbf{f}$ et $\mathbf{h}$. En admettant que l'amplitude des erreurs soit suffisamment faible et la linéarisation valable, il est admis que le résultat de l'EKF correspond aux paramètres statistiques d'une variable aléatoire gaussienne, à savoir :

1. l'espérance associée au vecteur d'état joint contenant la dernière pose du robot et la carte.

$$\left[\begin{array}{c} \hat{\mathbf{x}}_{t|t} \\ \hat{\mathbf{m}}_{t|t} \end{array} \right] \approx \mathbf{E}\left(\begin{array}{c} \mathbf{x}_t \\ \mathbf{m} \end{array} |\mathbf{z}_{0:t}, \mathbf{u}_{0:t}, \mathbf{x}_0 \right) \tag{1.56}$$

2. la matrice de variances-covariances associée :

$$\begin{aligned} \mathbf{P}_{t|t} &= \left[\begin{array}{cc} \mathbf{P}_{\mathbf{xx}} & \mathbf{P}_{\mathbf{xm}} \\ \mathbf{P}_{\mathbf{xm}}^T & \mathbf{P}_{\mathbf{mm}} \end{array} \right]_{t|t} \\ &\approx \mathbf{E}\left(\left(\begin{array}{c} \mathbf{x}_t - \hat{\mathbf{x}}_{t|t} \\ \mathbf{m} - \hat{\mathbf{m}}_{t|t} \end{array} \right) \left(\begin{array}{c} \mathbf{x}_t - \hat{\mathbf{x}}_{t|t} \\ \mathbf{m} - \hat{\mathbf{m}}_{t|t} \end{array} \right)^T |\mathbf{z}_{0:t}, \mathbf{u}_{0:t}, \mathbf{x}_0 \right) \end{aligned} \tag{1.57}$$

où l'indice $i|j$ indique que l'on désigne l'estimation de l'état à l'instant i avec la connaissance des mesures jusqu'à l'instant j. Les paramètres $\hat{\mathbf{x}}_{t|t}$, $\hat{\mathbf{m}}_{t|t}$ et $\mathbf{P}_{t|t}$ sont obtenus récursivement grâce aux deux étapes " classiques " du filtre de Kalman étendu rappelées ci-dessous.

Phase de prédiction : permet de déduire l'état et la matrice de variances-covariances *a priori*.

$$\hat{\mathbf{x}}_{t|t-1} = \mathbf{f}_t\left(\hat{\mathbf{x}}_{t-1|t-1}, \mathbf{u}_t\right) \tag{1.58}$$

$$\hat{\mathbf{m}}_{t|t-1} = \hat{\mathbf{m}}_{t-1|t-1} \tag{1.59}$$

$$\mathbf{P}_{\mathbf{xx},t|t-1} = (\nabla_{\mathbf{x}}\mathbf{f}_t)\mathbf{P}_{\mathbf{xx},t-1|t-1}(\nabla_{\mathbf{x}}\mathbf{f}_t)^T + (\nabla_{\mathbf{u}}\mathbf{f}_t)\mathbf{U}_t(\nabla_{\mathbf{u}}\mathbf{f}_t)^T + \mathbf{Q}_t \tag{1.60}$$

$$\mathbf{P}_{\mathbf{xm},t|t-1} = (\nabla_{\mathbf{x}}\mathbf{f}_t)\mathbf{P}_{\mathbf{xm},t-1|t-1} \tag{1.61}$$

$$\mathbf{P}_{\mathbf{mm},t|t-1} = \mathbf{P}_{\mathbf{mm},t-1|t-1} \tag{1.62}$$

où $\mathbf{U}_t$ est la matrice de variances-covariances du vecteur de commande (lorsque ce dernier est incertain) et :

$$\nabla_{\mathbf{x}}\mathbf{f}_t = \left.\frac{\partial \mathbf{f}_t(\mathbf{x},\mathbf{u})}{\partial \mathbf{x}}\right|_{\substack{\mathbf{x}=\hat{\mathbf{x}}_{t|t-1}\\ \mathbf{u}=\mathbf{u}_t}} \quad \text{et} \quad \nabla_{\mathbf{u}}\mathbf{f}_t = \left.\frac{\partial \mathbf{f}_t(\mathbf{x},\mathbf{u})}{\partial \mathbf{u}}\right|_{\substack{\mathbf{x}=\hat{\mathbf{x}}_{t|t-1}\\ \mathbf{u}=\mathbf{u}_t}}$$

Phase de correction : permet de déduire l'état et la matrice de variances-covariances *a posteriori* grâce au vecteur d'observation.

$$\begin{bmatrix} \hat{\mathbf{x}}_{t|t} \\ \hat{\mathbf{m}}_{t|t} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{x}}_{t|t-1} \\ \hat{\mathbf{m}}_{t|t-1} \end{bmatrix} + \mathbf{W}_t \left[\mathbf{z}_t - \mathbf{h}\left(\hat{\mathbf{x}}_{t|t-1}, \hat{\mathbf{m}}_{t|t-1}\right)\right] \tag{1.63}$$

$$\mathbf{P}_{t|t} = \mathbf{P}_{t|t-1} - \mathbf{W}_t\mathbf{S}_t\mathbf{W}_t^T \tag{1.64}$$

où $\mathbf{S}_t$ est la matrice de variances-covariances de l'innovation et $\mathbf{W}_t$ le matrice de gain de Kalman :

$$\mathbf{S}_t = (\nabla_{\mathbf{xm}}\mathbf{h})\mathbf{P}_{t|t-1}(\nabla_{\mathbf{xm}}\mathbf{h})^T + \mathbf{R}_t \tag{1.65}$$

$$\mathbf{W}_t = \mathbf{P}_{t|t-1}(\nabla_{\mathbf{xm}}\mathbf{h})^T\mathbf{S}_t^{-1} \tag{1.66}$$

avec :

$$\nabla_{\mathbf{xm}}\mathbf{h} = \left[\left.\frac{\partial \mathbf{h}(\mathbf{x},\mathbf{m})}{\partial \mathbf{x}}\right|_{\substack{\mathbf{x}=\hat{\mathbf{x}}_{t|t-1}\\ \mathbf{m}=\hat{\mathbf{m}}_{t|t-1}}} , \left.\frac{\partial \mathbf{h}(\mathbf{x},\mathbf{m})}{\partial \mathbf{m}}\right|_{\substack{\mathbf{x}=\hat{\mathbf{x}}_{t|t-1}\\ \mathbf{m}=\hat{\mathbf{m}}_{t|t-1}}} \right]$$

Les deux phases du filtre de Kalman étendu nous permettent donc finalement d'obtenir les deux moments statistiques caractérisant totalement $p\left(\mathbf{x}_t, \mathbf{m}|\mathbf{z}_{0:t}, \mathbf{u}_{0:t}, \mathbf{x}_0\right)$ dans le cas gaussien.

1.4.2 Avantages et limites pratiques

L'EKF-SLAM possède plusieurs avantages ([Dissanayake *et al.*, 2001; Durrant-Whyte et Bailey, 2006]) :

- le déterminant de chaque sous-matrice de la matrice de variances-covariances de la carte décroît à chaque observation. Cela signifie que la taille des zones de confiance (des ellipses dans notre cas qui est le SLAM gaussien) diminue strictement à chaque itération,
- si le nombre d'observations de chaque amer est infini, la carte devient complètement corrélée (ce qui indique que les positions des amers ne peuvent être modifiées que toutes ensembles et de façon rigide),
- si le nombre d'observations de chaque amer est infini, la variance de chaque amer converge vers une limite déterminée uniquement par l'incertitude initiale sur la position du robot.

Ces résultats de convergence sont démontrés dans [Dissanayake *et al.*, 2001] pour le cas général du SLAM linéaire gaussien. Ils indiquent en outre que le maintien de la matrice de variances-covariances complète est primordial dans le problème.

Néanmoins, l'EKF-SLAM pose certains problèmes non négligeables ([Mei, 2007]) :

- l'EKF-SLAM utilise une *approximation linéaire*. Ceci entraîne nécessairement des erreurs et peut rendre le filtre inconsistant (ce qui se traduit par une sous-estimation de l'erreur commise),
- la *complexité algorithmique* de l'EKF-SLAM empêche sa mise en œuvre dans le cas de grandes cartes car les opérations de stockage et de mise à jour de la matrice de variances-covariances totale (état du robot + amers) requiert une complexité en $\mathcal{O}(M^2)$ (M désignant le nombre d'amers dans la carte). [11]

1.4.3 Consistance de l'EKF-SLAM

Nous abordons dans cette section les problèmes de consistance liés à l'EKF-SLAM. La consistance peut être vue comme la qualité de l'incertitude fournie par le filtre (à travers la matrice de variances-covariances) par rapport à l'erreur réelle. Mesurer la consistance d'un algorithme nécessite d'avoir une *vérité terrain* ou alors de travailler en *simulation*. La deuxième solution est souvent employée (cf. par exemple [Julier et Uhlmann, 2001; Bailey *et al.*, 2006a]).

Nous détaillons dans ce paragraphe les résultats théoriques et " pratiques " (simulés) issus de trois travaux différents (présentés par ordre chronologique). Nous allons voir que tous ces travaux tendent à montrer que l'EKF-SLAM est un algorithme inconsistant.

1.4.3.1 Effets non désirés de l'EKF-SLAM ([Julier et Uhlmann, 2001])

Dans [Julier et Uhlmann, 2001], les auteurs mettent théoriquement en évidence le caractère inconsistant de l'EKF-SLAM dans le cas d'un véhicule immobile. Ils considèrent un véhicule parfaitement immobile (et des amers parfaitement immobiles eux aussi). L'équation d'évolution du filtre est donc triviale [12] et la matrice de variances-covariances associée est *nulle*. Les autres informations dont on dispose sont :

- $\hat{\mathbf{x}}_{0|0}$: l'espérance mathématique correspondant à l'état initial du robot,
- $\mathbf{P}_{0|0}$: la matrice de variances-covariances associée à $\hat{\mathbf{x}}_{0|0}$,
- $\mathbf{z}_{t,(i)} = \mathbf{h}\left(\mathbf{x}_t, \mathbf{m}_{(i)}\right) + \mathbf{v}_{t,(i)}$: la fonction d'observation de l'amer (i),

11. Cette matrice se " remplit " très rapidement. On ne peut donc pas appliquer d'optimisation liées à la structure de la matrice (comme cela est le cas pour l'insertion des matrices creuses) pour le calcul des inverses.

12. $\mathbf{x}_{t+1} = \mathbf{x}_t$, où $\mathbf{x}_t$ désigne l'état du robot (position et orientation à à l'instant t).

– $\mathbf{m}_{(i)} = \mathbf{g}\left(\mathbf{x}_t, \mathbf{z}_{t,(i)}, \mathbf{v}_{t,(i)}\right)$: la fonction permettant d'initialiser l'amer (i), [13]

– $\mathbf{R}_{t,(i)}$: la matrice de variances-covariances de $\mathbf{v}_{t,(i)}$.

Etant donné que les observations ne donnent qu'une information relative entre la position des amers et celle du robot, aucune information absolue ne peut être déduite sur la position du robot si il n'y a aucun mouvement. Les observations ne peuvent servir qu'à placer les amers et en aucun cas à modifier l'estimation de la pose du robot. Néanmoins, Julier et Uhlmann démontrent le théorème suivant :

Théorème 1.5 Si *l'état d'un robot mobile est initialisé avec une matrice de variance-covariance non-nulle et que les amers sont initialisés grâce à la fonction* $\mathbf{g}(.)$ *(en appliquant les règles classiques de propagation pour initialiser la matrice de variances-covariances associée)*
Alors *l'estimation de l'état du robot demeurera inchangée* si et seulement si *:*

$$\nabla_{\mathbf{x}}\mathbf{h} + (\nabla_{\mathbf{m}}\mathbf{h})(\nabla_{\mathbf{x}}\mathbf{g}) = 0 \tag{1.67}$$

à chaque instant.

Le théorème 1.5 est démontré en utilisant le fait qu'une partie de l'état est inchangée après la phase de prédiction (qui elle ne change pas l'état) *si et seulement si* le gain de Kalman associé à l'état est nul. [14]

Les auteurs appliquent ce résultat sur un cas pratique 2D. Le robot est paramétré par ses coordonnées cartésiennes et son orientation, exprimés dans un repère absolu et chaque amer est paramétré par ses coordonnées cartésiennes exprimées dans le même repère :

$$\mathbf{x}_t = \begin{bmatrix} x_t & y_t & \theta_t \end{bmatrix}^T \qquad \text{et} \qquad \mathbf{m}_{(i)} = \begin{bmatrix} x_{(i)} & y_{(i)} \end{bmatrix}^T \tag{1.68}$$

Par ailleurs, le modèle de mesure considéré est de type " *range and bearing* " : la distance entre le robot et chaque amer $(r_{t,(i)})$ ainsi que l'orientation relative entre le robot et chaque amer $(\phi_{t,(i)})$ sont disponibles : [15]

$$\mathbf{z}_{t,(i)} = \begin{bmatrix} r_{t,(i)} \\ \phi_{t,(i)} \end{bmatrix} = \begin{bmatrix} \sqrt{(x_{(i)} - x_t)^2 + (y_{(i)} - y_t)^2} \\ \tan^{-1}\left(\frac{y_{(i)} - y_t}{x_{(i)} - x_t}\right) - \theta_t \end{bmatrix} + \underbrace{\begin{bmatrix} v1_{t,(i)} \\ v2_{t,(i)} \end{bmatrix}}_{\mathbf{v}_{t,(i)}} \tag{1.69}$$

La fonction $\mathbf{g}(.)$ permettant d'initialiser la position des amers vaut quant à elle :

$$\mathbf{g}(\mathbf{x}_t, \mathbf{z}_{t,(i)}, \mathbf{v}_{t,(i)}) = \begin{bmatrix} x_{(i)} \\ y_{(i)} \end{bmatrix} = \begin{bmatrix} x_t + (r_{t,(i)} - v1_{t,(i)})\cos(\theta_t + \phi_{t,(i)} - v2_{t,(i)}) \\ y_t + (r_{t,(i)} - v1_{t,(i)})\sin(\theta_t + \phi_{t,(i)} - v2_{t,(i)}) \end{bmatrix} \tag{1.70}$$

13. Dans cette étude, on suppose que l'amer peut être initialisé à l'aide d'une seule mesure.
14. Ceci signifie que l'on a aucune information permettant d'améliorer l'estimation de l'état.
15. Un laser donne ce type d'informations.

Supposons désormais que l'état du robot est resté fixe jusqu'à l'instant t et qu'il réobserve l'amer (i) à l'instant $t+1$. Tous calculs faits, l'équation 1.67 devient :

$$-\begin{bmatrix} 0 & 0 & (\hat{x}_{(i)}-\hat{x}_t)\sin(\hat{\theta}_t+\phi_{t+1,(i)}) - (\hat{y}_{(i)}-\hat{y}_t)\cos(\hat{\theta}_t+\phi_{t+1,(i)}) \\ 0 & 0 & -\frac{(\hat{y}_{(i)}-\hat{y}_t)}{r_{t+1,(i)}}\sin(\hat{\theta}_t+\phi_{t+1,(i)}) - \frac{(\hat{x}_{(i)}-\hat{x}_t)}{r_{t+1,(i)}}\cos(\hat{\theta}_t+\phi_{t+1,(i)}) + 1 \end{bmatrix}^T = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}^T \quad (1.71)$$

Dans l'équation 1.71, $r_{t+1,(i)}$ et $\phi_{t+1,(i)}$ sont les *mesures* effectuées à l'instant $t+1$. Par ailleurs, $\hat{x}_t$,$\hat{y}_t$ et $\hat{\theta}_t$ sont les estimations de l'état du robot à l'instant t (ce qui correspond aussi à la prédiction à l'instant $t+1$). $\hat{x}_{t,(i)}$ et $\hat{y}_{t,(i)}$ sont les dernières estimations de la position des amers (qui correspondent également avec la prédiction de leur position à $t+1$). Ces deux dernières valeurs peuvent se mettre sous la forme suivante :

$$\begin{cases} \hat{x}_{t,(i)} &= \hat{x}_t + \hat{r}_{t,(i)}\cos(\hat{\theta}_t+\hat{\phi}_{t,(i)}) \\ \hat{y}_{t,(i)} &= \hat{y}_t + \hat{r}_{t,(i)}\sin(\hat{\theta}_t+\hat{\phi}_{t,(i)}) \end{cases} \quad (1.72)$$

où $\hat{r}_{t,(i)}$ et $\hat{\phi}_{t,(i)}$ s'obtiennent en appliquant la fonction $\mathbf{h}(.)$ avec les estimations de l'état du robot et de l'amer à l'instant t.

En combinant les équations 1.71 et 1.72, on obtient la condition suivante :

$$\begin{bmatrix} 0 & 0 & \cos(\hat{\theta}_t+\hat{\phi}_{t,(i)})\sin(\hat{\theta}_t+\phi_{t+1,(i)}) - \sin(\hat{\theta}_t+\hat{\phi}_{t,(i)})\cos(\hat{\theta}_t+\phi_{t+1,(i)}) \\ 0 & 0 & 1+\frac{\hat{r}_{t,(i)}}{r_{t,(i)}}\Big(\sin(\hat{\theta}_t+\hat{\phi}_{t,(i)})\sin(\hat{\theta}_t+\phi_{t+1,(i)}) - \cos(\hat{\theta}_t+\hat{\phi}_{t,(i)})\cos(\hat{\theta}_t+\phi_{t+1,(i)})\Big) \end{bmatrix}^T = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}^T \quad (1.73)$$

Dans le premier membre de l'équation 1.73, on remarque que :

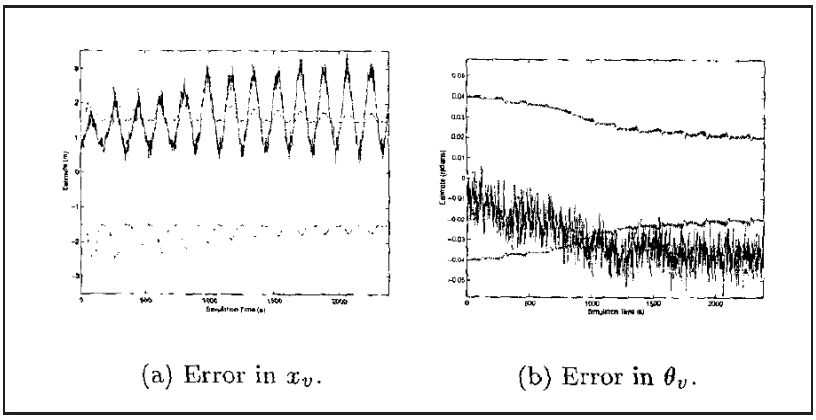
- le bloc de gauche est toujours égal à zéro. [16] Ceci indique que les deux premières composantes de l'état du robot ne sont pas changées. Ceci correspond à la position du robot,
- il y a deux composantes non nulles (en général) sur la troisième colonne. Il y a donc mise à jour de l'orientation du robot dès que l'innovation est non nulle. Il faudrait donc en principe que les deux termes soient nuls pour que l'estimation de l'orientation demeure inchangée.

  Dans un premier temps, il faut que $\hat{\theta}_t+\hat{\phi}_{t,(i)} = \hat{\theta}_t+\phi_{t+1,(i)}$, soit $\hat{\phi}_{t,(i)} = \phi_{t+1,(i)}$ (composante " en haut à droite ") : la direction mesurée de l'amer doit correspondre avec la direction estimée courante. Autrement dit, la seconde composante du vecteur d'innovation courant [17] doit être nulle.

  Si la condition précédente est vérifiée, il n'est donc pas nécessaire d'annuler le dernier terme de

16. Ceci provient du fait que la matrice jacobienne de $\mathbf{h}$ par rapport à la position du robot est l'opposée de la matrice jacobienne de $\mathbf{h}$ par rapport à la position de l'amer.

17. Etant donné que la prédiction de $\phi_{t+1,(i)}$ est égale à $\hat{\phi}_{t,(i)}$, le terme $\phi_{t+1,(i)} - \hat{\phi}_{t,(i)}$ correspond bien à la seconde composante du vecteur d'innovation à l'instant $t+1$.



(a) Error in x_v. (b) Error in θ_v.

FIGURE 1.4 – Erreur de l'EKF sur 3200s (source : [Julier et Uhlmann, 2001])

la matrice de poids étant donné qu'il est multiplié par 0. [18]

En conséquence, on remarque que l'estimation de l'orientation du robot reste inchangée *si et seulement si* la nouvelle mesure angulaire de l'amer concorde parfaitement avec la direction de la dernière position estimée de l'amer. Récursivement, on en déduit facilement que **la mesure d'angle doit constamment être égale à la mesure angulaire initiale** (ce qui n'est pas respecté dès que la sortie du capteur est bruitée). On constate alors une mise à jour de l'orientation du robot ainsi qu'une diminution de la variance correspondante qui n'ont pas lieu d'être étant donné que le robot ne mesure d'aucune façon son orientation absolue. Ce problème est décrit dans [Julier et Uhlmann, 2001] comme une inconsistance due à la *structure même* de l'EKF-SLAM. Il ne s'agit en rien d'un problème numérique qui peut être résolu par des méthodes de type " inflation des matrices de variances-covariances " [19]. Nous avons pu vérifier par nous-même ce phénomène à l'aide d'une implémentation `Matlab`.

Enfin, d'autres inconsistances sont mises en évidence dans cette publication, mais de manière expérimentale. Un test est effectué pour observer le comportement du filtre par rapport à un changement de position connu dans le modèle. Le résultat est inconsistant. Il est ensuite montré (toujours en simulation) que pour le cas d'un véhicule en mouvement suivant une trajectoire simple (cercle), le filtre est consistant au début, mais devient inconsistant sur le très long terme (figure 1.4).

18. On pourra remarquer que si on a la première composante de l'innovation qui est nulle, ie $\hat{r}_{t,(i)} = r_{t,(i)}$, alors il est quand même nécessaire d'avoir $\hat{\phi}_{t,(i)} = \phi_{t+1,(i)}$ pour annuler la " seconde source " de mise à jour.

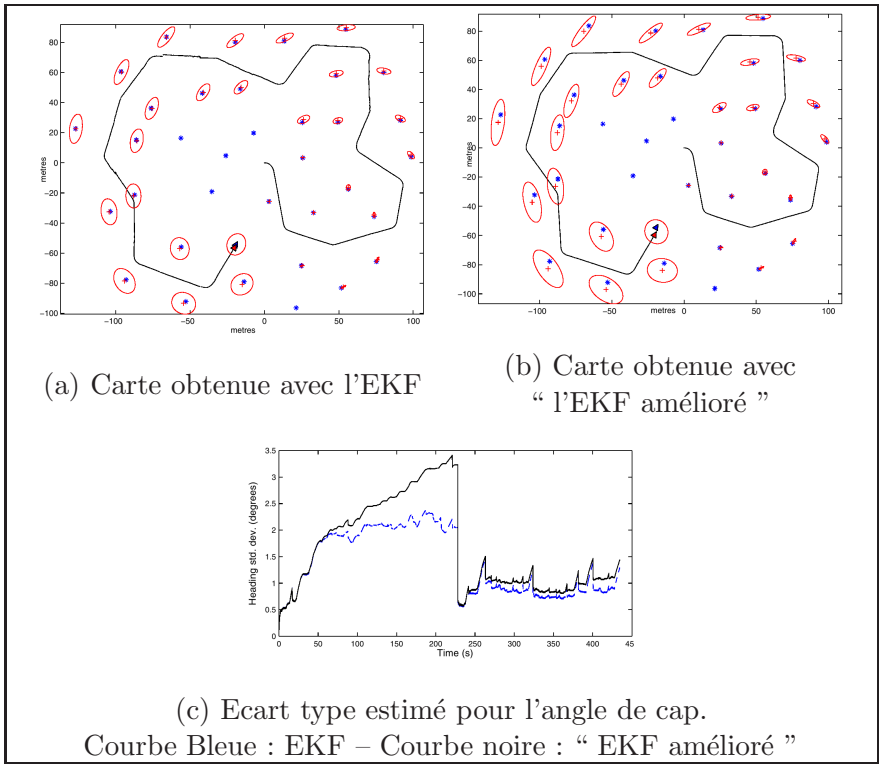
19. Il s'agit d'augmenter les valeurs des covariances dans les matrices $\mathbf{R}_t$ et $\mathbf{Q}_t$ de sorte à augmenter la variance du résultat final.

1.4.3.2 Etude empirique de la consistance de l'EKF-SLAM [Bailey *et al.*, 2006a]

Dans [Bailey *et al.*, 2006a], les auteurs exposent le problème de la consistance de l'EKF-SLAM à travers plusieurs simulations. Ils donnent deux symptomes d'inconsistance de l'algorithme :

1. Un gain d'information trop important : le filtre " croit posséder plus d'information que ce qu'il n'a vraiment " pour faire les mises à jour. Cet aspect est traité en détail dans la suite de ce paragraphe.

2. Des sauts importants dans les estimations en comparaison avec les valeurs du vecteur d'innovation. Ce phénomène est peu abordé et non expliqué dans l'article, nous ne le détaillons pas dans la suite.



(a) Carte obtenue avec l'EKF

(b) Carte obtenue avec " l'EKF amélioré "

(c) Ecart type estimé pour l'angle de cap.
Courbe Bleue : EKF – Courbe noire : " EKF amélioré "

FIGURE 1.5 – Résultats qualitatifs de consistance ([Bailey *et al.*, 2006a]) – On peut remarquer que les ellipses de confiance associées aux covariances de l'EKF sont un peu plus petites que celles de l'EKF amélioré (comparaisons de figures (a) et (b)) alors qu'on utilise strictement le même jeu de données. La figure (c) montre plus clairement l'inconsistance de l'EKF : l'incertitude associée à l'orientation se stabilise alors qu'on est encore en phase exploratoire. Ce problème n'est pas visible pour l'EKF amélioré.

Nous présentons les aspects " gain d'information " liés à la consistance pour un véhicule en mouvement. Les auteurs comparent l'EKF-SLAM classique et un " EKF-SLAM amélioré " dans lequel les matrices jacobiennes sont évaluées avec les vraies valeurs de l'état et de la commande. Cette évaluation est effectuée en simulation. [20] Dans les deux cas, on ne peut pas trouver les vraies statistiques de la densité de probabilité (à cause de l'approximation linéaire). Néanmoins, deux remarques qualitatives importantes sont formulées (cf. figure 1.5) :

1. les ellipses de confiance fournies par l'EKF sont plus petites que celles fournies par " l'EKF-SLAM amélioré ".
2. lorsque le robot évolue dans une zone non cartographiée (jusqu'à 220s sur la figure 1.5c), la variance de l'état du robot est censée augmenter. [21] Pourtant, dans le cas de l'EKF classique, la variance estimée sur l'orientation s'arrête d'augmenter à partir d'un certain seuil. Ceci signifierait que le filtre possède de l'information pour améliorer le cap (alors qu'il n'en a pas). " L'EKF-SLAM amélioré ", quant à lui, ne possède pas ce problème. Les auteurs de [Bailey *et al.*, 2006a] font remarquer que si on ajoute une mesure directe du cap absolu dans le vecteur $\mathbf{z}$ (de telle sorte à maintenir l'incertitude sur le cap toujours petite), alors les résultats des deux filtres sont presque identiques. Cette propriété semble indépendante des valeurs de bruit de mesure et de modèle.

Ces deux remarques **laissent penser** que l'EKF-SLAM est inconsistant, et ce dès que la variance sur l'orientation du robot est trop grande.

Une étude numérique plus poussée est ensuite proposée dans [Bailey *et al.*, 2006a]. L'EKF-SLAM sera dit consistant si à chaque instant, la différence entre la matrice de variances-covariances de l'estimation et la matrice de variances-covariances réelle est semi-définie positive :

$$\mathbf{P}_{t|t} - \mathbf{P}_t \geq 0 \tag{1.74}$$

où $\mathbf{P}_t$ désigne la variance réelle associée à $p\left(\mathbf{x}_t, \mathbf{m} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right)$.

Il est en général impossible de connaître la vraie matrice de variances-covariances, même dans le cas de simulations. Par contre, si la vraie valeur de l'état est connue, on peut calculer l'*erreur quadratique d'estimation normalisée* (NEES [22]) à chaque instant :

$$\epsilon_t = \left(\mathbf{x}_t - \hat{\mathbf{x}}_{t|t}\right)^T \mathbf{P}_{t|t}^{-1} \left(\mathbf{x}_t - \hat{\mathbf{x}}_{t|t}\right) \tag{1.75}$$

20. Le modèle de mesure utilisé est le même que précédemment. Par ailleurs, le modèle d'état est le modèle classique du tricycle commandé par l'angle de braquage et la vitesse des roues arrière.
21. Seule l'observation d'anciens amers peut permettre de faire diminuer ou stagner la variance de l'état.
22. *Normalised estimation error squared* en anglais.

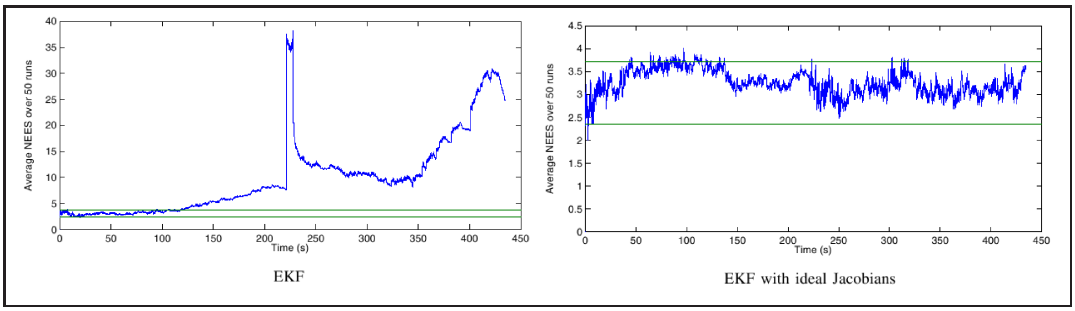


FIGURE 1.6 – NESS moyen concernant l'état du robot ([Bailey *et al.*, 2006a]) – Les bandes vertes indiquent la zone dans laquelle 95% des valeurs doivent être comprises pour que le filtre soit consistant. Le graphique de gauche fait bien apparaître le caractère *trop optimiste* de l'EKF. En revanche, l'EKF amélioré semble parfaitement consistant : ni trop optimiste, ni trop pessimiste.

Sous hypothèse gaussienne sur l'état, ϵ_t suit une loi du χ^2 à $\dim(\mathbf{x}_t)$ degrés de liberté. La consistance de l'EKF-SLAM sera finalement déduite en effectuant N tirages de Monte-Carlo (pour les bruits) sur les mêmes données parfaites de base, et en calculant pour tout t :

$$\bar{\epsilon}_t = \frac{1}{N}\sum_{i=1}^{N} \epsilon_{i,t} \tag{1.76}$$

On peut définir un intervalle qui délimite la région à 95% de probabilité pour $\bar{\epsilon}_t$ si le filtre est consistant. Ainsi, si les valeurs de ϵ_t trouvées sont trop petites par rapport à la borne inférieure de l'intervalle, le filtre sera pessimiste. Si au contraire, ϵ_t est souvent supérieur à la borne supérieure de l'intervalle défini, alors le filtre sera trop optimiste et donc *inconsistant*.

Dans [Bailey *et al.*, 2006a], ce critère est utilisé pour comparer l'EKF-SLAM avec d'autres variantes (dont " l'EKF amélioré "). La figure 1.6 présente les résultats obtenus avec la même simulation que précédemment et $N = 50$; la zone à 95% de probabilité pour $\bar{\epsilon}_t$ y est représentée en vert. On peut voir que l'EKF-SLAM devient inconsistant vers 100s, et ce lorsque l'incertitude concernant l'angle de cap devient beaucoup plus petite que celle de " l'EKF amélioré ". Parallèlement, on constate que l'EKF amélioré est **toujours consistant**. En conséquence, on désignera ce filtre par *EKF idéal* dans la suite de ce manuscrit.

Finalement, les auteurs de [Bailey *et al.*, 2006a] font remarquer que lorsqu'on utilise une mesure explicite du cap (ce qui permet de maintenir la variance du cap à une valeur " faible "), l'EKF-SLAM est consistant.

1.4.3.3 Observabilité associée au système défini par l'EKF ([Huang *et al.*, 2008])

Nous présentons enfin une étude plus récente des problèmes de consistance du filtre de Kalman étendu. Celle-ci est basée sur l'étude du rang de la matrice d'observabilité. Dans un premier temps, les auteurs de [Huang *et al.*, 2008] rappellent les principaux résultats d'observabilité du SLAM. Ensuite, ils présentent les propriétés du système associé au filtre de Kalman étendu idéal. Les propriétés du filtre de Kalman étendu classique sont ensuite présentées, concluant à une inconsistance du filtre de l'EKF. Enfin, une modification de l'algorithme original est proposée.

Les hypothèses utilisées dans [Huang *et al.*, 2008] sont :

– Le robot se déplace dans le plan et a trois degrés de liberté (deux en translation et un en rotation). On a :

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

– Le robot n'observe qu'un seul amer ponctuel paramétré par ses coordonnées cartésiennes 2D :

$$\mathbf{m} = \begin{bmatrix} x_A & y_A \end{bmatrix}^T$$

1.4.3.3.1 Observabilité du SLAM

Nous avons vu dans la section 1.3 que le problème du SLAM est inobservable. Dans [Huang *et al.*, 2008], les auteurs utilisent ce résultat en précisant que quel que soit l'ordre utilisé pour la dérivée de Lie, la dimension du noyau de la matrice d'observabilité est toujours égale à 3. [23] Une base de ce noyau est donnée par :

$$\mathbf{n_1} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \qquad \mathbf{n_2} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \qquad \text{et} \qquad \mathbf{n_3} = \begin{bmatrix} -y_t \\ x_t \\ 1 \\ -y_A \\ x_A \end{bmatrix} \tag{1.77}$$

Les vecteurs $\mathbf{n_1}$ et $\mathbf{n_2}$ définis dans l'équation 1.77 indiquent que l'on peut ajouter n'importe quelle constante aux coordonnées x et y du robot et de l'amer. Ceci traduit le fait qu'il existe une **translation globale inobservable**. Le vecteur $\mathbf{n_3}$ est plus complexe à interpréter. Dans [Huang *et al.*, 2008], on montre qu'une petite rotation de $\delta\theta$ de l'ensemble du vecteur d'état introduit un décalage de $\delta\theta\mathbf{n_3}$ après linéarisation au premier ordre. Ainsi, la présence du vecteur $\mathbf{n_3}$ dans le noyau de la matrice d'observabilité traduit l'existence d'une rotation globale inobservable.

En conséquence, les auteurs de [Huang *et al.*, 2008] montrent que dans le cas présent, le SLAM est **inobservable avec un " rang d'inobservabilité " égal à 3**. De plus, on retrouve dans le noyau

23. Sauf configurations particulières dégénérées qui ne sont pas traitées.

de la matrice d'observabilité la notion intuitive (introduite dans la section 1.3) qui nous dit que pour un ensemble de mesures données, on a une infinité de solutions possibles qui ne diffèrent que par une **transformation rigide à 3 degrés de liberté** (deux degrés de liberté en translation et un degré de liberté en rotation).

1.4.3.3.2 Observabilité associée à l'EKF-SLAM

Après avoir montré les propriétés d'observabilité associées au SLAM, les auteurs de [Huang *et al.*, 2008] investiguent les propriétés associées au système discret défini par le filtre de Kalman étendu. Pour cela, ils calculent la matrice d'observabilité associée au système linéarisé (tel que définie dans [Hermann et Krener, 1977]) pour trois cas :

1. **L'EKF idéal** : les points de linéarisation utilisés sont les véritables valeurs des états. On peut montrer que la matrice d'observabilité associée à ce filtre a un noyau de dimension 3 dont les vecteurs $\mathbf{n_1}$, $\mathbf{n_2}$ et $\mathbf{n_3}$ forment une base. On retrouve bien le même propriétés que pour le système générique de SLAM.

2. **L'EKF classique** : on respecte point par point l'algorithme présenté au paragraphe 1.4. Dans ce cas, la matrice d'observabilité associée au filtre a un noyau de dimension 2 engendré par les vecteurs $\mathbf{n_1}$ et $\mathbf{n_2}$. Les erreurs dues à l'introduction de points de linéarisation non parfaits ont **changé les propriétés d'observabilité** du SLAM. Ainsi, le vecteur $\mathbf{n_3}$ (correspondant à la rotation générale du plan) ne fait plus partie de l'espace inobservable.

3. **Un filtre de Kalman modifié : le *First Estimates Jacobian EKF (FEJ-EKF)***. Il s'agit d'une version modifiée de l'EKF dans laquelle les points de linéarisation associés aux amers ne sont jamais modifiés. Lorsqu'un amer est initialisé, le point de fonctionnement pour le calcul de la matrice jacobienne associée à la mesure n'est pas affiné au fur et à mesure des estimations (contrairement à l'EKF classique). Dans ce cas, le noyau de la matrice d'observabilité associée est de dimension 3 et on retrouve exactement l'espace engendré par les vecteurs $\mathbf{n_1}$, $\mathbf{n_2}$ et $\mathbf{n_3}$.

Les auteurs de [Huang *et al.*, 2008] illustrent leur propos en simulation, en effectuant une série de tirages de Monte-Carlo et en calculant le critère NESS ainsi que l'erreur quadratique moyenne. Ils montrent que le FEJ-EKF donne pratiquement les mêmes performances que le filtre de Kalman étendu idéal, alors que le filtre de Kalman étendu classique donne de trop petites variances.

1.4.3.4 Conclusion et remarque quant à la consistance

Dans ce paragraphe, nous avons montré quelques résultats de consistance concernant l'EKF-SLAM. Bien que ceux-ci aient été obtenus dans le cas du SLAM *range and bearing*, nous admettons qu'ils puissent être étendus au cas du SLAM visuel. Ces résultats montrent que l'algorithme est inconsistant : **les enveloppes d'incertitudes estimées par le filtre sont en général trop optimistes et ne donnent donc pas une bonne approximation de l'amplitude de l'erreur de filtrage.**[24] D'après les études de [Julier et Uhlmann, 2001] et [Bailey *et al.*, 2006a], il semblait que le problème d'inconsistance se manifestait surtout au niveau du gain d'information estimé quant à l'orientation du robot. Ceci a été confirmé par l'étude théorique de [Huang *et al.*, 2008] qui indique que le filtre de Kalman étendu rend artificiellement observable l'espace engendré par le vecteur $\mathbf{n_3}$. Cet espace permet d'interpréter au premier ordre de petites rotations globales. Ainsi, le filtre de Kalman étendu " croit " bénéficier d'une information qui n'existe pas, ce qui rend le filtre inconsistant.

Le premier moyen trouvé pour annuler cette inconsistance réside dans l'utilisation de l'EKF idéal, ce qui est impossible en pratique. Un résultat plus intéressant est donné dans [Huang *et al.*, 2008] avec l'utilisation du FEJ-EKF. Ce filtre maintient constant les points de fonctionnement associés aux amers dans le calcul des matrices jacobiennes. Ainsi, les auteurs de cette publication montrent que la consistance n'est pas directement liée au fait d'utiliser les points exacts de linéarisation, mais liée au fait de **maintenir l'ensemble des points de linéarisation associés aux amers constant**. Conserver cette propriété semble donc plus efficace que d'affiner les points de linéarisation pour les rendre plus proches de la solution idéale.

Enfin, on peut noter que ce problème d'inconsistance est un problème structurel lié à la définition du SLAM même. L'étude d'observabilité ne prend pas en compte le niveau de bruit des capteurs. Ainsi, les méthodes d'inflation de covariances utilisées dans quelques algorithmes ([Julier, 2003]) sont inutiles. Elles permettront au mieux de masquer le problème de consistance pendant un nombre limité d'itérations. Par ailleurs, cette étude ne prend pas en compte la qualité du point de fonctionnement et donc de l'estimation non-linéaire. Ainsi, le problème d'inconsistance continue à être présent même avec une initialisation de très bonne qualité et des estimations très proches de la valeur réelle.

Remarque 1.4 (Utilisation d'un capteur d'orientation)

En plus de montrer les problèmes d'inconsistance de l'EKF-SLAM, cette étude illustre l'intérêt d'un

24. Les zones de confiances sont en général des lieux où la probabilité que la solution soit contenue dedans est de $x\%$. Dans le cas de l'EKF, le critère NESS calculé montre que ce $x\%$ n'est en général pas respecté.

capteur absolu d'orientation (magnétomètre par exemple). En effet, les résultats présentés dans [Bailey et al.*, 2006a] montrent que l'inconsistance de l'EKF est constatée en pratique lorsque l'incertitude sur l'orientation du robot franchit un certain seuil. Ainsi, la présence d'un capteur donnant une orientation absolue permet de contenir l'incertitude de l'orientation du robot. Par ailleurs, la présence d'un tel capteur rend directement observable l'orientation du robot. Ainsi, le vecteur* $\mathbf{n_3}$ *disparaitraît du noyau de la matrice d'observabilité générale du SLAM (seule la translation globale resterait inobservable). En conséquence, le filtre de Kalman étendu ne présenterait plus d'incohérence sur ses propriétés d'observabilité. Ceci peut expliquer les bons résultats obtenus dans [Bailey* et al.*, 2006a] lorsque les auteurs utilisent un EKF classique dans lequel le vecteur de mesure est augmenté d'une mesure absolue de l'orientation du robot. Néanmoins, nous cherchons dans cette thèse à résoudre le problème du SLAM sans utiliser de capteurs absolus (dont la qualité peut dépendre de l'environnement) ; nous n'avons donc pas implanté d'algorithme utilisant un capteur d'orientation absolu, mais les propriétés théoriques que l'on peut obtenir grâce à ce type de capteur semble être une piste intéressante de réflexion future.*

1.5 Variantes du filtre de Kalman étendu

Nous présentons dans cette section une liste non exhaustive de variantes de l'EKF-SLAM. Il s'agit en général d'ajouter de petites différences dans le but d'améliorer les performances du filtre de Kalman étendu.

1.5.1 Filtre d'information tronqué

Une première variante du filtre de Kalman étendu consiste à simplifier la matrice de variances-covariances de sorte à alléger les coûts de calculs liés à son inversion. Cette matrice peut en effet atteindre une dimension élevée et a pour propriété d'être pleine en général. Pour cela, certains auteurs travaillent directement dans l'espace d'information et profitent du fait que la matrice d'information (ie. l'inverse de la matrice de variances-covariances) possède de nombreux éléments proches de zéro. L'idée est alors d'annuler ces coefficients, ce qui a pour effet de rendre la matrice éparse et beaucoup plus facile à inverser.

Ce type d'approche peut être trouvé dans [Thrun *et al.*, 2004] et ne constitue qu'une approximation de l'EKF. Cela conduit à supprimer certaines corrélations entre amers. Ce filtre délivre des résultats moins précis et moins consistants que l'EKF classique et n'est donc pas approfondi.

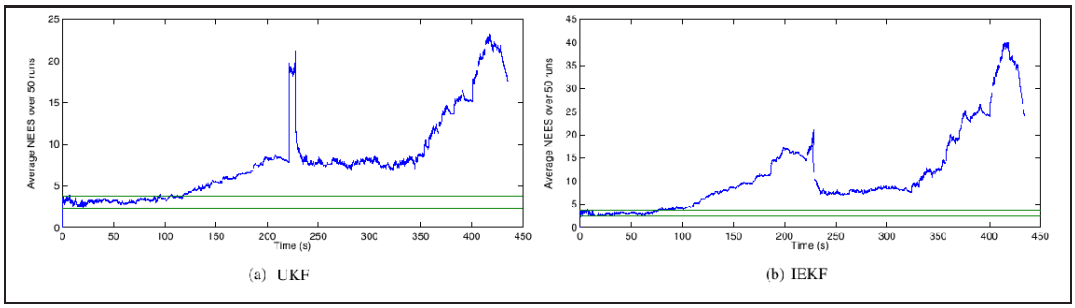


FIGURE 1.7 – NESS moyen concernant l'état du robot ([Bailey *et al.*, 2006a]) pour l'UKF et l'IEKF

1.5.2 Filre de Kalman sans parfum et EKF itératif

Le filtre de Kalman sans parfum (*Unscented Kalman Filter* ou UKF) et l'EKF itératif (*Iterated Extended Kalman Filter* ou IEKF) sont deux alternatives classiques au filtre de Kalman étendu.

Le principe de l'UKF est d'effectuer un **échantillonnage déterministe** du vecteur d'état. Celui-ci est réalisé en utilisant les différentes colonnes de la matrice de Cholesky. Selon Simon Julier, l'UKF est équivalent à un développement limité à l'ordre 2 alors que le filtre de Kalman étendu n'effectue qu'un développement limité à l'ordre 1 ([Julier et Uhlmann, 1997]). Cette variante donne en général de meilleurs résultats que l'EKF classique.

Le principe de l'IEKF, quant à lui est d'itérer à plusieurs reprises l'EKF à chaque instant dans le but d'optimiser la qualité des points de linéarisation. Ainsi, les équations de la phase de correction sont appliquées plusieurs fois jusqu'à convergence du résultat. A la première itération, le point de linéarisation utilisé est l'état obtenu après la prédiction. L'application de la phase classique de correction nous fait donc retrouver le résultat de l'EKF à la fin de la première itération. Ensuite, ce résultat est utilisé à nouveau pour affiner le calcul de la matrice jacobienne de la fonction $\mathbf{h}$ et ainsi de suite. On peut dès lors remarquer que le point de linéarisation utilisé pour les amers sera différent à chaque instant. Ainsi, on s'attend à trouver les même problèmes d'observabilité que ceux décrits dans [Huang *et al.*, 2008].

En pratique, ces deux filtres permettent dans certains cas d'améliorer les résultats obtenus par l'EKF. Néanmoins, il a été montré dans [Bailey *et al.*, 2006a] que ces deux filtres restent largement inconsistants. Le calcul du NESS effectué montre une très légère amélioration de la consistance dans le cas de l'UKF, mais on reste loin de la bande idéale que le filtre de Kalman idéal respecte (cf. figure 1.7). En conséquence, nous avons décidé de ne pas approfondir plus en détail ces méthodes.

1.5.3 FEJ-EKF

Nous avons présenté dans la section précédente le FEJ-EKF introduit dans [Huang *et al.*, 2008]. Ce filtre permet d'obtenir de bonnes propriétés de consistance grâce au fait qu'il maintient de bonnes propriétés d'observabilité. Néanmoins, le fait de ne jamais ajuster les points de fonctionnement peut s'avérer dangereux. En effet, le filtre ne sera valable que si les contraintes de linéarisation sont satisfaites. Même s'il n'est pas nécessaire que le point de fonctionnement soit " extrêmement " proche de la valeur réelle, il faut pouvoir assurer que la linéarisation reste valide (dans le cas contraire, les équations deviennent fausses et le système modélisé ne correspondra pas à la réalité). Dans les simulations de [Huang *et al.*, 2008], un capteur de type *range and bearing* est modélisé : les amers peuvent être initialisés précisément dès la première itération. Dans le cas du SLAM monoculaire, nous verrons que l'initialisation est plus délicate. Il paraît alors indispensable de pouvoir corriger les points de linéarisation si l'on se rend compte que ceux-ci sont mauvais. Ceci sera encore plus vrai lors de l'utilisation de données réelles. Malgré ses bonnes propriétés théoriques, ce filtre paraît trop sensible aux conditions d'initialisation pour être utilisé dans notre cas.

1.6 Résolution par filtre particulaire : le FastSLAM

Nous présentons dans cette section la résolution du problème du SLAM probabiliste avec un filtre particulaire Rao-Blackwellisé. Comme pour la présentation de l'EKF-SLAM, nous présentons dans un premier temps le principe général et les équations. Ensuite, nous listons les principaux avantages et inconvénients de cette méthode, avant de donner des résultats de convergence et de consistance.

Les résultats qui suivent sont issus de la thèse de Montemerlo ([Montemerlo, 2003]). Ils peuvent être également trouvés de façon plus concise dans les publications [Montemerlo *et al.*, 2002, 2003].

1.6.1 Principe

1.6.1.1 Introduction

Nous avons vu précédemment que l'EKF-SLAM a une complexité algorithmique importante du fait qu'elle croît avec le carré du nombre d'amers M. En 2003, ce poids calculatoire était considéré critique[25] ; il était alors très intéressant de développer un algorithme de SLAM pouvant traiter de nombreux amers et de grandes trajectoires. L'algorithme de FastSLAM a été introduit dans cette optique et utilise une implémentation astucieuse de l'algorithme de *filtrage particulaire*.

25. Bien plus que les problèmes de consistance et d'observabilité qui n'étaient que peu abordés.

L'algorithme de filtrage particulaire classique consiste à effectuer un échantillonnage aléatoire de *tout l'état* (dans le cas du SLAM, il s'agit de l'état du robot et des amers). Chaque échantillon est tiré selon une densité de probabilité choisie à l'avance,[26] une étape de pondération utilisant le vecteur de mesure permet ensuite de donner plus d'importance aux échantillons (nommés par la suite *particules*) les plus probables. Une description détaillée concernant le fonctionnement général des filtres particulaires pourra être trouvée dans [Doucet *et al.*, 2004; Joly *et al.*, 2009].

L'algorithme de filtrage particulaire classique n'est pas utilisable dans le cas du SLAM. En effet, cet algorithme impose d'échantillonner selon toute la dimension de l'état (à savoir la position du robot et les positions des amers). Le nombre de particules nécessaires pour représenter convenablement l'espace des solutions devient trop important : les temps de calculs et l'espace mémoire nécessaires seraient rédhibitoires.

L'algorithme de FastSLAM est basé sur une factorisation du problème, qui permet ensuite d'utiliser une variante du filtrage particulaire : le *filtre particulaire Rao-Blackwellisé*. Ce filtre est utilisé lorsque la densité de probabilité d'une partie de l'état peut être connue analytiquement conditionnellement au reste de l'état. Dans ce cas, l'échantillonnage particulaire n'intervient que sur la seconde partie de l'état ([Doucet *et al.*, 2000]). Dans le cas du FastSLAM, l'idée sera de n'échantillonner que la position du robot.

1.6.1.2 Factorisation du problème de SLAM

Dans sa thèse, Montemerlo ne cherche pas à évaluer la densité de probabilité définie dans l'équation 1.52 page 27, mais $p\left(\mathbf{x}_{0:t}, \mathbf{m} | \mathbf{z}_{0:t}, \mathbf{u}_{0:t}\right)$. Ceci revient à raisonner en termes de trajectoire ; cela est classique avec les filtres particulaires.

On peut montrer que la structure du SLAM permet de factoriser la densité précédente de la manière suivante :

$$p\left(\mathbf{x}_{0:t}, \mathbf{m} | \mathbf{z}_{0:t}, \mathbf{u}_{0:t}\right) = \underbrace{p\left(\mathbf{x}_{0:t} | \mathbf{z}_{0:t}, \mathbf{u}_{0:t}\right)}_{post.\ trajectoire} \underbrace{\prod_{i=1}^{M} p\left(\mathbf{m}_{(i)} | \mathbf{x}_{0:t}, \mathbf{z}_{0:t}, \mathbf{u}_{0:t}\right)}_{estimateur\ amers} \tag{1.78}$$

La factorisation présentée dans l'équation 1.78 montre que conditionnellement à la **trajectoire** du robot, chaque amer est indépendant des autres. Cette propriété est illustrée sur la figure 1.2 (page 27). Ceci est la base de l'algorithme de FastSLAM.

26. Qu'il convient de bien choisir... le choix de cette loi étant un facteur déterminant quant à la qualité du résultat.

1.6.1.3 FastSLAM

Le principe du FastSLAM est de tirer parti de la factorisation précédente dans l'utilisation d'un filtre particulaire à N particules.

Lorsque la densité d'une partie de l'état peut se déduire analytiquement de la densité de la seconde partie de l'état (par un filtre de Kalman linéaire dans le cas idéal), on peut limiter l'échantillonnage d'importance à la seconde partie de l'état et tirer partie de la relation existante pour estimer la première partie de l'état. Il s'agit de la Rao-Blackwellisation ([CASELLA et ROBERT, 1996; Doucet *et al.*, 2000]).

Dans le cas du SLAM, cette propriété est utilisée : les densités de probabilités des amers sont obtenues conditionnellement à la trajectoire du robot [27] (et ce de manière indépendante étant donné le caractère multiplicatif dans l'équation 1.78). En conséquence, *l'échantillonnage particulaire ne porte que sur l'état du robot.* Ensuite, un filtre de Kalman étendu permet d'obtenir une estimation des amers pour chaque particule. [28]

Plus intuitivement, le principe du FastSLAM est de générer stochastiquement N hypothèses [29] concernant la trajectoire du robot. Pour chacune de ces hypothèse, on applique un filtre de Kalman par amer en utilisant le vecteur de mesures. Chaque hypothèse est enfin pondérée en fonction de sa vraisemblance avec les mesures.

1.6.1.4 Stratégies d'échantillonnage

Nous présentons ici deux algorithmes de FastSLAM. Ils diffèrent par leur stratégie d'échantillonnage (méthode pour générer la trajectoire) :

FastSLAM 1.0 : le principe est d'échantillonner en utilisant la chaîne de Markov $p\left(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t\right)$. Lorsque le bruit associé à la chaîne de Markov est gaussien et additif, on a $p\left(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t\right) = \mathcal{N}\left(\mathbf{f}\left(\mathbf{x}_{t-1}, \mathbf{u}_t\right), \mathbf{Q}_t\right)$. Dans ce cas, pour prolonger une particule donnée, il suffit d'appliquer la fonction $\mathbf{f}$ (avec comme arguments la dernière valeur de la particule et le vecteur d'entrées) puis d'ajouter un échantillon généré selon $\mathcal{N}\left(\mathbf{0}, \mathbf{Q}_t\right)$.

Le calcul des poids (non normalisés) est aisé avec ce type de filtre. Pour la particule $[m]$, on a :

$$w_t^{[m]} \propto w_{t-1}^{[m]} p\left(\mathbf{z}_t|\mathbf{x}_{0:t}^{[m]}, \mathbf{z}_{0:t-1}, \mathbf{u}_{0:t}\right)$$

27. Ceci revient à considérer que la trajectoire est certaine au moment de l'estimation de la position des amers.
28. Ici, la résolution n'est pas exacte étant donné que l'on utilise un EKF.
29. La façon de générer les hypothèses est décrite dans le paragraphe suivant.

Cette stratégie n'utilise donc pas le vecteur de mesures pour échantillonner. Ceci peut être problématique si le modèle est ponctuellement mauvais (glissement non modélisé par exemple) ; dans ce cas, les particules seront toutes biaisées et le vecteur de mesure ne servira qu'à donner un poids important à la *moins mauvaise* particule.

FastSLAM 2.0 : le principe est d'échantillonner en utilisant le vecteur de mesures, ce qui permet d'améliorer la robustesse à des erreurs de modèle. Pour étendre la particule $[m]$, on va appliquer un filtre de Kalman de la façon suivante :

- étendre $\mathbf{x}_{t-1}^{[m]}$ avec le modèle (fonction $\mathbf{f}$). Le résultat obtenu est une prédiction $\hat{\mathbf{x}}_t^{[m]}$ associée à la matrice de variances-covariances de modèle $\mathbf{Q}_t$.[30]
- appliquer une étape de correction de Kalman avec le vecteur de mesures. Le résultat obtenu est une estimation de l'état avec une matrice de variances-covariances.
- effectuer un tirage aléatoire gaussien avec pour paramètre le résultat de l'étape précédente.

Le calcul des poids est légèrement plus compliqué que pour le FastSLAM 1.0. On pourra se referer à [Montemerlo, 2003] pp. 82 et 83 pour plus de détails.

Cette stratégie d'échantillonnage améliore la qualité des échantillons. En conséquence, le nombre de particules nécessaires pour représenter correctement la trajectoire du robot est plus faible que pour le FastSLAM 1.0.

Remarque 1.5

L'étape de Kalman nécessite des inversions de matrices. Dans le pire des cas, on peut observer simultanément tous les amers. Néanmoins, la matrice de variances-covariances associée aux amers et à la prédiction de la position est diagonale par blocs, la complexité est donc linéaire avec le nombre d'amers et non quadratique.

1.6.2 Avantages et limites pratiques

Le FastSLAM possède plusieurs avantages importants :

- l'approximation linéaire sur le modèle de mouvement est évitée,
- la complexité algorithmique est plus faible que pour l'EKF-SLAM. Dans le cas du fastSLAM 1.0 par exemple, on a un EKF à appliquer par particule et par amer. Etant donné que chaque filtre de Kalman a une dimension fixe, la complexité est au pire en $\mathcal{O}(NM)$. Ensuite, des améliorations algorithmiques permettent de la faire descendre en $\mathcal{O}(N\log(M))$,

30. Une particule seule a toujours une variance nulle. Ceci explique pourquoi seulement $\mathbf{Q}_t$ intervient dans la matrice de variances-covariances.

– il est possible de résoudre explicitement le problème d'associations des données durant le filtrage. Nous n'avons pas mentionné cet aspect dans le paragraphe précédent. Le lecteur intéressé se réfèrera à [Montemerlo, 2003] pour plus de détails. Rappelons tout de même que ceci n'est pas possible avec l'EKF-SLAM.

Par ailleurs, il existe un résultat de convergence pour le FastSLAM 2.0 :

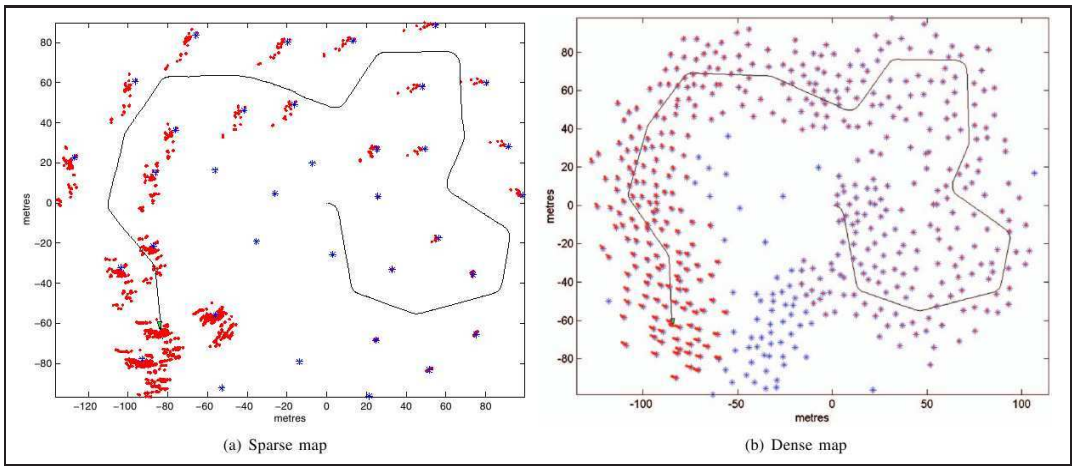
Théorème 1.6 *L'algorithme linéaire gaussien de FastSLAM converge vers la vraie carte avec $M = 1$ particule si tous les amers sont observés une infinité de fois et que la localisation d'un d'entre eux est connue à l'avance.*
Si aucun amer n'est connu à l'avance, la carte sera correcte à une constante près appliquée uniformément sur tous les amers.

La démonstration de ce théorème peut être trouvée dans [Montemerlo, 2003] pp 88–92. Ce résultat montre qu'il est possible de converger vers la bonne carte *sans maintenir les corrélations entre les amers*. Néanmoins, l'intérêt pratique de ce résultat est faible :

– la démonstration est faite pour le cas linéaire gaussien,
– il est seulement fait état de convergence finale. On n'a aucune information quant à la consistance au cours du processus de filtrage.

Malgré son avantage certain en termes de complexité algorithmique, le FastSLAM possède des inconvénients qui font que cette solution n'est pas idéale :

– le phénomène de dégénérescence des particules est problématique et nécessite de régulièrement rééchantillonner. Le moment adéquat n'est pas évident à déterminer. Dans sa thèse, Montemerlo effectue cette étape à chaque instant. D'autres auteurs définissent un seuil sur le nombre de particules effectives ([Hue *et al.*, 2002]),
– le rééchantillonnage diminue la diversité des particules, ce qui pose problème car on raisonne en termes de trajectoire : l'élimination des particules conduit à rendre déterministe les parties plus anciennes de la trajectoire. La conséquence est la perte des informations de corrélation entre les amers anciennement observés,
– de part son aspect particulaire, le **résultat final** de l'algorithme de FastSLAM n'est pas clair. En effet, on ne possède qu'un jeu de particules pondérées pour approcher une densité de probabilité ; il reste à effectuer l'estimation finale. On peut choisir de retourner la particule avec le plus fort poids pour l'estimation, ou encore d'effectuer une moyenne pondérée. Par ailleurs, l'obtention d'une information sur la qualité du résultat est délicate. Par exemple, comment donner



(a) Sparse map (b) Dense map

FIGURE 1.8 – Les deux cartes utilisées (bleues) avec les résultats d'un filtre particulaire à 1000 particules (rouge) ([Bailey *et al.*, 2006b])

une incertitude sur la position si on n'utilise que la particule la plus forte ? De plus, les matrices de variances-covariances associées à la carte sont données **conditionnellement** à la trajectoire de la particule. Elles ne reflètent pas directement l'incertitude globale sur les amers. La fusion de toutes les " cartes particulaires " peut être très délicate, surtout si on laisse l'algorithme effectuer l'association des données (l'association des données est valable pour une particule donnée ; certaines particules peuvent par exemple avoir plus d'amers que d'autres si certains appariements n'ont pas été trouvés). **En général, cet aspect n'est pas abordé dans la littérature**.

1.6.3 Consistance

Nous présentons dans ce paragraphe une étude de la consistance de l'algorithme de FastSLAM 2.0 réalisée dans [Bailey *et al.*, 2006b]. Deux points sont abordés :

1. la diversité des particules
2. la capacité de l'algorithme à être consistant

Pour les deux aspects, tous les résultats sont obtenus en simulation, en testant 4 configurations censées illustrer les cas d'utilisation possibles. Les auteurs testent deux densités de carte (cf. figure 1.8) et deux nombres de particules ($N = 100$ et $N = 1000$) Dans ce qui suit, les résultats de simulations sont obtenus avec les mêmes modèles de prédiction et de mesure que dans l'étude de consistance de l'EKF-SLAM (modèle de type tricycle et capteur de type *range and bearing*).

1.6.3.1 Diversité des particules

Par " diversité ", nous entendons le nombre d'hypothèses restantes parmi les hypothèses particulaires initiales. La perte de diversité des particules vient du fait que le filtre élimine les particules de poids trop faible et duplique celles de poids fort. Ceci contribue à éliminer des hypothèses pour le placement de certains amers, donnant alors " l'impression " que la précision de ces amers est améliorée.

Les auteurs montrent empirement ce problème en représentant le nombre d'hypothèses intiales encore présentes à chaque instant (cf. figure 1.9). Ainsi, il ne reste plus que quelques hypothèses sur le placement des amers initiaux en moins de 40s. Ceci posera problème si la boucle doit être fermée et que l'hypothèse unique retenue pour ces amers ne correspond pas parfaitement au moment de la fermeture de boucle.

Par ailleurs, les auteurs de la publication remarquent —*sans expliquer*— que cette décroissance est plus rapide dans le cas d'une carte dense, lorsqu'il y a donc beaucoup de mesures.[31] Ce résultat peut s'expliquer de manière analogue à celui du " mauvais comportement " bien connu des filtres particulaires lorsque les mesures sont trop précises par rapport au modèle. Dans notre cas, chaque mesure permet de mettre à jour le poids de chaque particule. Le caractère indépendant des mesures fait que cette mise à jour de poids est *multiplicative*. En conséquence, une particule peu probable aura plus d'incréments multiplicatifs proches de zéro dans le cas d'une carte dense ; le déséquilibre des poids est donc plus important lorsque le nombre de mesures est important. Ceci explique intuitivement pourquoi la diversité diminue plus rapidement lorsqu'on ajoute des mesures.

1.6.3.2 Consistance

Une étude de la consistance analogue à celle présentée pour l'EKF-SLAM est faite par les auteurs de [Bailey *et al.*, 2006b]. Etant donné que le résultat d'un filtre particulaire n'est qu'un jeu de particules, nous présentons d'abord quelles données sont utilisées pour effectuer les tests de consistance :

Estimation de l'état : calcul de la moyenne pondérée des particules

Incertitude de l'estimation : calcul de la variance empirique des particules avec la moyenne pondérée des écarts quadratiques à la moyenne.

Remarque 1.6

Ce calcul de variance n'est effectué que sur l'état. Ceci est réalisable car la dimension de l'état est la

31. Ce résultat est présenté comme non intuitif alors qu'il s'explique bien.

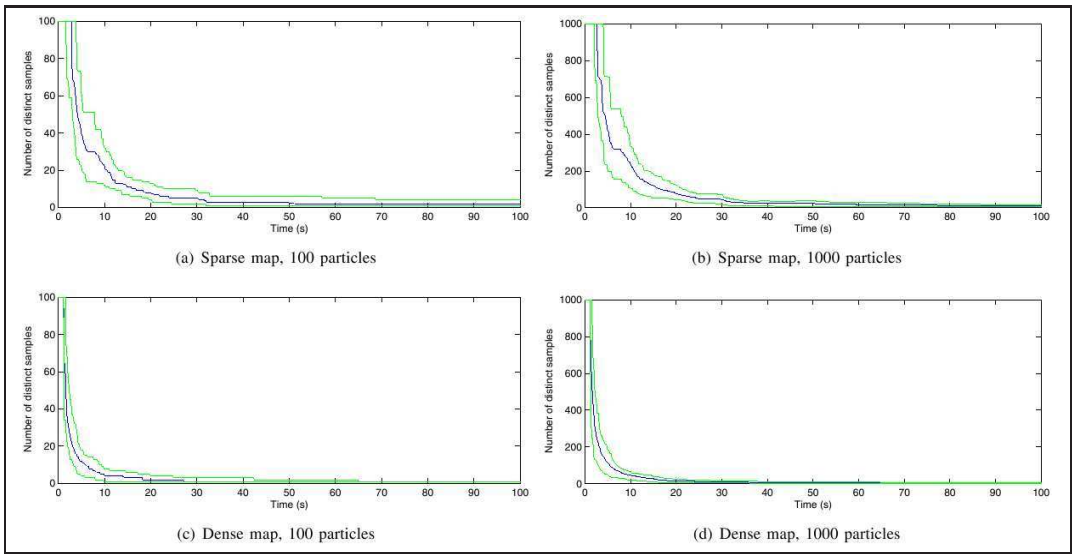


FIGURE 1.9 – Perte de diversité des particules calculée sur 50 simulations : valeur moyenne, bornes max et min ([Bailey *et al.*, 2006b])

même pour toutes les particules. Nous rappelons que ce calcul n'est pas faisable pour l'estimation de la variance globale des amers lorsque les associations ne sont pas connues (chaque particule peut avoir une association complètement différente, et les états des amers n'ont aucun lien entre eux).

La figure 1.10 compare la variance de l'erreur d'estimation (sur 50 simulations) à la variance moyenne fournie par le filtre. On remarque que le filtre est inconsistant très rapidement. Au bout de 200s, la variance prédite baisse brutalement. Ceci correspond à la fermeture de boucle : pratiquement toutes les hypothèses sont éliminées. Seules quelques particules très concentrées subsistent, ce qui provoque la baisse de la variance prédite.

La figure 1.10 met également en évidence le fait qu'un nombre plus élevé de particule permet de rendre le filtre plus consistant. Enfin, une carte moins dense permet d'obtenir de meilleurs résultats. Ces résultats semblent être liés à la diversité des particules.

Un calcul de consistance avec l'indice NESS est également effectué dans la publication étudiée et montre clairement l'inconsistance du filtre.

1.6.3.2.1 Conclusion sur la consistance

Au final, l'algorithme FastSLAM est inconsistant. La raison qui semble expliquer ce phénomène est **la perte de diversité des particules**. Cette perte de diversité entraîne un oubli de l'incertitude

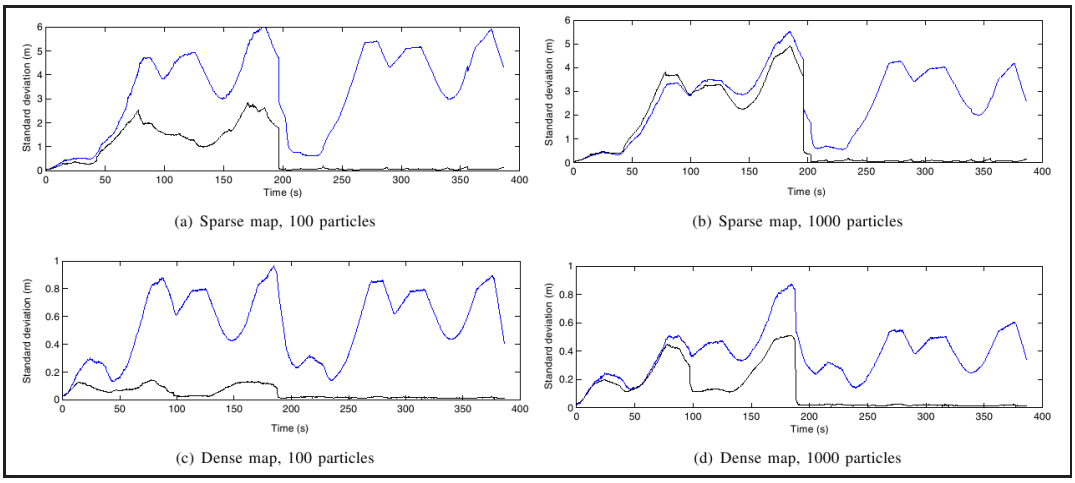


FIGURE 1.10 – Variance de l'erreur du filtre (courbe bleue) et variance prédite (courbe noire) [Bailey *et al.*, 2006b])

du début de la trajectoire.

Ce mauvais résultat du filtre particulaire est logique. Le problème de dégénérescence des particules est en effet bien connu. En pratique, il n'a pas d'influence sur le résultat pour les systèmes qui **oublient exponentiellement leur passé**. Ce n'est malheureusement pas le cas du SLAM : les anciennes erreurs subsistent dans la carte.

Ainsi, les auteurs de [Bailey *et al.*, 2006b] estiment que le FastSLAM n'est valable que pour l'initialisation et ne doit pas être utilisé pendant très longtemps. Pourtant, l'intérêt du FastSLAM est sa rapidité en termes de temps de calculs pour les grandes cartes. **Utiliser le FastSLAM uniquement pour l'initialisation apparaît en contradiction avec la problématique qui a motivé sa découverte**. En conséquence, cet algorithme n'a pas été utilisé dans la suite de la thèse.

1.7 Résolution par lissage probabiliste : le SAM

Nous présentons dans cette section l'algorithme de GraphSLAM. Il s'agit d'un algorithme dit de SAM (**Smoothing and Mapping**) dont le but est de lisser **l'ensemble de la trajectoire** sous hypothèse gaussienne (au lieu de ne filtrer que la position courante comme l'EKF). On va donc chercher à retrouver la densité de probabilité associée à la concaténation de l'ensemble des positions du robot et des amers. Nous verrons que ceci est possible dans un temps raisonnable grâce à l'utilisation des propriétés de la **matrice d'information** associée à la densité cherchée. Cette section est divisée en trois paragraphes. Nous décrivons dans un premier temps l'algorithme de GraphSLAM. Cette description

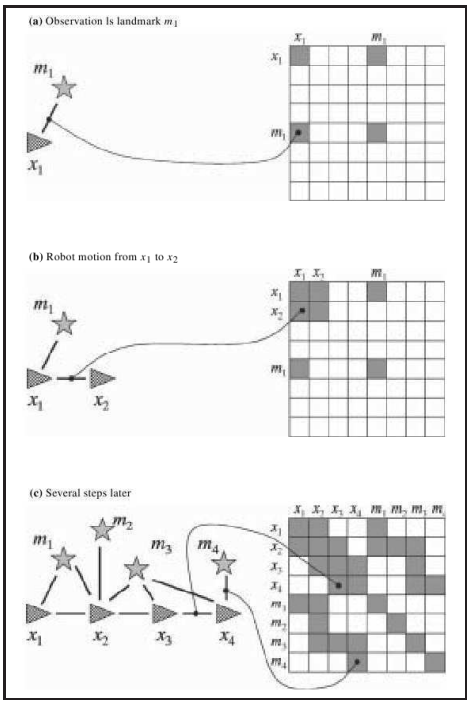


FIGURE 1.11 – Construction incrémentale de la matrice d'information totale ([Thrun et Montemerlo, 2006])

est essentiellement tirée de [Thrun et Montemerlo, 2006]. Nous verrons alors les avantages obtenus grâce à l'utilisation des paramètres d'informations. Nous détaillons ensuite les avantages et limites pratiques de ce type de méthodes. Nous faisons enfin quelques remarques concernant la consistance de cet algorithme.

1.7.1 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 (propriété qui était déjà utilisée dans le cadre du FastSLAM). 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 du système est construite de manière incrémentale (cf. figure 1.11). 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 incrémentalement la matrice d'information ; ceci a pour effet d'ajouter des termes dans la matrice d'information (cf. figure 1.12).

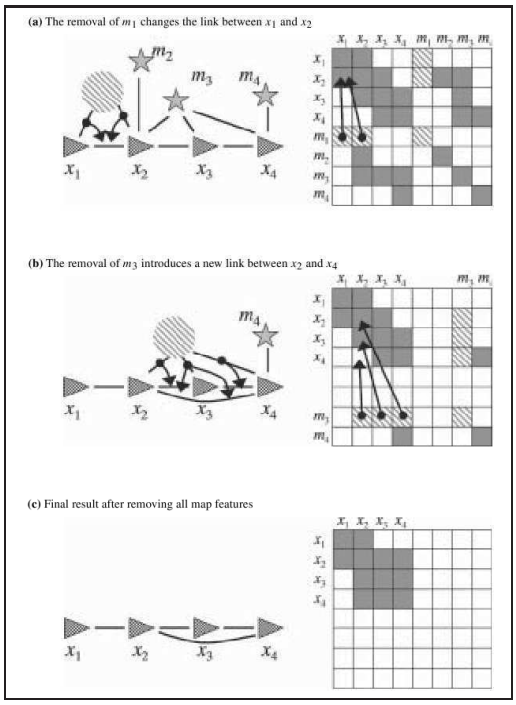


FIGURE 1.12 – Marginalisation des amers pour retrouver la pose uniquement. Cette opération entraîne l'ajout de liens entre les positions ([Thrun et Montemerlo, 2006]) — Les flèches représentent le processus de marginalisation et les disques les amers marginalisés

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.

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

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

$$p\left(\mathbf{x}_{0:t}, \mathbf{m} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right) = \frac{p\left(\mathbf{z}_t | \mathbf{x}_{0:t}, \mathbf{m}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}\right) p\left(\mathbf{x}_{0:t}, \mathbf{m} | \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}\right)}{p\left(\mathbf{z}_t | \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}\right)} \tag{1.79}$$

soit :

$$p\left(\mathbf{x}_{0:t}, \mathbf{m} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right) = \eta \; p\left(\mathbf{z}_t | \mathbf{x}_{0:t}, \mathbf{m}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}\right) p\left(\mathbf{x}_{0:t}, \mathbf{m} | \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}\right) \tag{1.80}$$

où η est une constante de normalisation. En appliquant l'équation 1.45 sur le premier facteur de l'équation 1.80, on obtient :

$$p\left(\mathbf{x}_{0:t}, \mathbf{m} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right) = \eta \; p\left(\mathbf{z}_t | \mathbf{x}_t, \mathbf{m}\right) p\left(\mathbf{x}_{0:t}, \mathbf{m} | \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}\right) \tag{1.81}$$

En appliquant la règle de Bayes, le second facteur dans l'équation 1.81 peut s'écrire :

$$p\left(\mathbf{x}_{0:t}, \mathbf{m}|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}\right) = p\left(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{m}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}\right) p\left(\mathbf{x}_{0:t-1}, \mathbf{m}|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}\right) \tag{1.82}$$

En utilisant l'équation 1.43, l'équation 1.82 devient :

$$p\left(\mathbf{x}_{0:t}, \mathbf{m}|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}\right) = p\left(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t\right) p\left(\mathbf{x}_{0:t-1}, \mathbf{m}|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}\right) \tag{1.83}$$

On peut désormais insérer le résultat de l'équation 1.83 dans l'équation 1.81 pour obtenir la formulation récursive suivante :

$$p\left(\mathbf{x}_{0:t}, \mathbf{m}|\mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right) = \eta\ p\left(\mathbf{z}_t|\mathbf{x}_t, \mathbf{m}\right) p\left(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t\right) p\left(\mathbf{x}_{0:t-1}, \mathbf{m}|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1}\right) \tag{1.84}$$

On peut alors montrer par récurrence que :

$$p\left(\mathbf{x}_{0:t}, \mathbf{m}|\mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right) = \eta'\ p\left(\mathbf{x}_0, \mathbf{m}\right) \prod_{k=1}^{t} p\left(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k\right) p\left(\mathbf{z}_k, |\mathbf{x}_k, \mathbf{m}\right) \tag{1.85}$$

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 :

$$p\left(\mathbf{x}_{0:t}, \mathbf{m}|\mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right) = \eta'\ p\left(\mathbf{x}_0, \mathbf{m}\right) \prod_{k=1}^{t} \left[p\left(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k\right) \prod_{i \in \mathcal{M}_t} p\left(\mathbf{z}_{k,(i)}, |\mathbf{x}_k, \mathbf{m}_{(i)}\right) \right] \tag{1.86}$$

Remarque 1.7 (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.*

1.7.1.2 Logarithme de la densité *a posteriori*

Calculons le logarithme de la densité calculée dans l'équation 1.86. En tenant compte de la remarque 1.7, nous obtenons :

$$\log p\left(\mathbf{x}_{0:t}, \mathbf{m} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right) = \text{const.} + \log p\left(\mathbf{x}_0\right) + \sum_{k=1}^{t} \left[\log p\left(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{u}_k\right) \sum_{i \in \mathcal{M}_k} \log p\left(\mathbf{z}_{k,(i)}, | \mathbf{x}_k, \mathbf{m}_{(i)}\right) \right] \tag{1.87}$$

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 :

$$\left\{ \begin{array}{rcl} p\left(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{u}_t\right) & \propto & \exp\left(-\frac{1}{2} \left(\mathbf{x}_t - \mathbf{f}_t(\mathbf{x}_{t-1}, \mathbf{u}_t)\right)^T \mathbf{Q}_t^{-1} \left(\mathbf{x}_t - \mathbf{f}_t(\mathbf{x}_{t-1}, \mathbf{u}_t)\right) \right) \\ p\left(\mathbf{z}_{t,(i)}, | \mathbf{x}_t, \mathbf{m}_{(i)}\right) & \propto & \exp\left(-\frac{1}{2} \left(\mathbf{z}_{t,(i)} - \mathbf{h}(\mathbf{x}_t, \mathbf{m}_{(i)})\right)^T \mathbf{R}_t^{-1} \left(\mathbf{z}_{t,(i)} - \mathbf{h}(\mathbf{x}_t, \mathbf{m}_{(i)})\right) \right) \end{array} \right. \tag{1.88}$$

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 :

$$\boxed{ \begin{array}{rcl} \log p\left(\mathbf{x}_{0:t}, \mathbf{m} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right) & = & \text{const.} - \frac{1}{2} \Big[\mathbf{x}_0^T \mathbf{\Omega}_0 \mathbf{x}_0 \\ & & + \sum_{k=1}^{t} \left(\mathbf{x}_k - \mathbf{f}_k(\mathbf{x}_{k-1}, \mathbf{u}_k)\right)^T \mathbf{Q}_t^{-1} \left(\mathbf{x}_k - \mathbf{f}_k(\mathbf{x}_{k-1}, \mathbf{u}_k)\right) \\ & & + \sum_{k=1}^{t} \sum_{i \in \mathcal{M}_k} \left(\mathbf{z}_{k,(i)} - \mathbf{h}(\mathbf{x}_k, \mathbf{z}_{k,(i)})\right)^T \mathbf{R}_t^{-1} \left(\mathbf{z}_{k,(i)} - \mathbf{h}(\mathbf{x}_k, \mathbf{z}_{k,(i)})\right) \Big] \end{array} } \tag{1.89}$$

Dans l'équation précédente, le premier terme ($\mathbf{x}_0^T \mathbf{\Omega}_0 \mathbf{x}_0$) traduit la connaissance *a priori* de la position intiale du robot. Dans notre cas, nous supposons connue la position initiale du robot (ce qui revient à supposer que $p\left(\mathbf{x}_0\right)$ est un Dirac. Pour calculer la densité de probabiltié conditionnée à $\mathbf{x}_0$, nous pouvons au choix :

- Appliquer le théorème de conditionnement des variables aléatoires gaussiennes sur la matrice d'information (décrit dans le paragraphe suivant à l'équation 1.104)
- Supposer que l'on a une connaissance " très grande " sur $\mathbf{x}_0$, ce qui revient à choisir $\mathbf{\Omega}_0$ diagonale avec de très grandes valeurs numériques (et en déduire le bon ξ_0 en fonction de la condition initiale choisie).

1.7.1.3 Expansion de Taylor

L'équation 1.89 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 $\mu^{\mathbf{x}}_{0:t}$ une trajectoire de linéarisation supposée proche de la trajectoire réelle et $\mu^{\mathbf{m}}$ un point de linéarisation pour la carte. Nous supposons $\mu^{\mathbf{x}}_{0:t}$ et $\mu^{\mathbf{m}}$ disponibles ; nous discuterons de la manière de les obtenir dans le paragraphe 1.7.1.5.

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

$$\left\{ \begin{array}{rcl} \mathbf{f}_t(\mathbf{x}_{t-1}, \mathbf{u}_t) & \approx & \mathbf{f}_t(\mu^{\mathbf{x}}_{t-1}, \mathbf{u}_t) + \mathbf{G}_t \left(\mathbf{x}_{t-1} - \mu^{\mathbf{x}}_{t-1}\right) \\ \mathbf{h}(\mathbf{x}_t, \mathbf{m}_{(i)}) & \approx & \mathbf{h}(\mu_t, \mu_{(i)}) + \mathbf{H}^{\mathbf{x}}_{t,(i)} \left(\mathbf{x}_t - \mu^{\mathbf{x}}_t\right) + \mathbf{H}^{\mathbf{m}}_{t,(i)} \left(\mathbf{m}_{(i)} - \mu^{\mathbf{m}}_{(i)}\right) \end{array} \right. \tag{1.90}$$

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 1.89 peut se ré-écrire de la façon suivante :

$$\begin{aligned} \log p\left(\mathbf{x}_{0:t}, \mathbf{m} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right) = \text{const.'} - \tfrac{1}{2} \underbrace{\mathbf{x}_0^T \boldsymbol{\Omega}_0 \mathbf{x}_0}_{⓪} \\ + \sum_{k=1}^{t} \left\{ \begin{array}{l} \underbrace{-\tfrac{1}{2} \begin{bmatrix} \mathbf{x}_k \\ \mathbf{x}_{k-1} \end{bmatrix}^T \begin{bmatrix} \mathbf{I} \\ -\mathbf{G}_k^T \end{bmatrix} \mathbf{Q}_k^{-1} \begin{bmatrix} \mathbf{I} & -\mathbf{G}_k \end{bmatrix} \begin{bmatrix} \mathbf{x}_k \\ \mathbf{x}_{k-1} \end{bmatrix}}_{①} \\ + \begin{bmatrix} \mathbf{x}_k \\ \mathbf{x}_{k-1} \end{bmatrix}^T \underbrace{\begin{bmatrix} \mathbf{I} \\ -\mathbf{G}_k^T \end{bmatrix} \mathbf{Q}_k^{-1} \left(\mathbf{f}_k\left(\mu^{\mathbf{x}}_{k-1}, \mathbf{u}_k\right) + \mathbf{G}_k \mu^{\mathbf{x}}_{k-1}\right)}_{②} \\ + \sum_{i \in \mathcal{M}_k} \left\{ \begin{array}{l} \underbrace{-\tfrac{1}{2} \begin{bmatrix} \mathbf{x}_k \\ \mathbf{m}_{(i)} \end{bmatrix}^T \begin{bmatrix} \left(\mathbf{H}^{\mathbf{x}}_{k,(i)}\right)^T \\ \left(\mathbf{H}^{\mathbf{m}}_{k,(i)}\right)^T \end{bmatrix} \mathbf{R}_k^{-1} \begin{bmatrix} \mathbf{H}^{\mathbf{x}}_{k,(i)} & \mathbf{H}^{\mathbf{m}}_{k,(i)} \end{bmatrix} \begin{bmatrix} \mathbf{x}_k \\ \mathbf{m}_{(i)} \end{bmatrix}}_{③} \\ + \begin{bmatrix} \mathbf{x}_k \\ \mathbf{m}_{(i)} \end{bmatrix}^T \underbrace{\begin{bmatrix} \left(\mathbf{H}^{\mathbf{x}}_{k,(i)}\right)^T \\ \left(\mathbf{H}^{\mathbf{m}}_{k,(i)}\right)^T \end{bmatrix} \mathbf{R}_k^{-1} \left(\mathbf{z}_{k,(i)} - \mathbf{h}\left(\mu^{\mathbf{x}}_k, \mu^{\mathbf{m}}_{(i)}\right) + \begin{bmatrix} \mathbf{H}^{\mathbf{x}}_{k,(i)} & \mathbf{H}^{\mathbf{m}}_{k,(i)} \end{bmatrix} \begin{bmatrix} \mu^{\mathbf{x}}_k \\ \mu^{\mathbf{m}}_{(i)} \end{bmatrix}\right)}_{④} \end{array} \right\} \end{array} \right\} \end{aligned} \tag{1.91}$$

Dans l'équation 1.91, 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 1.91 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 :

$$\log p\left(\mathbf{x}_{0:t}, \mathbf{m} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right) = \text{const.'} - \frac{1}{2} \begin{bmatrix} \mathbf{x}_{0:t} \\ \mathbf{m} \end{bmatrix}^T \mathbf{\Omega} \begin{bmatrix} \mathbf{x}_{0:t} \\ \mathbf{m} \end{bmatrix} + \begin{bmatrix} \mathbf{x}_{0:t} \\ \mathbf{m} \end{bmatrix}^T \xi \tag{1.92}$$

La matrice $\mathbf{\Omega}$ de l'équation 1.92 est construite à partir du terme ⓪ et des termes de type ① et ② de l'équation 1.91. Le vecteur d'information ξ, quant à lui, est construit grâce aux termes de type ② et ④ de l'équation 1.91.

1.7.1.4 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 1.92 construits, on pourrait essayer de retrouver l'équivalent en termes de matrice de variances-covariances et d'espérance mathématique. La matrice de variances-covariances totale serait pleine. Les auteurs de [Thrun et Montemerlo, 2006] 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* :

$$p\left(\mathbf{x}_{0:t}, \mathbf{m} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right) = p\left(\mathbf{x}_{0:t} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right) p\left(\mathbf{m} | \mathbf{x}_{0:t}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right) \tag{1.93}$$

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 :

$$\mathbf{\Omega} = \begin{bmatrix} \mathbf{\Omega}_{\mathbf{x}_{0:t}, \mathbf{x}_{0:t}} & \mathbf{\Omega}_{\mathbf{x}_{0:t}, \mathbf{m}} \\ \mathbf{\Omega}_{\mathbf{m}, \mathbf{x}_{0:t}} & \mathbf{\Omega}_{\mathbf{m}, \mathbf{m}} \end{bmatrix} \tag{1.94}$$

$$\xi = \begin{bmatrix} \xi_{\mathbf{x}_{0:t}} \\ \xi_{\mathbf{m}} \end{bmatrix} \tag{1.95}$$

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

Etant donné que les densités de probabilités 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 :

$$\tilde{\mathbf{\Omega}} = \mathbf{\Omega}_{\mathbf{x}_{0:t},\mathbf{x}_{0:t}} - \mathbf{\Omega}_{\mathbf{x}_{0:t},\mathbf{m}}\mathbf{\Omega}^{-1}_{\mathbf{m},\mathbf{m}}\mathbf{\Omega}_{\mathbf{m},\mathbf{x}_{0:t}} \tag{1.96}$$

$$\tilde{\xi} = \xi_{\mathbf{x}_{0:t}} - \mathbf{\Omega}_{\mathbf{x}_{0:t},\mathbf{m}}\mathbf{\Omega}^{-1}_{\mathbf{m},\mathbf{m}}\xi_{\mathbf{m}} \tag{1.97}$$

L'équation 1.91 nous montre que la matrice $\mathbf{\Omega}_{\mathbf{m},\mathbf{m}}$ est diagonale par blocs :

$$\mathbf{\Omega}_{\mathbf{m},\mathbf{m}} = \text{diag}\left(\mathbf{\Omega}_{(1),(1)} , \dots , \mathbf{\Omega}_{(j),(j)} , \dots , \mathbf{\Omega}_{(N),(N)} \right) \tag{1.98}$$

où N représente le nombre total d'amers et :

$$\mathbf{\Omega}_{(j),(j)} = \sum_{k\in\tau(j)} \left(\mathbf{H}^{\mathbf{m}}_{k,(i)}\right)^T \mathbf{R}_k \left(\mathbf{H}^{\mathbf{m}}_{k,(i)}\right) \tag{1.99}$$

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 :

$$\mathbf{\Omega}_{\mathbf{x}_{0:t},\mathbf{m}} = \left[\begin{array}{cccc} \mathbf{\Omega}_{\mathbf{x}_{0:t},(1)} & \cdots & \mathbf{\Omega}_{\mathbf{x}_{0:t},(j)} & \cdots & \mathbf{\Omega}_{\mathbf{x}_{0:t},(N)} \end{array} \right] \tag{1.100}$$

avec :

$$\mathbf{\Omega}_{\mathbf{x}_{0:t},(j)} = \left[\begin{array}{c} \mathbf{\Omega}_{\mathbf{x}_1,(j)} \\ \vdots \\ \mathbf{\Omega}_{\mathbf{x}_k,(j)} \\ \vdots \\ \mathbf{\Omega}_{\mathbf{x}_t,(j)} \end{array} \right] \quad \text{et} \quad \left\{ \begin{array}{lcll} \mathbf{\Omega}_{\mathbf{x}_k,(j)} & = & \left(\mathbf{H}^{\mathbf{x}}_{k,(i)}\right)^T \mathbf{R}^{-1}_k \left(\mathbf{H}^{\mathbf{m}}_{k,(i)}\right) & \textbf{si } k \in \tau(j) \\ \mathbf{\Omega}_{\mathbf{x}_k,(j)} & = & \mathbf{0} & \textbf{sinon} \end{array} \right. \tag{1.101}$$

Au final, on a :

$$\boxed{\begin{array}{rcl} \tilde{\mathbf{\Omega}} & = & \mathbf{\Omega}_{\mathbf{x}_{0:t},\mathbf{x}_{0:t}} - \sum_{j=1}^{N} \mathbf{\Omega}_{\mathbf{x}_{0:t},(j)} \left(\mathbf{\Omega}_{(j),(j)}\right)^{-1} \mathbf{\Omega}_{(j),\mathbf{x}_{0:t}} \\ \tilde{\xi} & = & \xi_{\mathbf{x}_{0:t}} - \sum_{j=1} \mathbf{\Omega}_{\mathbf{x}_{0:t},(j)} \left(\mathbf{\Omega}_{(j),(j)}\right)^{-1} \xi_{(j)} \end{array}} \tag{1.102}$$

avec :

$$\xi_{(j)} = \sum_{k\in\tau(j)} \left\{ \left(\mathbf{H}^{\mathbf{m}}_{k,(j)}\right)^T \mathbf{R}^{-1}_k \left(\mathbf{z}_{k,(i)} - \mathbf{h}\left(\mu^{\mathbf{x}}_k, \mu^{\mathbf{m}}_{(j)}\right) + \left[\begin{array}{cc} \mathbf{H}^{\mathbf{x}}_{k,(j)} & \mathbf{H}^{\mathbf{m}}_{k,(j)} \end{array} \right] \left[\begin{array}{c} \mu^{\mathbf{x}}_k \\ \mu^{\mathbf{m}}_{(j)} \end{array} \right] \right) \right\} \tag{1.103}$$

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

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

$$\boxed{\begin{array}{rcl} \tilde{\mathbf{\Omega}}^{\mathbf{m}|\mathbf{x}} & = & \mathbf{\Omega}_{\mathbf{m},\mathbf{m}} \\ \tilde{\xi}^{\mathbf{m}|\mathbf{x}} & = & \xi_{\mathbf{m}} + \mathbf{\Omega}_{\mathbf{m},\mathbf{x}_{0:t}}\mathbf{x}_{0:t} \end{array}} \tag{1.104}$$

1.7.1.4.3 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. Soient $\hat{\mathbf{x}}_{0:t}$ et $\mathbf{\Sigma}^{\mathbf{x}}$ les paramètres cherchés, on a :

$$\boxed{\begin{array}{rcl} \mathbf{\Sigma}^{\mathbf{x}} & = & \tilde{\mathbf{\Omega}}^{-1} \\ \hat{\mathbf{x}}_{0:t} & = & \hat{\mathbf{\Sigma}}^{\mathbf{x}}\tilde{\xi} \end{array}} \tag{1.105}$$

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

$$\begin{array}{rcl} \hat{\mathbf{\Sigma}}^{\mathbf{m}} & = & \mathbf{\Omega}^{-1}_{\mathbf{m},\mathbf{m}} \\ \hat{\mathbf{m}} & = & \mathbf{\Sigma}^{\mathbf{m}}\left(\xi_{\mathbf{m}} + \mathbf{\Omega}_{\mathbf{m},\mathbf{x}_{0:t}}\hat{\mathbf{x}}_{0:t}\right) \end{array} \tag{1.106}$$

Sachant que la matrice $\mathbf{\Omega}_{\mathbf{m},\mathbf{m}}$ 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) :

$$\begin{array}{rcl} \mathbf{\Sigma}_{(j)} & = & \mathbf{\Omega}^{-1}_{(j),(j)} \\ \hat{\mathbf{m}}_{(j)} & = & \mathbf{\Sigma}_{(j)}\left(\xi_{(j)} + \mathbf{\Omega}_{(j),\mathbf{x}_{0:t}}\hat{\mathbf{x}}_{0:t}\right) \quad = \quad \mathbf{\Sigma}_{(j)}\left(\xi_{(j)} + \mathbf{\Omega}_{(j),\tau(j)}\hat{\mathbf{x}}_{\tau(j)}\right) \end{array} \tag{1.107}$$

Les auteurs de [Thrun et Montemerlo, 2006] 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.

1.7.1.5 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.

1.7.1.5.1 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 [32]).

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

Remarque 1.8

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.*

1.7.1.5.2 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.

1.7.1.6 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 ([Dellaert et Kaess, 2006]). Dans ce dernier cas, des optimisations sont effectuées afin de factoriser la matrice d'information et de réduire les temps de calcul.

1.7.2 Avantages et limites pratiques

Nous avons présenté dans cette section le principe des méthodes de SAM. Ces méthodes possèdent de nombreux avantages :

- l'intégralité de la trajectoire est remise à jour par l'algorithme. Ceci assure une bonne cohérence de l'ensemble de la trajectoire et évite les sauts souvent constatés dans le cas de l'EKF
- la matrice d'information utilisée est creuse. Les calculs pour la mise à jour de l'état peuvent être effectués rapidement (par exemple, l'algorithme de $\sqrt{\text{SAM}}$ ([Dellaert et Kaess, 2006; Kaess

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

et al., 2007]) utilise cette propriété pour effectuer une factorisation optimisée de la matrice d'information),

– pour une itération donnée, **le point de linéarisation associé aux amers est commun à l'ensemble de la trajectoire**. On peut donc espérer hériter des bonnes propriétés de consistance du FEJ-EKF, tout en ayant la possibilité d'affiner les points de linéarisation.

Cependant, cette méthode n'est pas parfaite. Ces principaux défauts résident essentiellement dans les temps de calculs :

– le fait de traiter l'intégralité de la trajectoire implique une augmentation considérable de la taille de l'état,

– Calculer la matrice de variances-covariances globale est très couteux en temps de calcul. En effet, même si la matrice d'information est largement éparse, son inverse ne l'est pas. Il conviendra donc de chosir précisement quels morceaux de la matrice de variances-covariances globale doivent être calculés afin d'optimiser son calcul.

1.7.3 Consistance du SAM

Nous n'avons pas trouvé dans la littérature d'étude intensive de la consistance du SAM, comme cela a été fait pour l'EKF-SLAM et le FastSLAM. Néanmoins, nous pouvons déjà remarquer que pour une itération donnée, le point de fonctionnement associé aux amers est commun à toutes les poses de la trajectoire. La condition nécessaire de consistance d'après [Huang *et al.*, 2008] est donc respectée. Dans le chapitre suivant, nous proposons une étude concernant la consistance de cet algorithme dans le cadre d'une comparaison avec un algorithme de SLAM déterministe (reprise de [Joly et Rives, 2008]). Nous montrerons alors que l'algorithme de SAM est consistant si les erreurs sont bien centrées.

1.8 Conclusion sur les méthodes de résolution probabilistes

Nous avons dans ce chapitre les bases du SLAM probabilistes ainsi que les principaux algorithmes d'estimation associés. Tous ont leurs avantages et inconvénients. L'EKF-SLAM permet de maintenir la corrélation totale entre les amers grâce à la matrice de variances-covariances. En revanche, sa complexité algorithmique est importante et il a été montré que cet algorithme est largement inconsistant. La plupart de ses variantes (dont l'EKF et l'IEKF) sont également inconsistantes. Il a été récemment montré que l'EKF change les propriétés d'observabilité du SLAM concernant l'orientation : ceci serait la cause principale de sous-estimation de l'erreur.

L'algorithme de FastSLAM, quant à lui, est moins couteux en calculs mais entraîne le problème de la dégénérescence des particules (qui est fondamentale étant donné qu'un système de SLAM n'est pas un système à oubli du passé).

Enfin, les algorithmes de lissage nous paraissent les plus adaptés. En effet, le fait d'utiliser l'intégralité de la trajectoire dans l'état nous permet dans un premier temps de corriger une éventuelle mauvaise estimation concernant l'état du robot (ce qui est quasiment impossible avec l'EKF-SLAM). Par ailleurs, cet algorithme utilise le caractère épars de la matrice d'information totale, ce qui peut permettre d'accélérer les temps de calculs. De plus, cet algorithme utilise à chaque itération l'intégralité des mesures avec un unique point de fonctionnement pour les amers (ce qui semble être la condition nécessaire de consistance d'après [Huang *et al.*, 2008]). Nous espérons donc que cet algorithme présentera de bonnes propriétés de consistance. En conséquence, parmi les approches probabilistes, **nous privilégions cet algorithme dans la suite de la thèse, malgré ses contraintes concernant la taille de l'état**.

Chapitre 2

Le SLAM déterministe

Sommaire

Dans le chapitre précédent, nous avons présenté plusieurs méthodes probabilistes permettant de résoudre le problème du SLAM. Ces méthodes visent à retrouver la densité de probabilité *a posteriori* de la dernière position du robot (ou la trajectoire) et de la carte de l'environnement, en général sous l'hypothèse de densités de départ gaussiennes. En raison des non linéarités des fonctions employées, ces méthodes effectuent des approximations :

- Dans le cas des méthodes de type EKF ou SAM, on linéarise les fonctions en supposant que le résultat restera gaussien,
- Dans le cas du FastSLAM, on approxime la densité de probabilité continue par un jeu de particules discrètes.

Les méthodes probabilistes représentent la très grande majorité des méthodes de résolution du SLAM. Nous présentons dans ce chapitre une méthode plus originale de résolution utilisant des notions d'analyse par intervalles. Ici, l'idée est de supposer que l'on dispose d'équations de modèle et de mesures affectées par un bruit **borné** dont un **intervalle contenant les bornes est connu**.[1] Ces équations seront alors utilisées afin de déduire une solution au problème du SLAM **à erreur bornée**.

Ce chapitre se divise en cinq sections. Nous présentons dans la section 2.1 les nouvelles hypothèses et notations associées au SLAM déterministe. Nous présentons ensuite les bases de l'analyse par intervalles (tirées des premiers chapitre de [Moore, 1979]), à savoir la redéfinition des opérateurs classiques et la façon d'évaluer les fonctions (section 2.2). Nous montrons dans la section 2.3 notre approche pour appliquer ces notions d'analyse par intervalles pour résoudre le problème du SLAM. Celle-ci a fait l'objet d'une publication dans un rapport de recherches INRIA ([Joly et Rives, 2008]). La méthode proposée est décrite de manière générique, sans définir de modèle d'évolution du robot ou de mesures (comme dans le cas du chapitre précédent). Nous présentons dans la section 2.4 une application de l'analyse par intervalles issue de la littérature et publiée en même temps que la nôtre. Enfin, nous concluons dans la section 2.5 par une discussion sur les avantages et inconvénients attendus du SLAM par intervalles.

1. Cet intervalle doit permettre d'assurer que le bruit est contenu dedans. Eventuellement, il peut être plus large que les bornes réelles (on dira alors que cet intervalle est pessimiste).

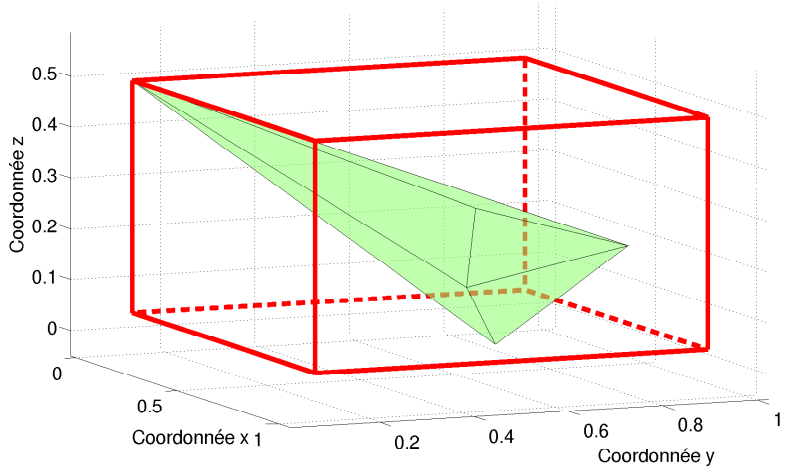


FIGURE 2.1 – Illustration du concept de boîte – Dans le cas multidimensionnel, les boîtes sont parallèles aux axes. Nous n'introduisons pas la notion de corrélation : les calculs géométriques seraient alors trop compliqués. Ainsi, si le résultat optimal d'une résolution déterministe conduirait au polytope vert, l'analyse par intervalle pourra au mieux obtenir la boîte rouge. La solution pourraît être améliorée par l'utilisation de l'union de plusieurs boîtes (formant ainsi un sous-pavage approximant mieux le polytope) ; cet aspect n'a pas été abordé dans ces travaux.

2.1 Formulation du problème

2.1.1 Notations et hypothèses

Les notations utilisées concernant les différents états sont les mêmes que celles définies dans le chapitre précédent (section 1.2 p.1.2). On retrouve donc $\mathbf{x}_t$, $\mathbf{u}_t$, $\mathbf{m}_{(i)}$, $\mathbf{z}_{t,(i)}$, $\mathbf{z}_t$ pour décrire respectivement l'état du robot, le vecteur de commandes, le vecteur d'état de l'amer (i), le vecteur de mesures de l'amer (i) à l'instant t et l'ensemble du vecteur de mesures à t.

Par ailleurs, nous introduisons de nouvelles notations et notions relatives à la représentation par intervalles :

- Soit $x \in \mathbb{R}$ une variable réelle. Sa représentation en analyse par intervalles est notée $I_x = [\underline{x}, \overline{x}]$. Par hypothèse, la probabilité de l'évènement $x \in I_x$ est égale à 1 (l'intervalle I_x est dit consistant),
- La taille de l'intervalle I_x est notée $t(I_x)$ et vaut $t(I_x) = \overline{x} - \underline{x}$,
- Soit $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T \in \mathbb{R}^n$ $(n > 1)$ une variable multidimensionnelle. Sa représentation en analyse par intervalles est une boîte notée $\mathbf{I_x} = I_{x_1} \times I_{x_2} \times \cdots \times I_{x_n}$.

Remarque 2.1 (“ Topologie ” des boîtes)

La définition de boîtes pour le cas multidimensionnel implique que les intervalles associés à chaque

composante du vecteur soient indépendants. On ne prendra pas en compte des configurations plus complexes telles que des polytopes par exemple (cf. figure 2.1). Ainsi, l'estimation d'un état peut être donnée par le centre de la boîte, et les incertitudes sont fournies composante par composante. La notion de corrélation n'intervient pas dans la description par boîtes.

Les quatre hypothèses présentées dans le chapitre précédent (page 25) peuvent être transposées à l'analyse par intervalles. Néanmoins, elles ne sont pas toutes utilisées :

1. Sachant des boîtes *a priori* sur le vecteur de commandes $\mathbf{u}_t$ et sur l'état $\mathbf{x}_{t-1}$ du robot, il est possible de prédire une boîte $\mathbf{I}_{\mathbf{x_t}}$ pour l'état du robot à l'instant t,
2. La seconde hypothèse (concernant l'independance des erreurs de mesures) **n'est pas utilisée**. Nous avons juste besoin de connaître des bornes concernant les erreurs. Celles-ci peuvent être très bien être corrélées, avec par exemple la présente d'un biais systématique,
3. La mesure à l'instant k ne dépend que de l'état du robot à l'instant k et de la position des amers.
4. A chaque instant, la mesure de l'amer (i) est indépendante des mesures des autres amers et ne dépend que de l'amer (i) et de l'état du robot.

2.1.2 SLAM par intervalles

Dans le cas du SLAM par intervalles, nous cherchons une boîte pour l'ensemble des états du robot ($\mathbf{x}_{0:t}$) et des amers ($\mathbf{m}$) en supposant connues des boîtes pour les mesures $\mathbf{z}_{0:t}$ et les vecteurs de commandes $\mathbf{u}_{1:t}$.

Pour satisfaire la première hypothèse énoncée au paragraphe précédent, nous supposons qu'il existe une fonction $\mathbf{f}$ telle que :

$$\mathbf{x_t} = \mathbf{f}\left(\mathbf{x}_{t-1}, \mathbf{u}_t\right) + \nu^{\mathbf{f}}_{t+1} \tag{2.1}$$

où on suppose connue une boîte $\mathbf{I}_{\nu^{\mathbf{f}}_{t+1}}$ contenant le vecteur $\nu^{\mathbf{f}}_{t+1}$ (ie. l'erreur de modèle) et une boîte $\mathbf{I}_{\mathbf{u}_t}$ contenant le vecteur $\mathbf{u}_t$ (ce qui permet de prendre en compte une erreur sur le vecteur de commandes).

Pour satisfaire les troisième et quatrième hypothèses, on suppose qu'il existe une fonction $\mathbf{h}$ telle que :

$$\mathbf{z}_{t,(i)} = \mathbf{h}(\mathbf{x}_t, \mathbf{m}_{(i)}) + \nu^{\mathbf{z}}_{t,(i)} \tag{2.2}$$

où on suppose connue une boîte $\mathbf{I}_{\nu^{\mathbf{z}}_{t,(i)}}$ contenant le vecteur $\nu^{\mathbf{z}}_{t,(i)}$ (ie. l'erreur de mesure). La quatrième hypothèse implique que chaque mesure d'amer ne dépend que de l'amer considéré et de l'état du robot (équation 2.2). Elle implique également que les erreurs $\nu^{\mathbf{z}}_{t,(i)}$ soient indépendantes entre les amers. **Dans**

l'analyse par intervalles, cette dernière propriété n'est pas utilisée. Seule la bornitude des erreurs (et la connaissance de bornes majorantes) est nécessaire.

Pour résoudre le problème du SLAM avec l'analyse par intervalles, nous allons utiliser successivement les fonctions **f** et **h** définies dans l'équation 2.2. Cette utilisation est décrite dans la section 2.3. Avant cela, nous décrivons les principales notions d'analyse par intervalles.

2.2 Notions d'analyse par intervalles

Nous présentons dans cette section les notions de base permettant d'évaluer des fonctions avec des intervalles. Il est indispensable de maîtriser ces notions avant d'appliquer les méthodes de résolution de SLAM par intervalles. Nous présentons dans un premier temps l'arithmétique classique des intervalles (addition, soustraction,...). Dans un second temps, nous précisons le traitement des fonctions trigonométriques (celles-ci étant fréquemment utilisées dans le cadre du SLAM [2]). Enfin, nous expliquons comment traiter les autres types de fonctions.

2.2.1 Opérateurs de base

L'analyse par intervalles nécessite de redéfinir l'arithmétique classique et la façon d'évaluer les fonctions. 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 opérations de base doivent être définies de sorte à ce que l'on puisse garantir que la nouvelle variable est dans l'intervalle résultant. Ainsi, il faut toujours prendre en compte le pire des cas possibles. En conséquence, les opérateurs de base sont définis comme suit :

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

$$\begin{cases} \underline{z} &= \underline{x} + \underline{y} \\ \overline{z} &= \overline{x} + \overline{y} \end{cases} \tag{2.3}$$

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

$$\begin{cases} \underline{z} &= \underline{x} - \overline{y} \\ \overline{z} &= \overline{x} - \underline{y} \end{cases} \tag{2.4}$$

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

$$\begin{cases} \begin{cases} \underline{z} &= \underline{x}.\underline{y} \\ \overline{z} &= \overline{x}.\underline{y} \end{cases} & \text{si} \quad \underline{x} > 0 \quad \text{et} \quad \underline{y} > 0 \\ \begin{cases} \underline{z} &= \overline{x}.\underline{y} \\ \overline{z} &= \overline{x}.\overline{y} \end{cases} & \text{si} \quad \underline{x} > 0 \quad \text{et} \quad \underline{y} < 0 \quad \text{et} \quad \overline{y} > 0 \\ \{ \quad \dots \end{cases} \tag{2.5}$$

2. Voir les chapitres suivants où nous décrivons en détail les fonctions utilisées.

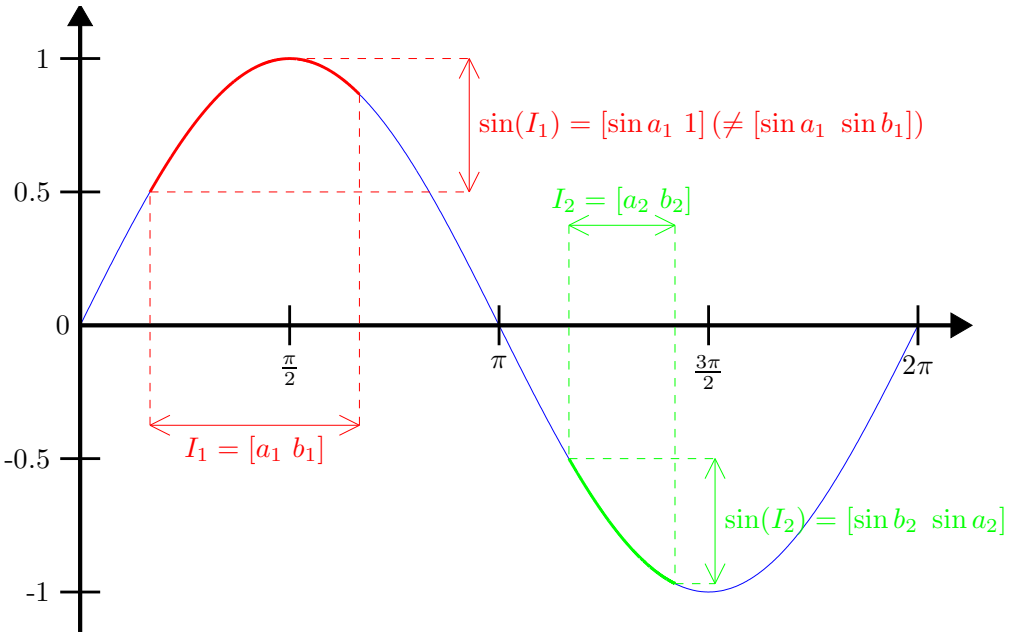


FIGURE 2.2 – Evaluation de la fonction *sinus* avec des intervalles – L'évaluation de la fonction *sinus* avec des intervalles dépend de la localisation de l'intervalle. Si ce dernier contient une des bosses de la fonction, l'intervalle final n'est pas donné directement par l'application de la fonction aux bornes de l'intervalle (cas de l'intervalle I_1 représenté en rouge). Si l'intervalle de départ est dans une partie monotone de la fonction, il suffit d'appliquer la fonction aux bornes de l'intervalle (cas de l'intervalle I_2 en vert).

La multiplication peut finalement se résumer par :

$$\left\{ \begin{array}{rcl} \underline{z} & = & \min\left(\underline{x}.\underline{y}, \underline{x}.\overline{y}, \overline{x}.\underline{y}, \overline{x}.\overline{y}\right) \\ \overline{z} & = & \max\left(\underline{x}.\underline{y}, \underline{x}.\overline{y}, \overline{x}.\underline{y}, \overline{x}.\overline{y}\right) \end{array} \right. \tag{2.6}$$

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

$$\left\{ \begin{array}{rcll} \underline{z} & = & \min\left(\underline{x}/\underline{y}, \underline{x}/\overline{y}, \overline{x}/\underline{y}, \overline{x}/\overline{y}\right) & \text{si} \quad \underline{y}.\overline{y} > 0 \\ \overline{z} & = & \max\left(\underline{x}/\underline{y}, \underline{x}/\overline{y}, \overline{x}/\underline{y}, \overline{x}/\overline{y}\right) & \\ \underline{z} & = & -\infty & \text{sinon} \\ \overline{z} & = & +\infty & \end{array} \right. \tag{2.7}$$

2.2.2 Fonctions trigonométriques

Nous décrivons dans ce paragraphe le cas des fonctions trigonométriques *sinus*, *cosinus* et *tangente*. Soit $x \in \mathbb{R}$ et $I_x = [\underline{x}, \overline{x}] \subset \mathbb{R}$ un intervalle contenant x. L'évaluation des fonctions trigonométriques en I_x donne :

sinus : il s'agit d'une fonction non monotone et périodique dont l'évaluation avec des intervalles n'est pas triviale. On distingue 5 cas :

1. si la taille de I_x est plus grande que 2π, le résultat est l'intervalle $[-1, 1]$,
2. si I_x contient $\pi/2$ (modulo 2π) **et ne contient pas** $3\pi/2$ (modulo 2π), alors le résultat est l'intervalle $[\min(\sin \underline{x}, \sin \overline{x}), 1]$ (c'est le cas représenté en rouge sur la figure 2.2),
3. si I_x contient $3\pi/2$ (modulo 2π) **et ne contient pas** $\pi/2$ (modulo 2π), alors le résultat est l'intervalle $[-1, \max(\sin \underline{x}, \sin \overline{x})]$,
4. si I_x contient $\pi/2$ (modulo 2π) **et contient également** $3\pi/2$ (modulo 2π), alors le résultat est l'intervalle $[-1, 1]$,
5. si I_x ne contient pas $\pi/2$ (modulo π), alors le résultat est l'intervalle défini par $[\min(\sin \underline{x}, \sin \overline{x}), \max(\sin \underline{x}, \sin \overline{x})]$ (c'est le cas représenté en vert sur la figure 2.2) : on est dans un voisinage monotone de la fonction).

Dans le cas de la fonction *sinus*, l'évaluation avec des intervalles a nécessité une étude préalable des variations de la fonction.

cosinus : on distingue les mêmes cas qu'avec la fonction *sinus*, mais avec un déphasage de $\pi/2$.

tangente : la fonction tangente est périodique et croissante par morceaux. On distingue deux cas :

1. si I_x contient $\pi/2$ (modulo π) alors le résultat vaut $]-\infty, +\infty[$,
2. sinon le résultat vaut $[\tan \underline{x}, \tan \overline{x}]$.

2.2.3 Autres fonctions

Les autres fonctions usuelles (*exponentielle*, *logarithme*,...) peuvent être redéfinies en termes d'intervalles. Dans le cas des fonctions *exponentielle* et *logarithme*, l'utilisation d'intervalles est triviale en raison de la monotonie de ces fonctions. D'autres fonctions usuelles peuvent également être évaluées à l'aide d'une étude des variations. C'est notamment le cas de la fonction carré $x \mapsto x^2$. Une évaluation naïve consisterait à appliquer l'équation 2.5. Or, l'évaluation avec l'intervalle $[-2, 1]$ de la fonction carré avec l'équation 2.5 retourne $[-2, 4]$. Cette solution n'est pas optimale : elle ne prend pas en compte le fait qu'un carré est toujours positif (la solution optimale serait alors $[0, 4]$).

En conséquence, l'évaluation optimale avec des intervalles (dans le sens où l'intervalle résultant est le plus petit possible) nécessite l'étude des variations de la fonction considérée. Ceci est relativement facile à mettre en œuvre pour le cas des fonctions simples. Pour le cas des fonctions plus complexes ou des problèmes de grandes dimension, cette approche ne peut pas être utilisée. En conséquence, on décomposera ces fonctions sous la forme $f \circ g \circ h \circ ...$ de sorte à n'appliquer que des fonctions

élémentaires ou connues. Cette méthode d'évaluation ne permet pas de garantir l'optimalité des intervalles (l'exemple du paragraphe suivant l'illustrera pour un cas simple).

2.2.4 Importance de l'écriture formelle

Nous avons vu dans les deux paragraphes précédents les opérations de base avec des intervalles. 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 de la solution I_z calculée, de I_y et de l'utilisation de $x = z-y$. La " simple " structure de groupe associée à l'addition dans le cas des variables réelles est en effet perdue dans le cas de l'évaluation avec des intervalles :

- le seul élément neutre de l'addition est l'intervalle $[0, 0]$
- pour un intervalle donné $[\underline{x}, \overline{x}]$ non dégénéré (ie. $\underline{x} \neq \overline{x}$) , le seul élément qui permettrait de retrouver l'élément neutre d'après l'équation 2.3 est $[-\underline{x}, -\overline{x}]$. Or, $-\underline{x} > -\overline{x}$: $[-\underline{x}, -\overline{x}]$ n'est donc pas intervalle. Il n'existe donc pas d'opposé à un intervalle quelconque (avec l'addition) : la structure de groupe est perdue.

Ces changements concernant les propriétés classiques des opérateurs invitent à être prudent quant à l'écriture formelle des expressions. Ainsi, deux écritures équivalentes d'une même expression (au sens de l'évaluation dans $\mathbb{R}$) peuvent conduire à deux évaluations différentes dans le cas des intervalles. Considérons par exemple la fonction $(x, y) \mapsto xy - x$ que l'on va appliquer avec $(x, y) \in I_x \times I_y = [0, 1] \times [0, 1]$. Nous distinguons pour cet exemple deux cas :

1. On écrit la fonction $(x, y) \mapsto xy - x$. Dans ce cas, l'intervalle résultant de $(x, y) \mapsto xy$ 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 $(x, y) \mapsto x(y-1)$. Dans ce cas, l'intervalle résultant de $y \mapsto y-1$ est $[-1, 0]$. Celui du produit de $y-1$ 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 $I_x \times I_y = [-1, 1] \times [-1, 1]$, 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. [3]

2.3 Application au SLAM

Nous présentons dans cette section l'approche générale utilisée pour résoudre le problème du SLAM à l'aide des outils d'analyse par intervalles. La méthode employée est articulée autour de deux étapes :

1. En supposant que l'on possède une boîte pour l'ensemble des amers et l'ensemble de la trajectoire jusqu'à l'instant t, on calcule une boîte pour l'état du robot à l'instant $t+1$ ainsi qu'une nouvelle boîte pour les amers (étape locale),
2. Une propagation globale permet ensuite de remettre à jour la trajectoire.

Nous présentons dans un premier temps la méthode permettant de passer de l'instant t à l'instant $t+1$. Ensuite, nous présentons la méthode utilisée pour traiter l'ensemble de la trajectoire.

2.3.1 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 $t+1$ lorsqu'on possède un intervalle de ces variables à l'instant t. Nous supposons donc disposer des informations suivantes :

$\mathbf{I}_{\mathbf{x}_t}$ **:** une boîte contenant $\mathbf{x}_t$.

$\mathbf{I}_{\mathbf{u}_t}$ **:** une boîte contenant $\mathbf{u}_t$.

$\mathbf{I}_{\mathbf{m}}$ **:** une boîte contenant $\mathbf{m}$.

$\mathbf{I}_{\nu^{\mathbf{f}}_{t+1}}$ **:** une boîte contenant $\nu^{\mathbf{f}}_{t+1}$.

$\mathbf{I}_{\nu^{\mathbf{z}}_{t+1}}$ **:** une boîte contenant $\nu^{\mathbf{z}}_{t+1}$.

2.3.1.1 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 $\mathbf{f}$ aux intervalles $\mathbf{I}_{\mathbf{x}_t}$, $\mathbf{I}_{\mathbf{u}_t}$ et $\mathbf{I}_{\nu^{\mathbf{f}}_{t+1}}$ (cf. équation 2.1). 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 $\mathbf{I}_{\mathbf{x}_{t+1}}$ contenant $\mathbf{x}_{t+1}$.

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

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

2.3.1.2.1 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 $\mathbf{h}$ 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 du vecteur de mesures 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, la dimension du vecteur utilisé en entrée est supérieure à la dimension du vecteur de mesures). Une connaissance *a priori* de l'état du robot et des amers est indispensable.

Dans la suite de ce paragraphe, on notera $\tilde{\mathbf{x}}$ la concaténation des vecteurs $\mathbf{x}_{t+1}$ et $\mathbf{m}$. Par ailleurs, n désignera dans la suite la dimension de $\mathbf{x}_{t+1}$, l la dimension du vecteur $\mathbf{m}$ (on a donc $\dim(\tilde{\mathbf{x}}) = n+l$) et q la dimension du vecteur $\mathbf{z}_t$.

2.3.1.2.2 Hypothèses effectuées Pour résoudre ce problème, nous effectuons les hypothèses suivantes :

- Nous possédons 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 $\mathbf{h}$ permet d'écrire :

$$\boxed{\forall i \in [1 \ldots n+l] \; \exists \left(j \in [1 \ldots q] \text{ et } g_i^j : \mathbb{R}^{n+l} \mapsto \mathbb{R} \right) \quad \text{tq } \tilde{x}_i = g_i^j(z_j, \tilde{x}_1, \ldots, \tilde{x}_{i-1}, \tilde{x}_{i+1}, \ldots, \tilde{x}_{n+l})} \tag{2.8}$$

  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 2.8 peut être vue comme une " condition d'observabilité " des variables $\tilde{x}_i$.

2.3.1.2.3 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 $\mathbf{I_z}$ pour $\mathbf{z}$ et un intervalle *a priori* $\mathbf{I_{\tilde{x}}}$ 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. Soit $\{g_i^1, \dots, g_i^{\gamma_i}\}$ 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 $\tilde{x}_i$:

$$I_{\tilde{x}_i} \leftarrow \left(\bigcap_{j=1}^{\gamma_i} I_{\tilde{x}_i}^j \right) \bigcap I_{\tilde{x}_i} \tag{2.9}$$

Nous pouvons alors déduire séquentiellement un intervalle pour toutes les composantes de $\tilde{\mathbf{x}}$. Ceci nous donnera un nouvel intervalle pour le vecteur $\tilde{\mathbf{x}}$ (qui sera par définition contracté par rapport à l'intervalle *a priori*). 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 2.1.

Théorème 2.1 (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 2.1 :

▲

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,

Fixer $\epsilon_1, \ldots, \epsilon_{n+l}$ # Seuils pour tester la convergence des composantes
$conv_1, \ldots, conv_{n+l} \leftarrow 0$ # Variables pour tester si une composante a convergé
$conv \leftarrow 0$
Tant que $conv = 0$ **faire**
| $I_{\mathbf{z}} \leftarrow I_{\mathbf{z}} \cap \mathbf{h}(I_{\tilde{\mathbf{x}}})$ # Réduction de $\mathbf{z}$ dès que possible grâce à l'*a priori* de $\tilde{\mathbf{x}}$
| **Pour** $i = 1 \ldots n + l$ **faire**
| | $I_{\tilde{x}_i}^{ancien} \leftarrow I_{\tilde{x}_i}$
| | $I_{\tilde{x}_i} \leftarrow$ mise à jour par équation 2.9
| | $decr_i \leftarrow t(I_{\tilde{x}_i}^{ancien}) - t(I_{\tilde{x}_i})$ # $decr_i < 0$ désigne la différence entre la longueur
| | # du nouvel intervalle et la longueur de l'ancien
| | **Si** $decr_i < \epsilon_i$ **alors**
| | | $conv_i \leftarrow 1$
| | **Sinon**
| | | $conv_i \leftarrow 0$
| | **Fin si**
| **Fin pour**
| $conv \leftarrow \prod_{i=1}^{n+l} conv_i$
Fin tant que
Retourner $I_{\tilde{x}_1}, \ldots, I_{\tilde{x}_{n+l}}$

Algorithme 2.1 – Algorithme de résolution de $\mathbf{h}(\tilde{\mathbf{x}}) = \mathbf{z}$

elle est par définition positive.

La suite $t(I_{\tilde{x}}^i)[j]$ est donc convergente. En conséquence, elle vérifie la propriété suivante :

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

Ainsi, pour ϵ_i fixé, il existe un N_i tel que $\big|t(I_{\tilde{x}}^i)[N_i + 1] - t(I_{\tilde{x}}^i)[N_i]\big| < \epsilon_i$, 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.

■

2.3.1.2.4 Exemple de résolution de $\mathbf{h}(\tilde{\mathbf{x}}) = \mathbf{z}$:

Soit la fonction **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 **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 2.8 :

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

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

L'exécution de l'algorithme est présentée dans le tableau 2.1. 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 **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 :

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

2.3.2 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).

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					

TABLE 2.1 – Exemple de résolution de $\mathbf{h}(\tilde{\mathbf{x}}) = \mathbf{z}$

Finalement, le traitement global consiste dans un premier temps à appliquer l'algorithme local de l'instant initial jusqu'à l'instant final. Ensuite, l'opération est répétée en utilisant les nouveaux intervalles comme information *a priori*, et ce jusqu'à la convergence des intervalles.

Remarque 2.2 (Propagation inverse des mesures)

Lors de l'application de l'algorithme global, on est amené à itérer sur l'ensemble de la trajectoire. A la fin de la première itération globale, les contractions les plus importantes se font sur les derniers amers vus : ceux-ci ont en général une influence sur la fin de la trajectoire. En conséquence, il paraît intéressant d'appliquer l'algorithme local de l'instant final jusqu'à l'instant initial après la première itération. Ensuite, nous proposons de repropager depuis l'instant initial jusqu'à l'instant final. Cette alternance est effectuée jusqu'à convergence. Le fait d'alterner entre propagation classique et propagation inverse permet en pratique d'accélérer la convergence de l'algorithme global.

2.3.3 Résumé de la " philosophie " de l'algorithme

Nous avons présenté dans cette section les principes du SLAM par intervalles. Le principe de l'algorithme est de d'abord calculer une boîte *a priori* concernant la position courante du robot. Ensuite, l'utilisation des mesures sert à **contracter les intervalles** dans le but de réduire le plus possible leur taille. Enfin, l'algorithme ne prend pas en compte intrinsèquement les corrélations : une mise à jour " performante " de la position des amers peut contribuer à diminuer l'incertitude sur la position du robot aux instant précédents, ce qui peut par ailleurs permettre de réduire l'incertitude associée à d'autres amers. Pour pallier ce problème, nous ré-estimons l'ensemble des boîtes de la trajectoire. Finalement, une étape de *propagation inverse* permet d'accélérer ce processus.

2.4 Exemple d'application ([Jaulin, 2009a])

A notre connaissance, il existe peu d'approches de SLAM basées sur l'analyse par intervalles. On pourra surtout citer les travaux de Luc Jaulin pour le cas du SLAM appliqué à la localisation de mines et d'un robot sous-marin ([Jaulin, 2009a]). Dans cette publication, l'auteur considère un sous-marin dont les entrées de modèle sont fournies par une centrale inertielle. La fonction **f** correspond à l'intégration des mesures de la centrale inertielle. Les mines sont quant à elles détectées et mesurées par un **sonar** : il s'agit d'une information de type *range and bearing* qui permet d'initialiser l'état des amers avec une seule itération. Par ailleurs, le sous-marin est équipé d'un GPS qu'il utilise à certains instants de la mission (il s'agit des moments où il remonte à la surface).

Les méthodes de résolution utilisées dans [Jaulin, 2009a] sont très proches de celles décrites dans la section précédente. On retrouve toujours l'idée de contraction des intervalles. Le principe de propagation inverse des données est également appliqué.

Les résultats obtenus sont comparés avec ceux d'un filtre de Kalman étendu classique (cf. figure 2.3). Les résultats obtenus en termes d'ellipses (pour le cas gaussien, ie. l'EKF) et d'intervalle d'incertitudes à 100% sont très similaires. Néanmoins, le SLAM par intervalles permet de garantir qu'**aucune approximation n'est réalisée dans le calcul de la solution**. Ceci n'est pas le cas avec l'EKF. En revanche, les résultats obtenus ne peuvent être obtenus qu'hors-ligne, après contraction des intervalles sur l'ensemble de la trajectoire. C'est également le cas de l'approche que nous avons présentée.

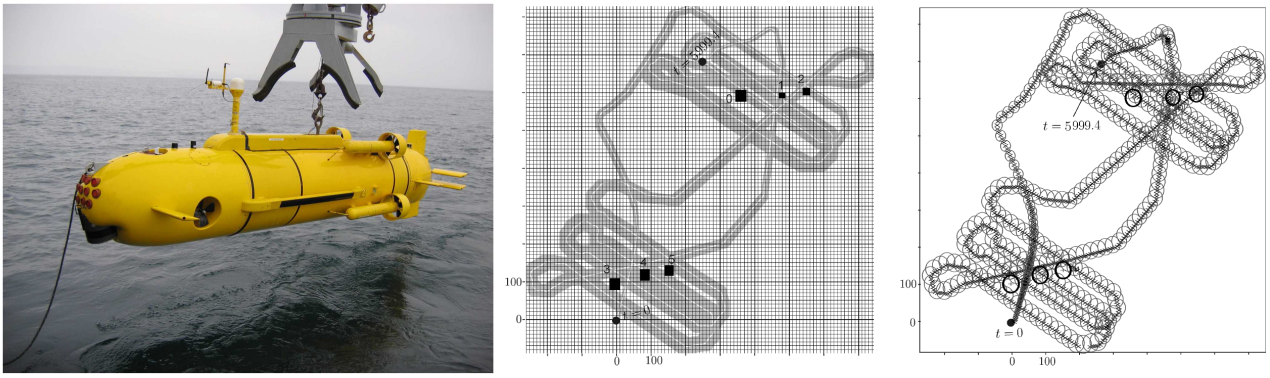


FIGURE 2.3 – SLAM par intervalles pour un sous-marin ([Jaulin, 2009a]) – A gauche : photo du sous-marin. Au milieu : résultat de l'analyse par intervalles. A droite : résultat du filtre de Kalman étendu.

2.5 Conclusion

Nous avons présenté dans ce chapitre une méthode générale de résolution du problème du SLAM utilisant des notions d'analyse par intervalles. Cette approche dispose de plusieurs avantages. Elle permet d'effectuer des preuves garantissant que le résultat obtenu contient de manière certaine la vraie solution. Contrairement aux méthodes probabilistes, aucune approximation n'est effectuée. En contrepartie, les méthodes d'évaluation des intervalles sont pessimistes : il s'agit de prendre en compte le pire des cas possibles. On s'attend donc naturellement à ce que les zones d'incertitudes soient beaucoup plus grandes avec ce type de méthode. Néanmoins, l'application montrée par Luc Jaulin dans [Jaulin, 2009a] montre qu'il est possible d'obtenir des résultats très satisfaisants en termes de taille des enveloppes d'incertitudes.

Le principal défaut de ces méthodes est que les preuves de consistance reposent sur une hypothèse très forte : **les erreurs sur les capteurs et le modèle d'évolution du robot ont des bornes connues par avance**. Il est en général difficile de garantir ce type de contrainte. Par exemple, la présence de mesures aberrantes peut provoquer des intersections nulles dans l'algorithme. Il existe néanmoins des possibilités de relâcher des contraintes lorsque ce type de cas se présente sous l'hypothèse que le nombre de mesures abérrantes est borné ([Jaulin, 2009b]). Dans ce cas, les preuves sur la consistance du résultat ne sont plus valables (car on est jamais sûr à 100% de relâcher les bonnes contraintes).

Finalement, les bons résultats obtenus dans [Jaulin, 2009a] sont obtenus dans un cas où le nombre de capteurs est important. En effet, on dispose d'une information de distance et d'orientation pour tous

les amers. De plus, le nombre d'amers est limité et leur signature dans le signal sonar est facilement reconnaissable et la mise en correspondance est faite " à la main " (ce qui permet d'éviter la présence de mises en correspondances aberrantes). Par ailleurs, la présence du GPS permet de limiter l'erreur sur la position du sous-marin à des endroits clés de la trajectoire (ce qui a pour effet de contraindre fortement l'ensemble de la trajectoire). Il conviendra donc d'étudier le comportement des méthodes par intervalles dans le cas du SLAM visuel sans GPS, pour lequel la redondance des mesures est nettement plus limitée. [4] C'est ce que nous proposons de faire dans le chapitre suivant.

4. Même si les outils théoriques sont les mêmes, il n'est plus du tout sûr que la qualité des résultats sera aussi bonne que celle de [Jaulin, 2009a].

Chapitre 3

Evaluation en simulation de l'algorithme de SAM et du SLAM par intervalles

Sommaire

Nous avons présenté dans les deux chapitres précédents un éventail de méthodes de résolution relativement large. Nous avons dans un premier temps présenté les principales méthodes probabilistes ainsi que leurs avantages et inconvénients. La méthode probabiliste qui nous a paru la plus apte à fournir des résultats **précis et consistants** est la méthode de SAM. Le chapitre précédent, quant à lui, était réservé à la présentation d'une méthode de SLAM utilisant des résultats d'analyse par intervalles. Ce type de méthode possède un atout conséquent : **si les erreurs de modèle et de mesures sont bornées et que les bornes sont connues et consistantes, alors le résultat sera toujours consistant**. Ainsi, ce type de méthode n'effectue aucune approximation (alors que la méthode de SAM effectue une linéarisation des équations, en admettant que le résultat suivra encore une loi gaussienne).

Nous proposons dans ce chapitre de comparer ces deux méthodes dans le cas précis du SLAM *bearing-only*. Les amers considérés seront des points dont on ne mesurera jamais directement la distance au robot. Il s'agit d'une classe de problèmes équivalente au SLAM visuel. Pour évaluer précisemment ces méthodes, nous décidons d'opérer en **simulation** sur un modèle d'évolution du robot " simple ". On aura ainsi une vérité terrain qui nous permettra de comparer objectivement les résultats. Par ailleurs, l'objectif de ce chapitre est d'évaluer deux méthodes de résolution de SLAM. Etant donné que les deux méthodes testées ne traitent pas le problème d'association des données, nous la supposerons connue d'avance et parfaite.

Ce chapitre est divisé en cinq sections. Nous décrivons dans un premier temps le cadre général de l'étude, à savoir les hypothèses sur les modèles utilisés et les équations communes aux deux méthodes d'estimation. Nous présentons ensuite les détails pratiques liés à l'utilisation de l'algorithme de SAM (section 3.2) puis ceux liés à l'algorithme de SLAM par intervalles (section 3.3). Le protocole complet des simulations ainsi que les résultats sont présentés dans la section 3.4. Nous concluons enfin notre analyse dans la section 3.5.

3.1 Cadre de l'étude

3.1.1 Introduction

Nous présentons dans cette section le cadre utilisé pour la comparaison des deux algorithmes. Il s'agit de celui du SLAM *bearing-only* : seuls les angles relatifs entre le robot et les amers sont mesurés (cf. figure 3.1). Cette problématique est très similaire à celle du SLAM visuel (typiquement, les donées en pixels fournies par les images peuvent être converties en mesures angulaires) : les résultats obtenus

dans ce chapitre sont donc pertinents dans l'optique d'une application au SLAM visuel.

Cette étude est réalisée dans le cas où **le robot se déplace dans un plan** et où **les amers sont des points 3D**. Dans la suite, nous désignons ce problème *SLAM 2D/3D*. L'état du robot est donné par les 3 composantes de sa pose : 2 pour la position et une pour l'orientation. Nous supposons que le robot est controlé en vitesses de translation et de rotation (exprimées dans le repère du robot). Par ailleurs, nous supposons qu'il existe une contrainte de **non-honolomie** sur le déplacement du robot (cf. paragraphe 3.1.2) : la vitesse instantanée de translation est alignée avec l'axe principal du robot (axe x sur la figure 3.1). Il s'agit d'une contrainte classique des robots à roues controlés par un modèle de type " char ".

Les deux méthodes comparées ne font pas les mêmes hypothèses concernant la nature des bruits. Néanmoins, on retrouve une partie commune dans la formulation des équations. En effet, les deux méthodes font intervenir la même représentation d'état et équations de modèle et de mesures $\mathbf{f}$ et $\mathbf{h}$:

$$\left\{ \begin{array}{rcl} \mathbf{x}_t & = & \mathbf{f}(\mathbf{x}_{t-1}, \mathbf{u}_t) + \nu_t^{\mathbf{f}} \\ \mathbf{z}_t & = & \mathbf{h}(\mathbf{x}_t, \mathbf{m}) + \nu_t^{\mathbf{z}} \end{array} \right. \tag{3.1}$$

La différence entre les deux approches se fait au niveau de la modélisation des vecteurs de bruits : $\nu_t^{\mathbf{f}}$ et $\nu_t^{\mathbf{z}}$ suivent des lois gaussiennes dans le cas du SAM alors que ces vecteurs sont bornés dans le cas de l'analyse par intervalles.

La suite de cette section contient 3 paragraphes. Nous décrivons dans un premier temps l'ensemble de la paramétrisation. Les deux paragraphes suivants sont consacrés à la description des fonctions $\mathbf{f}$ et $\mathbf{h}$.

3.1.2 Paramétrisation

Nous explicitons dans ce paragraphe les vecteurs $\mathbf{x}_t$, $\mathbf{u}_t$ et $\mathbf{z}_t$ utilisés. Ceux-ci font référence à la figure 3.1 page 87 :

$\mathbf{x}_t$: il s'agit de la pose du robot à l'instant t, à savoir la position du robot dans le plan (coordonnées absolues 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 identique au repère initial du robot.[1] On a :

$$\mathbf{x}_t = \left[\begin{array}{ccc} x_t & y_t & \theta_t \end{array} \right]^T \tag{3.2}$$

$\mathbf{u}_t$: 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

1. L'état du robot est donc initialisé à zéro, avec une incertitude nulle (cf. section 1.3).

à chaque instant t est dans le plan du véhicule. Il est repéré par ses deux composantes que nous exprimons 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 :

$$\mathbf{u}_t = \begin{bmatrix} V_t^x & V_t^y & \omega_t \end{bmatrix}^T \tag{3.3}$$

Nous supposons qu'une mesure de ce vecteur de commande est fournie par l'**odométrie du robot**, par le biais d'une vitesse de translation axiale (supposée être égale à V_t^x) et d'une vitesse de rotation (supposée être égale à ω_t). Enfin, la contrainte de non honolomie mentionnée au paragraphe précédent indique que la vitesse V_t^y est fixée à zéro : **localement, la trajectoire du robot est un arc de cercle tangent à l'axe x du robot ou une ligne droite le long de cet axe**. La contrainte de non honolomie $V_t^y = 0$ constitue une mesure virtuelle.

$\mathbf{m}_{(i)}$: 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 :

$$\mathbf{m}_{(i)} = \begin{bmatrix} x_{(i)} & y_{(i)} & z_{(i)} \end{bmatrix}^T \tag{3.4}$$

$\mathbf{z}_t$: 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 :

$$\mathbf{z}_{t,(i)} = \begin{bmatrix} \alpha_{t,(i)} & \beta_{t,(i)} \end{bmatrix}^T \tag{3.5}$$

Remarque 3.1 (Paramétrisation des amers)

Dans ce chapitre, les amers sont paramétrés par leurs coordonnées cartésiennes 3D. Nous verrons que ceci implique d'initialiser les amers après plusieurs itérations. Il existe d'autres méthodes de représentation. On pourra par exemple citer la représentation utilisant l'inverse de la profondeur des amers ([Civera et al.*, 2008]) qui permet (sous certaines hypothèses) d'initialiser les amers dès le premier instant où ils sont vus. Néanmoins, cette représentation présente des inconvénients ; elle implique notamment d'augmenter la taille du vecteur d'état associé aux amers. Entre outre, chaque amer est exprimé par rapport à une position de référence différente, ce qui rend l'interprétation des résultats délicate (notamment pour le cas de la représentation par intervalles). Le but de cet chapitre est de*

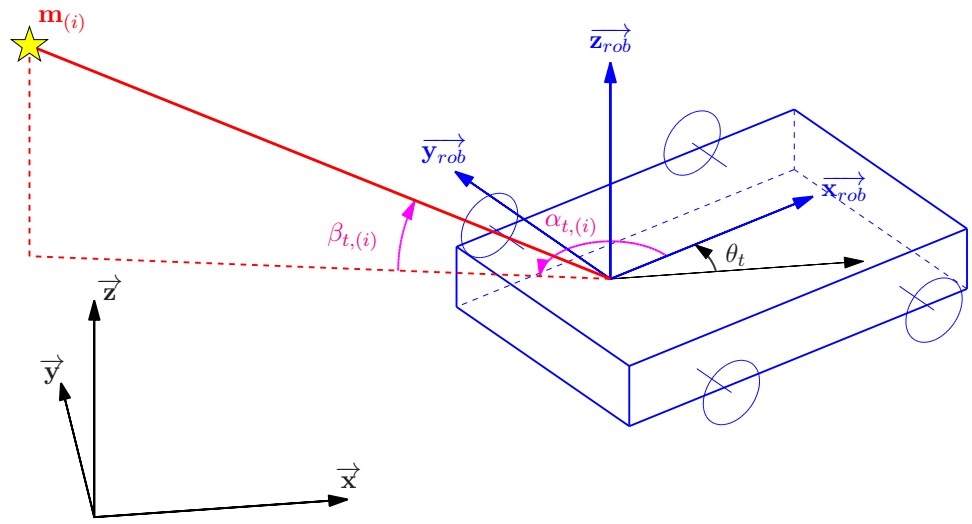


FIGURE 3.1 – Notations du SLAM 2D/3D

comparer les deux méthodes d'estimation. Dans ce but, il paraît souhaitable d'utiliser les modèles les plus " simples " possibles.

3.1.3 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 3.6 :

$$\left\{ \begin{array}{rcl} \dot{x} & = & V^x \cos\theta - V^y \sin\theta \\ \dot{y} & = & V^x \sin\theta + V^y \cos\theta \\ \dot{\theta} & = & \omega \end{array} \right. \tag{3.6}$$

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 3.6 nous donne la fonction $\mathbf{f}$:

$$\left\{ \begin{array}{rcl} x_{t+1} & = & x_t + \mathrm{sinc}\left(\frac{\omega_t dt}{2}\right)\left[V_t^x dt \cos\left(\theta_t + \frac{\omega_t dt}{2}\right) - V_t^y dt \sin\left(\theta_t + \frac{\omega_t dt}{2}\right)\right] \\ y_{t+1} & = & y_t + \mathrm{sinc}\left(\frac{\omega_t dt}{2}\right)\left[V_t^x dt \sin\left(\theta_t + \frac{\omega_t dt}{2}\right) + V_t^y dt \cos\left(\theta_t + \frac{\omega_t dt}{2}\right)\right] \\ \theta_{t+1} & = & \theta_t + \omega_t dt \end{array} \right. \tag{3.7}$$

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

Deux sources d'erreurs interviennent dans le système 3.7 :

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 3.1).

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 3.7 peut être écrite de la façon suivante :

$$\left\{ \begin{array}{lcl} x_{t+1} & = & x_t + \mathrm{sinc}\left(\frac{d\omega_t}{2}\right)\left[ds^x_t \cos\left(\theta_t + \frac{d\omega_t}{2}\right) - ds^y_t \sin\left(\theta_t + \frac{d\omega_t}{2}\right)\right] \\ y_{t+1} & = & y_t + \mathrm{sinc}\left(\frac{d\omega_t}{2}\right)\left[ds^x_t \sin\left(\theta_t + \frac{d\omega_t}{2}\right) + ds^y_t \cos\left(\theta_t + \frac{d\omega_t}{2}\right)\right] \\ \theta_{t+1} & = & \theta_t + d\omega_t \end{array} \right. \tag{3.8}$$

Remarque 3.2

Cette équation de modèle fait appel à deux entrées mesurées (ds^x_t et $d\omega_t$) et une entrée virtuelle (ds^y_t fixée à zéro).

3.1.4 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 $\mathbf{h}$) :

$$\left\{ \begin{array}{lcl} \alpha_{t,(i)} & = & \mathrm{arctan2}\left(\begin{array}{ccc} y_{(i)} - y_t & , & x_{(i)} - x_t \end{array} \right) - \theta_t \\ \beta_{t,(i)} & = & \arctan\left(\frac{z_{(i)}}{\sqrt{(x_{(i)}-x_t)^2 + (y_{(i)}-y_t)^2}}\right) \end{array} \right. \tag{3.9}$$

Remarque 3.3 (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 (sauf aux deux pôles de la sphère).

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.

3.2 Mise en application du SAM

Nous présentons dans cette section l'application du SAM au cas du SLAM 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.

3.2.1 Matrices jacobiennes des fonctions f et h

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

3.2.1.1 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 :

$$\mathbf{F}_{t+1} = \begin{bmatrix} 1 & 0 & -ds_t^x \text{sinc}\left(\frac{d\omega_t}{2}\right) \sin\left(\theta_t + \frac{d\omega_t}{2}\right) \\ 0 & 1 & ds_t^x \text{sinc}\left(\frac{d\omega_t}{2}\right) \cos\left(\theta_t + \frac{d\omega_t}{2}\right) \\ 0 & 0 & 1 \end{bmatrix} \tag{3.10}$$

Remarque 3.4 (Absence de ds_t^y)
Le terme ds_t^y n'apparaît pas dans l'équation 3.10 car il a été préalablement remplacé par zéro.

3.2.1.2 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 $\mathbf{H}^{\mathbf{x}}_{t,(i)}$:

$$\mathbf{H}^{\mathbf{x}}_{t,(i)} = \begin{bmatrix} \frac{y_{(i)}-y_t}{(x_{(i)}-x_t)^2+(y_{(i)}-y_t)^2} & \frac{-(x_{(i)}-x_t)}{(x_{(i)}-x_t)^2+(y_{(i)}-y_t)^2} & -1 \\ \mathbf{H}^{\mathbf{x}}_{t,(i)}(2,1) & \mathbf{H}^{\mathbf{x}}_{t,(i)}(2,2) & 0 \end{bmatrix} \tag{3.11}$$

avec :

$$\begin{cases} \mathbf{H}^{\mathbf{x}}_{t,(i)}(2,1) &= \frac{z_{(i)}(x_{(i)}-x_t)}{\sqrt{(x_{(i)}-x_t)^2+(y_{(i)}-y_t)^2}\left((x_{(i)}-x_t)^2+(y_{(i)}-y_t)^2+z_{(i)}^2\right)} \\ \mathbf{H}^{\mathbf{x}}_{t,(i)}(2,2) &= \frac{z_{(i)}(y_{(i)}-y_t)}{\sqrt{(x_{(i)}-x_t)^2+(y_{(i)}-y_t)^2}\left((x_{(i)}-x_t)^2+(y_{(i)}-y_t)^2+z_{(i)}^2\right)} \end{cases} \tag{3.12}$$

3.2.1.3 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 $\mathbf{H}^{\mathbf{m}}_{t,(i)}$:

$$\mathbf{H}^{\mathbf{m}}_{t,(i)} = \begin{bmatrix} \frac{-(y_{(i)}-y_t)}{(x_{(i)}-x_t)^2+(y_{(i)}-y_t)^2} & \frac{x_{(i)}-x_t}{(x_{(i)}-x_t)^2+(y_{(i)}-y_t)^2} & 0 \\ \mathbf{H}^{\mathbf{m}}_{t,(i)}(2,1) & \mathbf{H}^{\mathbf{m}}_{t,(i)}(2,2) & \frac{\sqrt{(x_{(i)}-x_t)^2+(y_{(i)}-y_t)^2}}{(x_{(i)}-x_t)^2+(y_{(i)}-y_t)^2+z_{(i)}^2} \end{bmatrix} \tag{3.13}$$

avec :

$$\left\{ \begin{array}{rcl} \mathbf{H^m_{t,(i)}}(2,1) & = & -\mathbf{H^x_{t,(i)}}(2,1) \\ \mathbf{H^m_{t,(i)}}(2,2) & = & -\mathbf{H^x_{t,(i)}}(2,2) \end{array} \right. \tag{3.14}$$

3.2.2 Matrices de variances-covariances utilisées

Dans le cadre du SAM, toutes les erreurs sont supposées être gaussiennes, centrées et de matrice de variances-covariances connues. 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 :

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

3.2.2.1 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 $t+1$ n'est pas 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. Nous supposons que la fonction de modèle $\mathbf{f}$ est utilisée avec un vecteur d'entrées aléatoires. En effet, 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 : [2]

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

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

2. 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).

Dans l'algorithme général de SAM, on suppose 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 :

$$\mathbf{Q}_{t+1} = \mathbf{J}^{\mathbf{u}}_t \mathbf{Q}^{\mathbf{u}} \mathbf{J}^{\mathbf{u}T}_t + \mathbf{Q}^{\mathbf{f}} \tag{3.15}$$

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

$$\mathbf{J}^{\mathbf{u}}_t = \begin{bmatrix} \text{sinc}\left(\frac{d\omega_t}{2}\right)\cos\left(\theta_t + \frac{d\omega_t}{2}\right) & -\text{sinc}\left(\frac{d\omega_t}{2}\right)\sin\left(\theta_t + \frac{d\omega_t}{2}\right) & \mathbf{J}^{\mathbf{u}}_t(1,3) \\ \text{sinc}\left(\frac{d\omega_t}{2}\right)\sin\left(\theta_t + \frac{d\omega_t}{2}\right) & \text{sinc}\left(\frac{d\omega_t}{2}\right)\cos\left(\theta_t + \frac{d\omega_t}{2}\right) & \mathbf{J}^{\mathbf{u}}_t(2,3) \\ 0 & 0 & 1 \end{bmatrix} \tag{3.16}$$

avec :

$$\begin{cases} \mathbf{J}^{\mathbf{u}}_t(1,3) &= \frac{2ds^x_t}{d\omega^2_t}\cos\left(\theta_t + \frac{d\omega_t}{2}\right)\cdot\left(\frac{d\omega_t}{2}\cos\left(\frac{d\omega_t}{2}\right) - \sin\left(\frac{d\omega_t}{2}\right)\right) - \frac{ds^x_t}{2}\text{sinc}\left(\frac{d\omega_t}{2}\right)\cdot\sin\left(\theta_t + \frac{d\omega}{2}\right) \\ \mathbf{J}^{\mathbf{u}}_t(2,3) &= \frac{2ds^x_t}{d\omega^2_t}\sin\left(\theta_t + \frac{d\omega_t}{2}\right)\cdot\left(\frac{d\omega_t}{2}\cos\left(\frac{d\omega_t}{2}\right) - \sin\left(\frac{d\omega_t}{2}\right)\right) + \frac{ds^x_t}{2}\text{sinc}\left(\frac{d\omega_t}{2}\right)\cdot\cos\left(\theta_t + \frac{d\omega}{2}\right) \end{cases} \tag{3.17}$$

3.2.2.2 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 :

$$\forall i \in [1 \dots N] \quad \mathbf{R}_{t,(i)} = \begin{bmatrix} \sigma^2_\alpha & 0 \\ 0 & \sigma^2_\beta \end{bmatrix} \tag{3.18}$$

Remarque 3.5 (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 3.4, les erreurs sur les angles de gisement et d'élévation sont générées indépendamment.
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).

3.2.3 Initialisation

L'algorithme de SAM 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.

3.2.3.1 Initialisation de la trajectoire

L'initialisation de la trajectoire peut se faire grâce à la fonction de modèle. Admettons qu'une solution à l'instant t ait été calculée. Il est possible d'initialiser la trajectoire à $t+1$ en concaténant la trajectoire obtenue à l'instant t et l'état obtenu par la prédiction du modèle. Dans le cas de contraintes temps-réel, il est envisageable d'intialiser des " blocs " de trajectoire. Dans ce cas, on concaténerait la solution obtenue à t avec un bout de trajectoire correspondant à l'utilisation du modèle en boucle ouverte entre $t+1$ et $t+m$ (ceci permet d'éviter la mise à jour de l'ensemble de la matrice d'information à tous les instants si cela n'est pas nécessaire). Il convient cependant que m soit suffisamment petit pour que l'utilisation de la fonction $\mathbf{f}$ en boucle ouverte soit valable.

3.2.3.2 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.

3.2.3.2.1 Equations de base

Supposons qu'un amer (i) soit observé aux instants t_1 et t_2. La première équation du système 3.9 (page 88) permet d'écrire :

$$\frac{y_{(i)} - y_k}{x_{(i)} - x_k} = \tan(\alpha_{k,(i)} + \theta_k) \qquad (k \in \{t_1, t_2\}) \tag{3.19}$$

soit :

$$y_{(i)} \cos(\alpha_{k,(i)}+\theta_k) - x_{(i)} \sin(\alpha_{k,(i)}+\theta_k) = y_k \cos(\alpha_{k,(i)}+\theta_k) - x_k \sin(\alpha_{k,(i)}+\theta_k) \qquad (k \in \{t_1, t_2\}) \tag{3.20}$$

La seconde équation du système 3.9 permet quant à elle d'écrire :

$$\frac{z_{(i)}}{\sqrt{(x_{(i)} - x_k)^2 + (y_{(i)} - y_k)^2}} = \tan \beta_{k,(i)} \qquad (k \in \{t_1, t_2\}) \tag{3.21}$$

soit :

$$z_{(i)} = \tan \beta_{k,(i)} \sqrt{(x_{(i)} - x_k)^2 + (y_{(i)} - y_k)^2} \qquad (k \in \{t_1, t_2\}) \tag{3.22}$$

L'utilisation de l'équation 3.20 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 3.22. 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 3.20.
2. Initialiser $z_{(i)}$ à l'aide de l'équation 3.22 si $x_{(i)}$ et $y_{(i)}$ ont été initialisé.

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

Pour initialiser $x_{(i)}$ et $y_{(i)}$, l'équation 3.20 est utilisée aux instants t_1 et t_2, ce qui nous donne :

$$\begin{bmatrix} \sin(\alpha_{t_1,(i)} + \theta_{t_1}) & -\cos(\alpha_{t_1,(i)} + \theta_{t_1}) \\ \sin(\alpha_{t_2,(i)} + \theta_{t_2}) & -\cos(\alpha_{t_2,(i)} + \theta_{t_2}) \end{bmatrix} \cdot \begin{bmatrix} x_{(i)} \\ y_{(i)} \end{bmatrix} = \begin{bmatrix} x_{t_1} \sin(\alpha_{t_1,(i)} + \theta_{t_1}) - y_{t_1} \cos(\alpha_{t_1,(i)} + \theta_{t_1}) \\ x_{t_2} \sin(\alpha_{t_2,(i)} + \theta_{t_2}) - y_{t_2} \cos(\alpha_{t_2,(i)} + \theta_{t_2}) \end{bmatrix} \tag{3.23}$$

En posant $\gamma_1 = \alpha_{t_1,(i)} + \theta_{t_1}$ et $\gamma_2 = \alpha_{t_2,(i)} + \theta_{t_2}$, l'équation 3.23 nous permet de déduire $x_{(i)}$ et $y_{(i)}$:

$$\begin{bmatrix} x_{(i)} \\ y_{(i)} \end{bmatrix} = \frac{1}{\sin(\gamma_2 - \gamma_1)} \begin{bmatrix} -\cos\gamma_2 & \cos\gamma_1 \\ -\sin\gamma_2 & \sin\gamma_1 \end{bmatrix} \cdot \begin{bmatrix} x_{t_1} \sin\gamma_1 - y_{t_1} \cos\gamma_1 \\ x_{t_2} \sin\gamma_2 - y_{t_2} \cos\gamma_2 \end{bmatrix} \tag{3.24}$$

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

L'équation 3.24 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 3.24. 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. 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 :

$$\frac{1}{\sin(\gamma_2-\gamma_1)} = \frac{1}{\sin(\gamma_2^{approx}-\gamma_1^{approx}+\delta_{\gamma_2}-\delta_{\gamma_1})} \tag{3.25}$$

$$= \frac{1}{\sin(\gamma_2^{approx}-\gamma_1^{approx}) + (\delta_{\gamma_2}-\delta_{\gamma_1})\cos(\gamma_2^{approx}-\gamma_1^{approx})} \tag{3.26}$$

$$= \frac{1}{\sin(\gamma_2^{approx}-\gamma_1^{approx})} \cdot \frac{1}{1+(\delta_{\gamma_2}-\delta_{\gamma_1})\cdot\cot(\gamma_2^{approx}-\gamma_1^{approx})} \tag{3.27}$$

D'après l'équation 3.27, l'erreur sur $\frac{1}{\sin(\gamma_2-\gamma_1)}$ sera linéaire en δ_{γ_1} et δ_{γ_2} *si et seulement si* :

$$|(\delta_{\gamma_2}-\delta_{\gamma_1})\cdot\cot(\gamma_2^{approx}-\gamma_1^{approx})| \ll 1 \tag{3.28}$$

L'équation 3.28 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 $\pi/2$). Par hypothèse, les erreurs sur θ_{t_1}, θ_{t_2}, $\alpha_{t_1,(i)}$ et $\alpha_{t_2,(i)}$ suivent une loi gaussienne multivariée et centrée. Une matrice de variances-covariances pour θ_{t_1}, θ_{t_2} peut être déduite de la matrice d'information globale du système. Les erreurs associées au vecteur $\alpha_{t_1,(i)}$ et $\alpha_{t_2,(i)}$ sont indépendantes et de variances connues. Il est donc possible de déduire une matrice de variances-covariances pour le vecteur $[\delta_{\gamma_2}\ \delta_{\gamma_1}]^T$; par ailleurs, ce vecteur est d'espérance nulle. Soit $\mathbf{\Sigma}^\gamma$ la matrice de variances-covariances associée à $[\delta_{\gamma_2}\ \delta_{\gamma_1}]^T$ définie par :

$$\mathbf{\Sigma}^\gamma = \begin{bmatrix} \sigma_{\gamma_1}^2 & \rho_\gamma\sigma_{\gamma_1}\sigma_{\gamma_2} \\ \rho_\gamma\sigma_{\gamma_1}\sigma_{\gamma_2} & \sigma_{\gamma_2}^2 \end{bmatrix} \tag{3.29}$$

où $\sigma_{\gamma_1}^2$ et $\sigma_{\gamma_1}^2$ désignent les variances associées à δ_{γ_1} et δ_{γ_2}, et ρ_γ le coefficient de corrélation. En conséquence, la variance associée au terme $\delta_{\gamma_2}-\delta_{\gamma_1}$ est de $\sigma_{\gamma_1}^2+\sigma_{\gamma_2}^2-2\rho_\gamma\sigma_{\gamma_1}\sigma_{\gamma_2}$. De plus, ce terme est d'espérance nulle. Ainsi, la variance calculée permet donc de donner une idée assez précise des valeurs extrêmes que peut prendre $|\delta_{\gamma_2}-\delta_{\gamma_1}|$. Finalement, on acceptera d'initialiser l'amer *si et seulement si* :

$$\boxed{\sqrt{\sigma_{\gamma_1}^2+\sigma_{\gamma_2}^2-2\rho_\gamma\sigma_{\gamma_1}\sigma_{\gamma_2}} < \tfrac{1}{5}\,|\tan(\gamma_2^{approx}-\gamma_1^{approx})|} \tag{3.30}$$

En pratique, on choisira comme points d'initialisation le premier instant où l'amer a été observé, et le premier point satisfaisant l'équation 3.30. Choisir systématiquement le premier point où l'amer a été observé permet en général de minimiser $\sigma^2_{\gamma_1}$ (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 3.30, l'équation 3.24 est appliquée. Il est même possible d'évaluer d'une façon réaliste la variance de l'erreur commise par cette initialisation.

3.2.3.2.4 Initialisation de $z_{(i)}$

Une fois que les paramètres $x_{(i)}$ et $y_{(i)}$ sont initialisés, l'équation 3.22 peut être utilisée à l'instant t_1 et/ou t_2 pour en déduire $z_{(i)}$.

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

L'application de l'équation 3.22 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* :

$$\boxed{\sigma_\beta < \tfrac{1}{5}\left|\cot\left(\beta^{approx}_{t_1,(i)}\right)\right| \qquad \text{ou} \qquad \sigma_\beta < \tfrac{1}{5}\left|\cot\left(\beta^{approx}_{t_2,(i)}\right)\right|} \tag{3.31}$$

L'équation 3.31 indique que l'amer doit être vu suffisament loin de la verticale pour que son altitude puisse être initialisée. Dans les cas réels (caméras perspectives et omnidirectionnelles), ceci est toujours garanti : le champ couvert par la caméra n'est pas suffisamment large pour contenir ces valeurs.

3.3 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 3.3.2 traite de l'initialisation des amers. Nous présentons dans le paragraphe 3.3.3 un problème particulier dû à l'orientation des axes du repère. Nous verrons en effet que les propriétés de la solution calculée par l'analyse par intervalles ne sont pas isotropes. Nous proposons ensuite une méthode pour prendre en compte cet aspect.

3.3.1 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.

3.3.1.1 Equation de modèle

A chaque itération, l'état *a priori* est obtenu en utilisant l'équation de modèle 3.8, 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 3.6 (Intervalles des erreurs constants)
Tout comme pour le cas du SAM, 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.

3.3.1.2 Equation de mesure

En ce qui concerne l'équation de mesures, nous utilisons le système d'équations 3.9 rappelé ci-après :

$$\left\{ \begin{array}{rcl} \alpha_{t,(i)} & = & \arctan2\left(\; y_{(i)} - y_t \quad , \quad x_{(i)} - x_t \; \right) - \theta_t \\ \beta_{t,(i)} & = & \arctan\left(\frac{z_{(i)}}{\sqrt{(x_{(i)}-x_t)^2+((y_{(i)}-y_t)^2}} \right) \end{array} \right. \tag{3.32}$$

3.3.1.2.1 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 3.32 :

$$\left\{\begin{array}{rcl} x_t & = & x_{(i)} - \left(y_{(i)} - y_t\right) \cdot \cot\left(\theta_t + \alpha_{t,(i)}\right) \\ y_t & = & y_{(i)} - \left(x_{(i)} - x_t\right) \cdot \tan\left(\theta_t + \alpha_{t,(i)}\right) \\ \theta_t & = & \text{atan2}\left(\ y_{(i)} - y_t \ \ , \ \ x_{(i)} - x_t \ \right) - \alpha_{t,(i)} \\ x_{(i)} & = & x_t + \left(y_{(i)} - y_t\right) \cdot \cot\left(\theta_t + \alpha_{t,(i)}\right) \\ y_{(i)} & = & y_t + \left(x_{(i)} - x_t\right) \cdot \tan\left(\theta_t + \alpha_{t,(i)}\right) \end{array}\right. \tag{3.33}$$

En appliquant le système 3.33 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).

3.3.1.2.2 Traitement de la seconde équation de mesure

Le traitement de la seconde équation du système 3.32 permet d'écrire :

$$\left\{\begin{array}{rcl} z_{(i)} & = & \tan\beta_{t,(i)} \sqrt{(x_{(i)} - x_t)^2 + ((y_{(i)} - y_t)^2} \\ \left(x_{(i)} - x_t\right)^2 & = & \left(\cot\beta_{t,(i)} z_{(i)}\right)^2 - \left(y_{(i)} - y_t\right)^2 \\ \left(y_{(i)} - y_t\right)^2 & = & \left(\cot\beta_{t,(i)} z_{(i)}\right)^2 - \left(x_{(i)} - x_t\right)^2 \end{array}\right. \tag{3.34}$$

La première équation du système 3.34 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 les commentaires du paragraphe 2.2.4.

3.3.2 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 3.33 et 3.34 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 au paragraphe 3.2.3.2. 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 3.24 rappelée ci-après :[3]

$$\begin{bmatrix} x_{(i)} \\ y_{(i)} \end{bmatrix} = \frac{1}{\sin(\gamma_2 - \gamma_1)} \begin{bmatrix} -\cos\gamma_2 & \cos\gamma_1 \\ -\sin\gamma_2 & \sin\gamma_1 \end{bmatrix} \cdot \begin{bmatrix} x_{t_1}\sin\gamma_1 - y_{t_1}\cos\gamma_1 \\ x_{t_2}\sin\gamma_2 - y_{t_2}\cos\gamma_2 \end{bmatrix} \tag{3.35}$$

Tout comme pour le cas du SAM, l'instant t_1 sera celui de la première observation de l'amer. L'instant t_2, quant à lui, sera le premier instant pour lequel l'intervalle de $\sin(\gamma_2 - \gamma_1)$ ne contient pas zéro. Ainsi, l'équation 3.35 peut être appliquée avec des intervalles et nous donnera alors une boîte consistante contenant la position réelle de l'amer (i).

Remarque 3.7 (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 3.33 et 3.34 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 3.34.

3.3.3 Choix de l'orientation des axes

Nous avons présenté dans le chapitre précédent l'application basique de l'algorithme de SLAM par intervalles. Nous y avons vu que l'utilisation d'intervalles ne permet pas nécessairement d'obtenir la solution déterministe la plus optimale (voir la figure 2.1 page 67 qui présente le cas d'un tetraèdre encadré par une boîte). De plus, il n'existe pas qu'une seule boîte englobante possible dans le cas de la figure 2.1. Néanmoins, l'analyse par intervalles impose que les axes de la boîte soient alignés avec le repère, ce qui réduit finalement le nombre de boîtes possibles à une. Nous proposons dans ce paragraphe de montrer que cette dernière contrainte peut conduire à des solutions très sous-optimales. Une solution pour améliorer ces solutions est proposée ensuite.

3.3.3.1 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.

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

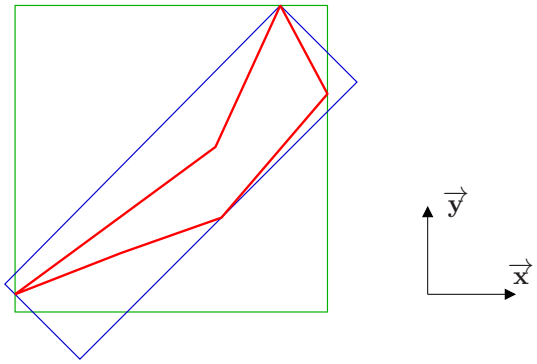


FIGURE 3.2 – Non optimalité de la représentation par rectangles — En rouge : un polytope contenant 100% des solutions possibles et supposé optimal — En vert : le plus petit rectangle aligné selon les axes du repère et contenant le polytope — En bleu : le plus petit rectangle tourné à 45 degrés et contenant le polytope

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 3.2). 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 3.2 (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 3.2.

Ainsi, la figure 3.2 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.

3.3.3.2 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

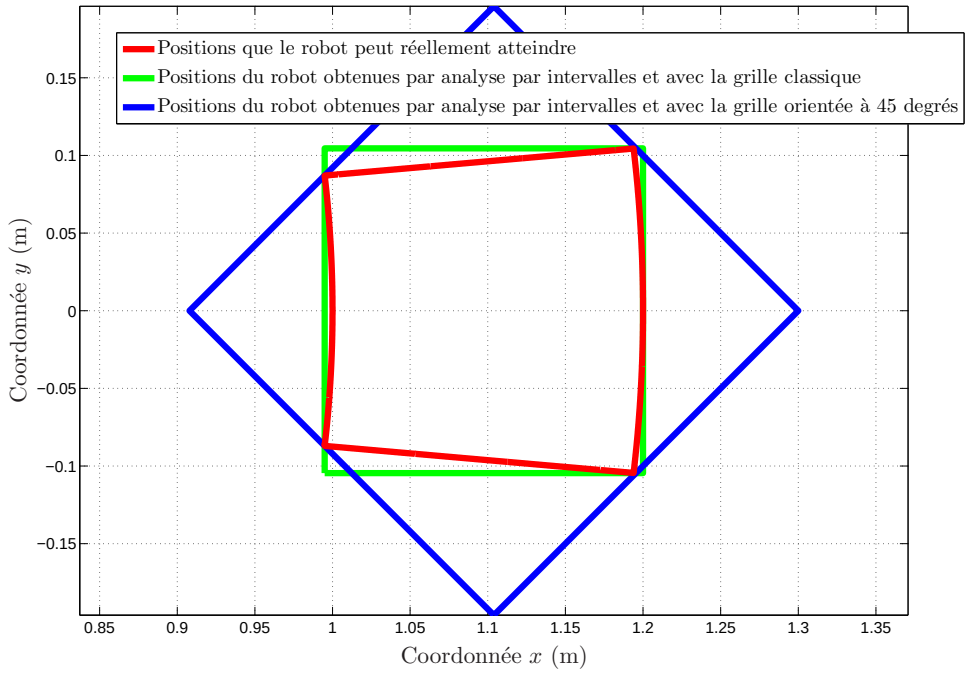


FIGURE 3.3 – Importance de l'orientation des rectangles représentant la position du robot

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 3.3) :

- 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.
- 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 3.3 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 termes d'aire du rectangle (0.044m^2 pour le rectangle vert contre 0.077m^2 pour le

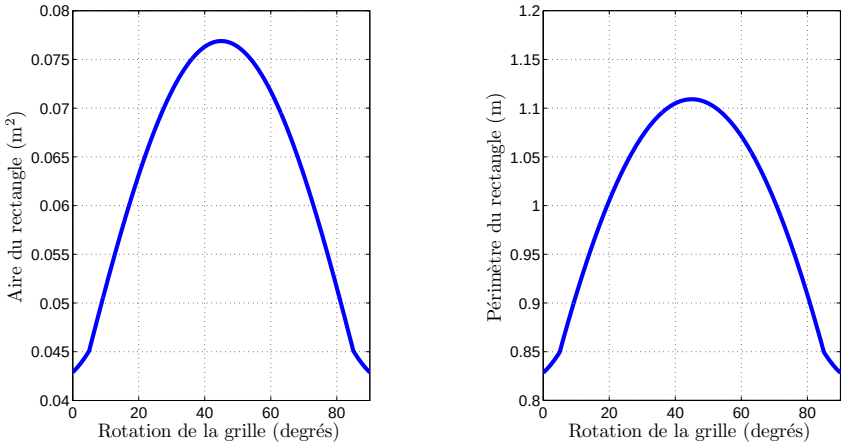


FIGURE 3.4 – Caractéristiques des rectangles obtenus en fonction de l'angle de rotation des rectangles — A gauche : les aires — A droite : les périmètres

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 $d\omega_1$

$$\left\{ \begin{array}{rcl} x_1 & = & ds_1^x \cdot \text{sinc}\left(\frac{d\omega_1}{2}\right) \cdot \cos\left(\theta_0 + \frac{d\omega_1}{2}\right) \\ y_1 & = & ds_1^x \cdot \text{sinc}\left(\frac{d\omega_1}{2}\right) \cdot \sin\left(\theta_0 + \frac{d\omega_1}{2}\right) \end{array} \right. \tag{3.36}$$

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 3.4). Les courbes de la figure 3.4 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 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 3.8 (Plage de valeurs considérée 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 $\theta_0 \in [0, \pi/2]$.

3.3.3.3 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 3.33, rappelée ci-après :

$$x_t = x_{(i)} - \left(y_{(i)} - y_t\right) \cdot \cot\left(\theta_t + \alpha_{t,(i)}\right) \tag{3.37}$$

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.

3.3.3.4 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 nécessite une étude analytique basée sur une disjonction des cas. Le problème du SLAM par intervalles a une dimension d'état trop importante pour envisager ce type de méthode. 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 prendre en compte cet aspect, 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 boîtes.

3. A la fin de la seconde application de l'algorithme, nous comparons les aires des nouvelles positions du robot 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.

3.3.3.5 Etude du gain apporté sur une simulation

Nous proposons dans ce paragraphe de présenter le gain obtenu en appliquant l'algorithme présenté au paragraphe précédent. 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 parfaitement respectée
- Erreur sur la vitesse mesurée : [4] 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 3.5 à 3.10. 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 caractère circulaire de la trajectoire permet de bien mettre en évidence l'importance du paramètre " orientation des boîtes ".

Enfin, on pourra remarquer sur la figure 3.6 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

4. 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.

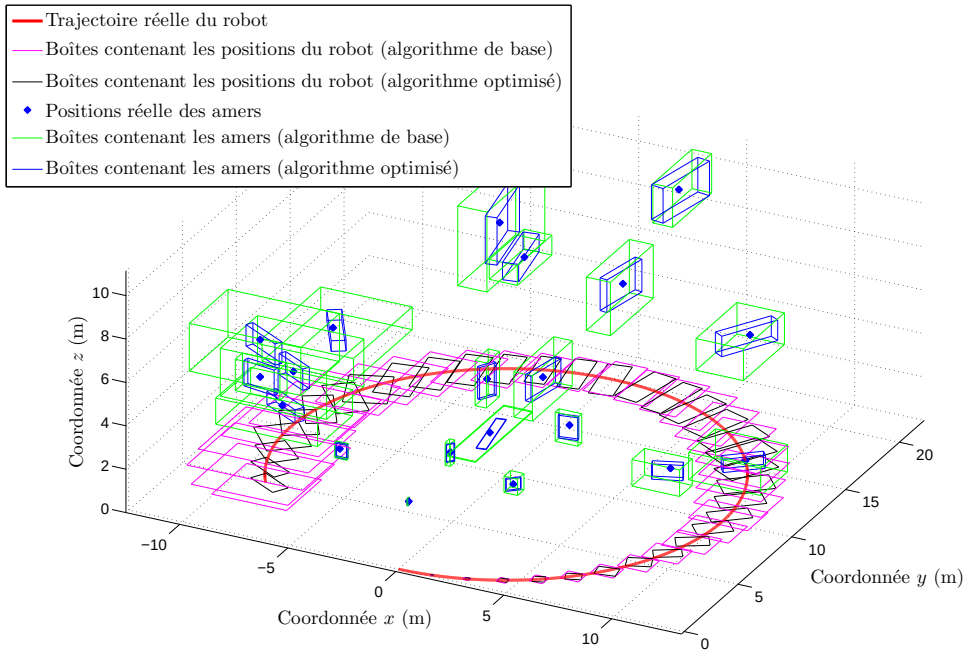


FIGURE 3.5 – Résultats globaux des deux algorithmes

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 vers cette position.

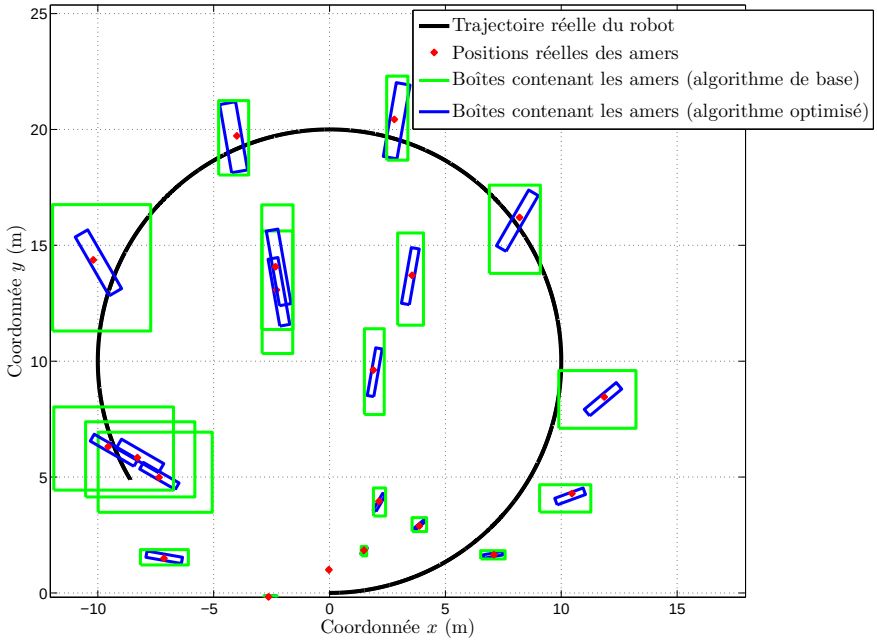


FIGURE 3.6 – Résultats des algorithmes concernant les amers

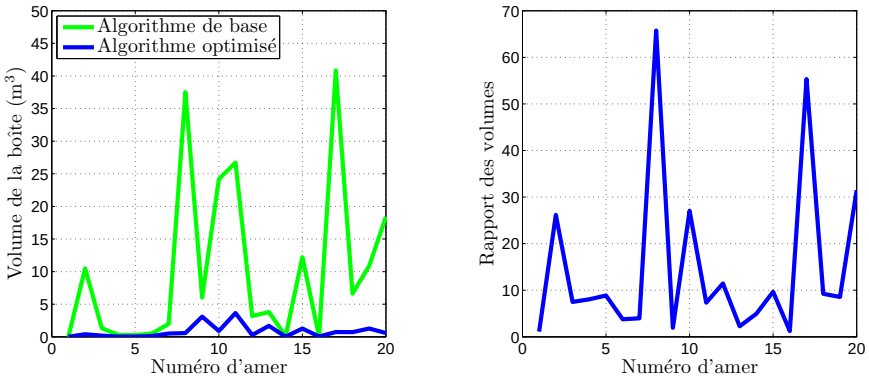


FIGURE 3.7 – A gauche : aires des rectangles obtenus pour les amers avec les deux algorithmes — A droite : rapport entre l'aire obtenue avec l'algorithme initial et l'aire obtenue avec l'algorithme optimisé (pour les amers)

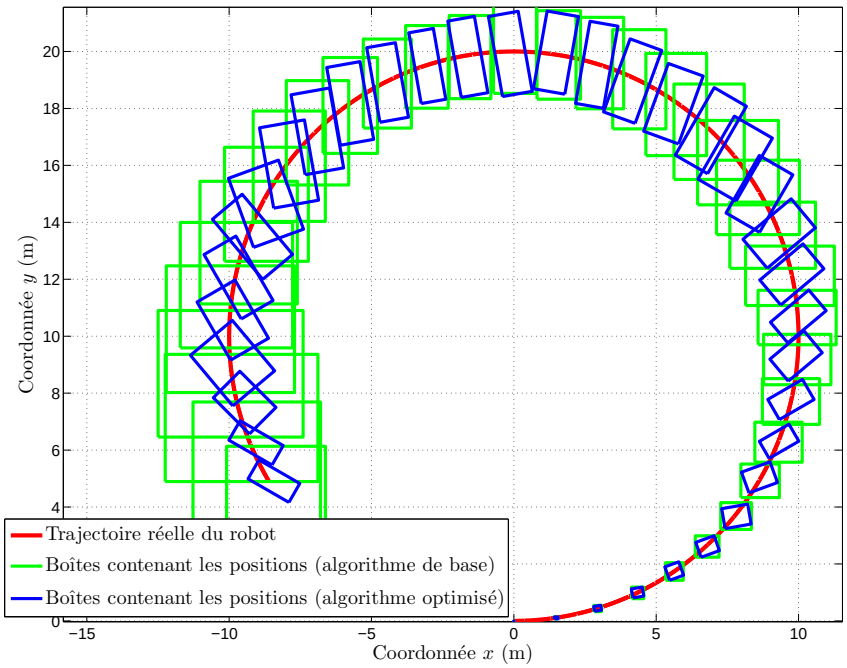


FIGURE 3.8 – Résultats des algorithmes concernant la position du robot (coordonnées x_t et y_t uniquement)

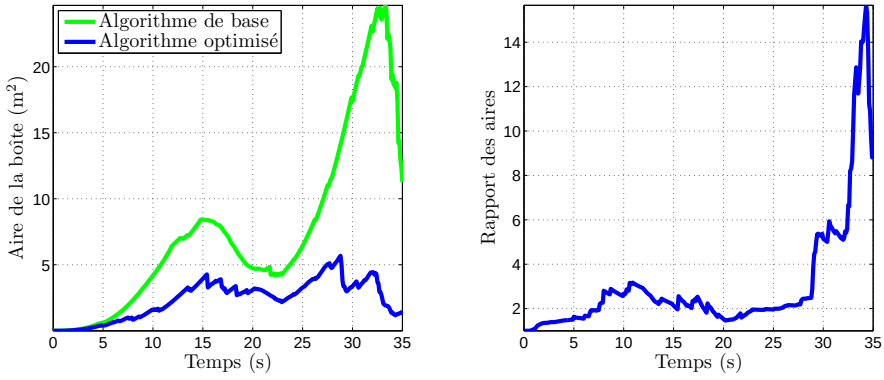


FIGURE 3.9 – A gauche : aires des rectangles obtenus pour les positions du robot avec les deux algorithmes — A droite : rapport entre l'aire obtenue avec l'algorithme initial et l'aire obtenue avec l'algorithme optimisé (pour les positions du robot)

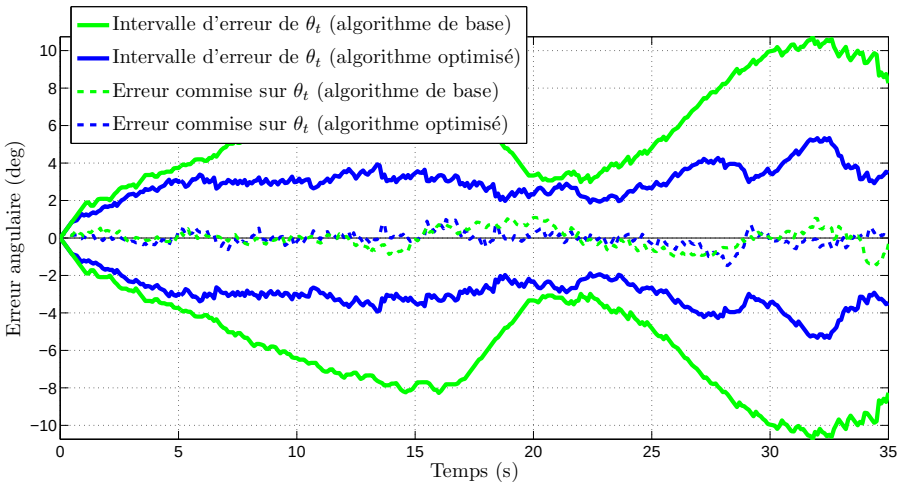


FIGURE 3.10 – Résultats des algorithmes concernant l'angle de cap du robot — Les traits pointillés représentent l'erreur d'estimation commise (différence entre le centre de l'intervalle et la solution réelle) – Les traits pleins représentent l'erreur maximale d'estimation prédite par l'algorithme (ie. ± la demi-longueur des intervalles) – Les traits pointillés sont systématiquement compris entre les traits pleins : les deux algorithmes sont donc consistants. On note néanmoins que l'algorithme optimisé fournit une enveloppe d'erreur plus petite.

3.4 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 3.4.2 les résultats obtenus pour le cas gaussien. Nous voyons ensuite les résultats obtenus lorsque les erreurs sont uniformes et centrées (paragraphe 3.4.3). Le paragraphe 3.4.5 présente les résultats obtenus lorsqu'on fait varier la visibilité des amers. Nous effectuons enfin quelques remarques sur la consistance du SAM dans le paragraphe 3.4.6.

3.4.1 Description des simulations

Nous présentons ici les caractéristiques communes à toutes les simulations.

3.4.1.1 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 représentées sur la figure 3.11.

3.4.1.2 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.

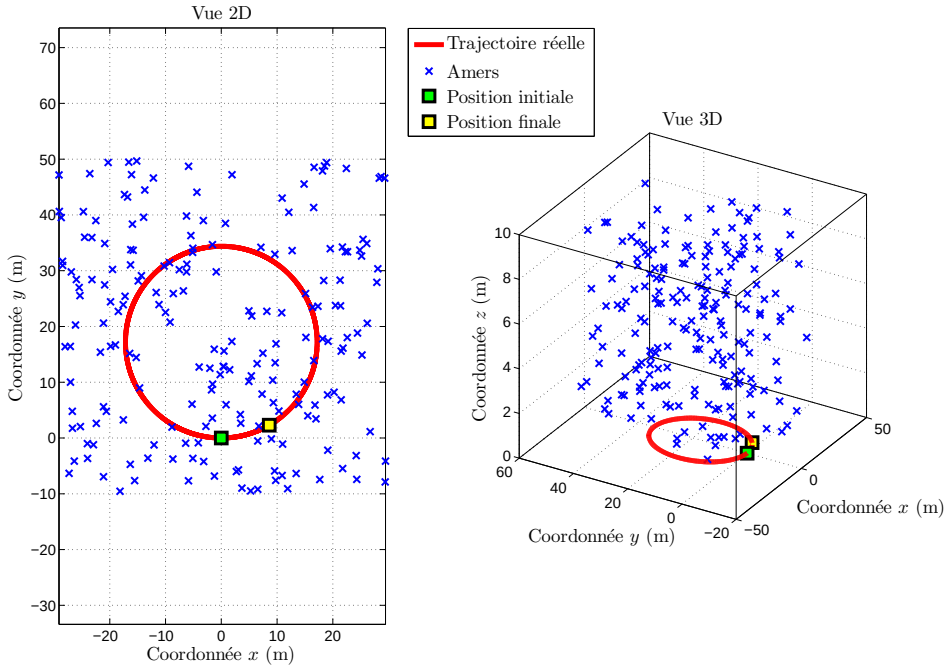


FIGURE 3.11 – Trajectoire réelle du robot et position réelle des amers — Le sens de rotation du robot est le sens trigonométrique — La position finale (carré jaune) est obtenue après un peu plus de deux tours

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

SAM : 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 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 3.4.5).

3.4.1.3 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 SAM, on s'attachera à vérifier si chaque position x, y (robot

à l'instant t et amers) est consistante en termes d'ellipse de confiance à 99% et si l'orientation l'est en termes 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 SAM, 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 C.

Remarque 3.9 (Limitations de l'étude concernant le SAM)
On remarquera en ce qui concerne le SAM 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 soi. D'ailleurs, la consistance des corrélations conditionne généralement la consistance de la solution finale.

3.4.2 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 SAM, et ensuite les résultats obtenus avec l'algorithme de SLAM par intervalles. Pour les deux cas, nous considérons les 4 scenarii présentés dans le tableau 3.1.

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

TABLE 3.1 – Scenarii pour le cas gaussien centré

3.4.2.1 SAM

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

Les résultats obtenus sont présentés sur les figures 3.12 à 3.16.

- La figure 3.12 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 3.13, 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.
- Les figures 3.14 et 3.15 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

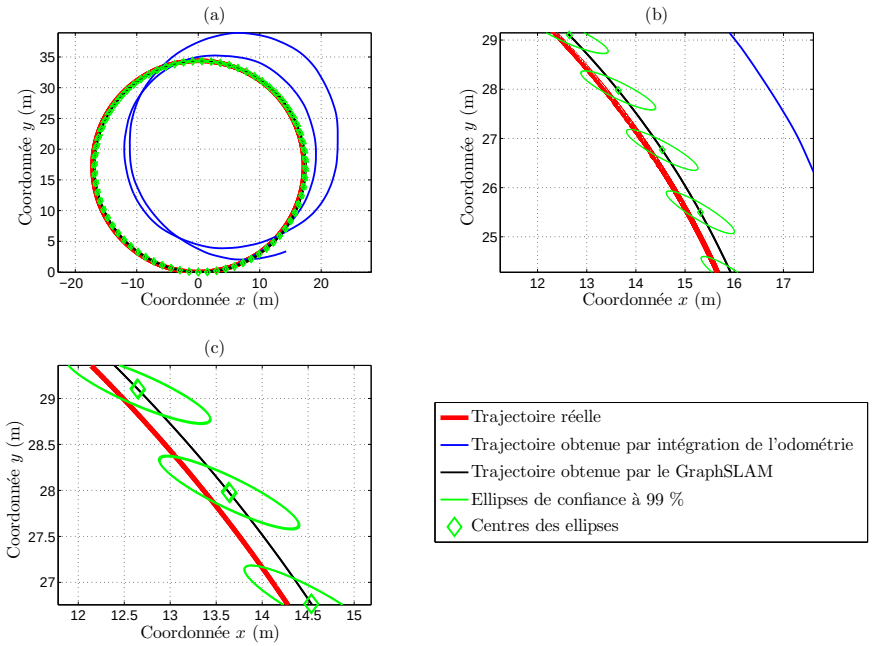


FIGURE 3.12 – Estimation des positions du robot obtenue avec le SAM sur le scénario 1 : (a) Vue globale — (b) Leger zoom — (c) Zoom important

faible (même si tous les amers sont vus en même temps et à toutes les positions).[5] Lorsqu'on s'éloigne du point de départ, on commence à accumuler des incertitudes. Même si l'algorithme de SAM 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 3.16 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 SAM 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 comportement de cet algorithme est donc très bon lorsque les hypothèses d'erreurs centrées et gaussiennes sont respectées.

5. 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.

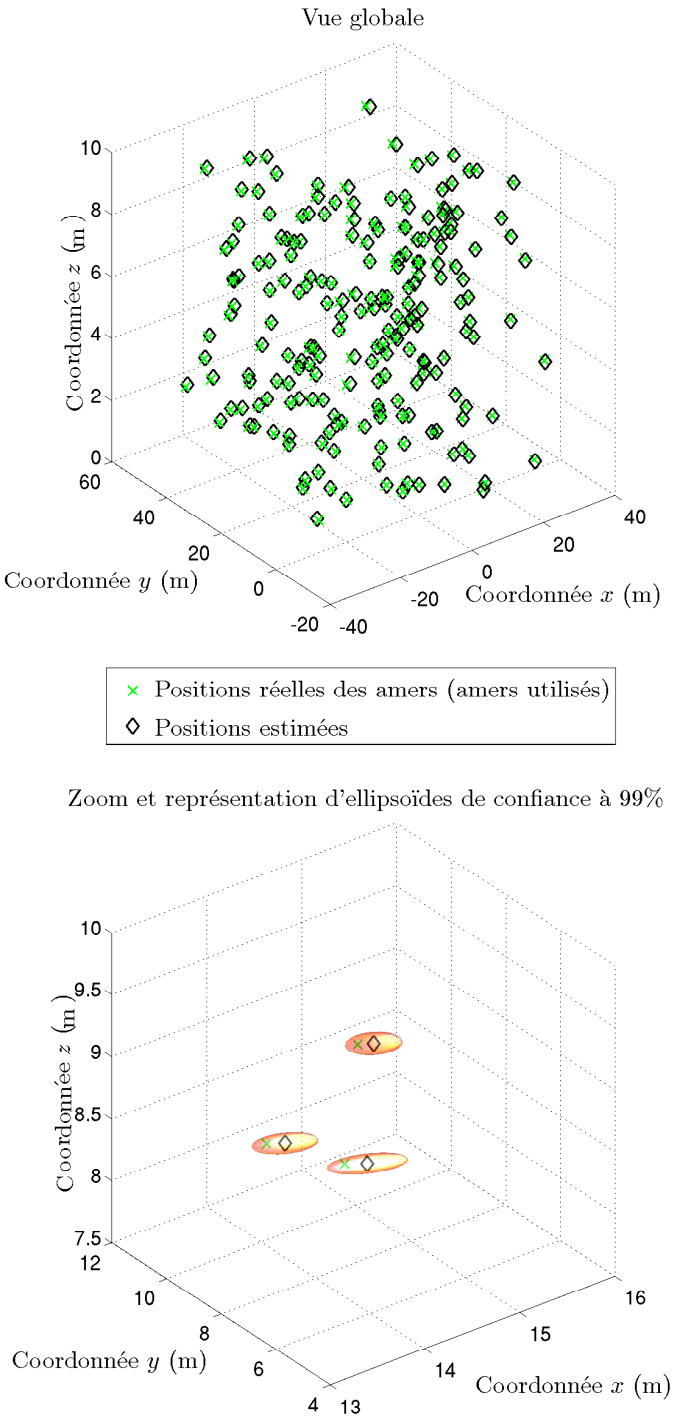


FIGURE 3.13 – Estimation des amers obtenue avec le SAM sur le scénario 1

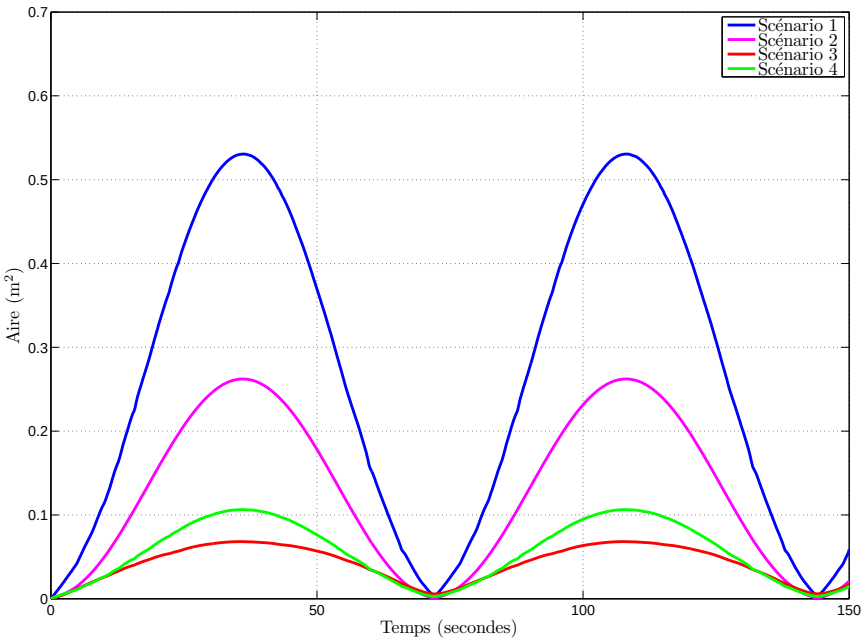


FIGURE 3.14 – Aires des ellipses de confiance à 99% concernant les positions du robot : scenarii 1 à 4

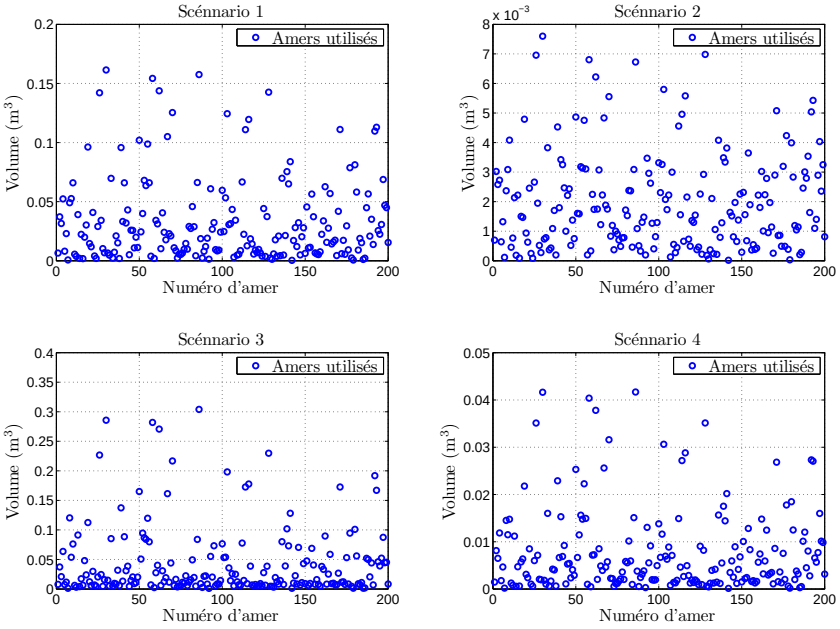


FIGURE 3.15 – Volumes des ellipsoïdes de confiance à 99% concernant les positions des amers : scenarii 1 à 4

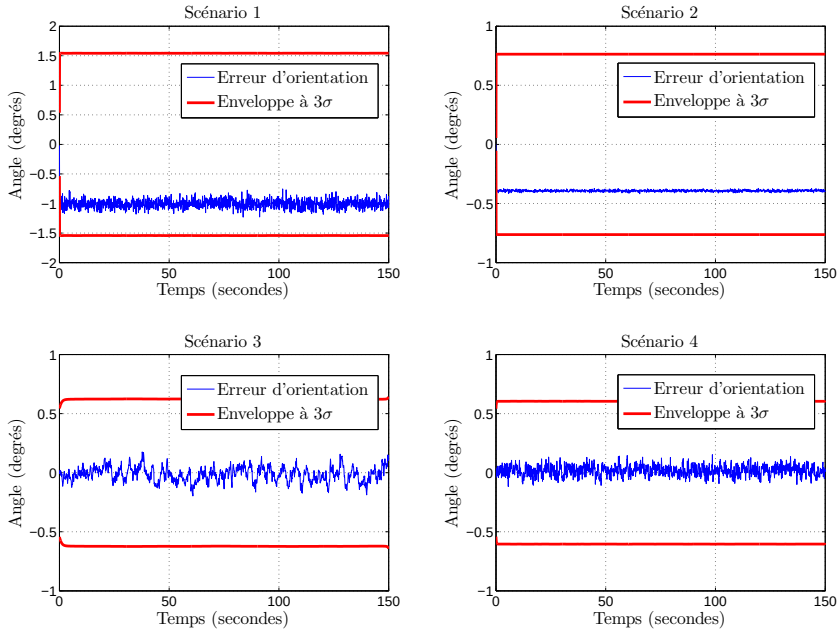


FIGURE 3.16 – Résultats du SAM concernant l'orientation du robot : scenarii 1 à 4

3.4.2.2 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 3.17 à 3.21. [6] 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. [7]

Les figures 3.17 à 3.19 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 3.20 et 3.21 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.

6. En ce qui concerne les amers, seuls 20 d'entre eux sont visualisés. Ceci permet de garder la figure lisible.
7. L'incertitude sur l'orientation atteint $+/-85$ degrés, et il est impossible ensuite d'estimer la fin de la trajectoire.

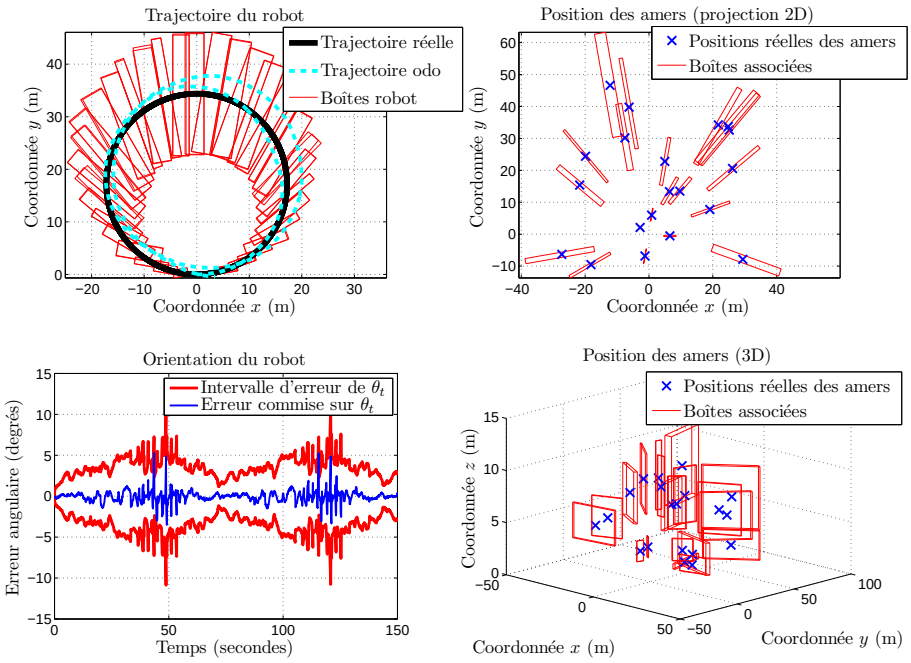


FIGURE 3.17 – Résultats du SLAM par intervalles : scénario 2 — Réglage à 4σ

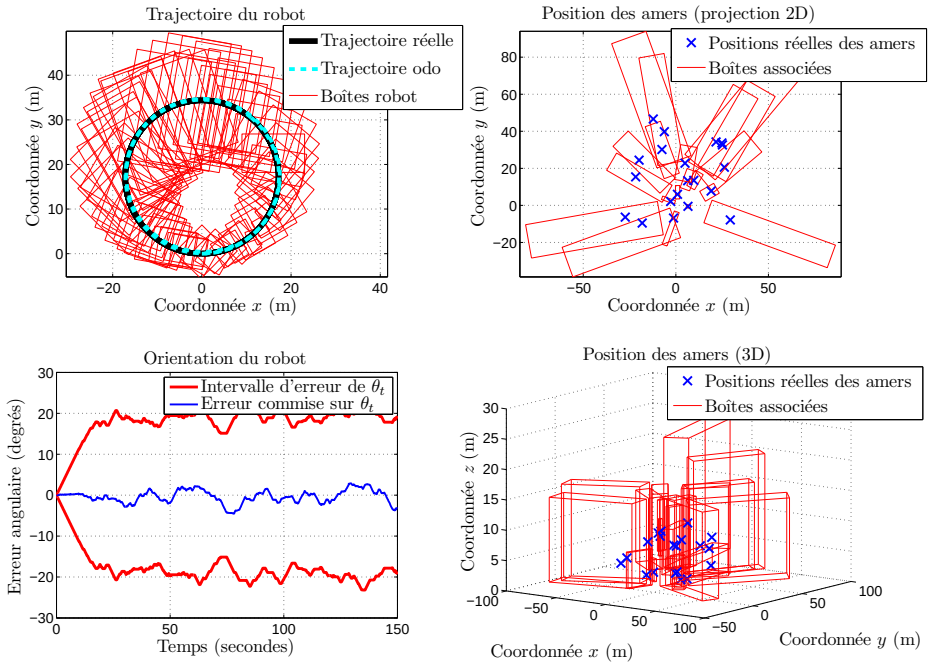


FIGURE 3.18 – Résultats du SLAM par intervalles : scénario 3 — Réglage à 4σ

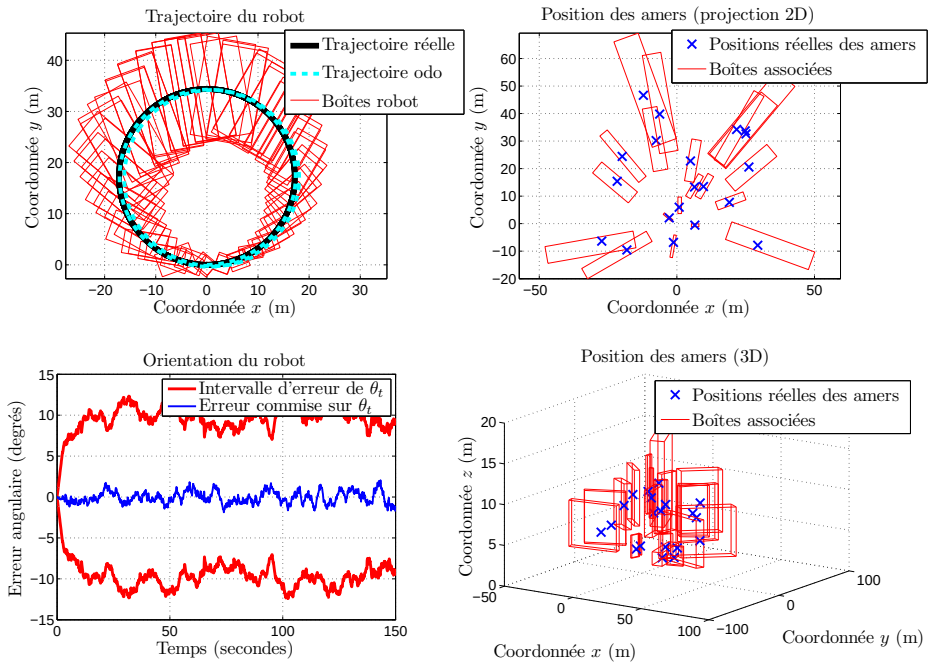


FIGURE 3.19 – Résultats du SLAM par intervalles : scénario 4 — Réglage à 4σ

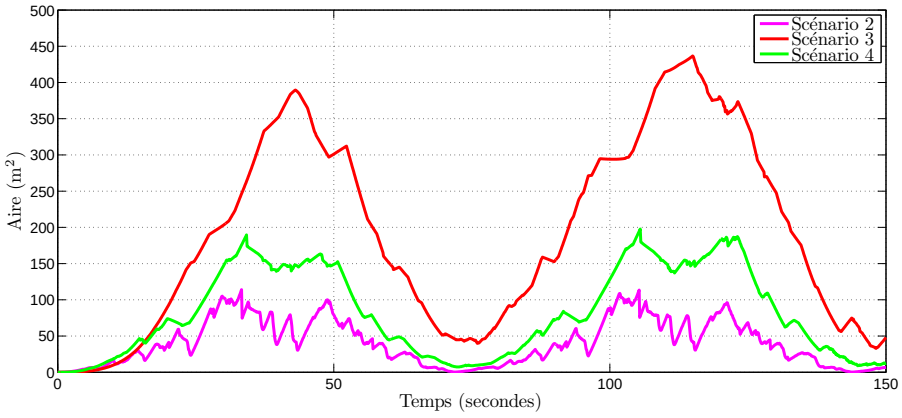


FIGURE 3.20 – Aires des boîtes concernant les positions du robot : scenarii 2 à 4 — Réglage à 4σ

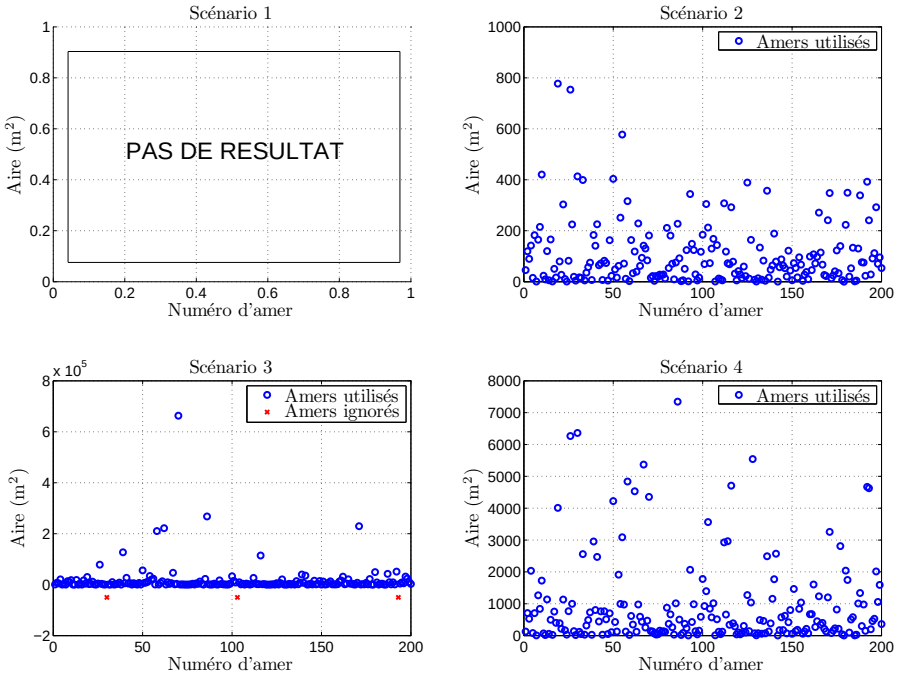


FIGURE 3.21 – Volumes des boîtes concernant les positions des amers : scenarii 1 à 4 — Réglage à 2σ

3.4.2.3 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.

Dans tous les cas, l'algorithme de SAM est supérieur à celui de SLAM par intervalles. Ainsi, lorsque les distributions des erreurs de nos capteurs sont gaussiennes et centrées, l'algorithme de SAM fournira d'excellents résultats pourvu que les paramètres soient connus (si ceux-ci sont trop optimistes, le SAM sera inconsistant. [8])

3.4.3 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 3.2.

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

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

TABLE 3.2 – Scenarii pour le cas uniforme centré

3.4.3.1 SAM

Nous présentons ici les résultats obtenus avec le SAM 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 alors :

$$V = \frac{a^2}{3} \tag{3.38}$$

En conséquence, les variances des filtres ont été réglées en utilisant l'équation 3.38 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 3.12, et ne sont pas représentées.

Les aires (resp. volumes) des ellipses (resp. ellipsoïdes) à 99% obtenues sont données sur les figures 3.22 et 3.23. 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 SAM lorsque les erreurs sont

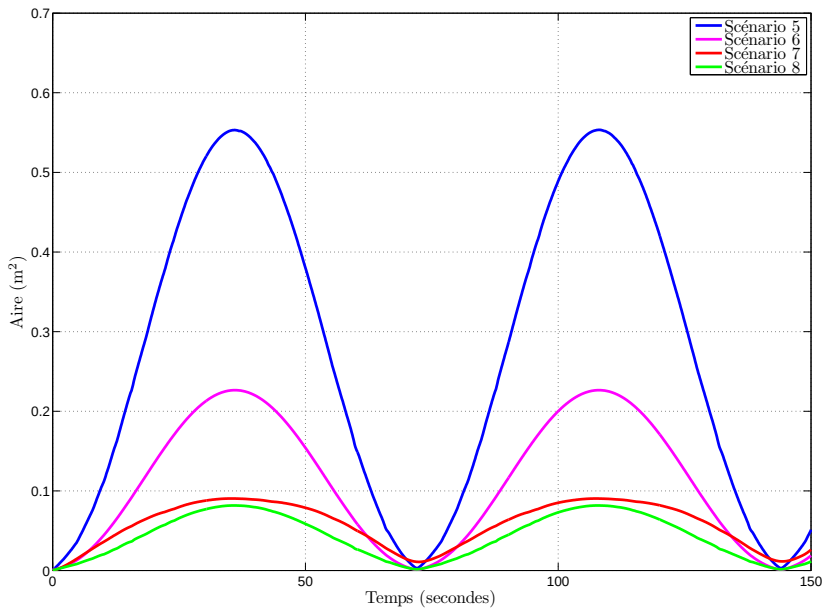


FIGURE 3.22 – Aires des ellipses de confiance à 99% concernant les positions du robot : scenarii 5 à 8

uniformes et centrées alors que l'on ne respecte plus l'hypothèse classique du SAM. 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 3.38. En pratique, ce type de réglage conduit à un résultat trop optimiste et inconsistant.

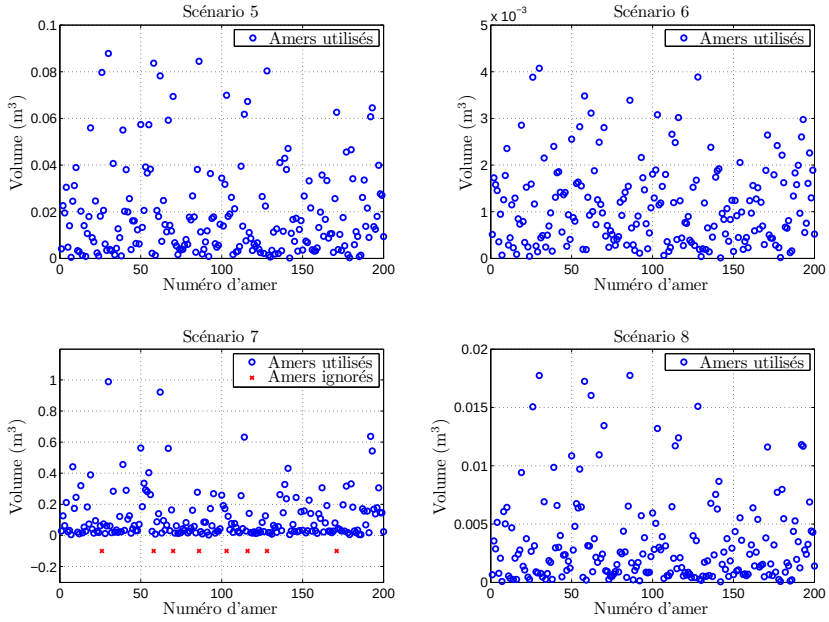


FIGURE 3.23 – Volumes des ellipsoïdes de confiance à 99% concernant les positions des amers : scenarii 5 à 8

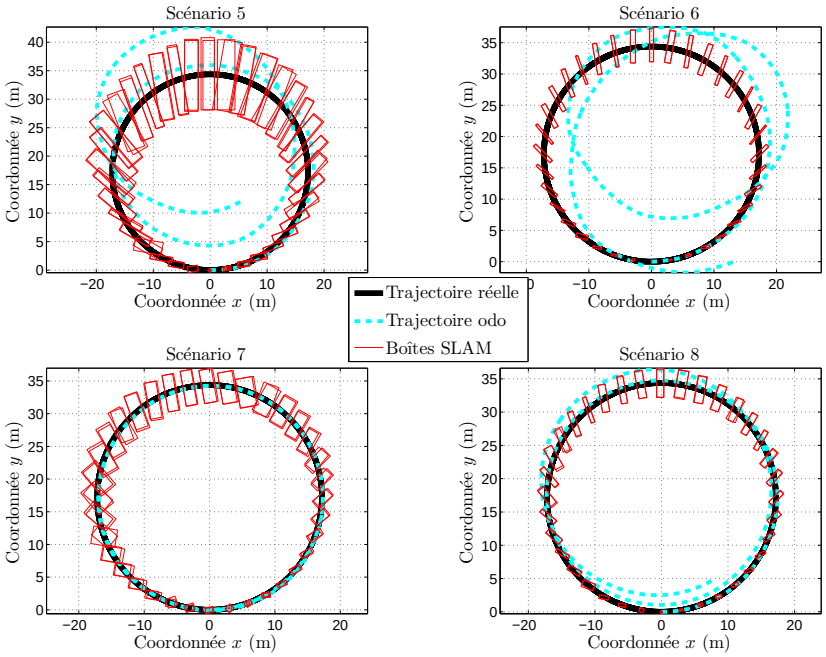


FIGURE 3.24 – Résultats du SLAM par intervalles en ce qui concerne la position du robot : scenarii 5 à 8

3.4.3.2 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 3.24 et 3.25 . Les aires et volumes obtenus sont quant à eux visibles sur les figures 3.26 et 3.27.

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 termes 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.

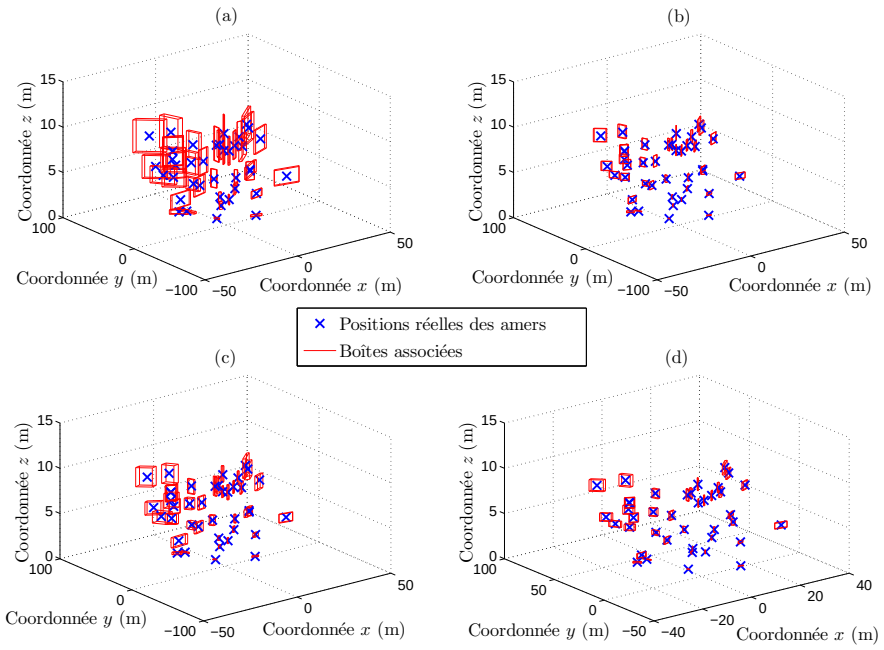


FIGURE 3.25 – Résultats du SLAM par intervalles en ce qui concerne la position des amers : scenarii 5 à 8

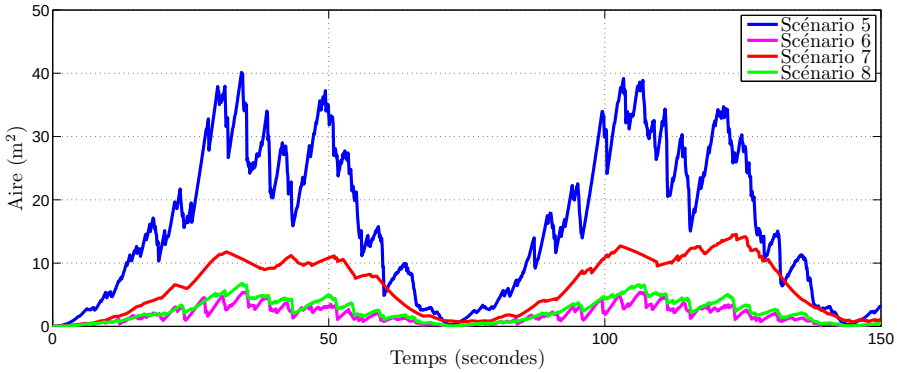


FIGURE 3.26 – Aires des boîtes concernant les positions du robot : scenarii 5 à 8

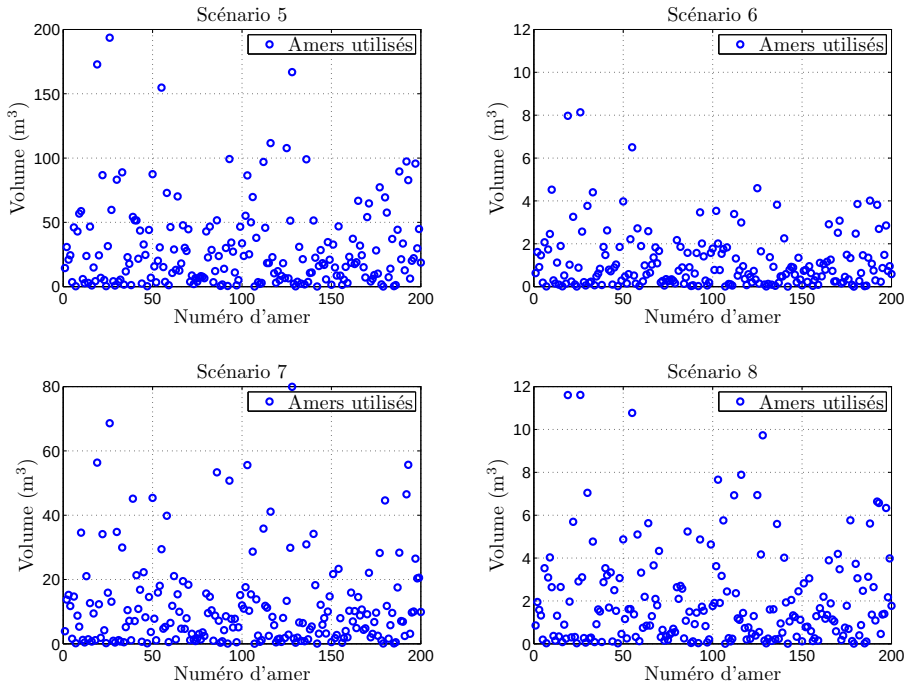


FIGURE 3.27 – Volumes des boîtes concernant les positions des amers : scenarii 5 à 8

3.4.3.3 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. 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 SAM. 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 SAM. Ceci est visible en comparant les figures 3.22 et 3.26 (aires associées aux positions du robot) ainsi que les figures 3.23 et 3.27 (volumes associés aux positions des amers). La figure 3.28 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.

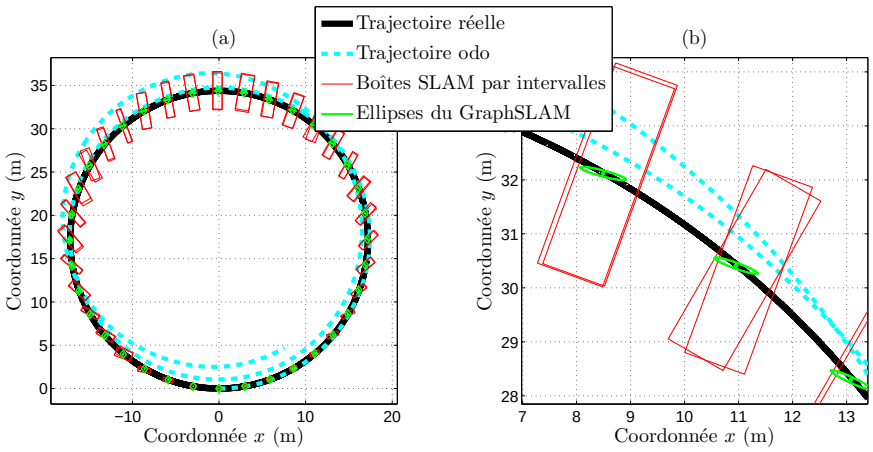


FIGURE 3.28 – Comparaison des boîtes du SLAM par intervalles et des ellipses à 99% du SAM : scénario 8 — (a) Vue globale — (b) Zoom

3.4.4 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 SAM. Les deux algorithmes sont testés avec les 4 scenarii présentés dans le tableau 3.3.

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 3.3).

3.4.4.1 SAM

Les résultats concernant le SAM 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

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 SAM.

TABLE 3.3 – Scenarii pour le cas biaisé

résultat retourné). Les figures 3.29 et 3.30 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 scénario (cf. figures 3.31 et 3.32). 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'algorithme 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**.

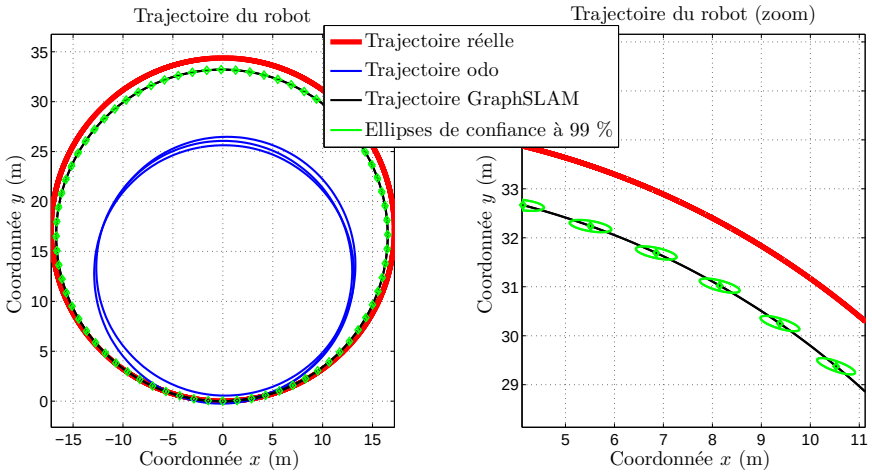


FIGURE 3.29 – Résultats du SAM en ce qui concerne la trajectoire du robot : scénario 9 — Réglage à 1σ

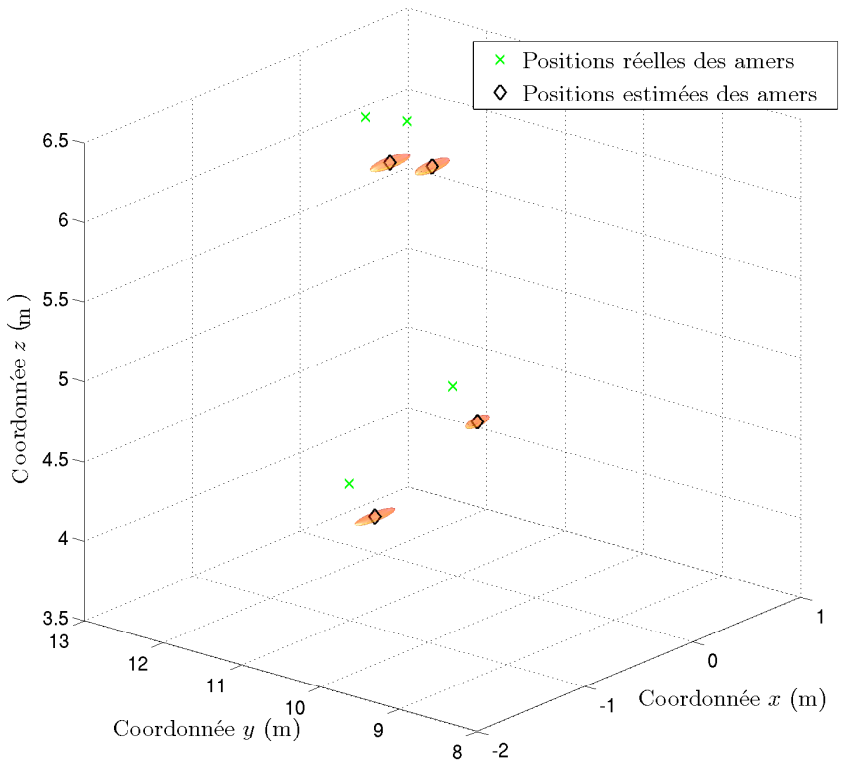


FIGURE 3.30 – Résultats du SAM en ce qui concerne la position des amers : scénario 9 — Réglage à 1σ

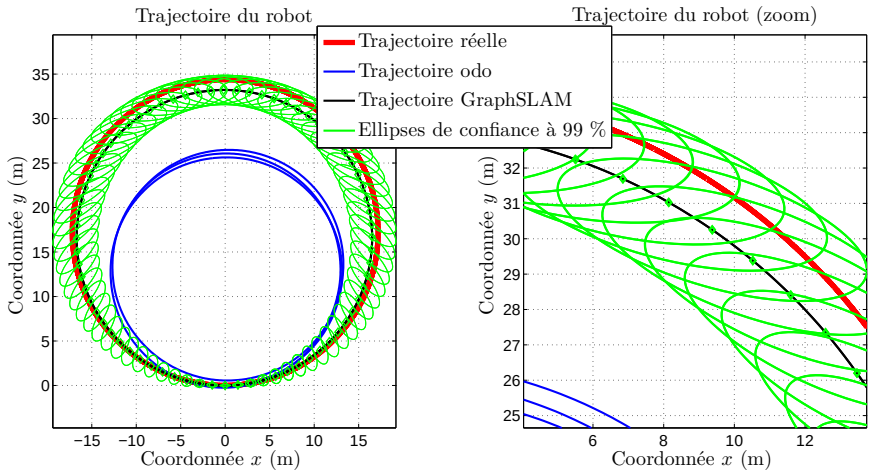


FIGURE 3.31 – Résultats du SAM en ce qui concerne la trajectoire du robot : scénario 9 — Réglage à 16σ

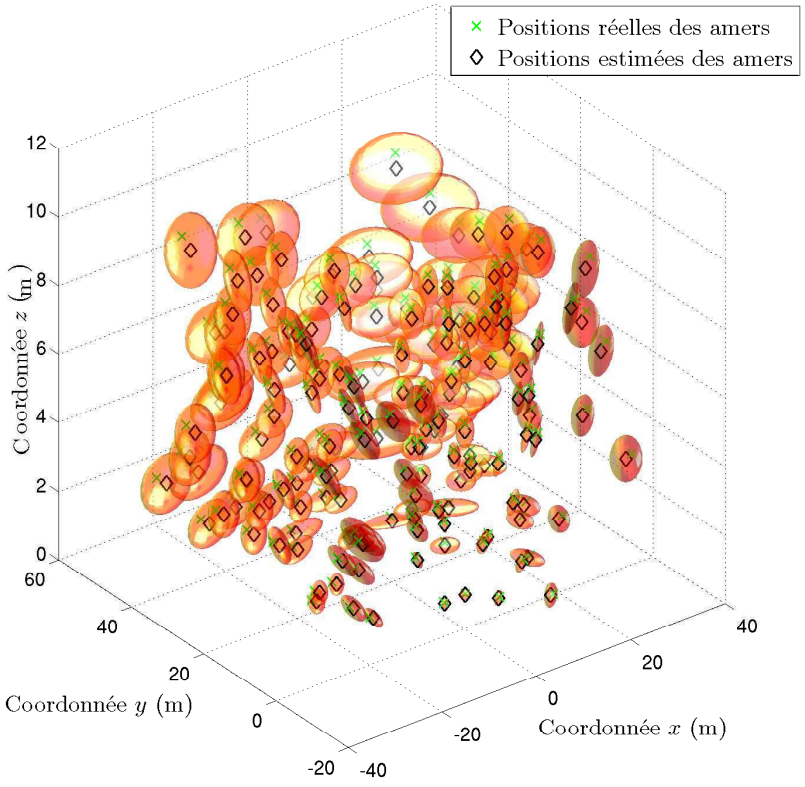


FIGURE 3.32 – Résultats du SAM en ce qui concerne la position des amers : scénario 9 — Réglage à 16σ

3.4.4.2 SLAM par intervalles

Nous présentons ici les résultats concernant le SLAM par intervalles. Les figures 3.33 à 3.36 montrent les boîtes obtenues pour l'état du robot et pour 40 amers. On peut constater que les résultats sont consistants comme cela était prévisible. La figure 3.36 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 3.37 et 3.38 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.

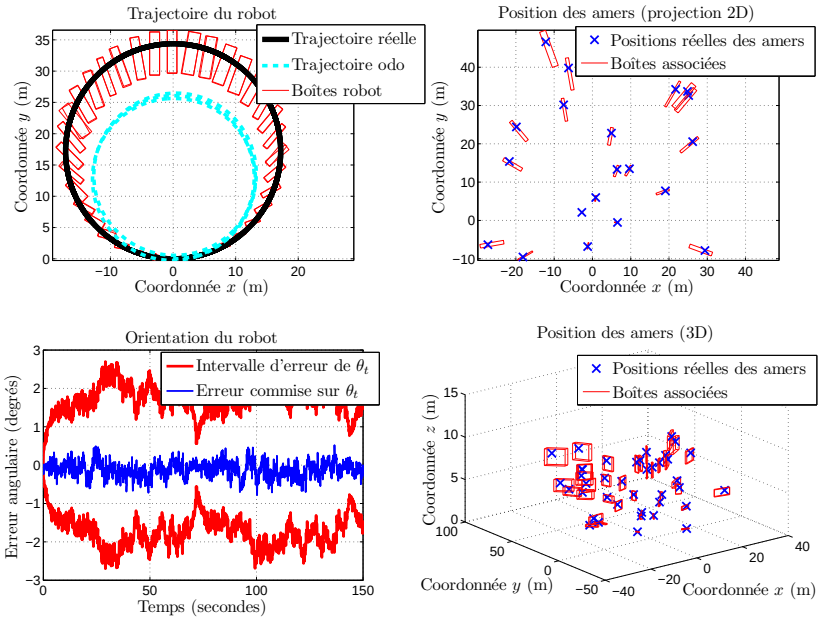


FIGURE 3.33 – Résultats du SLAM par intervalles : scénario 9

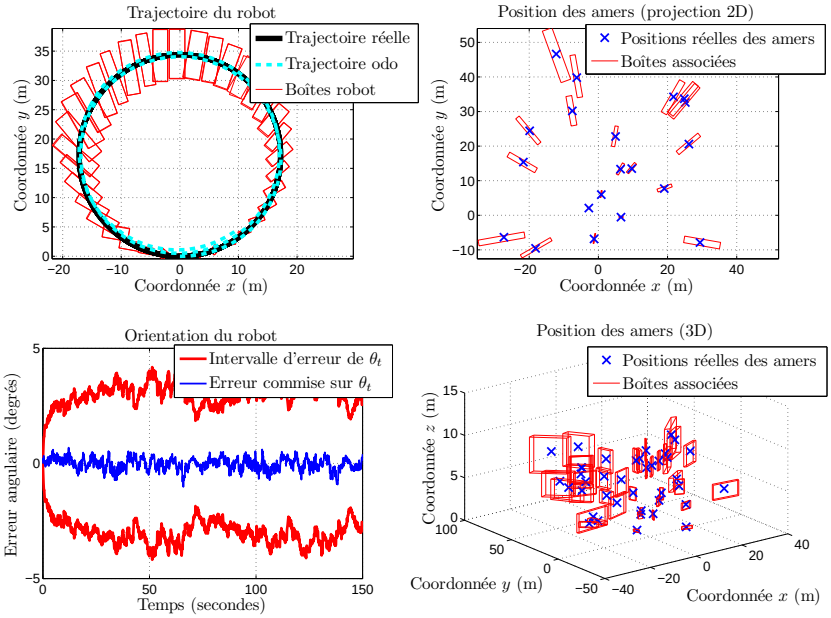


FIGURE 3.34 – Résultats du SLAM par intervalles : scénario 10

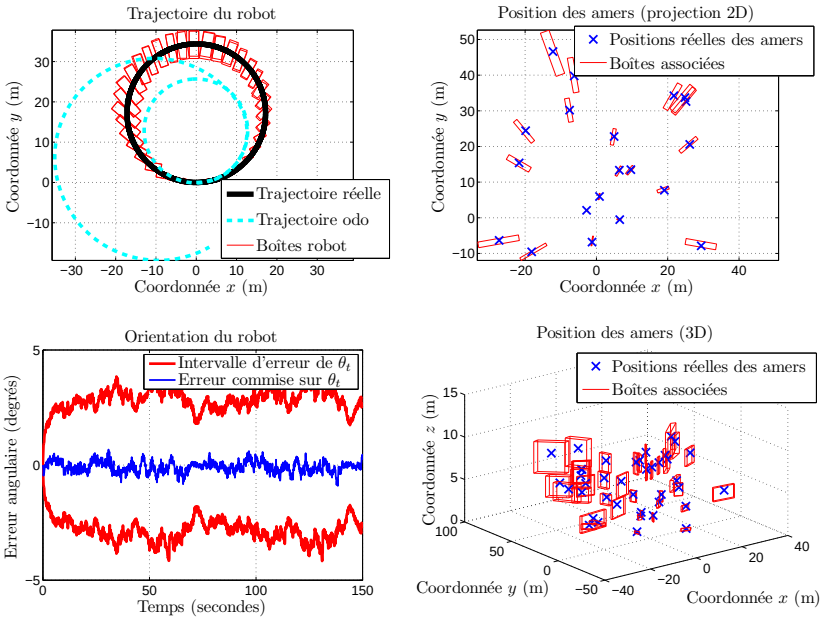


FIGURE 3.35 – Résultats du SLAM par intervalles : scénario 11

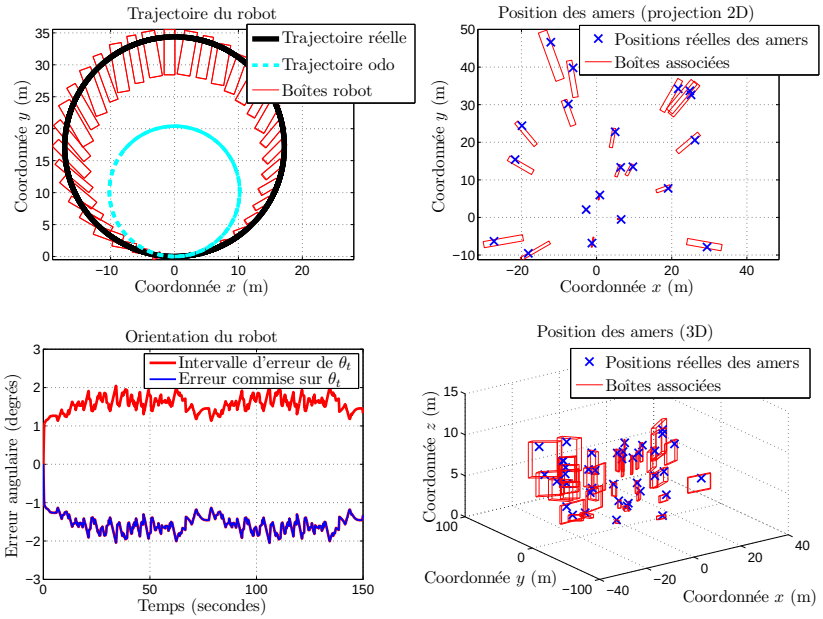


FIGURE 3.36 – Résultats du SLAM par intervalles : scénario 12

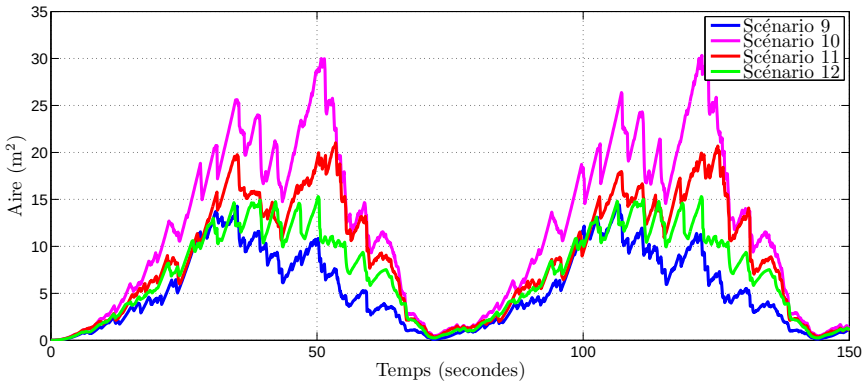


FIGURE 3.37 – Aires des boîtes concernant les positions du robot : scenarii 9 à 12

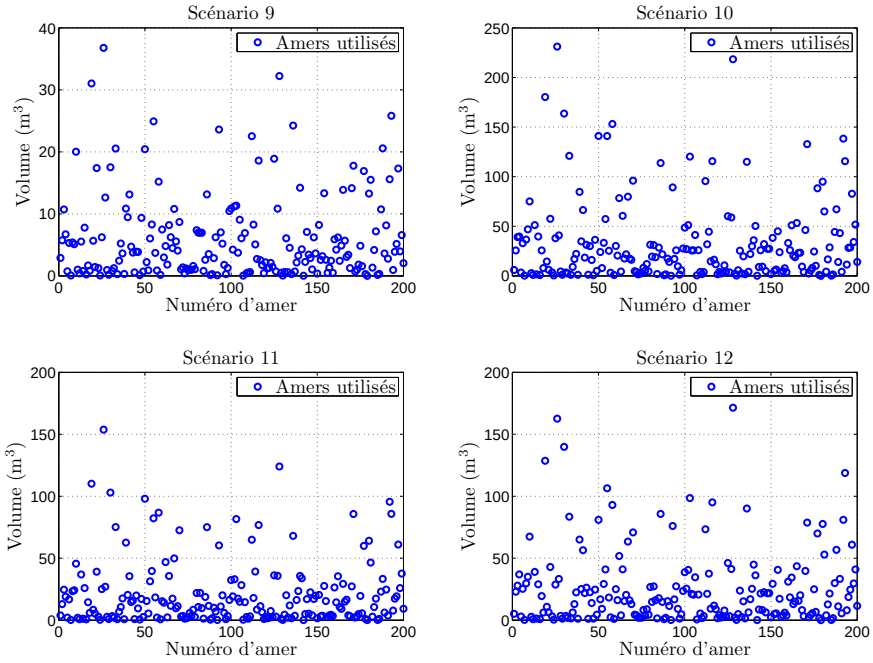


FIGURE 3.38 – Volumes des boîtes concernant les positions des amers : scenarii 9 à 12

3.4.4.3 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 SAM, l'algorithme de SLAM par intervalles n'a posé aucun souci en ce qui concerne le traitement des biais, qu'ils soient variables ou non.

3.4.5 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 3.39 à 3.42.

Les résultats concernant le SAM 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. [9]

9. 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

Par ailleurs, l'algorithme est toujours resté consistant.

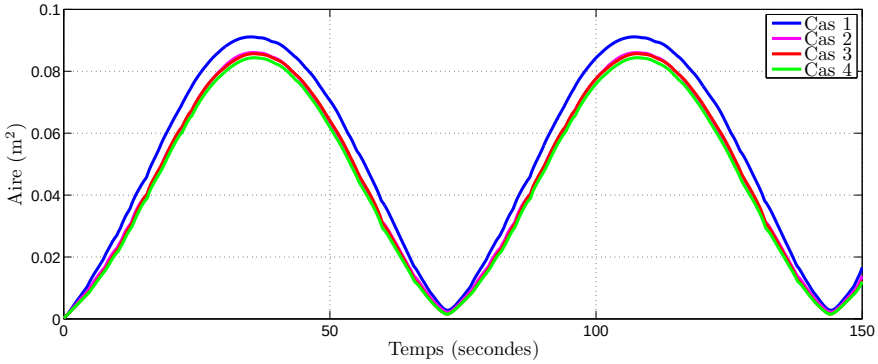


FIGURE 3.39 – Aires des ellipses de confiance à 99% concernant les positions du robot : scénario 8 visibilité diminuée (SAM)

reste donc du même ordre de grandeur.

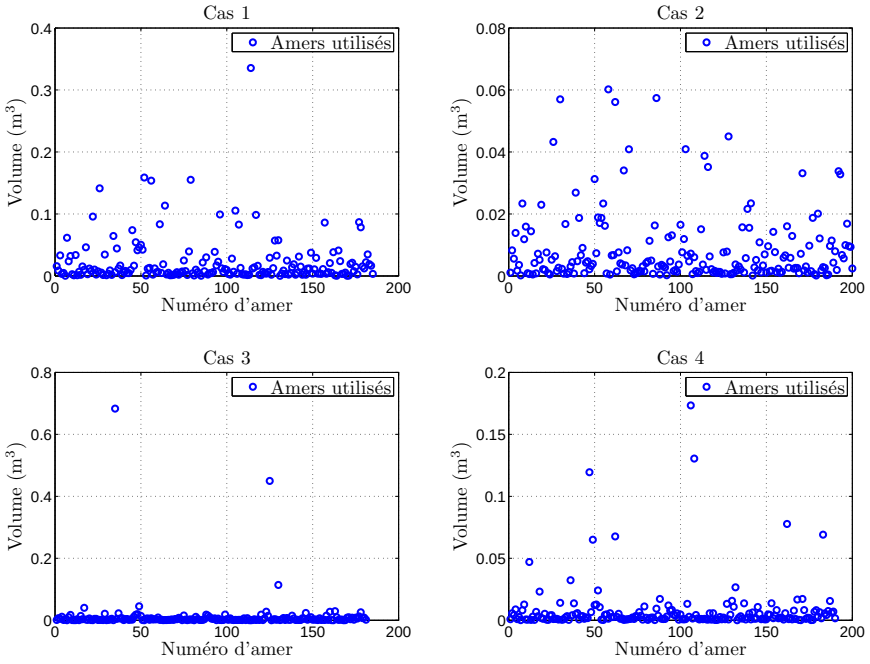


FIGURE 3.40 – Volumes des ellipsoïdes de confiance à 99% concernant les positions des amers : scénario 8 visibilité diminuée (SAM)

En revanche, les résultats du SLAM par intervalles sont nettement dégradés. Ceci est par ailleurs illustré sur les figures 3.43 et 3.44 (à comparer avec les figures 3.24 et 3.25 pages 123-124). 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 3.43 : 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.

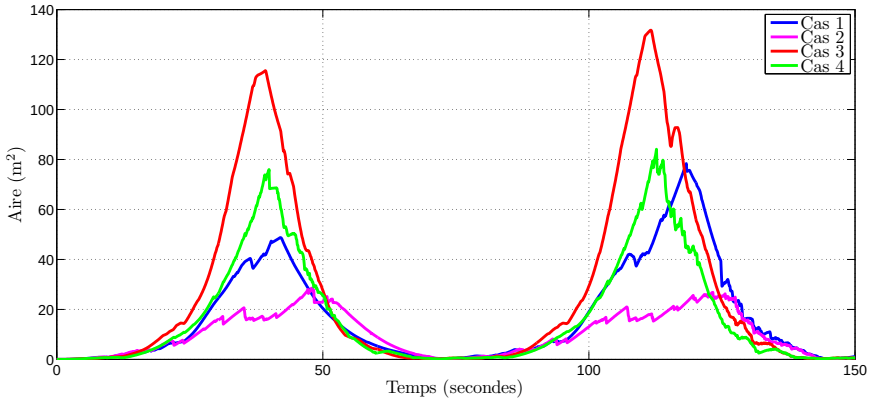


FIGURE 3.41 – Aires des boîtes concernant les positions du robot : scénario 8 visibilité diminuée (SLAM par intervalles)

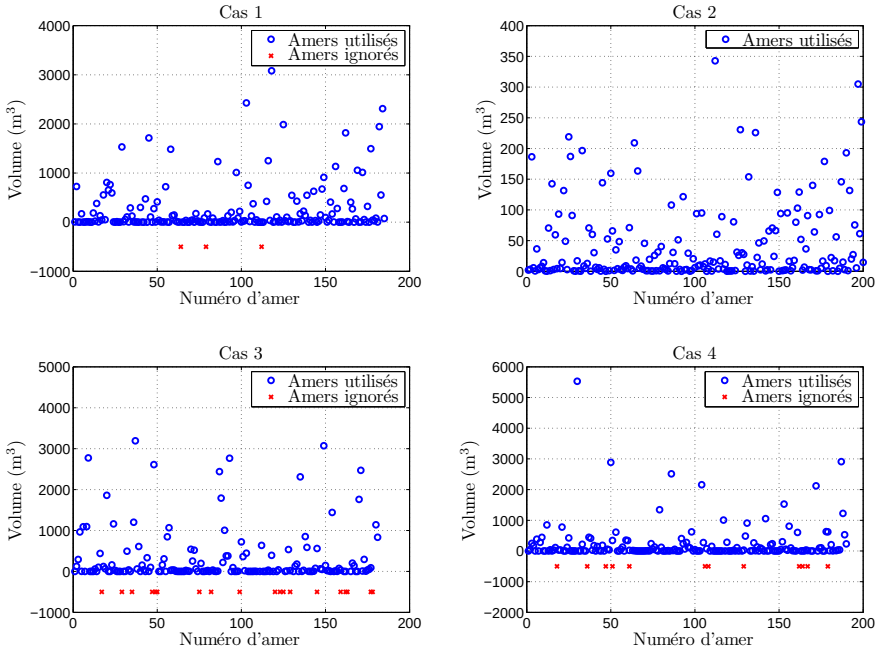


FIGURE 3.42 – Volumes des boîtes concernant les positions des amers : scénario 8 visibilité diminuée (SLAM par intervalles)

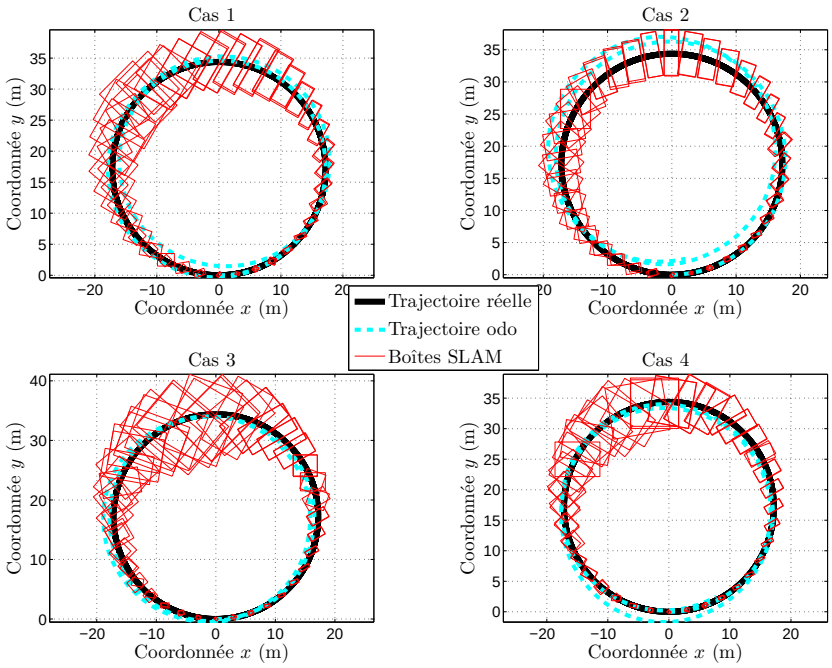


FIGURE 3.43 – Résultats du SLAM par intervalles concernant la trajectoire du robot : scénario 8 visibilité diminuée — Les trajectoires odométriques sont différentes car les échantillons de bruit le sont. En revanche, les paramètres sont identiques et les bruits toujours centrés. Le résultat n'est donc pas impacté par l'utilisation d'échantillons différents.

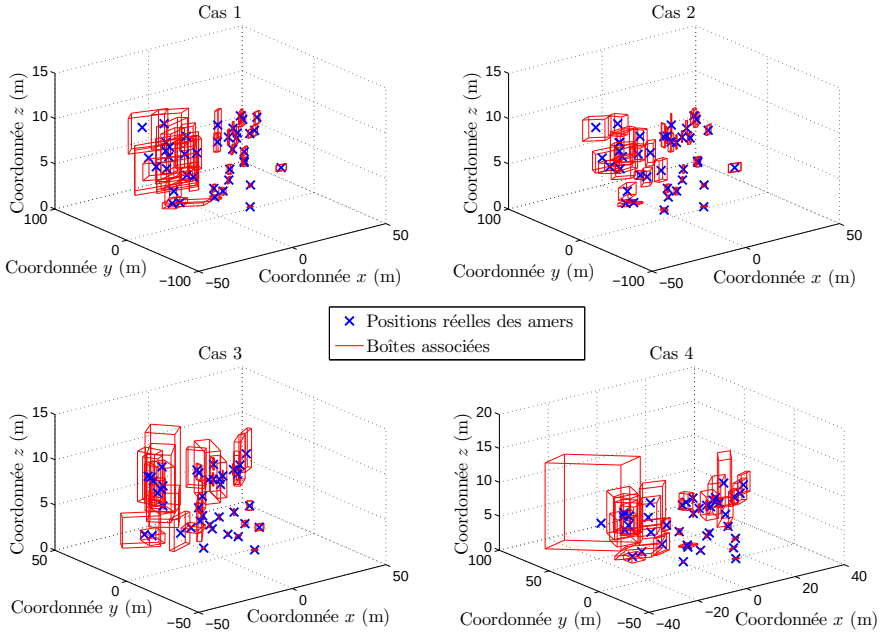


FIGURE 3.44 – Résultats du SLAM par intervalles concernant les positions des amers : scénario 8 visibilité diminuée

3.4.6 Remarque sur la consistance du SAM

Nous avons vu dans les paragraphes 3.4.2 et 3.4.3 que l'algorithme de SAM 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 [10]) pour les positions du robot. Cela revient à calculer pour chaque scénario la variable ϵ_t définie ainsi :

$$\epsilon_t = \left(\mathbf{x}_{t,Pos}^{Vrai} - \mathbf{x}_{t,Pos}\right)^T \mathbf{\Sigma}_{t,Pos}^{-1} \left(\mathbf{x}_{t,Pos}^{Vrai} - \mathbf{x}_{t,Pos}\right) \tag{3.39}$$

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 SAM 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 [Bailey *et al.*, 2006a] 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 3.45 montre que l'algorithme de SAM est consistant sans être trop pessimiste. Ceci confirme les résultats observés dans les paragraphes 3.4.2 et 3.4.3.

3.4.7 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 SAM 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 SAM 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

10. Normalised Estimation Error Squared

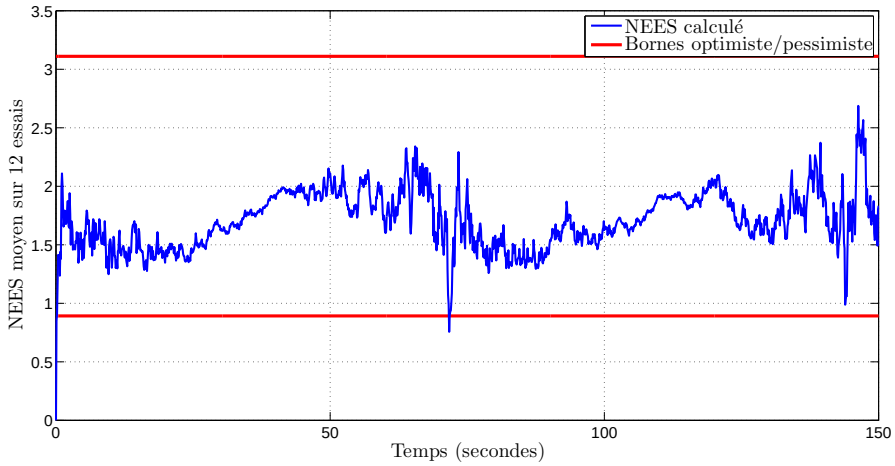


FIGURE 3.45 – Consistance du SAM par le calcul de l'erreur quadratique normalisée

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 SAM**.

La **supériorité du SLAM par intervalles a été mise en évidence lorsque les erreurs sur les capteurs sont biaisées**. Alors que le SAM 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 2.8 page 74) : 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 SAM, mais dans une moindre mesure ; le support de la densité de probabilité gaussienne n'étant pas borné, la présence de quelques mesures aberrantes [11] 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 travail).

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

11. La fréquence des points aberrants doit être compatible avec la loi gaussienne.

résultat reste exploitable. Dans le cas où les erreurs ne sont pas centrées, l'algorithme de SAM donnera en général des résultats exploitables [12] mais inconsistants.

3.5 Conclusion

Nous avons présenté dans ce chapitre la comparaison entre deux algorithmes de SLAM : un algorithme déterministe (le SLAM par intervalles) et un algorithme probabiliste (le SAM). 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 (SAM et SLAM par intervalles) et le problème d'orientation des boîtes (SLAM par intervalles).

Nous avons proposé un protocole de simulation visant à mettre en avant les principaux avantages et inconvénients des deux méthodes. Nous avons finalement montré qu'il n'existe pas d'algorithme beaucoup plus performant dans tous les cas. Néanmoins, on a pu remarquer que l'algorithme de SAM reste souvent consistant, et lorsqu'il ne l'est pas, les résultats ont une certaine cohérence. Le SLAM par intervalles, quant à lui, nécessite que toutes les mesures soient bornées :**aucune valeur aberrante n'est tolérée**. Dans le cas du SLAM visuel, cette dernière hypothèse est très difficile à assurer : la mise en correspondance image par image est un processus difficile. Ainsi, il paraît difficile de garantir que les hypothèses nécessaires à l'analyse par intervalles soient réunies.

L'approche de SAM nous paraît donc plus appropriée que l'analyse par intervalle pour le cas du SLAM visuel. Dans les expérimentations de Luc Jaulin ([Jaulin, 2009a]), on retrouve des résultats comparables entre méthode par intervalles et méthode probabiliste. Il convient de remarquer que ce cas est différent du nôtre. Le nombre de capteurs présent est plus important (il y a notament un GPS disponible à plusieurs endroits clés de la trajectoire). De plus, la mise en correspondance était effectuée à la main et garantie (dans le cas du SLAM visuel, il faudrait trouver un moyen pour éliminer les mauvaises mises en correspondances, ce qui n'est pas trivial). Enfin, le système de mesure associé aux amers comprenait à la fois l'angle et la distance (on parle alors de SLAM *range and bearing*). Ainsi, on peut remarquer que le rapport entre la quantité d'information obtenue par le robot et le nombre d'état à évaluer est nettement plus favorable dans le cas du sous-marin présenté dans [Jaulin, 2009a]. Dans le cas du SLAM visuel, la méthode semble pouvoir être employée, mais ses performances seraient moindres dans la plupart des situations.

12. 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.

Au final nous avons représenté un récapitulatif de toutes les méthode proposées dans cette première partie dans lee tableau 3.4. Nous choisirons pour la suite la méthode de SAM pour l'estimation. C'est la méthode qui nous semble réunir le plus d'avantages dans le but d'avoir une estimation consistante dans des conditions réalistes. Les autres méthodes probabilistes ne proposent pas les mêmes propriétés de consistance, et le SLAM par intervalles, quant à lui, paraît trop contraignant pour l'utilisation dans des conditions réelles.

	Complexité	Consistance	Exploitation du résultat	Observations
EKF-SLAM	−	− −	+	Les ellipses de confiance deviennent trop petites au fur et à mesure des itérations (même lorsque les bruits sont gaussiens centrés). Néanmoins, le résultats reste globalement exploitable.
Fast-SLAM	++	−	−	L'aspect particulaire ne permet pas de définir facilement d'ellipse d'erreur pour les amers. Par ailleurs la dégénérescence des particules peut être problématique en cas de fermeture de boucle.
SAM	−	+++	++	Le SAM délivre un résultat consistant tant que les bruits ne sont pas biaisés. Dans le cas de petits biais, le résultat devient inconsistant mais il reste assez cohérent. Les temps de calculs sont similaires à ceux de l'EKF.
SLAM par intervalles	−	++++	+ (Court terme) − − (Long terme) − − − (Intervalles mal connus)	Le SLAM par intervalles a tendance à être très pessimiste sur le long terme, rendant son résultat difficilement interprétable. Par ailleurs, les résultats sont dégradés lors de la présence de mesures aberrantes.

TABLE 3.4 – Récapitulatif des principales méthodes de SLAM visuel — La première colonne fournit une évaluation de la complexité des algorithmes. Les algorithmes les plus rapides à fournir un résultat sont mieux notés. – La seconde colonne, fournit des résultats concernant la consistance des algorithmes, ie. la capacité à fournir des enveloppes d'erreur qui ne soient pas trop petites. – La troisième colonne, quant à elle, évalue la facilité globale d'exploitation du résultat. Par exemple, si les enveloppes d'erreurs sont trop grandes, le résultat sera peu exploitable.

Deuxième partie

Résultats : du SLAM omnidirectionnel à 3 degrés de liberté au SLAM à 6 degrés de liberté

Chapitre 4

Introduction aux techniques de vision perspectives et omnidirectionnelles

Sommaire

Nous avons présenté dans la première partie du manuscrit diverses méthodes de filtrage. Pour tous les algorithmes présentés, nous avons essayé de rester le plus général possible. Tout au plus, nous avons introduit des modèles de mesures génériques (de type laser ou mesures d'angle). Dans cette seconde partie, nous présentons des résultats obtenus dans des situations réelles. Le capteur utilisé afin de percevoir l'environnement est une caméra omnidirectionnelle.

Dans ce chapitre, nous proposons de rappeler les bases de la vision par ordinateur. Nous verrons dans un premier temps la géométrie projective appliquée au cas perspectif (section 4.1). Nous verrons les principaux aspects concernant le modèle de projection planaire utilisé en vision par ordinateur ainsi que la détection et l'appariement de points d'intérêts. Nous montrons dans la seconde section les limitations de la vision classique dans le cadre d'une application au SLAM. Nous décrivons enfin dans la section 4.3 les principaux aspects de vision avec une caméra omnidirectionnelle.

Dans le cadre de cette thèse, nous n'avons pas développé de méthode originale de vision : nous nous inscrivons surtout en utilisateurs de méthodes existantes. Ainsi, le but de ce chapitre n'est pas de présenter de nouveaux algorithmes de traitement d'images. Il s'agit juste de donner un aperçu des méthodes existantes et de montrer le travail bas niveau nécessaire avant d'utiliser les méthodes de SLAM décrites. Le lecteur intéressé par plus de détails se reportera aux ouvrages et publications de références cités dans le chapitre.

4.1 Caméra perspective

Nous proposons dans cette section de décrire les divers aspects de la vision par ordinateur avec une caméra projective classique. Pour cela, nous décrivons d'abord le modèle sténopé de caméras. Nous présentons ensuite les bases de la géométrie épipolaire. Enfin, nous décrivons le type de primitive utilisé dans les images et quelques méthodes d'extraction et de suivi.

4.1.1 Modélisation d'une caméra perspective

En vision par ordinateur, on utilise l'approximation du modèle sténopé pour la caméra perspective classique (modèle *pinhole* dans la littérature anglo-saxone). Soit $\mathbf{C}$ le centre optique de la caméra. L'origine du repère de la caméra est prise en $\mathbf{C}$ et son troisième axe ($\mathbf{z}$) est pris dans la direction de l'axe optique (figure 4.1). Soit $\mathcal{I}$ le plan image, situé à une distance f du centre de la caméra (f désigne la distance focale de la caméra). L'image capturée par la caméra sera définie par la projection de la scène sur ce plan, après discrétisation sur la grille de pixels (définie par le capteur).

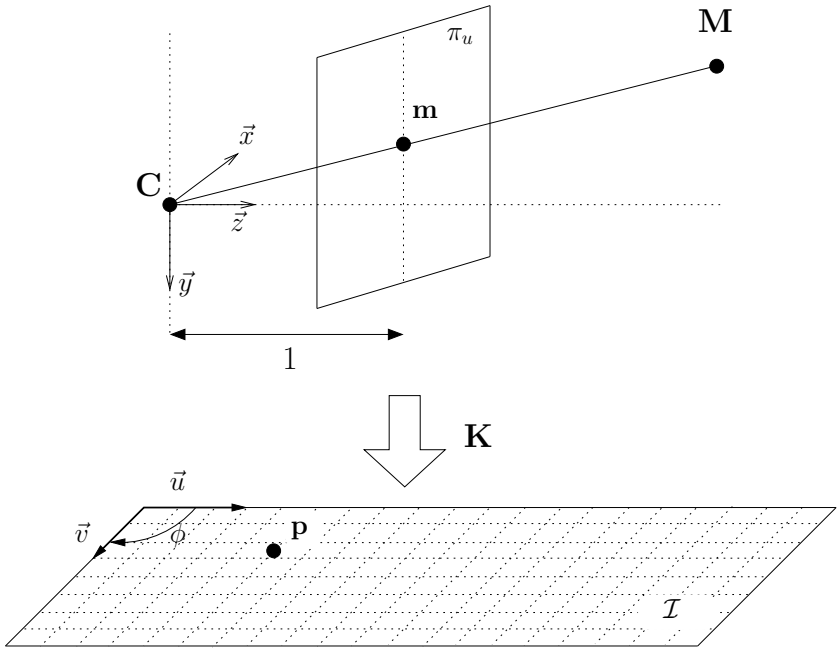


FIGURE 4.1 – Modèle sténopé de caméra : projection sur le plan unitaire (étape indépendante des paramètres de la caméra), puis transformation en pixels grâce à la matrice des paramètres intrinsèques.

Soit **M** un point de l'espace 3D vu par la caméra dont les coordonnées (x_M, y_M et z_M) sont données dans le repère de la caméra. Nous cherchons dans ce paragraphe à calculer les coordonnées en pixels de la projection de **M** dans l'image. On note **m** sa projection sur le plan unitaire (ie. le plan parallèle au plan image (noté π_u), mais à une distance unitaire du centre de la caméra), dont les coordonnées sont $\frac{x_M}{z_M}$, $\frac{y_M}{z_M}$ et 1. [1] Le point **m** n'est pas utilisé directement. La mesure finale en pixels (**p**) est donnée par la projection de **m** dans le repère du capteur :

$$\mathbf{p} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} fk_u & fk_u \cot\phi & u_0 \\ 0 & f\frac{k_v}{\sin\phi} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{x_M}{z_M} \\ \frac{y_M}{z_M} \\ 1 \end{bmatrix} \tag{4.1}$$

où on a défini les paramètres intrinsèques suivants :

k_u **et** k_v **:** facteurs d'agrandissement de l'image (exprimés en pixels/mm). En général, on a $k_u = k_v$ (cas classique d'une matrice de pixels carrés),

u_0 **et** v_0 **:** coordonnées de la projection du centre optique de la caméra dans le repère du capteur (exprimés en pixels)

ϕ **:** angle entre les axes du repère image. Les capteurs des caméras sont en général constitués d'une

1. **m** définit donc les coordonnées homogènes de **M** dans l'espace projectif. Il s'agit plus précisément des coordonnées homogènes **unitaires** (la troisième composante étant fixée à 1).

matrice de pixels carrés (on a alors généralement $\phi = \pi/2$).

La matrice définie dans l'équation 4.1 est invariante par multiplication de la distance focale par un scalaire α et division de k_u et k_v par ce même scalaire. On choisit donc de simplifier cette matrice en regroupant la distance focale réelle avec les facteurs d'agrandissement. On pose :

$$\mathbf{p} = \underbrace{\begin{bmatrix} f_u & sf_v & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{K}} \mathbf{m} \tag{4.2}$$

où on a défini :

- $f_u = fk_u$: la distance focale horizontale (exprimée en pixels),
- $f_v = \frac{fk_v}{\sin\phi}$: la distance focale verticale (exprimée en pixels),
- $s = \cot\phi$: le facteur d'obliquité (*skew factor* en anglais).

La plupart des caméras vérifient $f_u = f_v$ et $s = 0$. $\mathbf{K}$ est la matrice des paramètres intrinsèques : elle ne dépend que de la caméra. Elle peut être obtenue après calibration de la caméra à l'aide de mires ([Hartley et Zisserman, 2000]). Il existe également des méthodes d'autocalibration permettant de calibrer les caméras sans mire connue ([Faugeras *et al.*, 1992; Sturm, 1997]).

4.1.2 Géométrie épipolaire

La géométrie épipolaire est utilisée en vision stéréoscopique ou dans le cadre de séquences vidéos. Soient deux caméras de centres optiques respectifs $\mathbf{C_1}$ et $\mathbf{C_2}$ et dont les matrices des paramètres intrinsèques sont $\mathbf{K_1}$ et $\mathbf{K_2}$. Il peut s'agir de deux caméras physiquement différentes (cas par exemple d'un banc stéréoscopique) ou alors d'une même caméra que l'on déplace entre deux instants (dans ce cas, les deux caméras partagent les mêmes paramètres intrinsèques). On suppose par ailleurs que les deux caméras capturent la même scène de deux endroits différents.

Soit $\mathbf{M}$ un point capturé par les deux caméras (dont les coordonnées sont données dans le repère de la caméra 1). On note $\mathbf{m_1}$ le vecteur des coordonnées homogènes associées à la projection de $\mathbf{M}$ sur le plan unitaire de la caméra 1 (exprimé dans le repère de la caméra 1) et $\mathbf{m_2}$ le vecteur des coordonnées homogènes associées à la projection de $\mathbf{M}$ sur le plan unitaire de la caméra 2 (exprimé dans le repère de la caméra 2). Soient $\mathbf{R} = {}^{\mathbf{1}}\mathbf{R_2}$ et $\mathbf{t} = {}^{\mathbf{1}}\mathbf{t_2}$ les paramètres de la transformation entre les repères des caméras 1 et 2 (matrice de rotation et vecteur de translation). Les coordonnées de $\mathbf{M}$ dans le repère de la caméra 2 sont alors données par :

$${}^{\mathbf{2}}\mathbf{M} = \mathbf{R}^T (\mathbf{M} - \mathbf{t}) \tag{4.3}$$

La définition des coordonnées des vecteurs $\mathbf{m_1}$ et $\mathbf{m_2}$ (projection sur le plan unitaire $z = 1$) nous permet d'écrire :

$$\mathbf{m_1} = \alpha_1 \mathbf{M} \tag{4.4}$$

$$\mathbf{m_2} = \alpha_2 \mathbf{^2M} = \alpha_2 \mathbf{R}^T (\mathbf{M} - \mathbf{t}) \tag{4.5}$$

où α_1 et α_2 sont des constantes scalaires. Soit $[\mathbf{t}]_\times$ la matrice antisymétrique induite par $\mathbf{t}$, définie par :

$$[\mathbf{t}]_\times = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \qquad \left(\text{en supposant que } \mathbf{t} = \begin{bmatrix} t_x & t_y & t_z \end{bmatrix}^T\right) \tag{4.6}$$

$[\mathbf{t}]_\times$ permet alors de définir le produit vectoriel par $\mathbf{t}$ de manière matricielle ($\forall \mathbf{x} \in \mathbb{R}^3,\ [\mathbf{t}]_\times \mathbf{x} = \mathbf{t} \times \mathbf{x}$). En multipliant l'équation 4.5 à gauche par $\mathbf{m_1}^T [\mathbf{t}]_\times \mathbf{R}$ et en utilisant le fait que $\mathbf{M} = \mathbf{m_1}/\alpha_1$, on obtient :

$$\mathbf{m_1}^T [\mathbf{t}]_\times \mathbf{R} \mathbf{m}_2 = \alpha_2 \left(\frac{1}{\alpha_1} \underbrace{\mathbf{m_1}^T [\mathbf{t}]_\times \mathbf{m_1}}_{0} - \mathbf{m_1}^T \underbrace{[\mathbf{t}]_\times \mathbf{t}}_{0} \right) = 0 \tag{4.7}$$

Dans l'équation 4.7, la première accolade correspond à la définition matricielle du produit mixte entre les vecteurs $\mathbf{m_1}$, $\mathbf{m}_2$ et $\mathbf{m_1}$. Celui-ci est nul car il contient deux fois $\mathbf{m_1}$. La deuxième accolade, quant à elle, correspond au produit vectoriel de $\mathbf{t}$ par lui-même : il est donc nul.

Ainsi, l'équation 4.7 nous permet de définir la **contrainte épipolaire dans l'image normalisée** :

$$\mathbf{m_1}^T \mathbf{E} \mathbf{m_2} = 0 \tag{4.8}$$

où on a défini la matrice essentielle $\mathbf{E}$ par :

$$\mathbf{E} = [\mathbf{t}]_\times \mathbf{R} \tag{4.9}$$

La contrainte épipolaire définie dans le plan unitaire peut également être définie dans l'image. Soient $\mathbf{p_1}$ et $\mathbf{p_2}$ les coordonnées homogènes (unitaires) de la projection de $\mathbf{M}$ dans les images $\mathcal{I}_1$ et $\mathcal{I}_2$. Celles-ci sont liées à $\mathbf{m_1}$ et $\mathbf{m_2}$ par la relation :

$$\begin{cases} \mathbf{p_1} = \mathbf{K_1 m_1} \\ \mathbf{p_2} = \mathbf{K_2 m_2} \end{cases} \tag{4.10}$$

On peut alors utiliser les équations 4.9 et 4.10 pour définir la contrainte épipolaire dans les images. On a donc :

$$\mathbf{p_1}^T \mathbf{F} \mathbf{p_2} = 0 \tag{4.11}$$

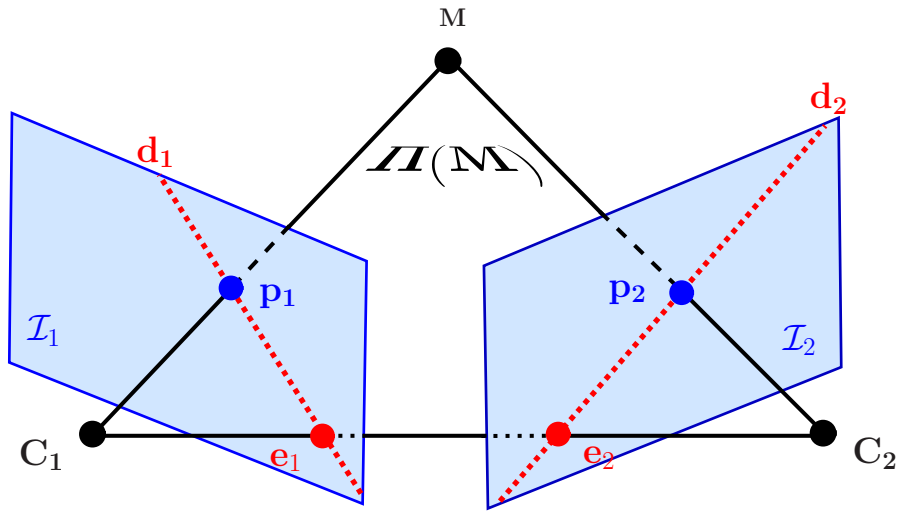


FIGURE 4.2 – La contrainte épipolaire (projetée dans les plans images)

où on a défini la matrice fondamentale $\mathbf{F}$ par :

$$\mathbf{F} = \mathbf{K_1}^{-T}[\mathbf{t}]_\times \mathbf{R}\mathbf{K_2}^{-1} \tag{4.12}$$

On définit également le plan épipolaire $\Pi(\mathbf{M})$ associé à $\mathbf{M}$ comme le plan passant par $\mathbf{M}$, $\mathbf{C_1}$ et $\mathbf{C_2}$ (cf. figure 4.2). L'intersection de $\Pi(\mathbf{M})$ dans les images 1 et 2 définit les droites épipolaires $\mathbf{d_1}$ et $\mathbf{d_2}$. Ces deux droites correspondent aux contraintes épipolaires. Si la transformation entre les deux caméras est connue, que la position de $\mathbf{M}$ est inconnue (mais que l'on a obtenu le point $\mathbf{p_1}$ dans l'image $\mathcal{I}_1$), alors l'observation de $\mathbf{M}$ dans $\mathbf{I}_2$ appartiendra à la droite $\mathbf{d_2}$. Les paramètres de $\mathbf{d_1}$ et $\mathbf{d_2}$ se déduisent facilement de la contrainte épipolaire (équation 4.12) :

$$\begin{cases} \mathbf{d_1} = \mathbf{F}\mathbf{p_2} & (\mathbf{p_2} \text{ est supposé connu}) \\ \mathbf{d_2} = \mathbf{F}^T\mathbf{p_1} & (\mathbf{p_1} \text{ est supposé connu}) \end{cases} \tag{4.13}$$

L'intérêt de la géométrie épipolaire est de fournir des contraintes entre différentes vues. Ainsi, dans le cas d'une séquence vidéo où les paramètres de la caméra sont constants ($\mathbf{K_1} = \mathbf{K_2}$), elle permet de calculer la position des points observés, la pose de la caméra, voire les deux. Les contraintes épipolaires sont également utilisées en calibration. Elle peut également servir pour calibrer la transformation géométrique entre deux caméras d'un banc stéréoscopique (avec en général $\mathbf{K_1} \neq \mathbf{K_2}$) Dans cette thèse, elles seront utilisées pour la mise en correspondance de points d'intérêt (un test basé sur cette contrainte permet d'éliminer les principaux mauvais appariements).

4.1.3 Extraction et suivi de primitives

4.1.3.1 Choix du type de primitive

Nous avons pu voir dans les chapitres précédents que notre approche de SLAM est paramétrique. Nous supposons que chaque amer est défini par un vecteur d'état, mesuré par le biais d'une fonction de mesures (fonction **h**).

Parmi les types de primitives géométriques pouvant être considérés, nous avons :

Des plans : ils peuvent être détectables à l'aide de contours fermés dans les images (et de propriétés des textures). Ils peuvent être paramétrés dans l'espace 3D à l'aide d'un point et du vecteur normal ou encore par leurs coordonnées de Plücker. En l'absence de connaissance *a priori* sur la scène, le vecteur de mesures associé à un plan n'est pas évident. En effet, un plan peut être **observé dans les images** par le biais d'un contour, d'une texture ou encore de points caractéristiques. Il faut par ailleurs garder à l'esprit que notre application finale est l'utilisation d'une caméra omnidirectionnelle. La détection des plans est alors encore plus difficile.

Des lignes : les lignes sont détectables facilement dans les images. Elles peuvent être paramétrées par un point et une direction ou par deux points s'il s'agit d'un segment. Comme pour le cas du plan, le vecteur de mesures associé à une ligne n'est pas trivial. Une ligne est en effet constituée d'une infinité de points. On pourrait par exemple utiliser une représentation paramétrique des droites dans les images. Par ailleurs, dans le cas des images omnidirectionnelles, les projections des lignes seront des courbes (exceptées les droites verticales qui seront des droites radiales), ce qui rend leur détection et leur suivi encore difficile et souvent instable.

Des points : ils sont détectables facilement dans les images. Par ailleurs, leur paramétrisation 3D est aisée (3 coordonnées dans l'espace) et le vecteur de mesure est trivial (les coordonnées d'un pixel, voire un couple d'angles si on considère une information de plus haut niveau [2]). Enfin, il est assez aisé de caractériser la matrice de variances-covariances associée au vecteur de mesures. Cette dernière est en effet directement liée à la qualité du détecteur utilisé pour extraire les points d'intérêts.

Parmi tous ces exemples, la primitive de type " points " nous paraît la plus appropriée pour notre application. En effet, les points 3D se projettent dans les images en point. Il s'agit de la seule primitive dont la représentation dans le vecteur d'état et l'expression du vecteur de mesures sont simples à

2. Il est facile de montrer qu'il existe une application bijective du plan image vers une partie des vecteurs unitaires.

manipuler ; il s'agit d'un critère important pour l'utilisation de l'algorithme de SAM. L'utilisation des primitives de type plans et lignes conduirait à des fonctions de projection dans l'image (fonction **h**) nettement plus compliquées. Ces deux dernières primitives sont par ailleurs difficiles à détecter dans le cas d'un capteur omnidirectionnel.

4.1.3.2 Méthodes d'extraction des points d'intérêts

De nombreuses méthodes existent quant à l'extraction de points d'intérêts. L'idée est en général de chercher des zones de fort gradient dans les images (les points d'intérêts étant des zones de forte discontinuité). Plusieurs types de points d'intérêts existent. On pourra notamment citer l'opérateur de Beaudet ([Beaudet, 1978]), le détecteur de coins de Moravec ([Moravec, 1977]) ou encore sur le détecteur Kitchen et Rosenfeld ([Kitchen et Rosenfeld, 1982]). Nous retiendrons surtout le détecteur de Harris et Stephens ([Harris et Stephens, 1977]), que nous proposons de décrire plus en détail.

Considérons un pixel (u_0, v_0) d'une image $\mathbf{I}$ (à chaque pixel (u, v) de l'image est associée un niveau de gris $\mathbf{I}(u, v)$). Calculons la différence entre l'image courante et l'image décalée de x et de y. Nous définissons $E_{u_0,v_0}(x, y)$ comme la somme pondérée (sur tous les pixels de l'image) de cette différence :

$$E_{u_0,v_0}(x, y) = \sum_u \sum_v w_0(u, v) \left(\mathbf{I}(u, v) - \mathbf{I}(u + x, v + y)\right)^2 \tag{4.14}$$

où w_0 est une fonction positive de $\mathbb{R}^2$ dans $\mathbb{R}$ qui est très proche de zéro (voire nulle) lorsque son argument est en dehors d'un voisinage $\mathcal{V}_0$ de (u_0, v_0).[3] Dans le cas du détecteur de Harris, on utilise une fonction gaussienne (définie de sorte à ce qu'on puisse la couper en dehors de $\mathcal{V}_0$ [4]) et centrée sur (u_0, v_0) afin de privilégier le centre dans le calcul de la moyenne :

$$w_0(u, v) = \exp\left(-\frac{1}{2\sigma^2}\left((u - u_0)^2 + (v - v_0)^2\right)\right) \tag{4.15}$$

où σ est un paramètre de " réglage " permettant de définir la taille de $\mathcal{V}_0$. La définition de ce masque gaussien permet de garantir l'isotropie dans le calcul de la somme. Par ailleurs, il permet de limiter les effets de bruits que l'on observe en utilisant des masques binaires. En supposant que x et y sont suffisamment petits, un développement au premier ordre non nul de l'équation 4.14 donne :

$$E_{u_0,v_0}(x, y) \approx \begin{bmatrix} x & y \end{bmatrix} \underbrace{\begin{bmatrix} < I_x^2 > & < I_x I_y > \\ < I_x I_y > & < I_y^2 > \end{bmatrix}}_{\mathbf{M}} \begin{bmatrix} x \\ y \end{bmatrix} \tag{4.16}$$

3. Dans la cas du détecteur de Moravec, on utilise directement la fonction indicatrice de $\mathcal{V}_0$ (0 ou 1).
4. Cela évite de calculer une moyenne sur toute l'image alors que presque tous les coefficients sont nuls.

avec :

$$\left\{\begin{array}{rcl} < I_x^2 > & = & \sum_u \sum_v w_0(u,v) \frac{\partial \mathbf{I}}{\partial x}(u,v) \\ < I_y^2 > & = & \sum_u \sum_v w_0(u,v) \frac{\partial \mathbf{I}}{\partial y}(u,v) \\ < I_x I_y > & = & \sum_u \sum_v w_0(u,v) \frac{\partial \mathbf{I}}{\partial x}(u,v) \cdot \frac{\partial \mathbf{I}}{\partial y}(u,v) \end{array}\right. \tag{4.17}$$

Il est facile de montrer que $(< I_x I_y >)^2 \leq < I_x^2 >< I_y^2 >$ et donc que la matrice $\mathbf{M}$ est symétrique positive. Elle possède donc deux valeurs propres positives. Si le voisinage $\mathcal{V}_0$ considéré contient un point d'intérêt (typiquement un coin), on s'attend à ce que la valeur $E_{u_0,v_0}(x,y)$ soit " grande " quel que soit le décalage (x,y) considéré. D'après l'équation 4.16, les deux valeurs propres de $\mathbf{M}$ doivent être " grandes " dans ce cas là. Si $\mathcal{V}_0$ contient un contour, on s'attend à ce que $E_{u_0,v_0}(x,y)$ soit important dans une direction privilégiée (orthogonale au contour) et très proche de zéro (le long du contour) dans la direction orthogonale. La diagonalisation de $\mathbf{M}$ doit faire apparaître ces deux directions. Dans ce cas, on doit avoir une valeur propre " grande " et l'autre proche de zéro. Enfin, si le voisinage ne contient pas de point d'intérêt ni de courbe, $E_{u_0,v_0}(x,y)$ doit être " assez petite " dans toutes les directions ; il en est donc de même pour les valeurs propres. Ainsi, le calcul des valeurs propres de la matrice $\mathbf{M}$ pour tous les pixels caractérise la présence de points d'intérêts. Il suffirait par exemple de chercher les maxima locaux de la fonction $\min(\lambda_1(u,v), \lambda_2(u,v))$, (où $\lambda_1(u,v)$ et $\lambda_2(u,v)$ sont les valeurs propres de la matrice $\mathbf{M}$ calculée au pixel (u,v)). Néanmoins, Harris et Stephens proposent une méthode pour éviter de calculer les valeurs propres de $\mathbf{M}$ pour chaque pixel. Pour cela, ils évaluent le critère défini par :

$$K(u_0,v_0) = \det \mathbf{M} - k \times \mathrm{trace}^2 \mathbf{M} \tag{4.18}$$

où $k < 1$ est un scalaire positif choisi " suffisamment petit mais pas trop ". Pour discuter de l'intérêt de cette expression, considérons λ_1 et λ_2 ($\lambda_1 \leq \lambda_2$) les deux valeurs propres de $\mathbf{M}$. On a alors $\det \mathbf{M} = \lambda_1 \lambda_2$ et $\mathrm{trace}^2 \mathbf{M} = (\lambda_1 + \lambda_2)^2$. Reprenons désormais les 3 cas vus au paragraphe précédent :

1. λ_1 et λ_2 " grands " (il y a un coin dans le voisinage du pixel). Dans ce cas, $\lambda_1 \lambda_2$ est grand aussi, et $(\lambda_1 + \lambda_2)^2$ également. Si k est suffisamment petit, de sorte que le terme $k(\lambda_1 + \lambda_2)^2$ soit négligeable devant $\lambda_1 \lambda_2$, alors $K(u_0,v_0)$ est largement positif. [5]

2. λ_1 " petit " et λ_2 " grand " (il y a un contour dans le voisinage du pixel). Dans ce cas, $\lambda_1 \lambda_2$ sera d'autant plus petit que λ_1 l'est et $(\lambda_1 + \lambda_2)^2$ sera beaucoup plus grand (de l'ordre de λ_2^2). Si k

5. Si λ_1 et λ_2 sont du même ordre de grandeur (ie. $\lambda_2/10 < \lambda_1 \leq \lambda_2$), alors le critère sera assurément positif dès que $k < 1/22$ (≈ 0.045).

n'est pas trop petit, $k(\lambda_1 + \lambda_2)^2$ restera plus grand que $\lambda_1\lambda_2$, et le tout sera négatif. [6]

3. $\lambda_1 \approx \lambda_2 \approx 0$ (zone uniforme). Dans ce cas, le critère sera également proche de zéro.

Ainsi, lorsque l'expression définie dans l'équation 4.18 est suffisamment grande, le voisinage du pixel considéré contient certainement un point d'intérêt. En général, l'algorithme final revient à extraire les maxima locaux du critère et à garder ceux supérieurs à un certain seuil. Par ailleurs, on décompose en général l'image en plusieurs fenêtres, dans lesquelles on limite le nombre de points (on ne garde que les N meilleurs scores) afin d'éviter d'avoir trop de points dans les zones fortement texturées. Concernant le réglage du paramètre k, la littérature préconise de le choisir proche de 0.04 ; il s'agit d'un résultat acquis par l'expérience [Ma *et al.*, 2003]).

4.1.3.3 Mise en correspondance de points d'intérêts

La mise en correspondance des points d'intérêts entre deux images est la base du suivi de ces points le long des séquences. Pour cela, plusieurs méthodes sont proposées dans la littérature. Nous proposons ici d'utiliser un descripteur associé à chaque point. Ce descripteur doit permettre de caractériser les points suffisamment finement pour ne permettre qu'un nombre limité de faux appariements, mais il doit également être invariant à un certains nombre de transformations pour prendre en compte les modifications de l'image autour du point pendant le déplacement de la caméra. Par exemple, considérer un simple patch de taille fixe autour du point serait une méthode non robuste. En effet, il faudrait prendre en compte les changement d'échelles (dans le cas où la distance au point varie) et les distorsions (dans le cas d'une rotation de la caméra).

Nous proposons d'utiliser le descripteur SIFT (*Scale Invariant Feature Transform*) pour les points d'intérêts ([Lowe, 2004]). Il s'agit d'une méthode de description locale de l'image invariante aux changements d'échelle, de perspective, d'éclairage et particulièrement robuste au bruit. Au final, le descripteur obtenu est un vecteur de dimension 128 représentant l'orientation du gradient dans le voisinage du point. La taille de ce vecteur fait que le descripteur SIFT est coûteux à calculer, rendant compliquée son implémentation pour des conditions temps réel. Enfin, on peut noter que cette approche a été évaluée comme l'une des plus performantes dans une étude comparative des descripteurs locaux ([Mikolajczyk et Schmid, 2005]). Nous ne détaillons pas plus en détails ce descripteur. Le lecteur intéressé pourra se repporter aux publications de références.

6. Si λ_1 est de deux ordres de grandeur au moins inférieur à λ_2 (ie. $0 < \lambda_1 \leq \lambda_2/100$), alors le critère sera assurément négatif dès que $k > 1/102$ (≈ 0.0098).

L'utilisation des descripteurs SIFT permet d'obtenir un premier jeu de mises en correspondances. Celui-ci n'est en général pas optimal et un nombre non négligeable d'associations aberrantes sont souvent présentes. Nous tentons de les éliminer par une approche de type RANSAC (*RANdom SAmple Consensus*). Il s'agit d'une approche non déterministe, introduite pour la première fois par Fischler et Bolles ([Fischler et Bolles, 1981]). Le principe de l'approche RANSAC est de sélectionner aléatoirement un certain pourcentage des mises en correspondances initiales. A partir de cet ensemble, on effectue une première approximation de la matrice fondamentale (on peut montrer qu'il faut au moins 9 mises en correspondances distinctes pour faire le calcul). Ensuite, on teste la contrainte épipolaire sur les mises en correspondances restantes. Si le nombre de points satisfaisant la condition (à un niveau de précision près) est suffisant, l'algorithme est arrêté. Sinon, il est possible que trop de points aberrants aient été pris en considération lors du calcul de la matrice fondamentale et un nouvel échantillonnage est effectué. L'algorithme s'arrête lorsqu'un consensus a été obtenu, séparant ainsi les bonnes mises en correspondances (*inliers*) des mauvaises (*outliers*).

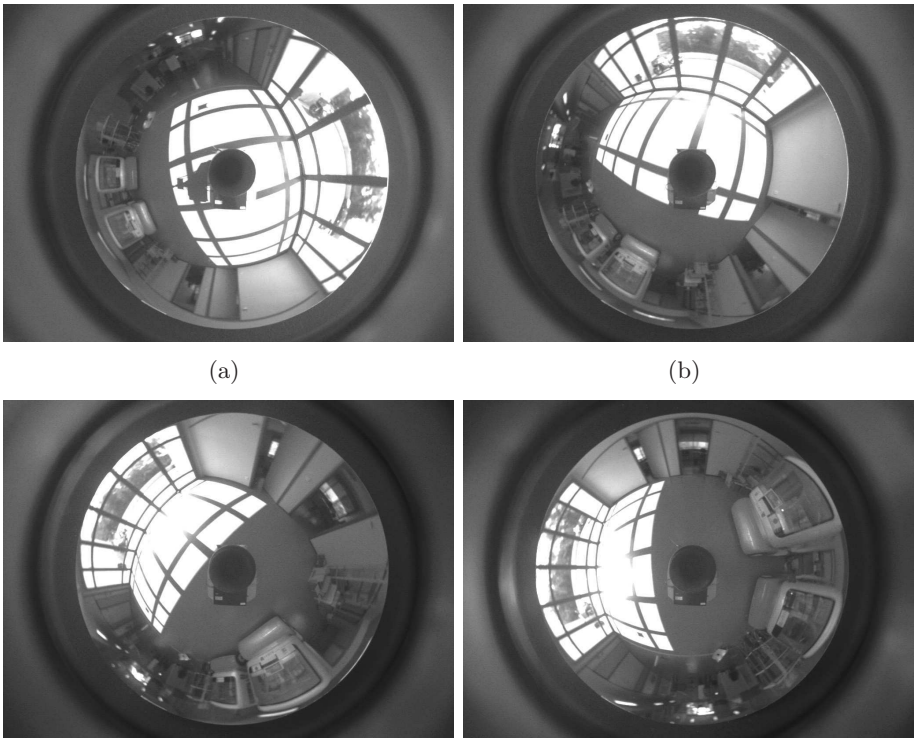
Il existe d'autres méthodes de mise en correspondances (utilisant des aspects géométriques, ou encore d'autres descripteurs...). Le but de ce chapitre n'est pas de toutes les décrire le lecteur intéressé pourra par exemple se référer à [Brandou, 2008] où plus de méthodes sont présentées.

4.2 Limitations de la vision perspective classique et technologies existantes pour les contourner

Nous avons présenté dans la section précédente les principaux aspects de la vision par ordinateur avec une caméra perspective classique. Nous proposons dans cette section de présenter les limitations de ce type de capteur. Nous verrons également les principales approches pour prendre en compte ces limitations.

Dans le cas de la vision perspective, nous avons utilisé la projection planaire : tous les points visibles sont projetés sur le plan unitaire. Certaines limitations apparaissent clairement. La première d'entre elle est que seul un demi-espace est " projetable " sur le plan. Cette limitation dans le modèle de projection est une conséquence logique du matériel utilisé : celui-ci ne permet de voir que ce qui est devant la caméra. En termes d'angle de gisement, cela implique **au mieux** une plage d'utilisation comprise dans l'intervalle $[-\pi/2, \pi/2]$. En pratique, il n'est pas possible d'atteindre un champ de vision aussi élevé.

Pourtant, il apparaît très intéressant dans le cadre du SLAM de pouvoir bénéficier d'un angle de



(a) (b)

(c) (d)

FIGURE 4.3 – Intérêt d'un champ de vision à 360 degrés — Dans le cas de ce demi-tour rapide, l'intégralité de la scène reste visible entre le début de la manœuvre (a) et la fin (d). Ceci permet d'obtenir une très bonne continuité dans le suivi des points. Le résultat du SLAM se retrouve alors mieux contraint que dans le cas où la moitié des points auraient disparus puis seraient réapparus.

vision élevé. Ceci permet de garder des points d'intérêt dans le champ de vision dans le cas de grandes rotations du robot. La figure 4.3 illustre cet aspect. On peut par ailleurs remarquer que la plupart des autres approches de SLAM (autres que visuelles) emploient des capteurs qui permettent de percevoir l'environnement à 360 degrés (c'est notamment le cas des approches utilisant des lasers rotatifs ou des ceintures d'ultrasons). Plusieurs approches technologiques permettent d'étendre le champ de vue dans le cas du SLAM visuel. Un état de l'art des différentes approches est proposé dans [Yagi, 1999]. Nous retiendrons surtout les 3 approches suivantes :

1. **Objectif très grand angle (caméra *fish-eye* en anglais) :** dans ce cas, la caméra utilisée permet de couvrir un champ de vision proche de 180 degrés, au prix d'importantes distorsions (cf. figure 4.4). Ce type de caméra est notamment utilisée dans [Royer *et al.*, 2007]. Néanmoins, les caméras *fish-eye* ne permettent pas une vision à 360 degrés, ce qui les rend finalement à peine meilleures que les caméras perspectives classiques.

2. **Plusieurs images perspectives :** dans ce cas, l'idée est d'utiliser plusieurs caméras perspectives classiques montées sur une " ceinture " pour avoir une image à 360 degrés (figure 4.5). L'inconvénient majeur est le traitement indispensable pour obtenir l'image finale. Par ailleurs, l'image obtenue étant composée de nombreuses images numériques classiques, l'image finale est de très haute résolution, ce qui rend les traitements bas niveaux très longs à réaliser. Enfin, il est nécessaire de calibrer les changements de repères entre les caméras. Pour ces raisons, nous choisissons de ne pas utiliser ce type de matériel pour notre application. Ce type de caméra est en revanche utilisée par l'IGN et également par l'entreprise Google pour son application *Street View.*[7]

3. **Caméra avec miroir convexe :** dans ce cas, la caméra filme un miroir convexe qui reflète la scène à 360 degrés. On a ainsi de véritables images omnidirectionnelles (comme sur la figure 4.3) au prix d'une distorsion importante dont il faut tenir compte (ce qui implique d'utiliser des techniques adaptées de traitement d'images). Par ailleurs, la résolution des images utilisées est raisonnable en comparaison à la résolution de la " super-image " que l'on peut obtenir avec la seconde méthode. Des photos de ce type de caméras sont présentées sur la figure 4.6. L'inconvénient majeur est le montage mécanique sujet aux vibrations (le mouvement du miroir par rapport à la caméra). Par ailleurs, les miroirs sont difficiles à usiner et il faut s'assurer que l'axe optique de la caméra soit aligné avec l'axe de révolution du miroir.

Au final, nous utiliserons dans la suite une caméra couplée à un miroir convexe ie. une **caméra omnidirectionnelle**. Les dynamiques que nous utiliserons dans nos expérimentations sont compatibles avec l'utilisation de ce type de caméra (en termes de vibrations). Nous consacrons donc la suite de cette section à la description des principaux aspects liés à la vision omnidirectionnelle.

4.3 Caméra omnidirectionnelle

Nous présentons dans cette section les principaux aspects liés à la vision omnidirectionnelle. Dans un premier temps, nous montrons les particularités de la modélisation et notamment l'utilisation de la projection sphérique. Nous évoquons ensuite les modifications à apporter quant aux traitements classiques.

7. Eventuellement, le système multi-caméras peut être remplacé par une caméra unique montée sur une tourelle *pan and tilt*. Les temps d'acquisitions sont alors longs (ce qui est incompatible avec une application en robotique mobile) et les traitements " d'assemblage " sont toujours présents.

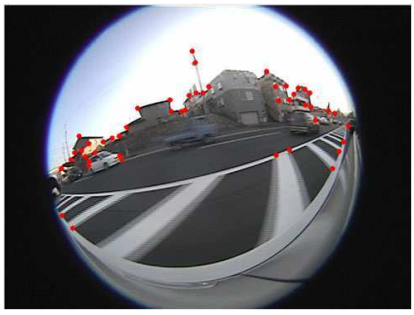


FIGURE 4.4 – Exemple d'image *fish-eye* ([Li et Shimomura, 2008])

(a) (b)

FIGURE 4.5 – Ceinture de caméras utilsée par l'IGN dans le cadre du projet Stereopolis ([Paparoditis, 2000]) — (a) Schéma du montage – (b) Exemple d'image très haute résolution obtenue (20 mégapixels)

(a) (b)

FIGURE 4.6 – Deux caméras omnidirectionnelles — (a) Montage utilisant un miroir hyperbolique – (b) Montage utilisant un miroir parabolique.

4.3.1 Modélisation de la vision omnidirectionnelle

4.3.1.1 Introductions aux différents modèles

Nous présentons ici la modélisation des capteurs **centraux catadioptriques**. Les capteurs sont dits centraux s'ils satisfont la contrainte **d'un centre de projection unique**. Ainsi, un point de l'espace 3D ne peut se projeter que sur un seul point de l'image. Il s'agit d'une contrainte qui limite la forme des miroirs utilisables. Par exemple, les miroirs sphériques ne satisfont pas cette contrainte (cf. figure 4.7). La présence de plusieurs centres de projection induit des propriétés particulières qu'il est difficile d'exploiter ; c'est pourquoi nous nous limitons à l'étude des capteurs centraux.

Les capteurs envisagés incluent la combinaison d'une caméra classique et d'un miroir dont la forme est une quadrique de révolution parmi :

1. **l'hyperboloïde**
2. **l'ellipsoïde**
3. **la paraboloïde** : dans ce cas, on ajoute une lentille télécentrique dans le montage car après projection sur le miroir tous les rayons sont parallèles à l'axe optique de la caméra ([Mei, 2007])
4. **un plan (cas dégénéré)** : dans ce cas, on se ramène au cas de la vision perpective classique. L'utilisation d'un miroir plan n'a donc aucune utilité pratique. Néanmoins, il existe un intérêt théorique qui apparaîtra au moment de la description du modèle unifié (paragraphe 4.3.1.2).

Les modèles de projection associés à ces quatre classes de capteurs sont présentés sur la figure 4.8. Il apparaît alors que l'utilisation de quatre modèles différents peut s'avérer pénalisant en termes de modélisation. João P. Barreto et Christopher Geyer ont montré dans leurs thèses qu'il est possible d'unifier ces classes de capteurs au sein d'un seul et unique modèle, au prix de la définition de nouveaux paramètres ([Barreto, 2003; Geyer, 2003]). Le modèle de projection utilisé est **sphérique**. Nous

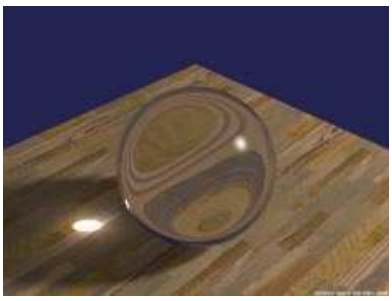


FIGURE 4.7 – Exemple de caustique due à la réfraction d'un rayon sur une sphère ([Swaminathan *et al.*, 2006])

présentons ici une définition légèrement modifiée du modèle de Barreto et Geyer ; il s'agit de la version proposée dans la thèse de Christopher Mei ([Mei, 2007]).

4.3.1.2 Modèle de projection unifié

Nous présentons dans ce paragraphe le modèle de projection unifié utilisé. L'ensemble des notations de ce paragraphe font référence à la figure 4.9. Soit $\mathbf{M}$ un point de l'espace 3D vu par le capteur catadioptrique et dont les coordonnées dans le repère du miroir (repère $\mathcal{F}_m$) sont $[x_M, y_M, z_M]^T$. La première étape de la projection consiste à définir la projection de $\mathbf{M}$ sur la sphère unité :

$$\begin{aligned} \mathbf{M}_S = \frac{\mathbf{M}}{\|\mathbf{M}\|} &= \left[\frac{x_M}{\sqrt{x_M^2+y_M^2+z_M^2}} \quad \frac{y_M}{\sqrt{x_M^2+y_M^2+z_M^2}} \quad \frac{z_M}{\sqrt{x_M^2+y_M^2+z_M^2}}\right]^T \\ &= \begin{bmatrix} x_S & y_S & z_S \end{bmatrix}^T \end{aligned} \tag{4.19}$$

Ensuite, les coordonnées de $\mathbf{M}_S$ sont exprimées dans le repère centré sur le point $\mathcal{C}_p$ dont les coordonnées dans le repère $\mathcal{F}_m$ sont $[0, 0, -\xi]^T$(cf. figure 4.9), où $\xi \in [0, 1]$ est un paramètre caractérisant la géométrie du miroir. Ceci définit donc le point $\mathcal{M}_S$ par :

$$\mathcal{M}_S = \begin{bmatrix} x_S & y_S & z_S + \xi \end{bmatrix}^T \tag{4.20}$$

Le point obtenu est alors projeté sur le plan unitaire en $\mathbf{m}$ (dans le repère centré sur $\mathcal{C}_p$) :

$$\mathbf{m} = \begin{bmatrix} \frac{x_S}{z_S+\xi} & \frac{y_S}{z_S+\xi} & 1 \end{bmatrix}^T \tag{4.21}$$

Les coordonnées en pixels dans le repère image sont finalement données par :

$$\mathbf{p} = \mathbf{K}\mathbf{m} \tag{4.22}$$

où $\mathbf{K}$ représente la matrice des paramètres intrinsèques d'une *caméra virtuelle* et vaut :

$$\mathbf{K} = \begin{bmatrix} \gamma_1 & s\gamma_1 & u_0 \\ 0 & \gamma_2 & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.23}$$

où u_0 et v_0 sont les coordonnées dans l'image de l'axe optique (correspondant au u_0 et v_0 de la caméra réelle), s le paramètre d'obliquité de la caméra (correspondant au s de la caméra réelle). γ_1 et γ_2 sont les distances focales (exprimées en pixels) de la caméra virtuelle : il s'agit des **distances focales généralisées**. Celles-ci dépendent de la géométrie du miroir et des distances focales de la caméra réelle. Les relations entre les paramètres du modèle unifié et les caractéristiques physiques du montage réel sont donnés dans le tableau **??** (dans le cas idéal où $f_u = f_v$ et donc $\gamma_1 = \gamma_2$).

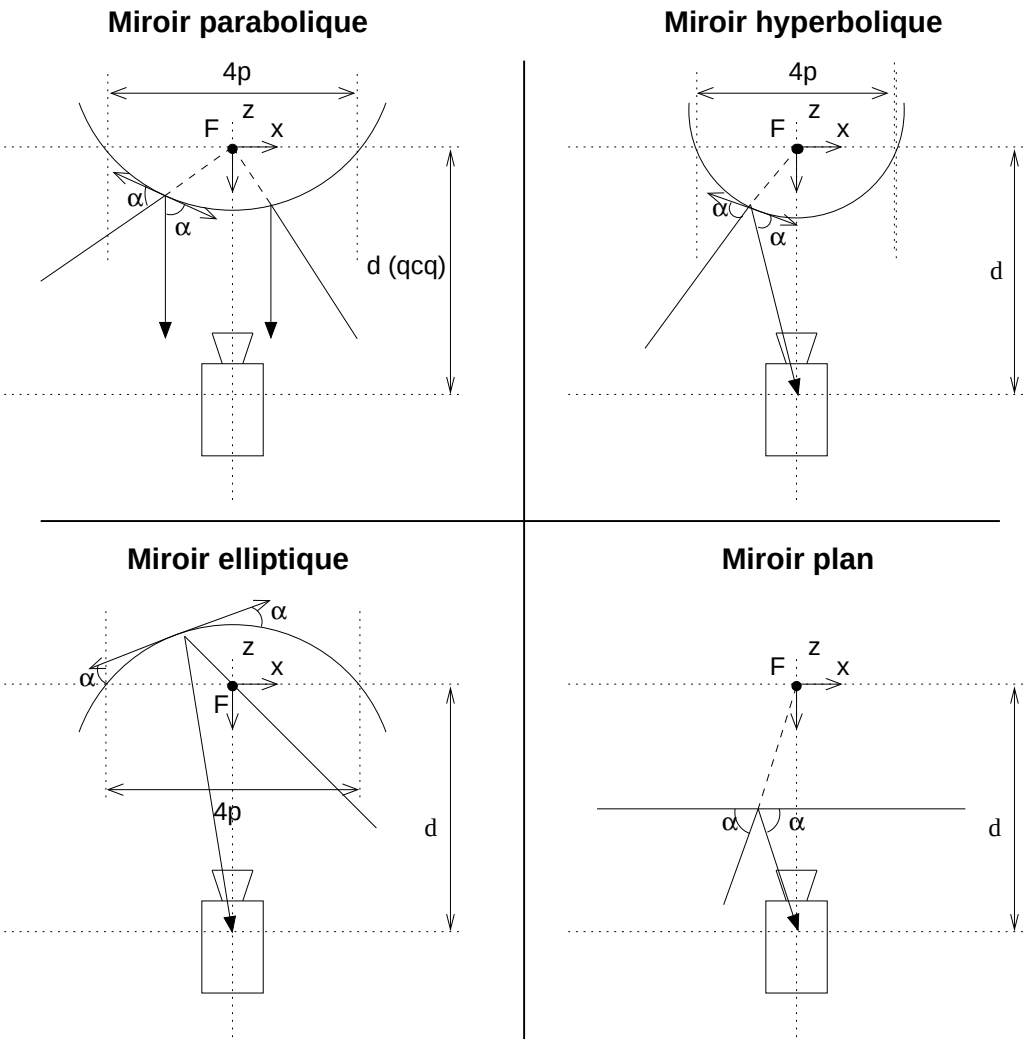


FIGURE 4.8 – Les classes de capteur catadioptriques centraux (adapté de [Mei, 2007]) — Pour chaque sous-figure, le point **F** désigne le foyer de la quadrique, et d la distance entre le foyer et le centre optique de la caméra. Dans le cas de la caméra parabolique, la distance d n'entre jamais en jeu dans les calculs car les rayons paraissent venir de l'infini (ce qui justifie l'utilisation de la lentille télécentrique)

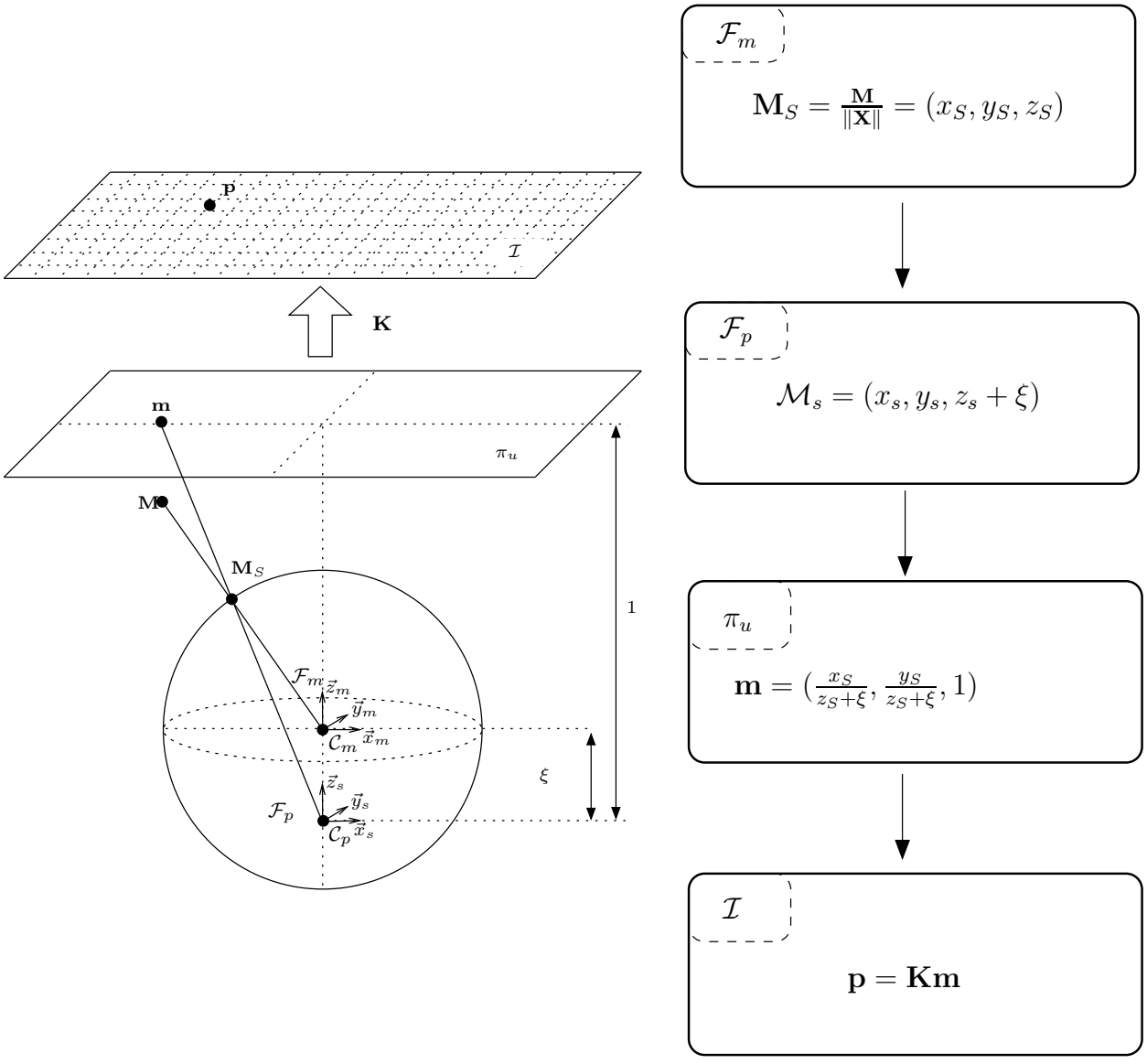


FIGURE 4.9 – Le modèle de projection unifié (adapté de [Mei, 2007])

	ξ	γ
Miroir parabolique	1	$-2pf$
Miroir hyperbolique	$\frac{df}{\sqrt{d^2+4p^2}}$	$\frac{-2pf}{\sqrt{d^2+4p^2}}$
Miroir elliptique	$\frac{df}{\sqrt{d^2+4p^2}}$	$\frac{2pf}{\sqrt{d^2+4p^2}}$
Miroir plan	0	-f
Cas perspectif	0	f
d : distance focale $4p$: latus rectum		

TABLE 4.1 – Paramètres du modèle unifié en fonction des paramètres physiques de la caméra et du miroir, dans le cas idéal où $f_u = f_v = f$ et donc $\gamma_1 = \gamma_2 = \gamma$

Remarque 4.1 (Calibration d'un capteur omnidirectionnel)

Le modèle de projection unifié nous permet de considérer la caméra omnidirectionnelle comme un capteur à part entière. Ainsi, les algorithmes de calibration utilisés estiment directement les paramètres γ_1, γ_2, ξ, s, u_0 et v_0 sans chercher à retrouver les paramètres géométriques du miroir ainsi que les paramètres de la caméra réelle ([Mei et Rives, 2006a,b; Mei, 2007]). Des méthodes d'autocalibration ont également été développées pour le cas omnidirectionnel ([Salazar-Garibay et Malis, 2009; Salazar-Garibay et al.*, 2009]) : les paramètres sont alors évalués grâce à un suivi de plan faisant appel à une méthode dense de minimisation.)*

Les étapes décrites précédemment permettent de calculer la position d'un point dans l'image lorsqu'on connaît sa position 3D. L'intérêt du modèle de projection unifié est qu'il permet de regrouper au sein d'un même modèle de projection sphérique l'ensemble des capteurs catadioptriques centraux. Par ailleurs, on peut remarquer que le cas de la caméra perspective planaire est un cas particulier très proche du capteur catadioptrique utilisant un miroir plan (différence d'un signe dans le modèle, due à la réflexion du rayon sur le miroir plan). On pourra également remarquer que l'intégralité de la sphère n'est pas utilisable : les points qui vérifient $z_S < -\xi$ ne sont pas utilisables (ie. toute la partie de la sphère sous le point $\mathcal{C}_p$ [8]). On retrouve alors bien la limitation du cas plan (un demi-espace non observable). Le cas du miroir parabolique est particulier puisque toute la sphère sauf le pôle du bas est observable ($\xi = 1$).

On remarquera enfin que la façon " classique " d'orienter une caméra omnidirectionnelle (axe $\mathbf{z}$ vers le " haut ") fait que cette limitation ne concerne que l'angle d'élévation. On conserve bien un gisement

8. Si on considère la partie inférieure de la sphère, on perd la bijectivité de projection sphère-image (il est alors impossible de de définir le modèle unifié inverse).

à 360 degrés (ce qui est le plus important en robotique terrestre où la plupart des déplacements sont réalisés " quasi-horizontalement ").

4.3.1.3 Modèle de projection unifié inverse

Nous avons présenté au paragraphe précédent le modèle de projection unifié direct (on part d'un point 3D et on le projette dans l'image). Considérons maintenant un pixel $\mathbf{p}$ de l'image donné en coordonnées homogènes. Pour retrouver les coordonnées du point associé sur la sphère, on commence par retrouver le point $\mathbf{m} = [x_m, y_m, 1]^T$ par multiplication à gauche par $\mathbf{K}^{-1}$: $\mathbf{m} = \mathbf{K}^{-1}\mathbf{p}$. On peut enfin montrer que les coordonnées du point associé sur la sphère ($\mathbf{M}_S$) sont données par :

$$\mathbf{M}_S = \begin{bmatrix} \frac{\xi+\sqrt{1+(1-\xi^2)(x_m^2+y_m^2)}}{x_m^2+y_m^2+1} x_m \\ \frac{\xi+\sqrt{1+(1-\xi^2)(x_m^2+y_m^2)}}{x_m^2+y_m^2+1} y_m \\ \frac{\xi+\sqrt{1+(1-\xi^2)(x_m^2+y_m^2)}}{x_m^2+y_m^2+1} - \xi \end{bmatrix} \tag{4.24}$$

4.3.2 Adaptation des descripteurs classiques et mise en correspondance

Nous avons présenté dans la section 4.1.3 le détecteur de Harris ainsi que le descripteur SIFT. Il est important de noter que ces opérateurs ont été définis directement dans l'espace de l'image et pour le cas de caméras perspectives classiques. Ceci a une légitimité certaine car le modèle de projection sous-jacent utilisé est un modèle planaire (on passe du plan unitaire au plan image par multiplication par une matrice). Dans le cas des images omnidirectionnelles, le modèle de projection sphérique est utilisé ; le passage de l'espace image à la sphère n'est plus linéaire. En conséquence, il apparaît en pratique que l'utilisation des opérateurs classiques ne donne pas de bons résultats dans le cas des images omnidirectionnelles. Par exemple, ceux-ci ne prennent pas en considération le fait que la résolution augmente au fur et à mesure que l'on s'éloigne du centre de l'image [9] (ceci est particulièrement visible sur les images de la figure 4.3).

Il paraît donc nécessaire de prendre en compte ces phénomènes lors de la détection des points d'intérêt et du calcul des descripteurs. Hicham Hadj-Habdelkader a montré dans son travail de post-doctorat à l'INRIA que les résultats pour la détection et mise en correspondance sont meilleurs lorsqu'on effectue les calculs sur la sphère ([Hadj-Abdelkader *et al.*]). Cela implique de redéfinir les gradients, noyaux de convolutions ainsi que tous les opérateurs dans l'espace sphérique. Par ailleurs, il se pose également le problème d'échantillonnage de la sphère. Nous ne décrivons pas les méthodes

9. Les objets sur les bords de l'image sont plus précis qu'au centre où tout se contracte.

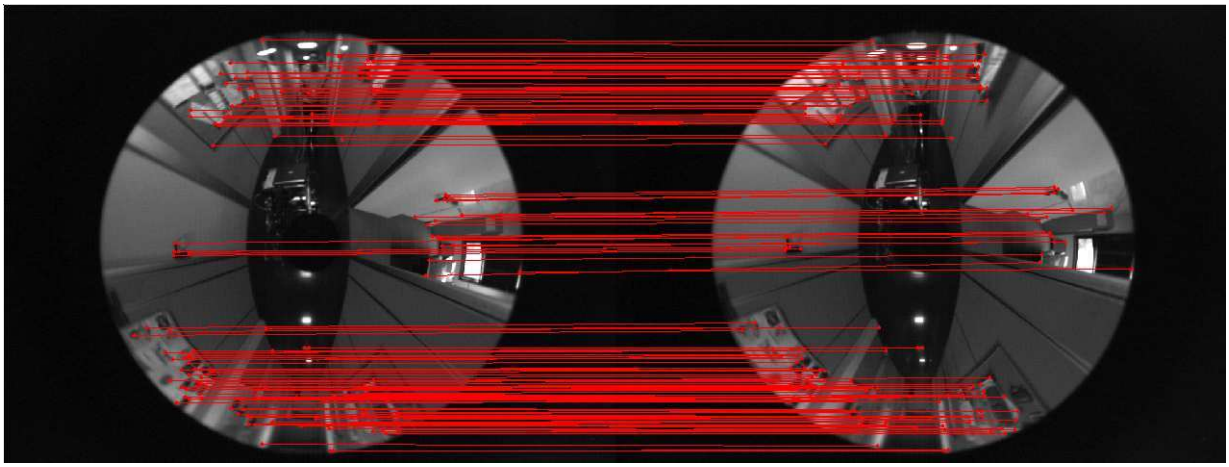


FIGURE 4.10 – Mises en correspondances de points entre deux images omnidirectionnelles.

employées pour adapter les détecteurs et descripteurs. Nous retiendrons seulement que la " philosophie " des algorithmes reste identique à celle décrite pour le cas perspectif. Le lecteur intéressé par les détails techniques pourra se référer à [Hadj-Abdelkader *et al.*].

Au final, notre approche pour le suivi des points consiste à détecter les points de Harris dans les images omnidirectionnelles. Un descripteur adapté est utilisé pour une première mise en correspondance. Enfin, un algorithme de RANSAC (adapté à la géométrie épipolaire du modèle sphérique de caméra) permet d'éliminer les principales mauvaises correspondances. Un exemple est présenté sur la figure 4.10.

4.4 Conclusion

Au final, nous avons présenté dans ce chapitre les principales notions de vision par ordinateur (classique et omnidirectionnelle). Nous avons donné les bases de la vision perspective classique. Nous avons montré ensuite les adaptations classiques effectuées dans le cadre de la vision omnidirectionnelle.

La méthode que nous avons proposée pour le suivi de points est une méthode très classique (détection de points de Harris puis utilisation d'un descripteur SIFT). Aujourd'hui, il existe des détecteurs et algorithmes de mises en correspondances plus performants. On pourra par exemple citer les points SURF et leurs dérivés ([Bay *et al.*, 2006; He *et al.*, 2009]), les améliorations des approches RANSAC ([Sattler *et al.*, 2009]), etc... Un état de l'art récent et conséquent peut être trouvé dans [Tuytelaars et Mikolajczyk, 2008]. Néanmoins, la plupart des méthodes proposées sont issues du traitement des images classiques. Pour l'utilisation avec les images omnidirectionnelles, une adaptation

non triviale est en général nécessaire.

En conclusion, il existe encore un travail important à effectuer avant d'obtenir un algorithme de suivi de points optimal. Dans la suite de ce document, nous utilisons les algorithmes proposés dans [Hadj-Abdelkader *et al.*] qui ont le mérite d'être adaptés aux images omnidirectionnelles. Cela nous a permis d'obtenir les résultats présentés dans les chapitres suivants. En pratique, on a noté que les algorithmes utilisés fournissent de bons résultats mais ont un temps de calcul non adapté aux contraintes temps réel. Le choix d'une méthode de suivi plus adaptée constitue un problème à part entière que nous n'avons pas résolu, notre travail de thèse étant essentiellement concentré sur les aspects haut-niveau liés au SLAM.

Dans la suite de cette thèse, on supposera donc le problème d'association des données résolu et découplé des méthodes de filtrage. On admettra par ailleurs qu'il peut exister un certain nombre de points aberrants dans les données que l'on essayera au mieux de filtrer.

Chapitre 5

Résultats de SLAM à 3 degrés de liberté

Sommaire

Nous présentons dans ce chapitre des résultats du SLAM à trois degrés de liberté obtenus avec les données acquises dans deux expérimentations. Ceux-ci ont été obtenus en utilisant l'algorithme de SAM décrit et évalué aux chapitres 1 et 3. Les expérimentations ont été effectuées avec deux robots évoluant **dans le plan** (le robot a deux degrés de liberté en translation et un en rotation) et dans des environnements d'intérieur. Nous utilisons une caméra omnidirectionnelle pour chaque expérimentation (miroir parabolique pour la première et miroir hyperbolique pour la seconde). Pour le traitement bas niveau des images, nous utilisons les techniques décrites au chapitre précédent. Par ailleurs, les informations visuelles seront augmentées par les données issues de l'odométrie.

Ce chapitre s'articule en cinq sections. Nous présentons dans la première section les principales caractéristiques des deux expérimentations. Nous y verrons la présentation générale des lieux et des trajectoires effectuées. Nous présenterons également dans cette section les modèles théoriques (équations d'évolution du robot et mesures) ainsi que le matériel utilisé. Nous présentons dans la section 5.2 l'algorithme utilisé pour le rejet de points aberrants. En effet, il existe encore de mauvaises mises en correspondances susceptibles de rendre l'algorithme inconsistant malgré l'utilisation du descripteur SIFT et de l'algorithme de RANSAC. Il faut donc prendre en compte cet aspect dans notre algorithme. La section 5.3, quant à elle, est consacrée à l'exposition et aux commentaires des résultats obtenus sur les deux jeux de données dans le cadre d'une implémentation Matlab hors ligne (les algorithmes n'ont pas été optimisés, le but est ici de vérifier la qualité des solutions obtenues). Nous verrons alors le bon comportement de l'algorithme sur des données réelles. Nous montrons dans la section 5.4 que l'algorithme de SAM peut être utilisé au delà du simple SLAM : nous verrons qu'il est possible de calibrer automatiquement la position de la caméra dans le repère du robot. Enfin, une synthèse des résultats est présentée en conclusion (section 5.5). Nous y montrerons les principaux avantages, inconvénients ainsi que les évolutions envisagées de l'algorithme et des conditions d'expérimentations.

5.1 Description des expérimentations

Cette section est consacrée à la description globale des expérimentations. Nous présentons dans un premier temps les conditions expérimentales ainsi que les trajectoires effectuées. Nous présentons ensuite le contexte théorique utilisé (équations de modèle et de mesures). Enfin, nous présentons le matériel utilisé : les robots (paragraphe 5.1.3) ainsi que les caméras (paragraphe 5.1.4).

5.1.1 Présentation générale

Dans la suite de ce chapitre, nous présentons deux expérimentations. Chacune d'entre elles a été réalisée avec un robot différent et dans un lieu différent. La première expérimentation a été réalisée avec l'ancien robot ANIS de l'INRIA Sophia Antipolis-Méditerranée dans les couloirs du bâtiment Borel. La seconde expérimentation, quant à elle, a été réalisée avec la nouvelle plateforme mobile (acquise par l'équipe ARobAS de l'INRIA Sophia Antipolis-Méditerranée) dans la halle robotique du bâtiment Kahn.

5.1.1.1 Expérimentation *Borel*

L'expérimentation réalisée dans le bâtiment Borel consiste en la réalisation d'une trajectoire plane avec le robot ANIS. Le lieu considéré est un **couloir** dans lequel la planéité est garantie. De plus, l'aspect couloir fait qu'il est facile de vérifier certaines propriétés géométriques permettant une comparaison avec une vérité terrain. En effet, les couloirs considérés ont une largeur fixe et contiennent de nombreux posters accrochés dessus (on s'attend à y trouver de nombreux points d'intérêts). De plus, la trajectoire contient plusieurs virages à angle droit : on s'attend donc dans le résultat final à retrouver la structure à angle droit des murs. En conséquence, la cohérence des résultats tirés de cette première expérimentation sera donc relativement facile à vérifier étant donné que la géométrie des lieux est suffisamment simple. Des images extraites de la séquence sont présentées sur la figure 5.1. La structuration des lieux apparaît clairement. On pourra également noter que le couloir utilisé forme un carré. Il y avait donc la possibilité d'effectuer une trajectoire contenant une *fermeture de boucle*. Néanmoins, un problème lors de l'acquisition des données a empêché l'exploitation de la partie de la séquence où l'on revient au point initial. [1] Enfin, une partie de la trajectoire est pratiquement dénuée d'informations visuelle : une partie des images de la séquence contient très peu de texture (figure 5.2).

5.1.1.2 Expérimentation *Kahn*

La séquence acquise dans le bâtiment Kahn est assez différente de la précédente. Ici, l'environnement est une halle robotique et il est beaucoup plus ouvert que les couloirs du bâtiment Borel. Au niveau structurel, on trouve un long mur et quelques angles droits. Mais l'ensemble des lieux ne permet pas d'accéder à une vérité terrain permettant une vérification de la cohérence aussi facile que pour l'expérimentation précédente.

1. Précisons également que cette séquence a été réalisée par Christopher Mei dans le cadre de sa thèse, avant même le début de ce travail.

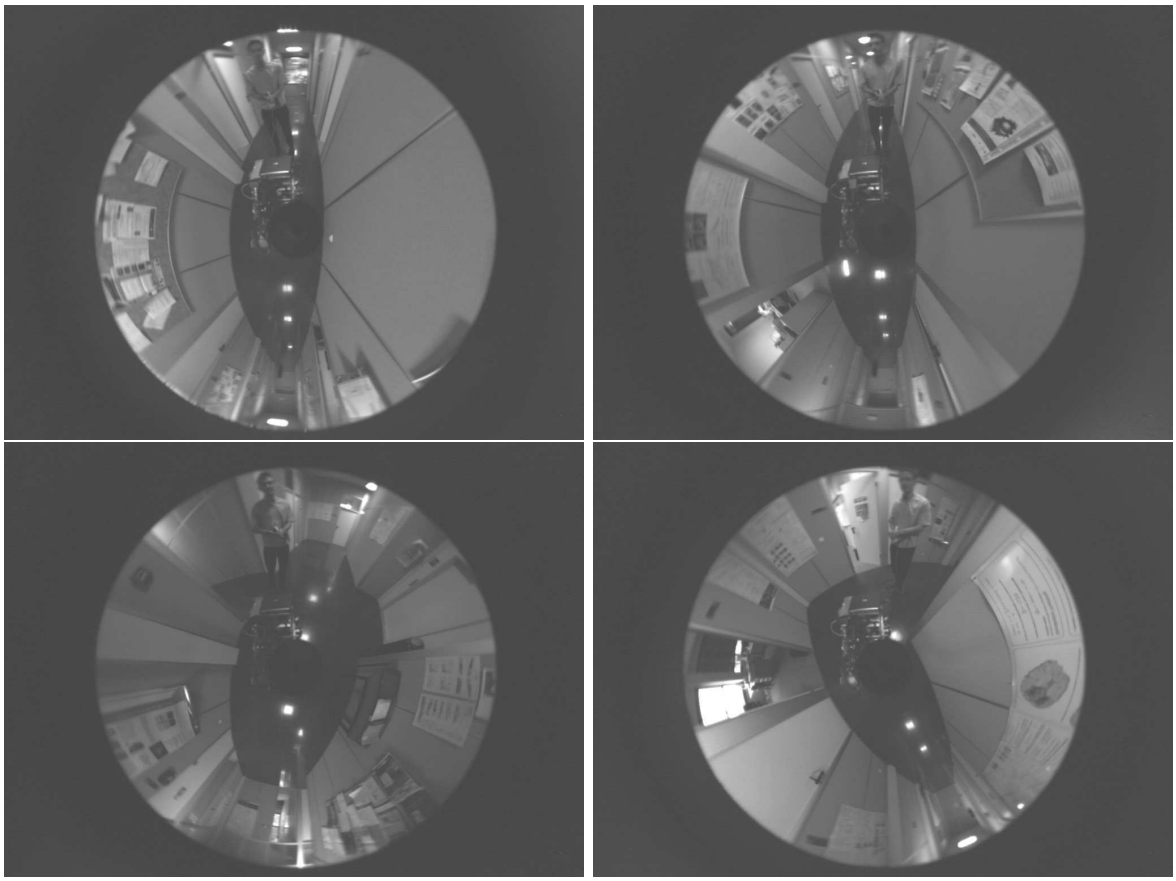


FIGURE 5.1 – Images extraites de la séquence acquise dans le bâtiment Borel

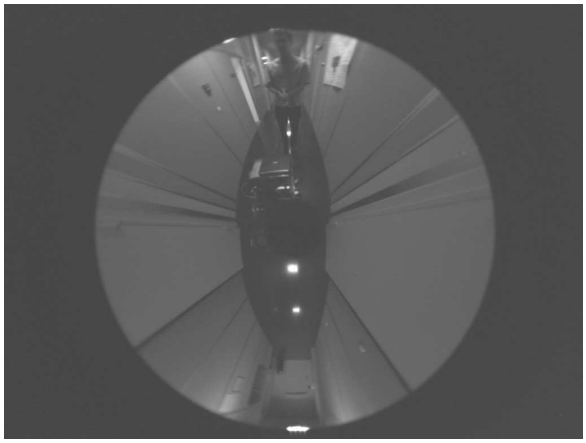
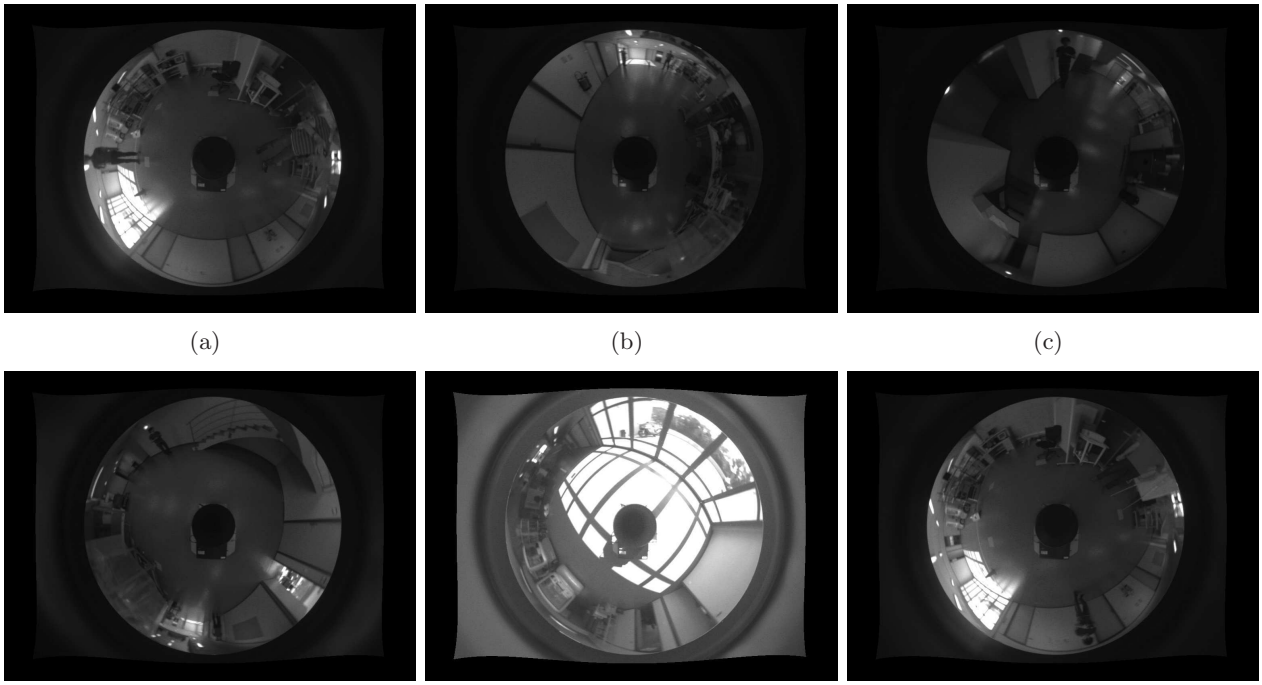


FIGURE 5.2 – Image faiblement texturée dans la séquence *Borel*



(a) (b) (c)

(d) (e) (f)

FIGURE 5.3 – Images extraites de la séquence acquise dans le bâtiment Kahn

Par ailleurs, nous avons effectué une trajectoire assez " tortueuse ", dont un demi-tour très serré. Des extraits de la séquence sont présentés sur la figure 5.3. On se rend compte ici de la diversité de l'environnement (même si on reste dans le cadre d'un environnement d'intérieur). On passe en effet de zones très dégagées (figure 5.3(a)) à des zones plus étroites comme celles sur la figure 5.3(c) (où un demi-tour très serré est effectué). De plus, la présence d'une baie vitrée (figure 5.3(e)) et de morceaux de couloir très sombres (figure 5.3(c)) fait que l'on a des différences d'éclairage très importantes le long de cette séquence. Pour pallier ce problème, nous avons utilisé un mode spécial de la caméra : le mode HDR (*High Dynamic Range*). Ce mode permet de compenser les zones surexposées afin de garder de l'information même dans les zones de fortes luminosité (typiquement la baie vitrée). Le prix de ce traitement est l'ajout de bruit dans les images, ce qui rend les traitements bas-niveaux moins fiables. Ainsi, cette seconde expérimentation nous rapproche plus des cas réels que la première. Enfin, nous avons fait en sorte que le début de la trajectoire (figure 5.3(a)) et la fin (figure 5.3(e)) soient au **même** endroit.[2] Cette donnée de *fermeture de boucle* nous permettra de contrôler la cohérence du résultat en l'absence de la possession d'une carte donnant une vérité terrain.

2. Nous avons fait démarrer l'expérimentation au niveau d'un **marqueur** du sol et avons piloté le robot jusqu'à ce marqueur à nouveau.

En revanche, nous n'avons pas utilisé la connaissance de cette fermeture de boucle dans l'algorithme. Ceci nous permet de tester les capacité de l'algorithme de SAM à superposer deux éléments physiquement identiques alors qu'ils sont détectés comme deux éléments différents. Néanmoins, un algorithme complet devrait introduire la possibilité de détecter automatiquement les fermetures de boucle. Pour cela, diverses approches existent dans la littérature. Les plus basiques utilisent le résultat de l'algorithme de filtrage pour prédire une possible fermeture de boucle. Des algorithmes plus élaborés cherchent des informations visuelles pertinentes dans les images ([Ho et Newman, 2006]). D'autres algorithmes utilisent des informations topologiques et des représentations issues des techniques d'indexation d'images pour détecter les fermetures de boucles de manière robuste. On pourra par exemple citer les méthodes basées sur les sacs de mots ([Angeli *et al.*, 2008, 2009]). Ces méthodes de fermeture de boucle n'ont été ni évaluées, ni implémentées dans ce travail de thèse.

5.1.2 Modèles théoriques

Nous donnons dans ce paragraphe la paramétrisation utilisée ainsi que les équations associées au modèle de déplacement du robot et au modèle de mesures.

5.1.2.1 Paramétrisation

La paramétrisation utilisée dans ce chapitre est la même que celle utilisée dans le chapitre 3 (page 85). Nous la rappelons ci-dessous :

$\mathbf{x}_t$: il s'agit de la pose du robot à l'instant t, à savoir la position du robot dans le plan (coordonnées absolues 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. La position du robot utilisée pour décrire l'état est le **centre des roues arrière** (l'équation d'intégration de l'odométrie est alors simplifiée). On a :

$$\mathbf{x}_t = \begin{bmatrix} x_t & y_t & \theta_t \end{bmatrix}^T \tag{5.1}$$

$\mathbf{u}_t$: le robot est vu comme un solide en mouvement, animé par un torseur cinématique qui sera utilisé comme vecteur de commandes. 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 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). Une mesure de ce vecteur de commandes est fournie par l'**odométrie du robot**, par le biais d'une vitesse de

translation axiale intégrée sur un cours laps de temps (ds_t) et d'une vitesse de rotation intégrée sur ce même laps de temps ($d\omega_t$). Enfin, notre robot a une contrainte de **non-honolomie** : sa vitesse transversale est nulle (aux légers glissements près). On a donc :

$$\mathbf{u}_t = \left[\begin{array}{ccc} ds_t & 0 & d\omega_t \end{array} \right]^T \tag{5.2}$$

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

$$\mathbf{m}_{(i)} = \left[\begin{array}{ccc} x_{(i)} & y_{(i)} & z_{(i)} \end{array} \right]^T \tag{5.3}$$

$\mathbf{z}_t$ **:** Pour cette série d'expérimentations, nous avons transformé les mesures brutes (ie coordonnées pixeliques) en angles de gisement ($\alpha_{t,(i)}$) et d'élévation ($\beta_{t,(i)}$).[3] **Ces angles correspondent aux coordonnées sphériques de la projection sur la sphère liée au repère de la caméra. Ce repère n'est pas confondu avec celui du robot.** On a donc :

$$\mathbf{z}_{t,(i)} = \left[\begin{array}{cc} \alpha_{t,(i)} & \beta_{t,(i)} \end{array} \right]^T \tag{5.4}$$

5.1.2.2 Equation d'évolution du robot

L'équation d'évolution permet de fournir une prédiction de l'état du robot à l'instant t connaissant l'état du robot à l'instant $t-1$ et les mesures de l'odométrie. L'équation utilisée est la même que celle décrite au chapitre 3 (équation 3.8 page 88). On a donc :

$$\left\{ \begin{array}{rcl} x_{t+1} & = & x_t + \mathrm{sinc}\left(\frac{d\omega_t}{2}\right) ds_t \cos\left(\theta_t + \frac{d\omega_t}{2}\right) \\ y_{t+1} & = & y_t + \mathrm{sinc}\left(\frac{d\omega_t}{2}\right) ds_t \sin\left(\theta_t + \frac{d\omega_t}{2}\right) \\ \theta_{t+1} & = & \theta_t + d\omega_t \end{array} \right. \tag{5.5}$$

La matrice jacobienne associée est identique à la matrice $\mathbf{F}_{t+1}$ donnée page 89.

5.1.2.3 Equation de mesures

L'équation de mesures associée est dans le cas réel légèrement différente de celle introduite dans le chapitre 3. En effet, les mesures sont effectuées dans le repère du miroir de la caméra. Celui-ci n'est pas confondu avec le centre de l'essieu arrière (repère du robot). Soient $\mathbf{t}$ et $\mathbf{R}$ le vecteur de translation et la matrice de rotation associés au changement de repère entre le repère du robot et le repère de la caméra.

3. Ceci nous permettra de vérifier que la variance associée à ces mesures n'est pas uniforme dans l'image (paragraphe 5.1.4.2).

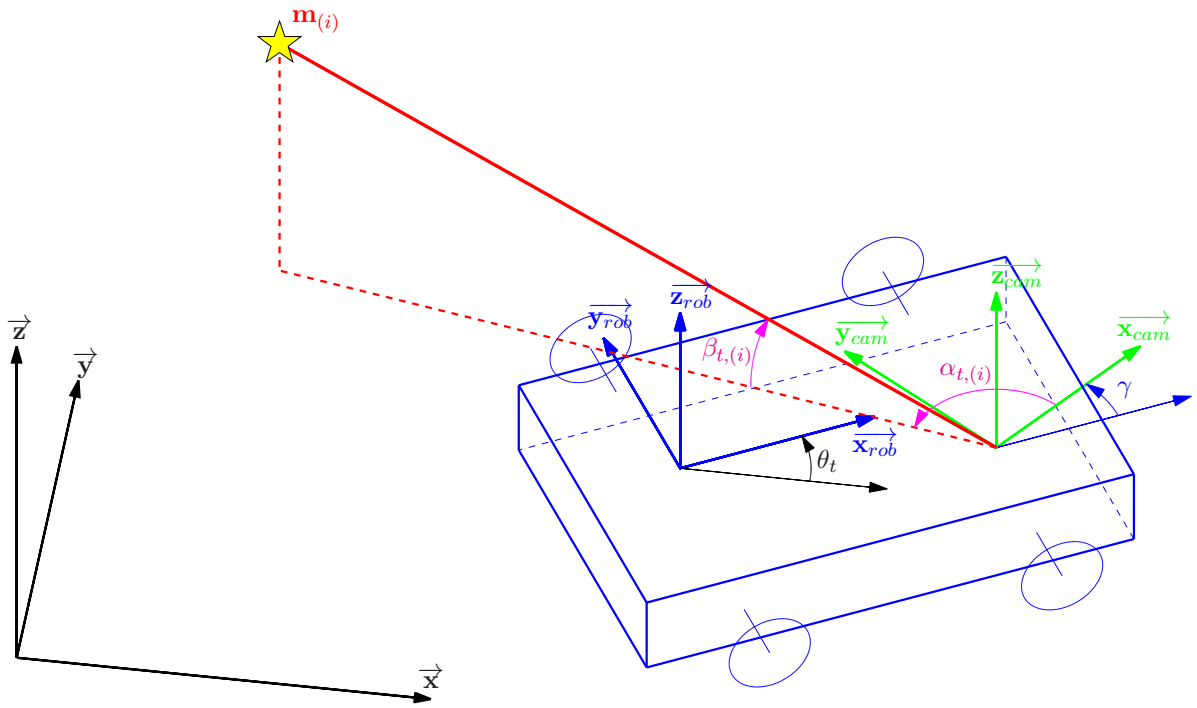


FIGURE 5.4 – Représentation des différents repères et paramètres dans le cas réel — Le repère global est représenté en noir. On retrouve en bleu le repère du robot, centré sur les roues arrière (tourné de l'angle de lacet θ_t par rapport au repère global). L'amer $\mathbf{m}_{(i)}$, quant à lui, est mesuré par rapport au repère associé à la caméra (en vert). Ce dernier repère est décalé par rapport au repère du robot et tourné d'un angle γ. Par convention, les repères de la caméra et du robot ont été placés à la même altitude.

Dans les deux expérimentations, la caméra est montée verticalement. [4] De plus, la coordonnée en altitude de la caméra n'a pas d'importance : elle génère un décalage constant de la trajectoire (dû au fait que la caméra est verticale). Ainsi, les altitudes des amers sont donnés par rapport au niveau en z de la caméra (ce qui n'a aucune importance en pratique). Ces considérations étant faites, on considère le changement de repère entre le robot et la caméra paramétré par le vecteur $[t_x\ t_y\ \gamma]^T$, où t_x et t_y désignent la partie en translation et γ la rotation entre les deux repères (figure 5.4). Finalement, l'équation de mesures est donc une version légèrement modifiée de l'équation 3.9 page 88. On a :

$$\begin{cases} \alpha_{t,(i)} = \arctan2\left(y_{(i)} - y_t - t_y\cos\theta_t - t_x\sin\theta_t \ , \ x_{(i)} - x_t - t_x\cos\theta_t + t_y\sin\theta_t \right) - \theta_t - \gamma \\ \beta_{t,(i)} = \arctan\left(\frac{z_{(i)}}{\sqrt{(x_{(i)} - x_t - t_x\cos\theta_t + t_y\sin\theta_t)^2 + (y_{(i)} - y_t - t_y\cos\theta_t - t_x\sin\theta_t)^2}} \right) \end{cases} \quad (5.6)$$

4. Pour le confirmer, on peut vérifier que les droites verticales dans l'espace correspondent bien à des droites radiales dans les images omnidirectionnelles sur les figures 5.1, 5.2 et 5.3.

La matrice jacobienne $\mathbf{H}_{t,(i)}$ est plus compliquée que celle obtenue dans les équations 3.11, 3.12, 3.13 et 3.14 (pages 89 et 90). Son calcul n'a que peu d'intérêt et n'est pas présenté ici.

5.1.2.4 Remarque quant à la paramétrisation

Nous avons défini le vecteur d'état comme le centre de l'essieu arrière du robot. Ceci a pour conséquence première de rendre l'équation d'évolution du robot relativement simple. La difficulté concernant le décalage de la caméra se trouve donc reportée sur l'expression du vecteur de mesures ainsi que sur sa matrice jacobienne.

Une autre solution aurait consisté à définir la position de la caméra dans le vecteur d'état au lieu de la position du centre de l'essieu arrière. De manière duale, on aurait gardé l'équation de mesures relativement simple (la même que celle du chapitre 3) mais l'équation d'évolution de la caméra aurait été plus compliquée (faisant intervenir le " transport " du torseur cinématique).

A priori, les deux solutions doivent fournir des résultats quasi-équivalents. Notre préférence pour la première méthode vient du fait qu'elle permet une comparaison directe entre le résultat du filtre et l'intégration de l'odométrie brute.[5] Dans le cas de la seconde solution, cette comparaison aurait nécessité l'utilisation d'un terme de transport dépendant de l'orientation du robot (qui est estimée par l'algorithme). La comparaison aurait été moins précise étant donné qu'elle dépendrait d'un terme qui est lui même estimé.

5.1.3 Plate-formes expérimentales

Nous présentons dans ce paragraphe le robot utilisé pour chacune des expérimentations. Nous donnons dans un premier temps le modèle odométrique utilisé puis les paramètres et aspects spécifiques de chaque robot.

5.1.3.1 Modèle odométrique

Les deux robots utilisés sont commandés en vitesses de translation et de rotation. Les actionneurs sont les deux roues motrices. La vitesse de translation est commandée avec la moyenne des vitesses des deux roues, et la vitesse de rotation avec la vitesse différentielle des roues. Les roues étant équipées de codeurs, l'estimation des vitesses (translation et rotation) intégrées entre les instants $t-1$ et t est donnée par :

$$\left\{ \begin{array}{rcl} ds_t & = & \frac{2\pi R}{2}\left(\delta g_t + \delta d_t\right) \\ d\omega_t & = & \frac{\pi R}{L}\left(\delta g_t - \delta d_t\right) \end{array} \right. \tag{5.7}$$

5. Ceci est intéressant pour évaluer l'apport de la caméra dans le calcul de la solution.

où δg_t et δd_t désignent les incréments d'angles des roues gauche et droite entre les instants $t-1$ et t, R le rayon des roues et L la distance entre les deux roues arrière. dl_t et dr_t sont obtenus grâce à un codeur. Plusieurs sources d'erreurs sont susceptibles d'intervenir dans le calcul de ds_t et $d\omega_t$:

Erreur des codeurs : les codeurs ont en effet une résolution limitée sur les 360 degrés. L'angle reporté n'est donc jamais exact,

Erreurs de calibration du robot : si la distance L et le rayon des roues sont mal mesurés, une erreur systématique est introduite dans l'utilisation de l'équation 5.7,

Glissement : dans ce cas, les valeurs calculées pour ds_t et $d\omega_t$ ne sont pas conformes au déplacement réel du robot.

Les deux premiers types d'erreur impliquent un mauvais calcul du vecteur $\mathbf{u}$ et sont donc modélisés dans la matrice de variances-covariances associée à $\mathbf{Q^u}$. Le dernier type d'erreur est associé à une erreur de l'équation d'intégration de l'odométrie et pris en compte dans la matrice $\mathbf{Q^f}$.

5.1.3.2 Paramètres ANIS

La première expérimentation utilise le robot ANIS. Il s'agit d'un robot construit à l'INRIA Sophia Antipolis-Méditerranée dans le cadre de la thèse de Roger Pissard-Gibollet ([Pissard-Gibollet, 1993]). Ce robot est doté de 4 roues (2 roues arrière équipées d'odomètres sur lesquelles s'appuie la commande et deux roues folles à l'avant). Il a été conçu pour les manipulations d'intérieur : toutes ses roues ont des **surfaces de contact très fines et sont indéformables**. Ainsi, le rayon R est constant et facilement mesurable, tout comme la distance L. Nous considérons donc ces deux paramètres parfaitement calibrés. L'odométrie de ce robot est donc particulièrement précise. Une photographie de ce robot (avec le montage de la caméra omnidirectionnelle) est présentée sur la figure 5.5.

Ces considérations étant faites, la matrice $\mathbf{Q^u}$ ne dépend que de l'incertitude associée aux codeurs. Soit $\mathbf{Q_{cod}} = \mathbf{I}\sigma_{cod}^2$ la matrice de variances-covariances associée aux erreurs angulaires des codeurs (nous considérons la même variance pour chaque codeur). La matrice de variances-covariances associée à ds_t et $d\omega_t$ est donnée par :

$$\mathbf{Q^{interm}} = 2\left(\sigma_{cod}\pi R\right)^2 \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{L^2} \end{bmatrix} \tag{5.8}$$

La matrice $\mathbf{Q^u}$ telle que définie dans la section 3.3 prend en compte la possibilité d'une faible vitesse de translation dans l'axe transversal (correspondant alors à un léger glissement). On admet que la contrainte de non-honolomie est pratiquement réalisée et que les erreurs de modèle sur le déplacement

FIGURE 5.5 – Photographie du robot ANIS — On peut voir les deux roues avant (roues folles en plastique avec une surface de contact très faible) ainsi que le montage vertical de la caméra omnidirectionnelle.

transversal sont au moins d'un ordre de grandeur inférieur à l'erreur commise sur ds_t. Au final, on utilise :

$$\mathbf{Q^u} = 2\left(\sigma_{cod}\pi R\right)^2 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1/100 & 0 \\ 0 & 0 & \frac{1}{L^2} \end{bmatrix} \tag{5.9}$$

Il est également nécessaire de définir la matrice $\mathbf{Q^f}$. Il s'agit de la matrice de variances-covariances associée au vecteur de bruit additif sur l'équation d'évolution ($\nu_t^{\mathbf{f}}$) (cf. section 3.3). Ce vecteur de bruit traduit les erreurs associées à l'utilisation de la fonction d'évolution. L'hypothèse principale effectuée est que le torseur cinématique du robot est constant entre deux instants d'intégration $t-1$ et t : le robot se déplace le long d'un arc de cercle ou en ligne droite. Cette hypothèse n'est pas parfaitement vraie. Intuitivement, plus le temps d'intégration est petit, plus l'hypothèse est correcte. Par ailleurs, il est possible que le robot ait subit des glissements que la matrice de variances-covariances $\mathbf{Q^u}$ ne peut modéliser. Dans ce cas aussi, la probabilité d'avoir un glissement important entre deux instants croît avec le temps d'intégration. Pour ces raisons, nous avons choisi de représenter $\nu_t^{\mathbf{f}}$ comme une marche

aléatoire centrée dont la matrice de covariance est donnée par :

$$\mathbf{Q}_t^{\mathbf{f}} = \begin{bmatrix} S_x\Delta t & 0 & 0 \\ 0 & S_y\Delta t & \\ 0 & 0 & S_\theta\Delta t \end{bmatrix} \tag{5.10}$$

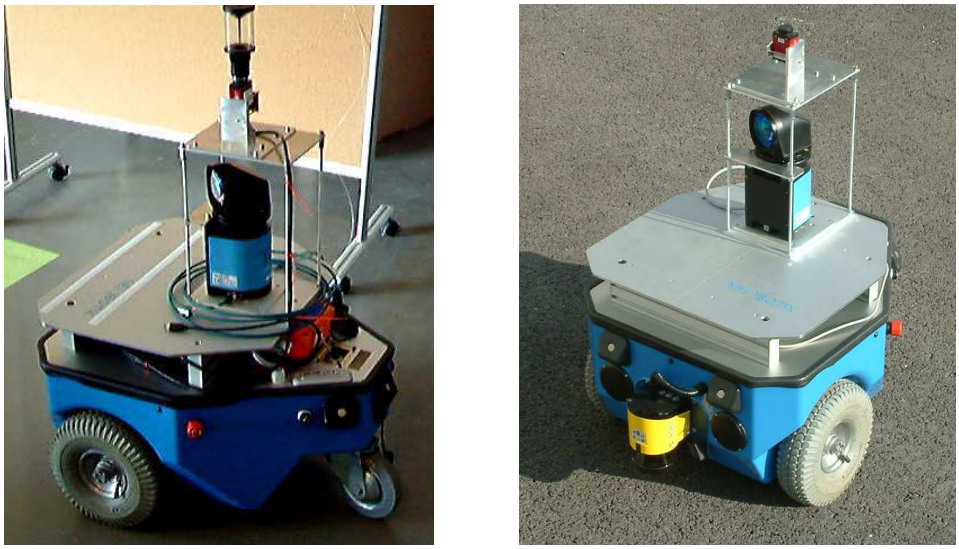
où S_x, S_y et S_θ sont les paramètres associés aux marches aléatoires sur les composantes x, y et θ. Δt désigne le temps écoulé entre les instants $t-1$ et t. Malgré la finesse de la surface de contact et l'aspect non déformable des roues, il n'y a pas eu de glissement ni de dérapage. Ceci est dû à la faible vitesse utilisée et à la bonne maîtrise des conditions expérimentales. En conséquence, on a choisi $S_x = S_y = 10^{-5}\text{m}^2.\text{s}^{-1}$ pour les composantes x et y. Ceci correspond à une variance possible de 10^{-5}m^2 (donc un écart type de 3.2mm) au bout d'une seconde d'intégration, ce qui nous semble raisonnable. Pour la composante θ, nous avons choisi $S_\theta = 5.10^{-6}\text{rad}^2.\text{s}^{-1}$, ce qui correspond à une variance de 5.10^{-6}rad^2 (donc un écart type de 0.12deg) au bout d'une seconde d'intégration.

Au final, les paramètres du robot ANIS font que l'on attend une odométrie pure assez précise. Le robot est en effet doté de roues dures à faible surface de contact, et il évolue à basse vitesse dans un environnement d'intérieur favorable. L'influence des erreurs de modèle est donc *a priori* très faible. L'intégralité des paramètres du robot est donnée dans le tableau récapitulatif 5.1 (page 188).

5.1.3.3 Paramètres Hannibal

La seconde expérimentation a été effectuée avec le robot Hannibal (dont des photos sont données sur la figure 5.6). Il s'agit d'un robot fabriqué par la société Neobotix et acheté par l'équipe-projet ARobAS de l'INRIA Sophia Antipolis-Méditerranée. Ce robot est doté de 3 roues (2 roues avant équipées d'odomètres sur lesquelles s'appuie la commande et une roue folle à l'arrière). Contrairement au robot ANIS, le robot Hannibal peut effectuer des expérimentations dans des environnements d'intérieur et d'extérieur.

Pour être capable d'évoluer dans des environnements extérieurs, Hannibal est équipé de **pneumatiques**. Ceci implique inévitablement la présence d'erreurs lors de la mesure des paramètres R et L servant à l'équation d'odométrie. En effet, la surface de contact entre une roue et le sol n'est pas ponctuelle et n'est pas connue à l'avance. Elle dépend de l'écrasement des roues (fonction de la charge portée par le robot et du gonflage des pneumatiques) et également de leurs épaisseurs. Ainsi, la distance L n'est pas triviale à évaluer précisément (étant donné que les surfaces de contacts sont larges et non connues) ; il en est de même du rayon R en raison de l'écrasement. Nous allons prendre en compte ces considérations dans le calcul de $\mathbf{Q}^{\mathbf{u}}$.



(a) (b)

FIGURE 5.6 – Photographies du robot Hannibal — (a) : On aperçoit la roue folle (petite) ainsi qu'une roue avant. Un laser rotatif à 360degrés (de grande portée) est monté sur la plateforme (non utilisé dans ce travail). Enfin, on voit en hauteur la caméra équipée de son objectif omnidirectionnel. – (b) : Photographie prise en extérieur. On aperçoit en jaune un laser de proximité. On voit également que l'épaisseur des roues avant est non négligeable.

Soient σ_R^2 et σ_L^2 les variances associées à la mesure du rayon des roues et de la longueur de l'essieu arrière. En admettant que l'erreur sur le rayon de la roue gauche est indépendante de l'erreur sur la roue droite, la matrice de variances-covariances associée au vecteur $[ds_t \; d\omega_t]^T$ (notée $\mathbf{Q}^{\mathbf{interm}}$) est donnée par linéarisation de la fonction autour des valeurs mesurées. En admettant que le rayon mesuré soit le même pour les deux roues, on obtient :

$$\mathbf{Q}^{\mathbf{interm}} = 4\pi^2 \begin{bmatrix} \frac{\sigma_{cod}^2}{2} R^2 + \frac{\sigma_R^2}{4}\left(\delta g_t^2 + \delta d_t^2\right) & \frac{\sigma_R^2}{4L}\left(\delta g_t^2 - \delta d_t^2\right) \\ \frac{\sigma_R^2}{4L}\left(\delta g_t^2 - \delta d_t^2\right) & \frac{\sigma_{cod}^2}{2L^2} R^2 + \frac{\sigma_R^2}{4L^2}\left(\delta g_t^2 + \delta d_t^2\right) + \frac{\sigma_L^2}{L^4} R^2 \left(\delta g_t - \delta d_t\right)^2 \end{bmatrix} \tag{5.11}$$

La matrice obtenue dans l'équation 5.11 n'est pas diagonale (contrairement à celle obtenue dans l'équation 5.8). De plus, la matrice dépend directement de l'incrément de rotation des roues. En remarquant que $\delta g_t - \delta d_t$ est proportionnel à $d\omega_t$ et que $\delta g_t^2 - \delta d_t^2$ est proportionnel à $ds_t d\omega_t$, l'équation 5.11 devient :

$$\mathbf{Q}^{\mathbf{interm}} = 2\left(\sigma_{cod}\pi R\right)^2 \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{L^2} \end{bmatrix} + \pi^2 \begin{bmatrix} \sigma_R^2\left(\delta g_t^2 + \delta d_t^2\right) & 4\frac{\sigma_R^2}{R^2} ds_t d\omega_t \\ 4\frac{\sigma_R^2}{R^2} ds_t d\omega_t & \frac{\sigma_R^2}{L^2}\left(\delta g_t^2 + \delta d_t^2\right) + \frac{\sigma_L^2}{L^2} d\omega_t^2 \end{bmatrix} \tag{5.12}$$

On retrouve bien la partie utilisée pour le robot ANIS, à laquelle s'ajoute un terme supplémentaire. On pourra remarquer que les termes de corrélation n'interviennent que lorsque le robot est en rotation.

De la même façon que pour le robot ANIS, on incorpore la composante de glissement transversal dans la matrice $\mathbf{Q^u}$, en admettant qu'un rapport de 10 sur le niveau des erreurs est réaliste. La matrice $\mathbf{Q^u}$ vaut finalement :

$$\mathbf{Q^u} = \begin{bmatrix} \mathbf{Q^{interm}}(1,1) & 0 & \mathbf{Q^{interm}}(1,2) \\ 0 & \frac{\mathbf{Q^{interm}}(1,1)}{100} & 0 \\ \mathbf{Q^{interm}}(2,1) & 0 & \mathbf{Q^{interm}}(2,2) \end{bmatrix} \tag{5.13}$$

Enfin, la matrice $\mathbf{Q^f}$ est définie de la même façon que pour la première expérimentation. Cette matrice ne dépend pas directement du matériel mais de la façon dont il est utilisé. Cette expérimentation a été réalisée à faible vitesse dans des conditions similaires à celles de l'expérimentation avec le robot ANIS.

5.1.4 Les caméras

Nous présentons dans ce paragraphe les données relatives aux caméras. Pour les deux expérimentations, on a utilisé une caméra omnidirectionnelle (avec miroir parabolique pour la première expérimentation et un miroir hyperbolique pour la seconde). La transformation des mesures pixels en angles est d'abord donnée. Nous décrivons ensuite les matrices de variances-covariances obtenues. Enfin, des détails quant à la calibration des caméras sont donnés.

5.1.4.1 Obtension du vecteur de mesures à partir d'une image

Les mesures utilisées sont des mesures de haut niveau. A chaque instant, le vecteur de mesures est constitué des angles de gisement des amers vus ($\alpha_{t,(i)}$) et angles d'élévation ($\beta_{t,(i)}$). L'extraction des mesures se fait après transformation des coordonnées pixels des points dans l'image en coordonnées angulaires grâce au modèle de projection inverse (equation 4.24 page 168) dont le processus est rappelé.

Soit $\mathbf{p}$ les coordonnées homogènes d'un point de l'image en pixel. On commence par retrouver le point $\mathbf{m} = [x_m \ y_m \ 1]^T$ (projection de $\mathbf{p}$ sur le plan unitaire) par $\mathbf{m} = \mathbf{Kp}$ où $\mathbf{K}$ est la matrice des paramètres intrinsèques de la caméra et vaut :

$$\mathbf{K} = \begin{bmatrix} \gamma_1 & \gamma_1 s & u_0 \\ 0 & \gamma_2 & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.14}$$

Les coordonnées du point sur la sphère sont données par :

$$\begin{bmatrix} x_S \\ y_S \\ z_S \end{bmatrix} = \begin{bmatrix} \frac{\xi+\sqrt{1+(1-\xi^2)(x_m^2+y_m^2)}}{x_m^2+y_m^2+1} x_m \\ \frac{\xi+\sqrt{1+(1-\xi^2)(x_m^2+y_m^2)}}{x_m^2+y_m^2+1} y_m \\ \frac{\xi+\sqrt{1+(1-\xi^2)(x_m^2+y_m^2)}}{x_m^2+y_m^2+1} - \xi \end{bmatrix} \tag{5.15}$$

où ξ est le paramètre du miroir. Enfin, les coordonnées sont obtenues par :

$$\begin{cases} \alpha &= -\arctan2(-x_S, -y_S) \\ \beta &= \arctan\left(\frac{-z_S}{\sqrt{x_S^2+y_S^2}}\right) \end{cases} \tag{5.16}$$

Remarque 5.1 (Commentaire de l'équation 5.16)

Les coordonnées finales obtenues dans l'équation 5.16 ne sont pas " habituelles ". On se serait attendu à obtenir $\alpha = \arctan2(y_S, x_S)$ *et* $\beta = \arctan\left(\frac{z_S}{\sqrt{x_S^2+y_S^2}}\right)$*. La raison de ce changement est la position du repère du miroir. En effet, l'axe* $\mathbf{z}$ *du miroir pointe vers l'extérieur du miroir (donc vers le bas), ce qui explique le signe moins dans l'expression de* β*. Par ailleurs, l'expression de* α *peut également s'écrire* $\arctan2(y_S, x_S) + \pi/2$ *(ramené entre* $-\pi$ *et* π*). Ce décalage est dû au fait que l'axe* $\mathbf{y}$ *de l'image est dirigé dans la direction de l'axe* $\mathbf{x}$ *du robot (le haut de l'image représente l'avant). La définition des angles de l'équation 5.16 permet de ne pas avoir à ajouter une matrice de rotation triviale entre le repère du robot et le repère de la caméra.*

5.1.4.2 Calcul analytique de la matrice de variances-covariances associée au vecteur de mesures

Nous proposons dans ce paragraphe de détailler le calcul de la matrice de variances-covariances associée au vecteur de mesures, ie. $\mathbf{R}_t$. On suppose que chaque point d'intérêt est extrait indépendemment des autres. Ainsi, $\mathbf{R}_t$ est choisie bloc-diagonale : $\mathbf{R}_t = \text{diag}\left(\mathbf{R}_{t,(1)}, \ldots, \mathbf{R}_{t,(N)}\right)$ (N désignant le nombre de points détectés). Chaque matrice $\mathbf{R}_{t,(i)}$ est de taille 2×2 et correspond à la matrice de variances-covariances associée à la mesure de l'amer (i).

Il est facile de remarquer que les images omnidirectionnelles n'ont pas une résolution constante en terme d'angles. Ainsi, l'angle solide capturé par un pixel est plus faible lorsque le pixel est situé sur les bords de l'image que au centre. Il paraît alors naturel que la matrice $\mathbf{R}_{t,(i)}$ depende de la position du point dans l'image.

Dans la suite, nous admettons que la matrice de variances-covariances associée à la mesure brute en pixels est constante et caractéristique du détecteur. Nous notons $\mathbf{R}_{raw}$ cette matrice. Par ailleurs, on suppose que l'erreur est isotropique. Nous posons donc $\mathbf{R}_{raw} = \sigma_{raw}^2\mathbf{I}$. En supposant que les erreurs soient suffisament petites, $\mathbf{R}_{raw}$ peut être propagée jusqu'à $\mathbf{R}_{t,(i)}$ par :

$$\mathbf{R}_{t,(i)} = \mathbf{J}_{\alpha\beta}\left(\mathbf{p}_{t,(i)}\right)\mathbf{R}_{raw}\left(\mathbf{J}_{\alpha\beta}\left(\mathbf{p}_{t,(i)}\right)\right)^T \tag{5.17}$$

où $\mathbf{J}_{\alpha\beta}$ est la matrice jacobienne de la fonction qui à un pixel associe le vecteur $[\alpha\ \beta]^T$, évaluée au pixel considéré ($\mathbf{p}_{t,(i)}$). Pour faciliter l'interprétation des résultats, nous choisissons d'exprimer $\mathbf{R}_{t,(i)}$

en fonction des angles à la place des coordonnées en pixels. Ceci est possible car il existe une bijection entre les pixels de l'image et les angles d'azimuth et d'élévations. [6] Une façon facile d'obtenir $\mathbf{J}_{\alpha\beta}$ en fonction des angles est de calculer l'inverse de $\mathbf{J_p}$, la matrice jacobienne de la fonction qui à un couple d'angle donné associe un pixel. L'équation 5.17 devient donc :

$$\mathbf{R}_{t,(i)} = \left(\mathbf{J_p}\left(\alpha_{t,(i)}, \beta_{t,(i)}\right)\right)^{-1} \mathbf{R}_{raw} \left(\mathbf{J_p}\left(\alpha_{t,(i)}, \beta_{t,(i)}\right)\right)^{-T} \tag{5.18}$$

où $\mathbf{J_p}\left(\alpha_{t,(i)}, \beta_{t,(i)}\right)$ est obtenu en dérivant la fonction définie par :

$$\begin{array}{rcl}]-\pi,\pi]\times]-\frac{\pi}{2},\frac{\pi}{2}[& \rightarrow & \mathbb{R}^2 \\ (\alpha,\beta) & \mapsto & \begin{bmatrix} \gamma_1 \frac{\cos\alpha\cos\beta}{-\sin\beta+\xi} + \gamma_1 s \frac{\sin\alpha\cos\beta}{-\sin\beta+\xi} + u_0 \\ \gamma_2 \frac{\sin\alpha\cos\beta}{-\sin\beta+\xi} + v_0 \end{bmatrix} \end{array} \tag{5.19}$$

Tous calculs faits, on obtient :

$$\left(\mathbf{J_p}(\alpha,\beta)\right)^{-1} = \begin{bmatrix} \frac{-\sin\alpha(\xi-\sin\beta)}{\gamma_1\cos\beta} & \frac{(\cos\alpha+s\sin\alpha)(\xi-\sin\beta)}{\gamma_2\cos\beta} \\ \frac{(\xi-\sin\beta)^2\cos\alpha}{\gamma_1(1-\xi\sin\beta)} & -\frac{(s\cos\alpha-\sin\alpha)(\xi-\sin\beta)^2}{\gamma_2(1-\xi\sin\beta)} \end{bmatrix} \tag{5.20}$$

Le résultat final s'obtient en substituant l'équation 5.20 dans l'équation 5.18 :

$$\mathbf{R}_{t,(i)} = \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix} \tag{5.21}$$

avec :

$$\begin{array}{rcl} r_{11} & = & \frac{\sigma^2_{raw}(\xi-\sin\beta_{t,(i)})^2}{\cos^2\beta_{t,(i)}} \left(\frac{\sin^2\alpha_{t,(i)}}{\gamma_1^2} + \frac{(\cos\alpha_{t,(i)}+s\sin\alpha_{t,(i)})^2}{\gamma_2^2}\right) \\ r_{12} & = & \frac{\sigma^2_{raw}(\xi-\sin\beta_{t,(i)})^3}{\cos\beta_{t,(i)}(1-\xi\sin\beta_{t,(i)})} \left(\frac{-\sin\alpha_{t,(i)}\cos\alpha_{t,(i)}}{\gamma_1^2} - \frac{(\cos\alpha_{t,(i)}+s\sin\alpha_{t,(i)})(s\cos\alpha_{t,(i)}-\sin\alpha_{t,(i)})}{\gamma_2^2}\right) \\ r_{21} & = & r_{12} \\ r_{22} & = & \frac{\sigma^2_{raw}(\xi-\sin\beta_{t,(i)})^4}{(1-\xi\sin\beta_{t,(i)})^2} \left(\frac{\cos^2\alpha_{t,(i)}}{\gamma_1^2} + \frac{(s\cos\alpha_{t,(i)}-\sin\alpha_{t,(i)})^2}{\gamma_2^2}\right) \end{array} \tag{5.22}$$

L'équation 5.22 montre que la matrice de variances-covariances associée à un amer dépend de sa position dans l'image. Par ailleurs, on peut remarquer que l'erreur sur le gisement peut être corrélée avec l'erreur sur l'élévation. Néanmoins, cette corrélation disparaît dès que $s = 0$ et $\gamma_1 = \gamma_2$. [7]. Dans ce cas, la matrice de variances-covariances obtenue vaut :

$$\mathbf{R}_{t,(i)} = \frac{\sigma^2_{raw}}{{\gamma_1}^2}\mathrm{diag}\left(\frac{\left(\xi-\sin\beta_{t,(i)}\right)^2}{\cos^2\beta_{t,(i)}}, \frac{\left(\xi-\sin\beta_{t,(i)}\right)^4}{\left(1-\xi\sin\beta_{t,(i)}\right)^2}\right) \tag{5.23}$$

On remarque alors que $\mathbf{R}_{t,(i)}$ ne dépend que de l'angle d'élévation. Ceci est une conséquence de la géométrie de révolution des miroirs considérés. Remarquons enfin que la variance de l'angle d'élévation

6. Excepté aux pôles de la sphère, mais cette région n'est jamais considérée en pratique.
7. Ce qui est en général le cas.

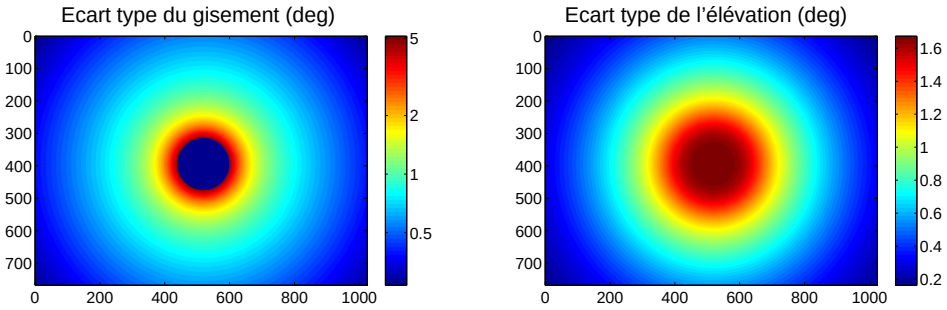


FIGURE 5.7 – Ecart-types des mesures en fonction du pixel considéré dans l'image pour le cas de la caméra parabolique de la première expérimentation. Sur l'image de gauche, le centre de la caméra n'est pas représenté car les valeurs deviennent trop importantes et tendent vers l'infini.

est toujours définie (on a toujours $1 - \xi \sin \beta_{t,(i)} \neq 0$ car $\xi < 1$). Ceci n'est pas le cas pour la variance de l'angle de gisement qui augmente jusqu'à l'infini lorsqu'on se rapproche des pôles de la sphère. [8] La figure 5.7 donne les covariances pour le cas du miroir parabolique avec $\sigma_{raw} = 4$pixels. [9] On voit bien le fait que le calcul de l'angle de gisement devient critique lorsqu'on se rapproche du centre de la caméra. En revanche, la qualité d'estimation de l'angle d'élévation reste globalement bonne en tout point sur la caméra malgré une légère dégradation au centre de l'image.

5.1.4.3 Paramètres de calibration

Les paramètres de calibrations utilisés ont été obtenus grâce à l'utilisation de mires de calibration et à la boîte à outils développée par Christopher Mei ([Mei, 2007]). Les paramètres concernant la position de la caméra dans le repère du robot ont été mesurés manuellement. On a au final :

Objectif parabolique :

- Point central : $(u_0, v_0) = (517.65, 393.66)$
- Distances focales : $\gamma_1 = \gamma_2 = -273.898$
- Obliquité nulle : $s = 0$
- Paramètre du miroir : $\xi = 1$
- Coordonnées du repère caméra dans le repère du robot : $t_x = 0.55$m, $t_y = -0.29$m, $\gamma = 0$deg

Objectif hyperbolique :

- Point central : $(u_0, v_0) = (521.12, 384.34)$
- Distances focales : $\gamma_1 = \gamma_2 = -266.1664$

8. Ceci est logique car le gisement n'est pas défini pour les élévations de $-\pi/2$ et $\pi/2$.
9. Les paramètres numériques associés à la caméra sont donnés dans le paragraphe suivant.

– Obliquité nulle : $s = 0$

– Paramètre du miroir : $\xi = 0.9$

– Coordonnées du repère caméra dans le repère du robot : $t_x = 0.1955\text{m}$, $t_y = 0\text{m}$, $\gamma = 0\text{deg}$

5.1.5 Conclusion quant aux conditions expérimentales

Nous avons proposé dans cette section une description la plus complète possible du protocole expérimental. Nous avons dans un premier temps présenté les caractéristiques des lieux et trajectoires effectuées, mettant ainsi en évidence les paramètres que l'on espère retrouver dans le résultat final (géométrie des murs pour la première expérimentation et l'aspect fermeture de boucle pour la seconde). Nous avons également introduit les modèles utilisés : ceux-ci sont très proches de ceux introduits au chapitre 3. Enfin, nous avons fourni une description détaillée de l'ensemble du matériel utilisé ainsi que les paramètres numériques nécessaires pour les implémentations de l'algorithme de SAM.

Un résumé complet des paramètres est donné dans le tableau comparatif 5.1.

		Première expérimentation	**Seconde expérimentation**
Descriptions générales		Utilisation du robot **ANIS** dans les couloirs du bâtiment Borel. Présence de nombreux posters sur les murs et géométrie des couloirs simple.	Utilisation du robot **Hannibal** dans la halle robotique du bâtiment Kahn. Environnement intérieur assez ouvert et complexe. Les positions initiales et finales coïncident.
ROBOTS	Paramètres géométriques	$R = 0.1\text{m}$ $L = 0.5\text{m}$	$R = 0.12\text{m}$ $L = 0.5\text{m}$
	Ecart-types liés à $\mathbf{Q^u}$	$\sigma_{cod} = \pi/180\text{rad}$ $\sigma_R = 0$ $\sigma_L = 0$	$\sigma_{cod} = \pi/180\text{rad}$ $\sigma_R = 2.10^{-3}\text{m}$ $\sigma_L = 0.0445\text{m}$
	Paramètres liés à $\mathbf{Q^f}$	$S_x = 10^{-5}\text{m}^2.\text{s}^{-1}$ $S_y = 10^{-5}\text{m}^2.\text{s}^{-1}$ $S_\theta = 5.10^{-6}\text{rad}^2.\text{s}^{-1}$	$S_x = 10^{-5}\text{m}^2.\text{s}^{-1}$ $S_y = 10^{-5}\text{m}^2.\text{s}^{-1}$ $S_\theta = 5.10^{-6}\text{rad}^2.\text{s}^{-1}$
CAMERAS	Position repères caméra	$t_x = 0.55\text{m}$ $t_y = -0.29\text{m}$ $\gamma = 0$	$t_x = -0.1955\text{m}$ $t_y = 0\text{m}$ $\gamma = 0$
	Paramètres intrinsèques	$(u_0, v_0) = (517.65, 393.66)$ $\gamma_1 = \gamma_2 = -273.898$ $s = 0 \qquad \xi = 1$	$(u_0, v_0) = (521.12, 384.34)$ $\gamma_1 = \gamma_2 = -266.1664$ $s = 0 \qquad \xi = 0.9$
	Ecart-types mesures brutes	$\sigma_{raw} = 4\text{pixels}$	$\sigma_{raw} = 4\text{pixels}$

TABLE 5.1 – Caractéristiques des deux expérimentations

5.2 Rejet des points aberrants

5.2.1 Introduction

L'utilisation des techniques visuelles présentées au chapitre 4 (descripteur SIFT couplé avec la méthode RANSAC) a une certaine robustesse mais a aussi ses faiblesses. En effet, de part son aspect aléatoire, la méthode de RANSAC ne garantit pas que les mises en correspondances sont parfaites, mais un pourcentage " faible " d'*outliers* (points aberrants).

Les mesures associées à ces *outliers* sont par définition très éloignées des mesures attendues. Ainsi, elles ne permettent plus de garantir l'hypothèse de petites erreurs nécessaire à l'utilisation de notre méthode de lissage non linéaire. La conséquence de l'intégration des *outliers* dans le filtre est :

- au mieux, un léger biais dans la solution,
- au pire, une divergence du filtre comme dans le cas extrême du scénario 12 dans les simulations du chapitre 3 (page 126).

Il paraît alors naturel d'essayer de supprimer du filtre ces points abérrants ; il en va de la consistance de la solution finale.

Les outliers peuvent être divisés en deux sources principales :

1. Les erreurs dans l'association des données. Le suivi des points a mis en correspondance deux points qui n'auraient pas dû l'être. Ceci peut arriver lorsqu'une zone de l'image possède un motif répété plusieurs fois. La méthode proposée au paragraphe suivant est en général assez robuste dans ce type de cas.

2. Les points sont en mouvement. Dans ce cas, l'algorithme de mise en correspondance ne fait pas d'erreur, mais l'hypothèse du filtre concernant la staticité de la scène n'est plus respectée. Ceci peut biaiser considérablement la solution, surtout si le mouvement est important et que le point est suivi " longtemps ". La méthode proposée au paragraphe suivant est un peu moins robuste dans ce cas, notamment lorsque les mouvements des points sont très lents.

Nous proposons d'utiliser une méthode classique de rejets des *outliers* basée sur la distance de Mahalanobis. Nous donnons les principaux aspects de la méthode dans le paragraphe suivant.

5.2.2 Utilisation de la distance de Mahalanobis

Le principe décrit ici consiste à filtrer les points considérés à chaque " étape de prédiction ". L'idée est alors d'effectuer une prédiction du vecteur de mesures et de vérifier si celui-ci ne s'éloigne pas

trop de la mesure effectuée. Une description précise des avantages et inconvénients théoriques de cette méthode peut être trouvée dans ([Joly, 2006]).

Supposons qu'à l'instant t nous disposons d'une solution sous la forme d'un vecteur d'information ξ et d'une matrice d'information $\mathbf{\Omega}$ (correspondant à toute la trajectoire depuis l'instant initial jusqu'à l'instant t). Grâce au théorème de marginalisation (théorème 1.3 p.20), il est possible de calculer les paramètres d'information puis la matrice de variances-covariances associée au vecteur $[\mathbf{x}_t^T, \tilde{\mathbf{m}}^T]^T$ (où $\tilde{\mathbf{m}}$ désigne la restriction de $\mathbf{m}$ à l'ensemble des amers visibles à l'instant $t+1$). Soient μ_t et $\mathbf{\Sigma}_t$ l'espérance mathématique et la matrice de variances-covariances obtenues après marginalisation :

$$\mu_t = \begin{bmatrix} \mu_{\mathbf{x}_t} \\ \mu_{\tilde{\mathbf{m}}} \end{bmatrix} \qquad \text{et} \qquad \mathbf{\Sigma}_t = \begin{bmatrix} \mathbf{\Sigma}_{\mathbf{x}_t\mathbf{x}_t} & \mathbf{\Sigma}_{\mathbf{x}_t\tilde{\mathbf{m}}} \\ \mathbf{\Sigma}_{\tilde{\mathbf{m}}\mathbf{x}_t} & \mathbf{\Sigma}_{\tilde{\mathbf{m}}\tilde{\mathbf{m}}} \end{bmatrix} \tag{5.24}$$

Connaissant le vecteur $\mathbf{u}_{t+1}$, une prédiction de l'état du système à l'instant $t+1$ est donnée par :

$$\mu_{t+1} = \begin{bmatrix} \mathbf{f}\left(\mu_{\mathbf{x}_t}, \mathbf{u}_{t+1}\right) \\ \mu_{\tilde{\mathbf{m}}} \end{bmatrix} \qquad \text{et} \qquad \mathbf{\Sigma}_{t+1} = \begin{bmatrix} \left(\nabla_{\mathbf{x}}\mathbf{f}\right)\mathbf{\Sigma}_{\mathbf{x}_t\mathbf{x}_t}\left(\nabla_{\mathbf{x}}\mathbf{f}\right)^T + \mathbf{Q}_t & \left(\nabla_{\mathbf{x}}\mathbf{f}\right)\mathbf{\Sigma}_{\mathbf{x}_t\tilde{\mathbf{m}}} \\ \mathbf{\Sigma}_{\tilde{\mathbf{m}}\mathbf{x}_t}\left(\nabla_{\mathbf{x}}\mathbf{f}\right)^T & \mathbf{\Sigma}_{\tilde{\mathbf{m}}\tilde{\mathbf{m}}} \end{bmatrix} \tag{5.25}$$

On peut alors déduire de l'équation 5.25 une prédiction du vecteur de mesures ainsi qu'une matrice de variances-covariances associée à la prédiction (grâce à la fonction $\mathbf{h}$, sa matrice jacobienne et les règles de propagation des matrices de variances-covariances). Ceci nous permet d'extraire pour chaque amer considéré la prédiction $\mathbf{z}_{t+1,(i)}^{pred}$ et la matrice de variances-covariances $\mathbf{R}_{t+1,(i)}^{pred}$. L'algorithme final de rejet est donné par :

1. Calcul de $\mathbf{z}_{t+1,(i)}^{pred}$ et $\mathbf{R}_{t+1,(i)}^{pred}$
2. Calcul de $\mathbf{e} = \mathbf{z}_{t+1,(i)}^{pred} - \mathbf{z}_{t+1,(i)}$
3. Comparaison de $test = \mathbf{e}^T \left(\mathbf{R}_{t+1,(i)}^{pred} + \mathbf{R}_{t+1,(i)}\right)\mathbf{e}$ à un seuil donné fonction de la loi du χ^2 à 2 degrés de liberté. Nous avons choisi un seuil correspondant à 99% (soit 9.21). Si $test > 9.21$, alors la mesure de (i) est considérée comme aberrante. Le suivi de ce point est donc arrêté.

5.3 Résultats de SLAM visuel

Nous présentons dans cette section les résultats obtenus avec l'algorithme de SAM sur les deux jeux de données. Le paragraphe 5.3.1 est consacré à la première expérimentation. Les résultats de la seconde expérimentation sont donnés dans le paragraphe 5.3.2 (page 196). Dans les deux cas, nous utilisons l'algorithme classique de SAM dont tous les paramètres ont été donnés dans les sections précédentes. Enfin, précisons que l'ensemble des résultats ont été obtenus en **post-traitements** et en

utilisant le maximum de points possible. Nous n'avons pas introduit de contrainte sur les temps de calculs.

5.3.1 Expérimentation Borel

5.3.1.1 Analyse qualitative et quantitative des résultats

Nous présentons dans ce paragraphe les résultats de SLAM obtenus avec les données de la première expérimentation. Dans un premier temps, on peut se rendre compte que la carte globale est très cohérente. Celle-ci est représentée sur la figure 5.8. On constate en effet que l'on retrouve bien la structure de couloir du lieu visité (alignement de points de part et d'autre de la trajectoire). Par ailleurs, on retrouve parfaitement les angles droits présents dans la réalité. En ce qui concerne la trajectoire, celle-ci semble cohérente avec la carte (déplacement " à peu près " au milieu des couloirs). On note un léger décalage avec l'intégration de l'odométrie pure (courbe verte sur la figure 5.8).

Quelques données quantitatives ont été extraites de cette carte afin de mettre en évidence la qualité des résultats. Pour cela, nous avons calculé par regression linéaire les équations des droites numérotées sur la figure 5.8. On a obtenu :

Droite 1 : $-0.0163x + 0.1291y - 0.9915 = 0$

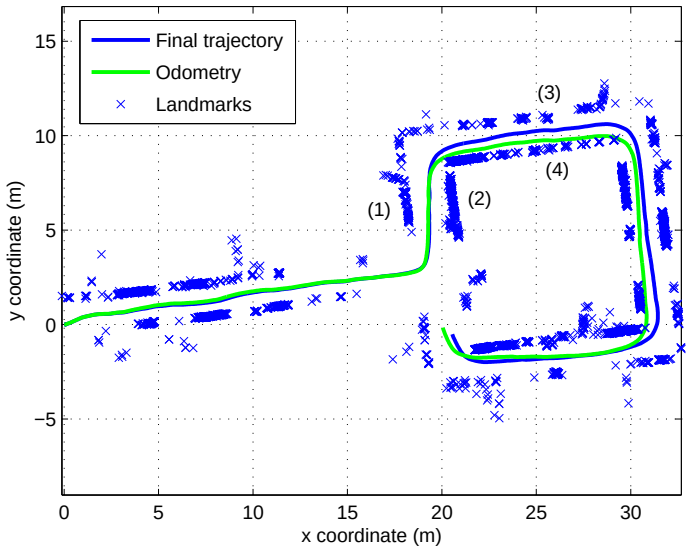
Droite 2 : $-0.0186x + 0.1614y - 0.9867 = 0$

Droite 3 : $0.0294x + 0.0035y - 0.9996 = 0$

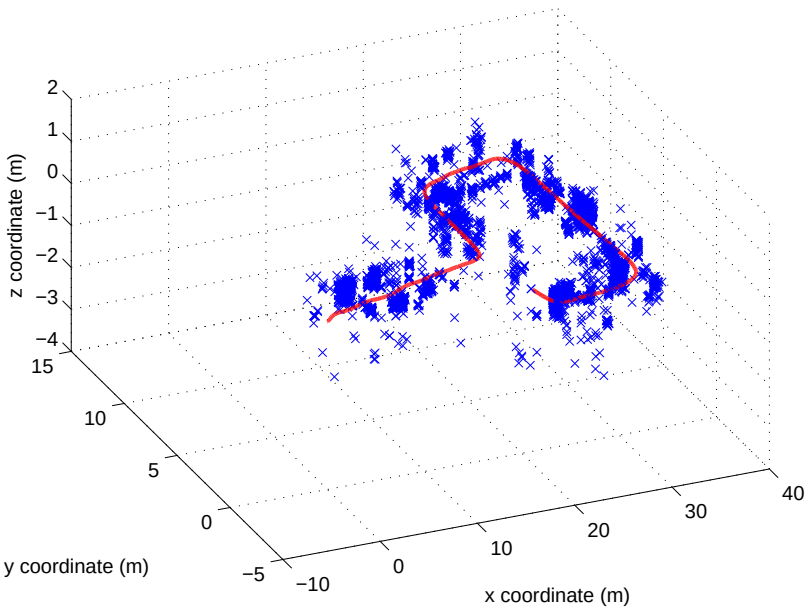
Droite 4 : $0.0310x + 0.0037y - 0.9995 = 0$

Ces équations nous confirment la structure de couloirs à angle droit que l'on visualise qualitativement sur la figure 5.8. En effet, on se rend compte que les droites 1 et 2 sont quasiments parallèles (le produit scalaire de leur vecteurs directeurs unitaires indique un angle de 0.62deg entre les deux droites). Il en est de même avec les droites 3 et 4 (avec un angle de 0.14deg). Nous avons également vérifié l'orthogonalité des murs. Le calcul des produits scalaires a permis de montrer que l'angle entre les droites 1 et 3 est de 90.54deg et l'angle entre les droites 2 et 4 est de 89.78deg. On retrouve quasiment les 90 degrés de la vérité terrain. Enfin, le calcul de la distance entre les droites 1 et 2 donne 1.832m (contre 1.794m en réalité) [10] et la distance entre les droites 3 et 4 donne 1.822m (contre 1.794m en réalité). Ceci confirme donc l'excellente qualité de la solution. Un résumé de ces résultats est fourni dans le tableau 5.2.

10. Vérité terrain mesurée au laser.



(a)



(b)

FIGURE 5.8 – Carte globale du bâtiment Borel obtenue par le robot ANIS — (a) Trajectoire estimée, intégration de l'odométrie pure et projection dans le plan $z = 0$ des amers estimés – (b) Vue 3D des éléments précédents

	Valeurs estimées	Vérités terrain	Erreurs relatives (%)
Angle entre les droites (1) et (2)	0.62 deg	0	Non définie
Distance entre les droites (1) et (2)	1.832m	1.794m	2.11
Angle entre les droites (3) et (4)	0.14 deg	0	Non définie
Distance entre les droites (3) et (4)	1.822m	1.794m	1.59
Angle entre les droites (1) et (3)	90.54 deg	90 deg	0.60
Angle entre les droites (2) et (4)	89.78 deg	90 deg	0.24

TABLE 5.2 – Informations quantitatives extraites de la carte du bâtiment Borel

5.3.1.2 Evolution des incertitudes au cours de l'expérimentation

En plus des qualités intrinsèques de la trajectoire et de la carte, nous avons étudié l'évolution des matrices de variances-covariances associées au système. On constate qu'il y a une augmentation significative de l'incertitude entre le début de l'expérimentation et la fin. Ceci est logique et dû au fait que l'algorithme est appliqué en boucle ouverte sans réobservation du début de la carte. Ainsi, l'ensemble de la solution subit un effet de marche aléatoire qui se traduit par une inflation progressive des matrices de variances-covariances. Celle-ci est néanmoins beaucoup plus faible que dans le cas où on aurait appliqué uniquement la solution odométrique. L'utilisation des mesures de l'environnement permet de limiter la vitesse d'augmentation des ellipses.

L'augmentation de l'incertitude peut être problématique. En effet, la taille des ellipses sur la figure 5.9(b) est considérable et les ellipses se chevauchent. Ainsi, si le résultat devait être fourni à un utilisateur humain, l'interprétation deviendrait délicate. Ceci provient du fait que l'estimation des amers est imprécise dans **le repère initial**. Néanmoins, cette représentation est le résultat de l'utilisation des blocs diagonaux de la matrice de variances-covariances du système. Une étude prenant en compte les corrélations permettrait de montrer que les incertitudes dans le repère courant du robot sont beaucoup plus faibles (une étude détaillée est proposée dans le chapitre 8). Par exemple, le fait que les ellipses des murs des couloirs se chevauchent donne l'impression que l'on ne peut pas évaluer précisément la largeur du couloir. Ceci n'est pas exact car les corrélations permettent de définir de manière très précise les mesures différentielles, malgré des estimations globales de moindre qualité. Si ces informations sont effectivement présentes dans les termes hors diagonaux de la matrice de variances-covariance, le problème de la représentation se pose :l'interprétation directe est délicate. De même, si le résultat du SLAM doit être utilisé pour un autre algorithme (par exemple pour le contrôle), il pourraît être délicat de devoir passer tous les termes de la matrice de variances-covariances, surtout pour le cas de systèmes de grande dimension. Cette problématique nous amènera à utiliser des cartes

locales (troisième partie du document).

5.3.1.3 Instabilité due à l'absence de données

Nous présentons dans ce paragraphe un problème particulier lié au manque de mesures durant une partie de l'expérimentation. Durant la première ligne droite de l'expérimentation, il y a eu une zone dans laquelle très peu de points étaient suivis (voire aucun à certaines itérations). Cet aspect avait été signalé dans la section 5.1 et illustré sur la figure 5.2 page 174. Cette zone correspond environ à la partie de la trajectoire comprise entre les abscisses 12m et 17m.

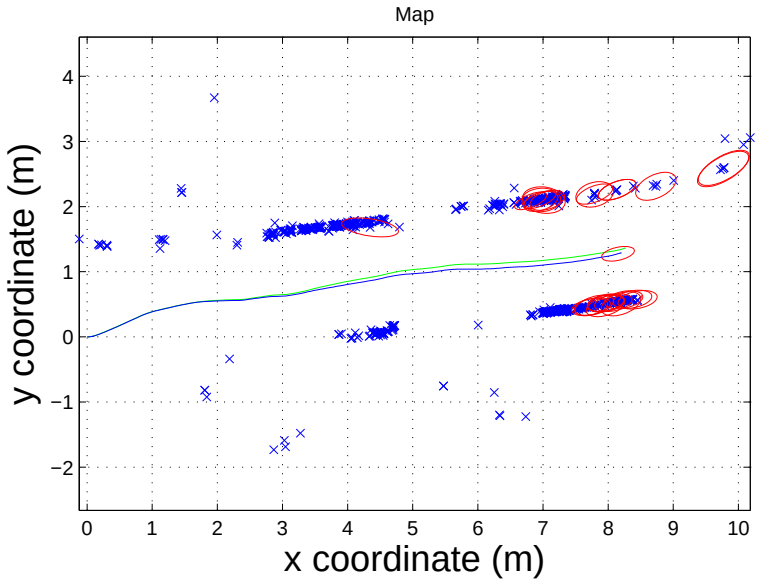
Tout au long de cette zone, l'information principale pour l'algorithme de lissage était l'odométrie. Il n'y avait pratiquement plus aucune contrainte liée à l'environnement. Ceci a posé un problème d'instabilité lorsque la dimension de l'état a considérablement augmenté : les valeurs numériques associées à l'information sur cette partie de la trajectoire deviennent trop faibles par rapport aux autres valeurs. On a en effet constaté vers les dernières itérations que la solution se déplaçait considérablement. Le début de la carte était pratiquement inchangé alors que l'ensemble de la fin de la carte subissait une rotation rigide autour de la zone à faible information (figure 5.10). Ceci est finalement assez logique : l'absence de mesures a provoqué une zone numériquement instable dans laquelle la densité de probabilité est assez diffuse. Le gradient devient alors trop faible pour que l'optimisation liée au SAM trouve un minimum global pertinent.

Concernant ce problème, on pourra remarquer que les propriétés locales sont conservées. En effet, l'instabilité ne s'opère que sur une seule partie de la trajectoire, donnant alors l'impression que la carte peut être découpée en deux blocs rigides. La taille des murs, et les angles restent localement inchangés. L'algorithme de SAM permet donc tout de même de garantir une cohérence locale forte (ce qui rejoint les remarques faites au paragraphe précédent concernant l'importance des liens locaux).

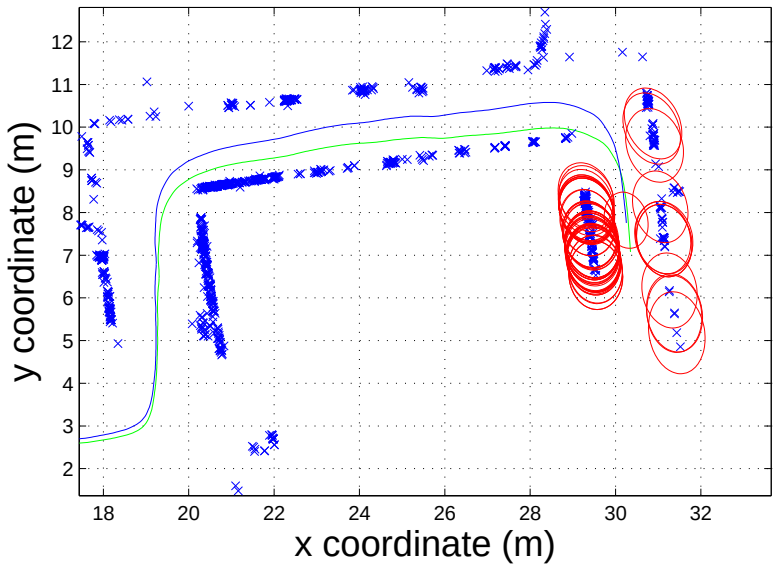
5.3.1.4 Bilan de ces premiers résultats

Le bilan de cette première expérimentation est très bon. Nous avons en effet réussi à obtenir une carte et une trajectoire dont la qualité semble qualitativement très bonne. Par ailleurs, certaines mesures quantitatives concernant la structure de l'environnement ont confirmé ces impressions. Le problème soulevé concernant l'instabilitié est très spécifique et dû à de mauvaises conditions dans une zone localisée de la trajectoire.

Finalement, la principale faiblesse de cette expérimentation est **sa relative facilité**. En effet, hormis la partie de la séquence problématique, le reste de la séquence comportait des images avec



(a)



(b)

FIGURE 5.9 – Evolution des ellipses d'incertitudes à 99% pour la séquence Borel. L'intégration de l'odométrie pure est représentée en vert. On ne représente que les ellipses associées à la position courante du robot et les amers courants — (a) Résultat obtenu au début de la trajectoire (expérimentation réalisée à 20%) – (b) Résultat obtenu après environ 70% de la trajectoire.

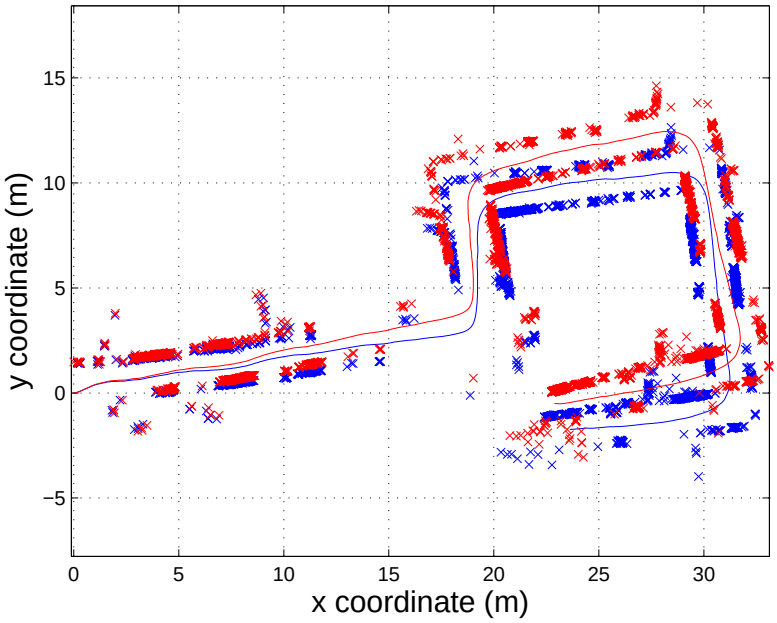


FIGURE 5.10 – Illustration du décalage à cause de la région sans amers

beaucoup de textures (les lieux étaient très structurés, avec notamment la présence de nombreux posters sur les murs). De plus, la plupart des points détectés l'étaient à une distance assez faible du robot, ce qui a contribué à la précision du résultat. Enfin, on peut remarquer que l'odométrie utilisée était très précise dans son ensemble. Ceci est mis en évidence sur la figure 5.3(a) : il y a très peu de décalage entre la trajectoire estimée par le SAM et l'intégration pure de l'odométrie. On peut alors se demander si la bonne qualité des résultats est le fait de cette odométrie. L'algorithme aurait-il été capable de corriger une odométrie plus défaillante ? Les résultats présentés au paragraphe suivant (obtenus avec le robot Hannibal) permettront de répondre à cette question.

5.3.2 Expérimentation Kahn

5.3.2.1 Analyse qualitative et quantitative des résultats

Nous présentons dans ce paragraphe les résultats obtenus lors de la seconde expérimentation. Les conditions de cette expérimentation sont plus difficiles que la précédente, notamment sur deux points :

1. L'odométrie : elle est moins précise que celle du robot ANIS à cause de la présence de pneumatiques,
2. L'environnement : celui-ci est beaucoup plus ouvert.

Le second point cité nous empêche également d'obtenir une vérité terrain. Pour compenser cette absence de vérité terrain, nous avons choisi de faire en sorte que la position finale à l'issue de l'expérimentation soit confondue avec la position initiale. De plus, tous les points vus à la fois à l'allée et au retour (typiquement le mur principal qui est longé) doivent se superposer.

La trajectoire et la carte obtenues (en projection dans le plan horizontal ainsi qu'en 3D) sont présentés sur la figure 5.11. On se rend compte que la trajectoire odométrique (en verte) est fortement décalée (le retour devant se faire quasiment parallèlement à l'aller). Ici, l'odométrie subit un biais conséquent lors des mouvements de rotation, dû à la présence de pneumatiques. Néanmoins, la trajectoire et la carte fournies par le filtre sont très bonnes. Premièrement, le mur le long de la partie droite paraît pratiquement rectiligne. Par ailleurs, on remarque sur la vue 3D des alignements verticaux de points régulièrement espacés correspondant à une baie vitrée (l'alignement obtenu ainsi que la régularité témoignent de la qualité de la solution).

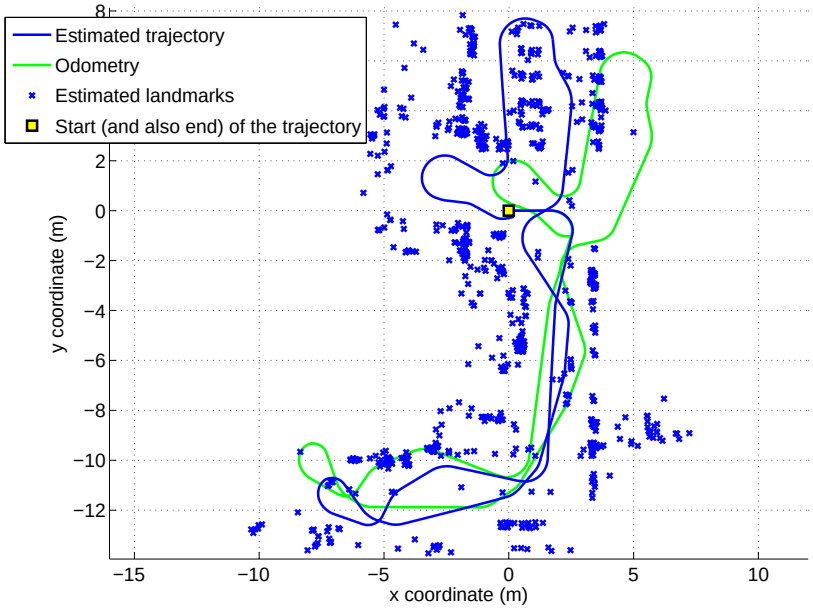
Enfin, la qualité de l'estimation est également visible après vérification de la fermeture de boucle. La figure 5.12 montre en effet que la position finale retournée par le filtre est très proche de la position intiale. Par ailleurs, la vérité terrain (ie. l'origine) est bien incluse dans l'ellipse de confiance à 99%. L'intégration de l'odométrie pure, quant à elle, est nettement plus éloignée.

5.3.2.2 Evolution des incertitudes au cours de l'expérimentation

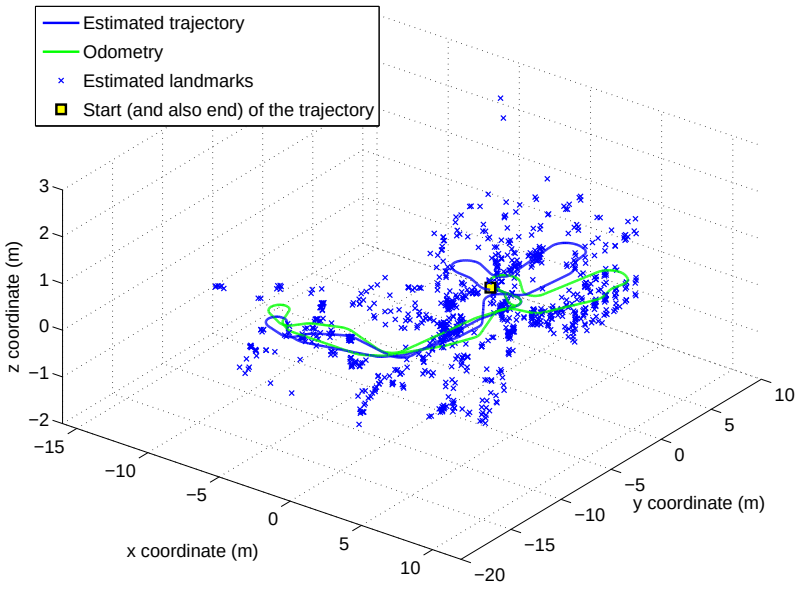
Tout comme dans l'expérimentation précédente, on constate une augmentation de la taille des enveloppes d'erreurs lorsqu'on avance dans l'expérimentation. Ceci est illustré sur la figure 5.13. On remarque que vers la fin de la trajectoire, toutes les ellipses de confiance ont une direction privilégiée. Ceci peut s'interpréter à cause de l'odométrie. En effet, la comparaison entre les figures 5.11(a) et 5.13(b) indique que l'orientation des ellipses à la fin de la trajectoire tend à retrouver l'erreur de l'odométrie pure (à une échelle moindre). Ainsi, le décalage global de l'odométrie se retrouve dans la direction des ellipses de confiance finales, ce qui est un élément supplémentaire montrant la cohérence de l'algorithme.

5.3.2.3 Bilan de ces seconds résultats

Le bilan de cette seconde expérimentation est également très bon. Nous avons en effet pu obtenir une solution tout à fait cohérente alors que l'odomètrie est largement défaillante. Même si nous n'avons pas de vérité terrain concernant les mesures de l'environnement, on retrouve des alignements cohérent ainsi qu'un angle droit (“ en bas à droite ” de la figure 5.11(a)). Enfin, la qualité du point final alors



(a)



(b)

FIGURE 5.11 – Carte globale du bâtiment Kahn et trajectoire du robot obtenues par le robot Hannibal — (a) Projection dans le plan 2D de la trajectoire estimée, de l'intégration de l'odométrie pure et des amers estimés – (b) Vue 3D des élément précédents.

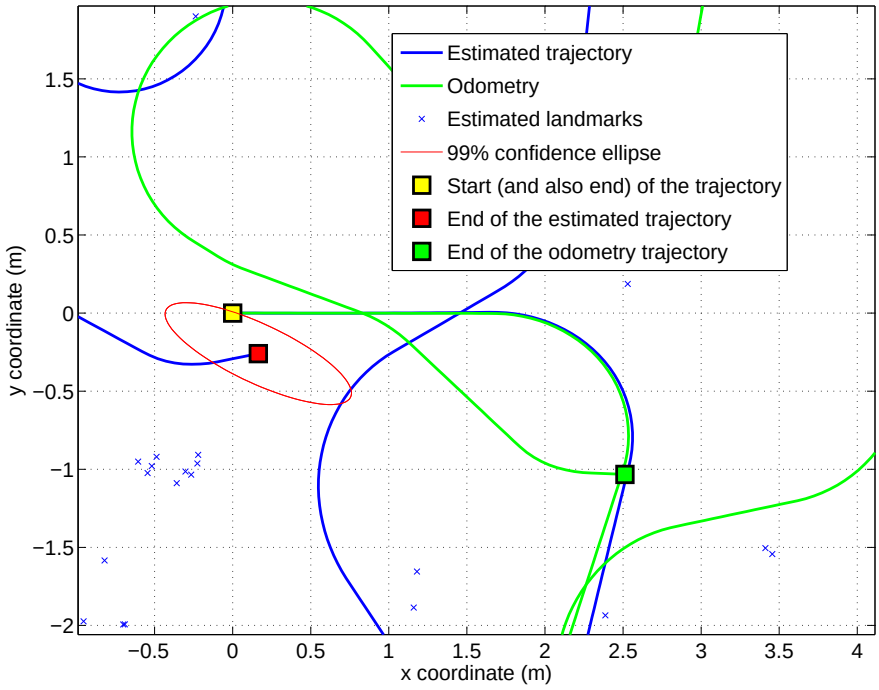
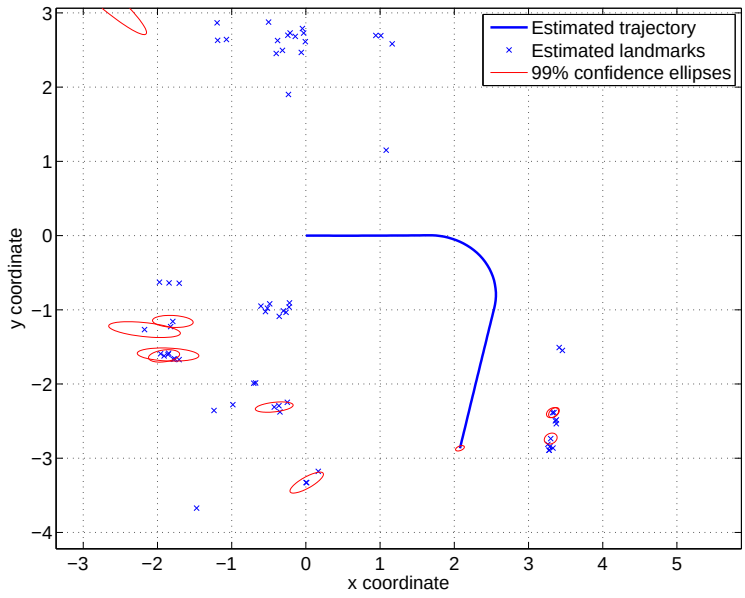


FIGURE 5.12 – Zoom sur la fin de la trajectoire. L'ellipse de confiance englobe bien la position initiale.

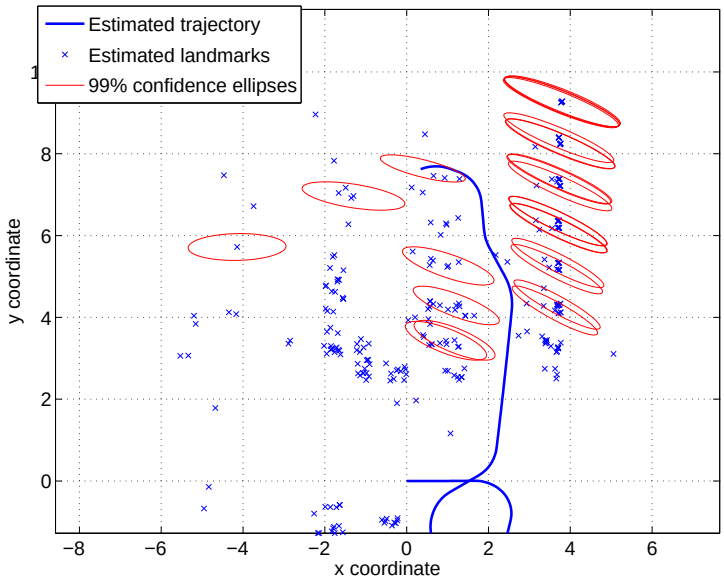
que l'on a pas remis en correspondance les points de l'environnement pouvant servir à la fermeture de boucle permet de conforter ces bons résultats.

5.3.3 Synthèse concernant les deux expérimentations

Finalement, les résultats présentés dans cette section se complètent bien. En effet, les premiers résultats ont permis de montrer de très bonnes propriétés quant à la précision métrique du résultat fourni par l'algorithme de SAM. Néanmoins, la simplicité de la scène et la bonne qualité de l'odométrie pouvaient laisser penser que ces résultats n'étaient pas significatifs. Ainsi, la seconde expérimentation permet de compléter le bilan : l'environnement ouvert et l'odométrie défaillante constituent des conditions expérimentales difficiles mais qui n'empêchent pas d'obtenir de très bons résultats.



(a)



(b)

FIGURE 5.13 – Evolution des ellipses d'incertitudes à 99% pour la seconde expérimentation

5.4 Utilisation du SLAM pour la calibration

Nous avons présenté dans la section précédente les premiers résultats utilisant l'algorithme de SAM avec 3 degrés de liberté, aidé par l'odométrie du robot. Ces premiers résultats se sont avérés très bons. Une première conséquence de ce résultat est que les propriétés d'observabilités sont bonnes, tout comme le conditionnement des équations.

Nous proposons dans cette section d'étendre ces résultats au cas où les paramètres de positionnement de la caméra ne sont pas connus. Ce type d'approche peut être intéressante dans le cas où on ne dispose pas de moyen précis de mesurer la position de la caméra par rapport aux roues motrices (cas d'une voiture par exemple).

Les nouveaux paramètres à estimer seront ainsi introduits dans le vecteur d'état utilisé dans le SAM. Nous espérons que les bons résultats de la section précédente traduisent une redondance que l'on pourra exploiter pour estimer ces paramètres.

Nous présentons dans un premier temps les aspects méthodologiques de ce nouveau problème (paragraphe 5.4.1). Nous y introduisons d'abord la notion d'observabilité pour le nouveau système ; nous y donnons ensuite les aspects pratiques pour modifier l'algorithme initial. Nous présentons enfin les résultats obtenus (paragraphe 5.4.2).

5.4.1 Méthodologie

5.4.1.1 Remarque quant à l'observabilité

Avant de modifier l'algorithme de SAM, il est nécessaire de vérifier que le nouveau problème posé est observable. Si ce n'est pas le cas, l'augmentation de l'état dans le filtre sera inutile (voire même néfaste). [11] Nous proposons dans ce paragraphe de donner une idée intuitive de ce fait grâce à une étude de l'équation de mesures, rappelée ici :

$$\left\{\begin{array}{rcl} \alpha_{t,(i)} & = & \arctan2\left(\; y_{(i)} - y_t - t_y\cos\theta_t - t_x\sin\theta_t \;\;,\; x_{(i)} - x_t - t_x\cos\theta_t + t_y\sin\theta_t \; \right) - \theta_t - \gamma \\ \beta_{t,(i)} & = & \arctan\left(\dfrac{z_{(i)}}{\sqrt{(x_{(i)}-x_t-t_x\cos\theta_t+t_y\sin\theta_t)^2+(y_{(i)}-y_t-t_y\cos\theta_t-t_x\sin\theta_t)^2}} \right) \end{array}\right. \tag{5.26}$$

Pour traiter de l'observabilité, nous supposons que toutes les mesures sont **parfaites** (odométrie et angles fournis par la caméra), que nous avons une condition initiale connue pour la position du robot. Nous cherchons à savoir s'il est possible de retrouver l'ensemble des état sous les hypothèses

11. Si un paramètres n'est pas observable, on devrait constater des pertes de rang dans la matrice de variances-covariance totale, ce qui peut provoquer des instabilités numériques lors de la marginalisation.

précédentes, dans le cas où nous avons un seul amer à estimer. La connaissance de l'odométrie dans ce problème fait que l'état du robot peut être considéré comme connu. Ainsi, il reste à savoir si on peut estimer les paramètres de l'amer et du repère caméra.

On peut alors montrer que dans le cas d'un système à un amer unique le système vérifie les propriétés d'inversion locale dès que l'on a 5 positions non cocycliques. Par extension, avec plus de 1 amer, le système restera observable car les paramètres de la caméra peuvent être supposés connus. Ainsi, **le système augmenté est observable, mais il est nécessaire d'effectuer au moins un changement de rayon de courbure dans la trajectoire**. Cette propriété a été démontrée dans l'annexe B (section B.4). Intuitivement, la qualité de l'estimation sera liée aux variations de rayon de courbure. Les deux expérimentations menées alternent entre ligne droite et virages serrés. Les conditions d'observabilité sont donc réunies.

5.4.1.2 Détails pratiques pour modifier l'algorithme initial

Pour introduire la nouvelle partie de l'état, nous avons établi quelques modifications. L'état considéré n'est plus défini par la concaténation de $\mathbf{x}_{1:t}$ et $\mathbf{m}$ mais est donné par $[\mathbf{x}_{1:t}^T, t_x, t_y, \gamma, \mathbf{m}^T]^T$. Techniquement, ceci implique d'ajouter 3 colonnes lors du calcul de la matrice jacobienne $\mathbf{H}$ associée aux mesures. Les calculs ne sont pas détaillés ici mais peuvent être facilement déduits du système 5.26.

Enfin, nous ajoutons un *a priori* concernant les paramètres de position de la caméra. [12] Ceci permet de fournir une contrainte sur la solution qui est nécessaire. En effet, nous avons vu que tant que le robot évolue en ligne droite [13] il est impossible de séparer l'information apportée à la position (x, y) des amers de celle sur la position de la caméra. Du point de vue du filtre, ce manque d'information se traduit par une perte de rang sur la matrice d'information $\mathbf{\Omega}$. On ne peut donc plus obtenir une estimation des amers par l'opération $\mathbf{\Omega}^{-1}\xi$, ce qui compromet l'utilisation du filtre. Définir un *a priori* sur la position du repère caméra (aussi grand soit-il) permet de rendre inversible la matrice d'information totale, ce qui permet d'obtenir la marginalisation de la position des amers. Enfin, l'étude précédente a montré qu'il existe une notion d'observabilité locale (le système ne peut être inversé analytiquement), ce qui justifie la connaissance d'un *a priori*. Par ailleurs, il n'est pas aberrant d'admettre que l'on peut grossièrement mesurer la position de la caméra dans le repère du robot avant le début d'une expérimentation.

12. Avant l'incrémentation de ξ et Ω grâce aux différentes matrices jacobiennes, la partie de l'état associée aux paramètres de la caméra est initialisée avec une valeur non nulle.

13. Ceci peut être fréquent en début d'expérimentation.

Soit $\mu^0_{cam} = [t^0_x\ t^0_y\ \gamma^0]^T$ une estimation *a priori* des paramètres du repère de la caméra. La matrice d'information initiale concernant ces paramètres est donnée par :

$$\mathbf{\Omega}^0_{cam} = \begin{bmatrix} \frac{1}{\sigma^2_{t_x}} & 0 & 0 \\ 0 & \frac{1}{\sigma^2_{t_y}} & 0 \\ 0 & 0 & \frac{1}{\sigma^2_{\gamma}} \end{bmatrix} \tag{5.27}$$

où σ_{t_x}, σ_{t_y} et σ_γ sont les écarts types associés à la connaissance *a priori* de la position du repère de la caméra dans le repère du robot. Ces paramètres ne doivent pas être infinis. Le vecteur d'information intial est quant à lui donné par :

$$\xi^0_{cam} = \mathbf{\Omega}^0_{cam}\mu^0_{cam} = \begin{bmatrix} \frac{t^0_x}{\sigma^2_{t_x}} & \frac{t^0_y}{\sigma^2_{t_y}} & \frac{\gamma^0}{\sigma^2_{\gamma}} \end{bmatrix}^T \tag{5.28}$$

Les paramètres numériques que nous avons utilisés sont :

- $t_x = t_y = \gamma = 0$: on donne comme *a priori* le centre de l'essieu arrière et un décalage angulaire nul. On sait pertinemment que ces valeurs ne sont pas exactes ; on aurait pu initialiser plus finement, mais nous préférons montrer que l'algorithme peut retrouver précisément les bonnes valeurs,
- $\sigma_{t_x} = \sigma_{t_x} = 1$m : ces valeurs permettent de définir une enveloppe d'incertitude qui englobe le robot,
- $\sigma_\gamma = 5$deg

Ainsi, la " zone de recherche " de la position de la caméra enveloppe globalement l'intégralité du robot (grâce aux écarts types à 1m). Par ailleurs, l'enveloppe à 99% associée à γ correspond à $[-15\ +15]$deg. Ces valeurs paraissent exploitables et englober largement les vraies valeurs, qui sont attendues autour de 10-20cm concernant les paramètres de translation et quelques degrés (tout au plus) concernant la rotation.

5.4.2 Résultats obtenus

Nous présentons dans ce paragraphe les résultats obtenus dans le cadre des deux expérimentations. Concernant le filtre, nous avons conservé les même réglages que dans le paragraphe précédent.

La figure 5.14 permet de comparer la solution obtenue par le nouvel algorithme à la solution obtenue par l'algorithme initial (sans prise en compte de la position de la caméra). On se rend compte que les cartes et trajectoires obtenues se superposent quasiment, ce qui montre la pertinence du nouvel algorithme. Par ailleurs, nous avons représenté sur la figure 5.15 l'évolution de l'estimation de t_x au cours des premiers instants ainsi que le rapport V/ω. On se rend compte que l'estimation

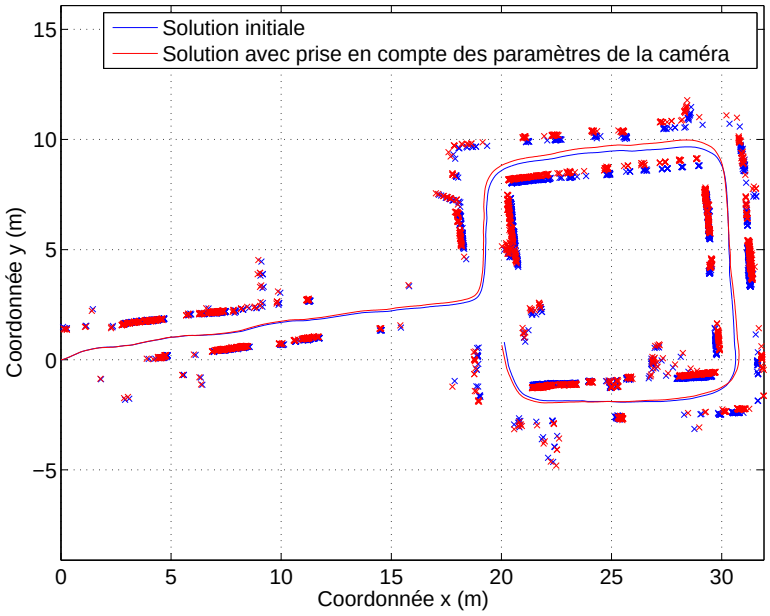


FIGURE 5.14 – Superposition des trajectoires et cartes du bâtiment Borel obtenues avec l'algorithme initial et l'algorithme évaluant la position de la caméra

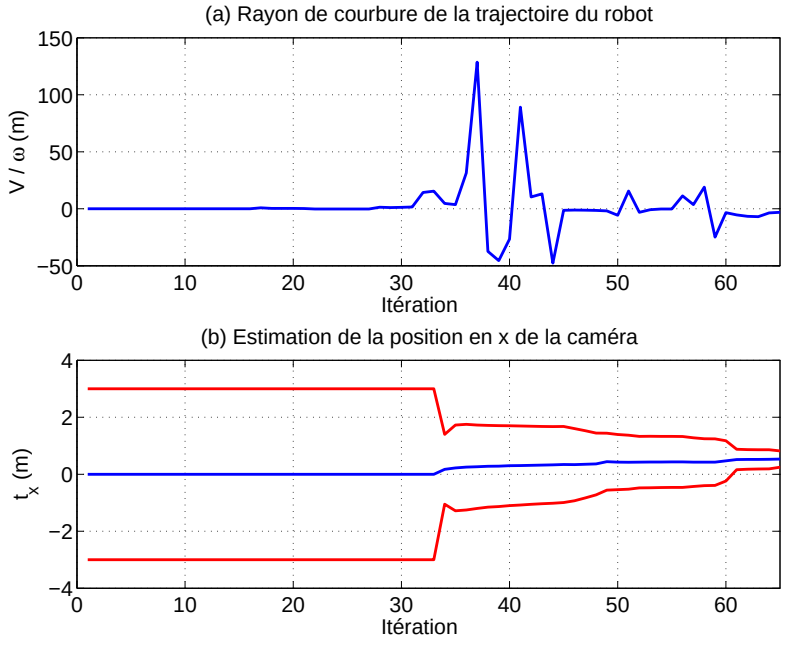


FIGURE 5.15 – Evolution du rayon de courbure et de l'estimation du paramètre t_x (avec son enveloppe à 99%)sur les premiers instants pour le cas de la séquence Borel

de t_x commence réellement (l'enveloppe de confiance à 99% se modifie largement) au moment où le rayon de courbure de la trajectoire se met à varier significativement. On retrouve alors bien le résultat d'observabilité attendu : l'estimation des paramètres de la caméra ne peut se faire que lorsque le rayon de courbure varie. Au final, les paramètres ont convergé vers :

- $t_x = 0.5575$m (intervalle à 3σ : $[0.5163, 0.5988]$m)
- $t_y = -0.2859$m (intervalle à 3σ : $[-0.3939, -0.1780]$m)
- $\gamma = 0.53$deg (intervalle à 3σ : $[0.2020, 0.8642]$deg)

Les valeurs obtenues sont très proches des valeurs réelles que l'on a mesurées : notre approche est donc parfaitement cohérente.

Des résultats similaires ont été obtenus avec la séquence du bâtiment Kahn (figures 5.16 et 5.17 [14]). Les paramètres ont finalement convergé vers :

- $t_x = -0.2046$m (intervalle à 3σ : $[-0.2121, -0.1971]$m)
- $t_y = -0.0227$m (intervalle à 3σ : $[-0.0416, -0.0039]$m)
- $\gamma = -0.0198$deg (intervalle à 3σ : $[-0.3349, 0.2953]$deg)

Tout comme pour la première expérimentation, les résultats obtenus concernant les paramètres de la caméra sont tout à fait cohérents avec les paramètres mesurés manuellement.

Au final, l'ajout de la position de la caméra dans les paramètres à estimer n'a pas posé de problème. Malgré une incertitude de départ très élevée en ce qui concerne la position de la caméra dans le repère du robot (de l'ordre du mètre en position), nous avons réussi à obtenir des zones de confiance à 99% de quelques centimètres qui sont tout à fait cohérentes avec les mesures effectuées. Par ailleurs, nous avons bien constaté que l'estimation des paramètres commence lorsque le rayon de courbure de la trajectoire varie. De plus, les résultats obtenus quant à la carte et à la trajectoire ne sont pratiquement pas modifiés par rapport à l'algorithme initial, ce qui appuie la qualité des résultats. Ces bons résultats reflètent finalement très bien les conclusions fournies par l'étude d'observabilité.

5.5 Conclusion

Nous avons présenté dans ce chapitre les résultats de SLAM visuel issus de deux expérimentations. L'algorithme initial proposé est basé sur le SAM et effectue la fusion entre les données issues de

14. Ici, la vitesse de rotation était exactement nulle sur les premiers instants, alors que la vitesse était constante. Le rayon de courbure était donc infini. Nous avons donc représenté directement les variations de ω. Sachant que V est maintenu quasi-constant sur toute l'expérimentation, les variations de ω correspondent également à des variations du rayon de courbure

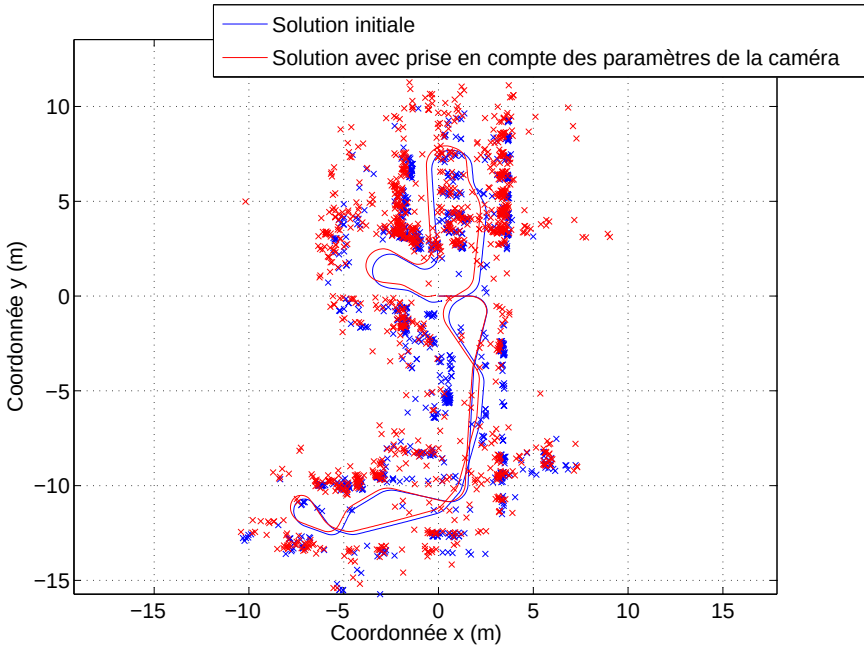


FIGURE 5.16 – Superposition des trajectoires et cartes du bâtiment Kahn obtenues avec l'algorithme initial et l'algorithme évaluant la position de la caméra

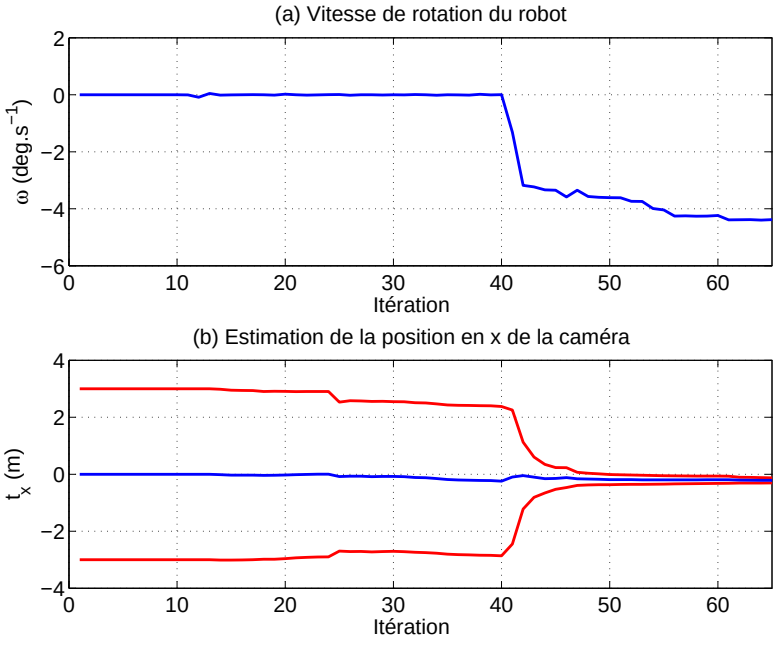


FIGURE 5.17 – Evolution du rayon de la vitesse de rotation et de l'estimation du paramètre t_x sur les premiers instants pour le cas de la séquence Kahn

l'odométrie et les points d'intérêt vus par la caméra omnidirectionnelle. Nous avons vu qu'il était possible d'améliorer largement la position fournie par l'odométrie pure (notamment dans le cas de la séquence Kahn). Nous avons également vu que notre approche de SAM peut conduire à des instabilités lorsque le nombre de mesures est insuffisant dans certaines zones. Nous verrons dans la troisième partie de ce manuscrit que l'on peut éviter ce problème avec la création de cartes locales (voir chapitre 8).

Par ailleurs, nous avons montré qu'il est possible d'estimer précisément la position de la caméra dans le repère du robot. Ceci peut être très utile dans le cas d'applications réelles où il est difficile d'effectuer des mesures précises sur la plate-forme). Une perspective directe de ces résultats serait l'estimation des paramètres de calibration de la caméra pendant le processus de SLAM.

Au final, ces résultats montrent que notre algorithme est parfaitement adapté aux environnements d'intérieurs et plans. Néanmoins, il sera mis en échec dans le cas de déplacements plus complexes. Nous proposons au chapitre suivant un algorithme permettant de traiter les mouvements du robot à 6 degrés de liberté.

Chapitre 6

Résultats de SLAM à 6 degrés de liberté

Sommaire

Nous avons présenté au chapitre précédent les résultats de SLAM visuel à 3 degrés de liberté et augmenté par des informations odométriques. L'algorithme utilisé nous a permis d'obtenir une excellente carte métrique ainsi que de développer une approche originale permettant de corriger la position de la caméra. Nous proposons dans ce chapitre de lever les deux principales limitations de l'algorithme du chapitre 5, à savoir :

1. l'utilisation de l'odométrie,
2. la limitation à 3 degrés de liberté.

Ainsi, on cherche dans ce chapitre à définir un algorithme de SLAM **purement visuel**. L'inconvénient immédiat dans l'absence d'odométrie est l'**absence complète de données métriques**. Ainsi, on sait par avance qu'on ne pourra obtenir une solution qu'**à un facteur d'échelle près**. Malgré ce manque d'information, nous verrons qu'il est possible d'obtenir des résulats tout à fait satisfaisant. Ces derniers ont été présentés à la conférence ICINCO'10 ([Joly et Rives, 2010]).

Ce chapitre s'articule en 5 sections. Nous présentons dans la section 6.1 les éléments du nouveau problème considéré (hypothèses et équations). Nous discutons ensuite de l'importance d'initialiser les amers dès le premier instant d'observation (section 6.2). Dans ce but, nous verrons deux approches couramment utilisées dans la littérature. La section 6.3, quant à elle, est dédiée à l'adaptation au SAM de l'algorithme de représentation des amers avec l'inverse de la profondeur. Des résultats sur données réelles sont présentées dans la section 6.4. Nous concluons ce chapitre par une synthèse des résultats montrant les avantages et limites pratiques des solutions proposées.

6.1 Nouveau problème

6.1.1 Introduction

Nous présentons dans cette section les nouveaux éléments liés à l'algorithme de SLAM visuel à 6 degrés de liberté. Nous introduisons dans un premier temps les nouvelles équations liées à l'évolution du robot et au vecteur de mesures (paragraphes 6.1.2 et 6.1.3). Nous formulons enfin quelques remarques quant à l'observabilité du nouveau système.

6.1.2 Mouvement de la caméra

6.1.2.1 Mouvement libre : plus de contrainte sur l'odométrie

Contrairement aux chapitres précédents, nous considérons maintenant une caméra libre de se déplacer dans l'espace suivant six degrés de liberté : trois en rotation et trois en translation. Aucune

mesure proprioceptive de ce déplacement n'est supposée disponible du fait de l'absence d'odométrie. Ainsi, on considère que l'évolution de la caméra se fait à l'aide d'un vecteur vitesse défini dans le repère global ainsi que d'un vecteur rotation défini dans le repère du robot. Ces deux vecteurs d'entrées sont supposés constants entre deux instants consécutifs t et $t+1$. Dans notre cas, les vecteurs de vitesses linéaire et angulaire ne sont pas connus. Pour cette raison, nous les introduisons dans le vecteur d'état pour les estimer.

Finalement, la paramétrisation choisie pour le modèle d'évolution de la caméra est donnée par :

$$\mathbf{x}_t = \begin{bmatrix} \mathbf{r}_t^T & \mathbf{v}_t^T & \mathbf{q}_t^T & \omega_t^T \end{bmatrix}^T \tag{6.1}$$

où on a défini :

- $\mathbf{r}_t = [x_t \ y_t \ z_t]^T$ la position de la caméra dans le repère global,
- $\mathbf{v}_t = [v_{xt} \ v_{y_t} \ v_{zt}]^T$ la vitesse de translation (constante) entre les instants t et $t+1$ et exprimée dans le repère global,
- $\mathbf{q}_t = [q_{1t} \ q_{2t} \ q_{3t} \ q_{4t}]^T$ le quaternion permettant de représenter la matrice de rotation du repère caméra. La matrice de rotation obtenue à partir de $\mathbf{q}_t$ est la matrice de rotation $\mathbf{R}^{WC}$ permettant de transformer les coordonnées d'un vecteur du repère de la caméra vers le repère du global,
- $\omega_t = [\omega_{xt} \ \omega_{y_t} \ \omega_{zt}]^T$ le vecteur de vitesse de rotation entre les instants t et $t+1$ et exprimé dans le repère du robot.

Remarque 6.1 (Utilisation des quaternions)
L'utilisation des quaternions peut paraître délicate. En effet, les quaternions sont des vecteurs de dimension 4 qui représentent des éléments de $SO(3)$: ils ne sont donc pas une représentation minimale du groupe de matrices de rotations. Pour assurer une cohérence, les quaternions sont normalisés. Néanmoins, l'estimation d'un quaternion dans un filtre peut conduire à un vecteur dont la norme est différente de 1 (à moins d'imposer une mesure virtuelle pour respecter la contrainte). En pratique, nous avons constaté que ceci ne pose pas de problème.

A chaque itération, les quaternions obtenus par l'algorithme de SAM sont pratiquement de norme unitaire (ce qui montre une certaine cohérence), si bien que le fait de les normaliser ne change pratiquement pas les valeurs numériques initiales. Nous avons donc choisi pour stratégie de renormaliser à chaque itération les quaternions obtenus avant de les utiliser à nouveau comme point de fonctionnement. Ainsi, les quaternions obtenus après transformation du vecteur et de matrice d'information en espérance ont toujours une norme très proche de 1.

Malgré ce problème lié à la représentation non minimale, l'utilisation des quaternions permet de définir toutes les matrices de rotation possibles, ce qui est impossible avec les représentations minimales qui contiennent toutes des singularités, comme par exemple les angles d'Euler. Enfin, les règles de dérivation et composition des quaternions sont beaucoup plus simples que dans le cas des représentations angulaires.

6.1.2.2 Nouvelle équation de modèle

La nouvelle équation permettant de prédire l'état entre les instants t et $t+1$ est basée sur l'intégration du vecteur de rotation instantanée et du vecteur vitesse entre ces deux instants.

Nous supposons qu'un vecteur d'entrées ($[\mathbf{V}_t^T\ \mathbf{\Omega}_t^T]^T$) [1] agit sur les vitesses. Ce vecteur de commande n'étant pas mesuré, nous le modélisons par un vecteur aléatoire gaussien centré de matrice de variances-covariances connue. Nous ferons attention au moment du " réglage " à faire en sorte que cette matrice puisse bien prendre en compte l'amplitude des accélérations que l'on admettra pour le robot.

Pour intégrer les vitesses et obtenir le vecteur position et le quaternion, nous supposons que les vitesses intégrées sont constantes entre les deux instants considérés. Ceci n'est pas tout à fait le cas, ce qui rend l'équation d'intégration " basique " (ie. à vitesse constante) légèrement inexacte. Pour prendre en compte ce fait, nous supposons que la nouvelle position et orientation du robot sont le résultat de l'intégration d'une vitesse constante légèrement différente de celle prévue. Nous notons $\nu_t^{\mathbf{v}}$ et ν_t^{ω} les perturbations considérées. Elles sont modélisées par des bruits blancs gaussiens centrés et indépendants de matrice de variances-covariances connue (en ordre de grandeur, la variance associée à ces bruits est bien inférieure à celle associée au vecteur $[\mathbf{V}_t^T\ \mathbf{\Omega}_t^T]^T$).

Finalement, la nouvelle équation d'évolution est donnée par :

$$\mathbf{x}_{t+1} = \begin{bmatrix} \mathbf{r}_{t+1} \\ \mathbf{q}_{t+1} \\ \mathbf{v}_{t+1} \\ \omega_{t+1} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_t + \left(\mathbf{v}_t + \frac{\mathbf{V}_t}{2} + \nu_t^{\mathbf{v}}\right)\Delta t \\ \mathbf{q}_t \otimes \mathbf{q}\left(\left(\omega_t + \frac{\mathbf{\Omega}_t}{2} + \nu_t^{\omega}\right)\Delta t\right) \\ \mathbf{v}_t + \mathbf{V}_t \\ \omega_t + \mathbf{\Omega}_t \end{bmatrix} \tag{6.2}$$

où $\mathbf{q}(\mathbf{u})$ désigne le quaternion associé au vecteur de rotation $\mathbf{u}$ et $\otimes$ l'opérateur permettant de multiplier deux quaternions (voir [Hartley et Zisserman, 2000] pour les formules élémentaires sur les quaternions).

La matrice de variances-covariances associée à ce modèle est donnée par :

$$\mathbf{Q}_t = \mathbf{J}_{\mathbf{u}}\mathbf{Q}_{\mathbf{u}}\mathbf{J}_{\mathbf{u}}^T \tag{6.3}$$

1. Ici $\mathbf{\Omega}_t$ ne désigne pas la matrice d'information associée au filtre.

avec

$$\mathbf{J_u} = \left[\frac{\partial \mathbf{x}_{t+1}}{\partial \mathbf{V}_t} \frac{\partial \mathbf{x}_{t+1}}{\partial \mathbf{\Omega}_t} \frac{\partial \mathbf{x}_{t+1}}{\partial \nu_t^{\mathbf{v}}} \frac{\partial \mathbf{x}_{t+1}}{\partial \nu_t^{\omega}} \right] \tag{6.4}$$

et

$$\mathbf{Q_u} = \begin{bmatrix} S_{\mathbf{V}} \Delta t \mathbf{I} & 0 & 0 & 0 \\ 0 & S_{\mathbf{\Omega}} \Delta t \mathbf{I} & 0 & 0 \\ 0 & 0 & S_{\nu^{\mathbf{v}}} \Delta t \mathbf{I} & 0 \\ 0 & 0 & 0 & S_{\nu^{\omega}} \Delta t \mathbf{I} \end{bmatrix} \tag{6.5}$$

où $S_{\mathbf{V}}$, $S_{\mathbf{\Omega}}$, $S_{\nu^{\mathbf{v}}}$ et $S_{\nu^{\omega}}$ sont les paramètres associés aux bruits considérés.

6.1.3 Vecteur de mesures

Dans ce chapitre, le vecteur de mesures est constitué des mesures en pixels fournies par la caméra. Soient $x_{(i)}^C$, $y_{(i)}^C$ et $z_{(i)}^C$ les coordonnées de l'amer (i) exprimées dans le repère de la caméra. La mesure en pixels est donnée par les étapes du modèle de projection unifiée rappelées ci-après :

1. Normalisation des coordonnées : $[x_{(i)}^S \ y_{(i)}^S \ z_{(i)}^S] = \frac{1}{\left(x_{(i)}^C\right)^2 + \left(y_{(i)}^C\right)^2 + \left(z_{(i)}^C\right)^2} [x_{(i)}^C \ y_{(i)}^C \ z_{(i)}^C]$
2. Correction avec le paramètre du miroir : $\mathbf{m} = \frac{1}{z_{(i)}^S + \xi} [x_{(i)}^S \ y_{(i)}^S \ z_{(i)}^S]$
3. Multiplication par la matrice des paramètres intrinsèques : $\mathbf{p} = \mathbf{K}\mathbf{m}$

6.1.4 Notions d'observabilité

Nous proposons dans ce paragraphe quelques remarques quant à " l'observabilité ". Il est dans ce cas très difficile de mener une étude d'observabilité aussi rigoureuse que celle menée pour le cas du SLAM à trois degrés de liberté. En effet, les commandes $\mathbf{V}_t$ et $\mathbf{\Omega}_t$ de l'équation 6.2 sont inconnues (dans le domaine continu, ces deux commandes correspondent à des accélérations). Dans ce cas, effectuer une étude d'observabilité impliquerait de reformuler le système continu de sorte à le rendre **affine en la commande**.[2] Cette étape est très délicate à cause de la présence d'une matrice de rotation à 3 degrés de liberté. Trouver une représentation des rotations qui permet de rendre le système affine en la commande est non trivial. Enfin, l'étude à réaliser serait alors nettement plus compliquée que celle présentée en annexe B.

Néanmoins, nous nous permettons de formuler quelques remarques communément admises :

– on retrouve les mêmes problèmes concernant la transformation globale rigide (trajectoire et carte) que dans le cas à 3 degrés de liberté. Fixer la condition initiale du robot avec une position

2. Avoir un système affine en la commande est indispensable pour réaliser une étude d'observabilité indépendante de la commande.

nulle et une matrice de rotation égale à l'identité (quaternion associé : $[1\ 0\ 0\ 0]^T$) permet de résoudre ce problème,

– l'ensemble des amers et de la trajectoire sera alors observable à **un facteur d'échelle près**, ceci étant dû à l'**absence de données métriques**.

6.2 Initialisation non retardée des amers

Dans les chapitres précédents, nous avons utilisé une approche retardée pour l'initialisation des amers. On attendait que le rapport entre la parallaxe de l'amer et l'incertitude associée soit favorable à une initialisation consistante. Dans le cas de notre algorithme de SLAM à 6 degrés de liberté, nous préférons utiliser une initialisation immédiate des amers.

Nous présentons dans le paragraphe 6.2.1 les motivations qui ont poussé à utiliser une telle approche. Les paragraphes 6.2.2 et 6.2.3 présentent deux algorithmes pertinents tirés de la littérature. Nous concluons enfin cette section en donnant la méthode finale retenue pour l'initialisation des amers.

6.2.1 Motivations

Nous présentons dans ce chapitre une méthode de SLAM purement visuelle. Le modèle d'évolution présenté dans l'équation 6.2 suppose que les vitesses de translation et de rotation sont constantes entre deux instants consécutifs (les entrées $\mathbf{V}_t$ et $\mathbf{\Omega}_t$ sont en effet considérées comme des bruits blancs gaussiens). Comparativement au cas du SLAM à trois degrés de liberté aidé par l'odométrie, l'équation de prédiction est beaucoup moins fiable. Lorsque la dynamique du robot change, l'algorithme de SLAM ne pourra pas le " voir " directement. Ainsi, le fait d'attendre une parallaxe suffisante peut conduire à une mauvaise initialisation si l'équation de modèle n'a pu prédire correctement la position courante du robot. Par ailleurs, il apparaît que lorsque les amers sont vus avec une faible parallaxe, ceux-ci peuvent apporter une information sur l'orientation du robot, même s'ils ne permettent pas de calculer la position du robot et qu'on ne peut pas calculer leur position.

En conséquence, de nombreuses publications traitant de l'initialisation et de la façon de représenter les amers proposent une classification des amers en deux groupes (voir [Trawny et Roumeliotis, 2006] par exemple) :

Amers à l'infini : les amers situés à une distance " infinie " peuvent servir de " boussole ". Bien qu'ils n'apportent pas d'information sur la position du robot, son orientation peut être calculée. En pratique, cet aspect était déjà visible dans la matrice jacobienne $\mathbf{H}$ dans le cas du SLAM à

3 degrés de liberté (équations 3.11 et 3.12 page 89) : les coefficients liés aux positions du robot et de l'amer tendent vers zéro lorsque la distance entre l'amer et le robot tend vers l'infini alors que celui associé à l'orientation du robot vaut -1. Dans le cas où des amers à distance " non infinie " sont vus, on montre que l'information qu'ils apportent est analogue à celle d'amers à l'infini tant que la parallaxe reste petite. C'est cette propriété que l'on essayera d'exploiter en initialisant les amers dès le premier instant où ils sont vus.

Amers à distance connue : les amers apportent à la fois une information sur l'orientation du robot ainsi que sur sa position (et leur propre position aussi). Lorsqu'un amer a été vu avec suffisamment de parallaxe, il passe du groupe précédent à ce groupe. Typiquement, on peut directement les représenter avec leur coordonnées euclidiennes.

Ainsi, même les amers qui n'ont pas été observés avec une parallaxe suffisante apportent une information sur l'état du système. En revanche, on ne peut pas les représenter de manière classique (introduire des coordonnées euclidiennes alors qu'on ne peut pas les observer rendrait le filtre inconsistant). Il est donc nécessaire d'effectuer un traitement spécifique afin de tirer partie des informations fournies par les amers " boussoles ".

Nous proposons dans la suite de cette section deux méthodes issues de la bibliographie qui ont été appliquées avec succès aux méthodes de SLAM par EKF. Nous discutons par ailleurs de la possible intégration avec une méthode de type SAM.

6.2.2 Filtre multi-hypothèses ([Solà, 2007])

Une première série de méthodes consiste à utiliser différentes hypothèses afin de prendre en compte l'incertitude quant à la profondeur à l'instant initial. Dans ce but, Joan Solá propose dans sa thèse une méthode pour intégrer toutes les hypothèses au sein d'un filtre de Kalman unique ([Solà *et al.*, 2005; Solà, 2007]). Il s'agit de la méthode FIS (*Federated Information Sharing*). Chaque hypothèse viendrait augmenter l'état et serait initialisée avec une espérance et une matrice de variances-covariances données. Enfin, un poids est associé à chaque hypothèse et mis à jour à chaque itération.

L'échantillonnage des hypothèses se fait avec la notion astucieuse de " rayon conique ". Lorsque le robot observe un amer, le cas idéal voudrait que l'amer appartienne au rayon partant dans la direction mesurée. Cette direction étant entachée d'une erreur, la zone où peut se situer l'amer est en réalité un cône infini (représenté en rouge sur la figure 6.1). L'idée est alors de placer les différentes hypothèses de sorte à remplir ce cône. Joan Solá montre que ceci est possible grâce à l'utilisation d'une suite

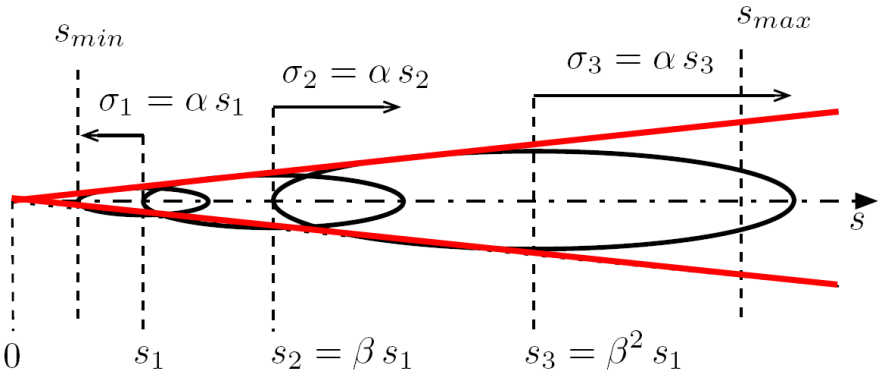


FIGURE 6.1 – Concept de rayon conique (adapté de [Solà, 2007])

géométrique pour définir les hypothèses de profondeur. Soit s_i la profondeur de l'hypothèse i et σ_i l'écart-type associé. Ces valeurs sont définies par :

- s_1 fixé et $\sigma_1 = \alpha s_1$. Ceci permet de définir la profondeur minimale considérée (s_{min}).
- $s_{i+1} = \beta s_i$ et $\sigma_i = \alpha s_i$.

Le paramètre β détermine la densité de la série d'hypothèses. Le paramètre α est quant à lui défini de sorte à ce que $s_1 - \sigma_1 = s_{min}$ (s_{min} étant un paramètre que l'on se donne). Enfin, le nombre d'hypothèses utilisées (N_g) est fonction de la valeur s_{max} que l'on souhaite pouvoir représenter. Les paramètres α, β, s_1 et N_g sont donc des paramètres à définir de sorte à représenter le plus fidèlement possible le rayon conique initial. Il est montré dans [Solà, 2007] que le couple $(\alpha, \beta) = (0.3, 3)$ donne de bonnes performances en pratique. Par ailleurs, il est montré dans [Lemaire *et al.*, 2005] que le choix de ces valeurs n'est pas critique. Les valeurs de s_1, σ_1 et N_g sont calculées en fonction des valeurs que l'on souhaite pour s_{min} et s_{max}.

Lorsque l'initialisation des rayons coniques est réalisée, chaque hypothèse considérée est traitée dans le filtre de Kalman étendu comme un amer. Ensuite, les poids sont mis à jour grâce aux innovations calculées. Les hypothèses dont le poids devient trop proche de zéro sont ensuite éliminées (figure 6.2).

D'autres méthodes multi-hypothèses pour initialiser les amers dès la première itération ont été développées comme, par exemple, l'utilisation d'une somme de gaussiennes ([Kwok et Dissanayake, 2004]). L'idée est alors d'utiliser plusieurs filtres en parallèle, d'associer un poids à chaque solution et d'éliminer ensuite les mauvaises hypothèses. Il s'agit de l'algorithme de GSF-SLAM (*Gaussian Sum Filter SLAM*). Le problème du GSF-SLAM réside dans l'utilisation d'un nombre élevé d'amers : **le nombre de combinaisons nécessaires pour représenter toutes les hypothèses interdit toute application en temps réel**. L'algorithme FIS, quant à lui, est plus léger étant donné qu'il n'utilise

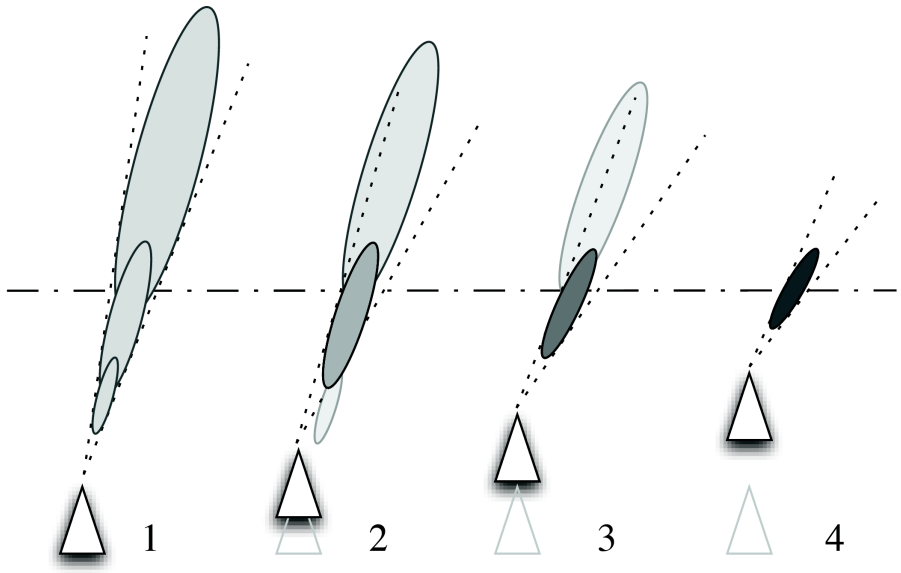


FIGURE 6.2 – Evolution des hypothèses de l'algorithme FIS en fonction des itérations ([Solà, 2007]). La ligne pointillée représente la position réelle.

qu'une seule matrice de variances-covariances. La figure 6.3 illustre les différences des algorithmes lors de l'introduction d'un nouvel amer.

6.2.3 Surparamétrisation avec l'inverse de la profondeur ([Civera *et al.*, 2008])

Nous présentons dans ce paragraphe une méthode permettant d'introduire les amers dans le filtre dès la première observation sans effectuer d'hypothèses multiples [Civera *et al.*, 2008; Montiel *et al.*, 2006]. Pour arriver à ce résultat, les auteurs utilisent une nouvelle représentation des amers basée sur l'inverse de leur profondeur. Il s'agit d'une représentation proche des coordonnées sphériques avec néanmoins deux différences :

1. Il s'agit d'une représentation robot-centrée. Si on veut pouvoir initialiser immédiatement les angles de gisement et d'élévation, il est nécessaire de centrer la représentation sur la position courante de la caméra,
2. On n'utilise pas la distance entre la position centrale et celle de la caméra mais l'**inverse de cette distance**.

Ainsi, un amer (i) est représenté par un vecteur à six dimensions :

$$\mathbf{y}_{(i)} = \begin{bmatrix} x^0_{(i)} & y^0_{(i)} & z^0_{(i)} & \theta_{(i)} & \phi_{(i)} & \rho_{(i)} \end{bmatrix}^T \tag{6.6}$$

où on a défini :

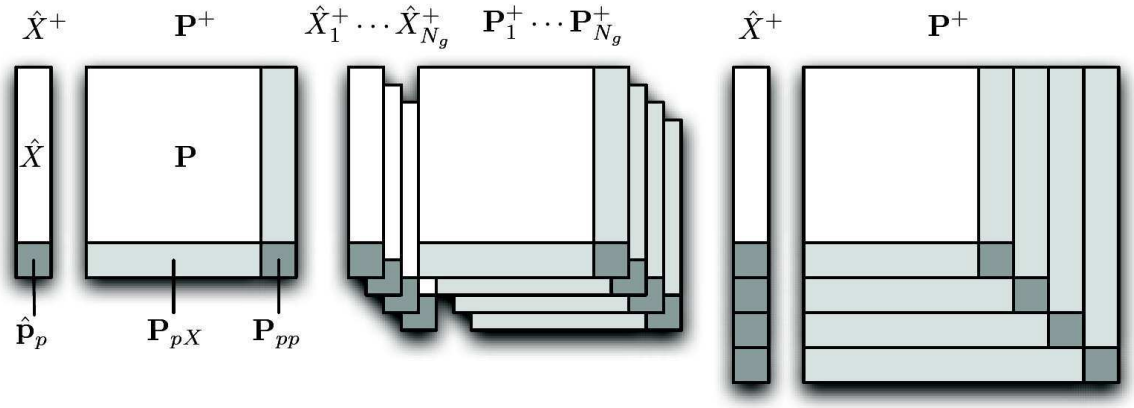


FIGURE 6.3 – Ajout d'un amer dans le cas de l'EKF classique, du GSF-SLAM et du FIS-SLAM ([Solà, 2007]) — Gauche : EKF classique, on introduit qu'une seule hypothèse lors de l'initialisation d'un amer (ne peut être réalisé de façon consistante que si l'amer a été vu avec une parallaxe suffisante) – Centre : GSF-SLAM, on introduit un nouveau filtre pour chaque hypothèse. Si on avait déjà N hypothèses, on en aurait alors $N.N_g$ après cette opération — Droite : FIS-SLAM, on augmente l'état par autant d'amers que d'hypothèses considérées.

- $x^0_{(i)}$, $y^0_{(i)}$, $z^0_{(i)}$ les coordonnées euclidiennes de la caméra au moment où l'amer a été observé pour la première fois,
- $\theta_{(i)}$, $\phi_{(i)}$ les angles de gisement et d'élévation de l'amer par rapport à la position de la caméra au moment de la première observation. Ces angles sont définis dans le repère global et définissent la direction de l'amer donnée par le vecteur unitaire $\mathbf{d}(\theta_{(i)}, \phi_{(i)})$,
- $\rho_{(i)}$ l'inverse de la profondeur entre l'amer et la position de la caméra à la première observation.

Ainsi, on peut retrouver les coordonnées euclidiennes de l'amer par :

$$\mathbf{m}_{(i)}^{Eucl} = \begin{bmatrix} x_{(i)} \\ y_{(i)} \\ z_{(i)} \end{bmatrix} = \begin{bmatrix} x^0_{(i)} \\ y^0_{(i)} \\ z^0_{(i)} \end{bmatrix} + \frac{1}{\rho_{(i)}} \underbrace{\begin{bmatrix} \cos\phi_{(i)} \cos\theta_{(i)} \\ \cos\phi_{(i)} \sin\theta_{(i)} \\ \sin\phi_{(i)} \end{bmatrix}}_{\mathbf{d}(\theta_{(i)}, \phi_{(i)})} \tag{6.7}$$

Les auteurs de [Civera *et al.*, 2008; Montiel *et al.*, 2006] justifient cette paramétrisation en étudiant la linéarité de l'équation de mesures associé au filtre de Kalman étendu. En effet, plus l'équation sera linéaire autour d'un point de fonctionnement, et plus l'hypothèse de propagation des densités de probabilités gaussiennes sera pertinente. Les auteurs comparent la linéarité des équations à l'aide d'un critère basé sur la dérivée seconde, et ce dans le cas où on paramètre avec la profondeur et dans le cas où on paramètre avec l'inverse de la profondeur. Les résultats montrent que :

- La linéarité n'est pas suffisante dans le cas où utilise directement la profondeur lorsque l'angle de parallaxe est trop faible,

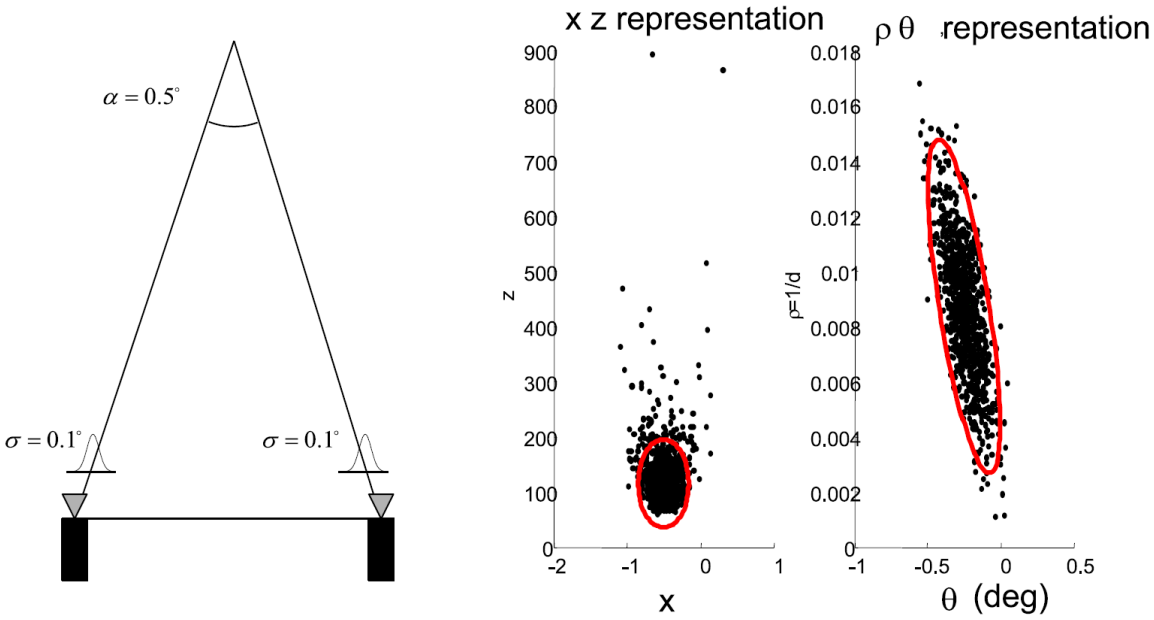


FIGURE 6.4 – Intérêt de la représentation par profondeur inverse ([Montiel *et al.*, 2006]) — Gauche : présentation des données simulées. Un point est vu de deux positions présentant une très faible parallaxe. Un nombre élevé d'échantillons de mesures gaussiens a été simulé afin d'observer la répartition de la reconstruction – Centre : répartition des reconstruction lorsque l'amer est paramétré par ses coordonnées euclidiennes. L'ellipse rouge correspond à la propagation des incertitudes après linéarisation sous hypothèse gaussienne (95% de confiance). La répartition des solutions n'est clairement pas gaussienne avec la paramétrisation en XY – Droite : paramétrisation de l'amer avec un angle et l'inverse de la profondeur. La répartition est alors gaussienne et concorde bien avec l'ellipse rouge.

– L'équation faisant intervenir l'inverse de la profondeur possède de bonnes propriétés de linéarité dès les très faibles parallaxes, et ce pour de larges valeurs de $\rho_{(i)}$. Par exemple, les auteurs de [Montiel *et al.*, 2006] montrent qu'elle est valide pour $\rho \in [0\ \rho_{max}]$ (la valeur de ρ_{max} dépendant des paramètres choisis). En conséquence, les distances acceptables pour l'amer sont comprises dans l'intervalle $[\frac{1}{\rho_{max}}\ +\infty]$. Cette paramétrisation permet donc de représenter conjointement des amers à distance raisonnable (pourvu que ρ_{max} ne soit pas trop petit) et des amers très loin.

Les résultats de cette étude sont illustrés sur la figure 6.4 pour le cas d'une simulation 2D avec faible parallaxe. On voit alors bien l'intérêt de la représentation par l'inverse de la profondeur.

Cette dernière propriété fait que la qualité de la solution dépend très peu de l'initialisation des $\rho_{(i)}$. Ainsi, les auteurs de [Civera *et al.*, 2008] montrent que l'on peut fixer des valeurs assez génériques pour la valeur initiale des $\rho_{(i)}$ (notée ρ_0) et l'écart-type associé (σ_ρ), du moment que les distances infinies sont clairement prises en compte. Par exemple, ils suggèrent d'utiliser $(\rho_0, \sigma_\rho = (0.1, 0.5)$, ce qui donne un intervalle de confiance à 95% de $[-0.9\ 1.1]$. Ceci permet de prendre en compte des

distances comprises entre 0.9m et l'infini, ce qui est assez réaliste pour une application de robotique mobile terrestre. [3]

La représentation avec l'inverse de la profondeur permet donc d'initialiser " facilement " les nouveaux amers. Elle permet également de prendre en compte simultanément tous les amers. Néanmoins, on pourra trouver trois inconvénients à cette représentation :

1. Il est nécessaire d'augmenter l'état. La représentation **n'est plus minimale**. Ceci peut poser des problèmes de stabilité pour l'algorithme de filtrage. Il existe une infinité d'états permettant de représenter une position euclidienne donnée : quelques soient les trois premières composantes choisies, il existera un triplet (ϕ, θ, ρ) permettant de représenter l'amer. Cette représentation introduit donc un problème d'observabilité évident,
2. L'augmentation de l'état implique une charge de calcul supplémentaire : la dimension associée aux amers est **doublée**, ce qui peut poser problème pour une implémentation temps réel. Néanmoins, une méthode permettant de changer de représentation a été présentée dans [Civera *et al.*, 2007]. Ainsi, lorsque la parallaxe devient suffisant, on peut passer d'une représentation en l'inverse de la profondeur à une représentation classique euclidienne des amers.
3. Théoriquement, les valeurs autorisées pour l'inverse de la profondeur doivent être strictement positives. Il est impossible d'introduire ce type de contrainte dans un filtre de Kalman étendu classique, les densités de probabilités gaussiennes étant non nulles sur tout $\mathbb{R}$. Ainsi, il est possible que certaines profondeurs inverses deviennent négatives lorsque les amers sont trop éloignés. Cela signifie que dans l'estimation, l'amer passe d'une distance infinie dans une direction à une distance infinie mais en direction inverse. Ce type de valeurs aberrantes peuvent éventuellement être supprimées du filtre ou réinitialisées. Une solution est proposée dans [Parsley et Julier, 2008] pour les traiter.

6.2.4 Conclusion et choix d'une méthode

Nous avons présenté dans les paragraphes précédents les deux façons les plus utilisées pour initialiser les amers dès le premier instant où ils sont vus. Nous avons au final choisi de considérer la représentation avec l'inverse de la profondeur pour notre algorithme de SAM à 6 degrés de liberté. Les raisons qui ont motivé ce choix sont :

3. Dans [Montiel *et al.*, 2006], les mêmes auteurs proposent une méthode légèrement différente pour initialiser ces valeurs. Ils font en sorte que l'intervalle à $\pm 2\sigma_\rho$ soit égal à $[0\ \frac{1}{d_{min}}]$ (d_{min} étant une valeur que l'on se fixe). Nous avons préféré retenir la version donnée dans la publication la plus récente.

- La représentation avec l'inverse de la profondeur permet de faire coexister les deux types d'amers définis dans le paragraphe 6.2.1 : à savoir les amers à l'infini et les amers à distance finie.
- Le filtre multi-hypothèses fait cohabiter dans le même filtre des amers inconsistants (ceux dont la profondeur est mauvaise) et des amers corrects. Il n'y a pas de garantie que l'influence des mauvais amers ne perturbera pas la solution,
- Les filtres multi-hypothèses classiques (somme de gaussiennes) sont trop lourds à mettre en œuvre à cause de la combinatoire associée aux hypothèses dès que l'on a plusieurs amers,
- Les filtres multi-hypothèses ne s'utilisent que pour une distance maximale finie fixée à l'avance,
- La représentation avec l'inverse de la profondeur peut être améliorée dans le cas du SAM. En effet, l'état est augmenté dans le cas de l'EKF car ce filtre ne conserve que la dernière position du robot. Dans le cas du SAM, toute la trajectoire est estimée à chaque itération. On peut donc *a priori* utiliser cette donnée pour ne pas surparamétriser l'état des amers.

En conclusion, le choix de la représentation inverse de la profondeur nous semble la solution la plus pertinente pour initialiser les amers dès la première itération. Nous présentons dans la section suivante les modifications apportées pour implémenter cette méthode dans le cas de notre algorithme de SAM.

6.3 Application de la méthode *inverse depth* au SAM

Nous présentons dans cette section les détails de notre algorithme de SAM utilisant la représentation inverse de la profondeur. Nous montrons dans un premier temps que nous pouvons obtenir un **algorithme utilisant une représentation minimale des amers** (paragraphe 6.3.1). Nous effectuons ensuite une comparaison des nouvelles propriétés avec celles de l'algorithme présenté dans [Civera *et al.*, 2008] (paragraphe 6.3.2).

6.3.1 Représentation minimale des amers

6.3.1.1 Principe

La représentation inverse de la profondeur nécessite 6 paramètres que l'on peut répartir dans deux vecteurs (cf. figure 6.5) :

1. $\mathbf{y}^1_{(i)} = [x^0_{(i)}\ y^0_{(i)}\ z^0_{(i)}]^T$ la position de la caméra au moment où l'amer est vu pour la première fois,
2. $\mathbf{y}^2_{(i)} = [\theta_{(i)}\ \phi_{(i)}\ \rho_{(i)}]^T$ les paramètres définissant le vecteur reliant $\mathbf{y}^1_{(i)}$ à l'amer.

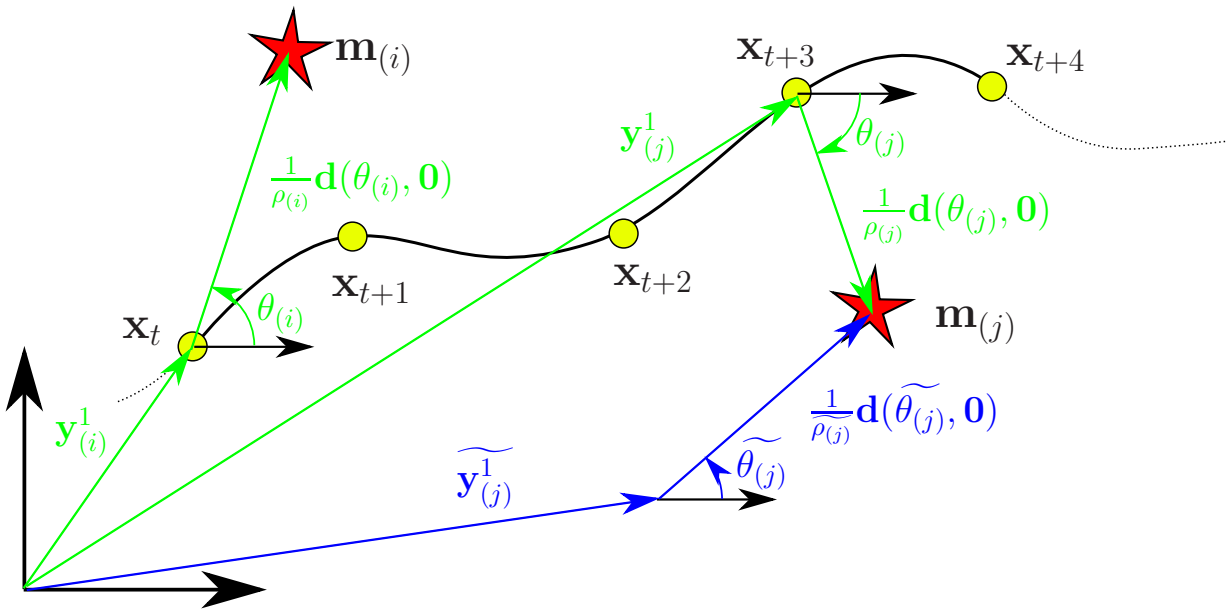


FIGURE 6.5 – Illustration de la représentation inverse de la profondeur dans le cas 2D — Dans cet exemple, l'amer (i) est vu pour la première fois à l'instant t. Il est donc paramétré par le vecteur $\mathbf{y}^1_{(i)}$ (que l'on prend égal à la valeur estimée pour $\mathbf{x}_t$) et par $\rho_{(i)}$, $\theta_{(i)}$ et $\phi_{(i)} = 0$. L'amer (j), quant à lui, est vu pour la première fois à l'instant $t+3$. Les vecteurs en vert représentent la paramétrisation classique avec l'inverse de la profondeur. Nous avons représenté en bleu une autre possibilité pour choisir $\mathbf{y}^1_{(j)}$. Il en résulte de nouvelles valeurs pour $\rho_{(j)}$ et $\theta_{(j)}$. Il est ainsi possible de fixer $\mathbf{y}^1_{(j)}$ comme on le souhaite. On pourra toujours trouver le vecteur $\mathbf{y}^2_{(j)}$ qui convient.

La position d'un amer est normalement entièrement définie par 3 paramètres. Supposons que le vecteur $\mathbf{y}^1_{(i)}$ soit connu et fixé à l'avance. Le vecteur $\mathbf{y}^2_{(i)}$ contient suffisamment d'information pour définir l'amer. Ainsi, on se rend compte que la seconde partie de la représentation permet de représenter entièrement tous les amers. Si on fixe le vecteur $\mathbf{y}^1_{(i)}$ à zéro, on retrouve la représentation sphérique classique (à l'exception du fait qu'on n'utilise pas la profondeur mais son inverse).

L'idée de notre algorithme est **de ne considérer que le vecteur $\mathbf{y}^2_{(i)}$ dans l'état**, étant donné que quel que soit le vecteur $\mathbf{y}^1_{(i)}$ utilisé, il existera un vecteur $\mathbf{y}^2_{(i)}$ permettant de représenter l'amer. L'équation de mesure prendra le vecteur $\mathbf{y}^1_{(i)}$ comme paramètre, mais restera une fonction de $\mathbb{R}^3$ dans $\mathbb{R}^2$. Si l'on se réfère à l'équation principale du SAM (équation 1.91 page 58), le vecteur $\mathbf{y}^1_{(i)}$ aura un rôle analogue au vecteur μ (qui ne sert que de point de fonctionnement) alors que le vecteur $\mathbf{y}^2_{(i)}$ fera partie intégrante du vecteur $\mathbf{m}$.

Théoriquement, on peut utiliser n'importe quelle valeur pour le vecteur $\mathbf{y}^1_{(i)}$ (voir le cas en bleu sur la figure 6.5). L'ensemble des mesures permettra alors de déterminer le vecteur $\mathbf{y}^2_{(i)}$. Néanmoins, si notre but est d'entrer l'amer dans le filtre dès la première itération, il est nécessaire de choisir une

valeur de $\mathbf{y}^1_{(i)}$ qui permettra d'initialiser le vecteur $\mathbf{y}^2_{(i)}$ suffisamment proche de la solution réelle. Dans ce but, on utilise la valeur estimée de la position de la caméra à l'instant d'intialisation. Aux itérations suivantes, on utilisera pour fixer $\mathbf{y}^1_{(i)}$ l'estimée courante de la position de la caméra au premier instant où l'amer est vu. [4]

Ensuite, lors de la mise à jour de l'algorithme de SAM, nous mettons à jour ces positions grâce à la nouvelle trajectoire obtenue. Cette dernière étape n'est théoriquement pas indispensable, mais elle permet d'optimiser la qualité des linéarisations des équations de mesures.

Au final, l'algorithme présenté dans ce paragraphe permet de ne conserver dans l'estimation que 3 paramètres pour chaque amer : **la représentation est donc minimale**. Nous verrons au paragraphe 6.3.2 que ceci constitue une différence théorique majeure par rapport à l'algorithme initial basé sur le filtre de Kalman étendu. On pose donc pour la suite de ce chapitre :

$$\mathbf{m}_{(i)} = \begin{bmatrix} \phi_{(i)} & \theta_{(i)} & \rho_{(i)} \end{bmatrix}^T \tag{6.8}$$

6.3.1.2 Equation de mesures complète

Nous présentons dans ce paragraphe la nouvelle équation de mesures. Pour cela, nous cherchons à retrouver les coordonnées euclidiennes de l'amer exprimées dans le repère courant associé à la caméra. Ces coordonnées sont ensuite projetées dans l'image omnidirectionnelle.

Les coordonnées de l'amer (i) dans le repère de la caméra sont données par :

$$\mathbf{m}^C = \mathbf{R}^{CW}_t \left(\mathbf{m}^{Eucl} - \mathbf{r}_t \right) \tag{6.9}$$

où $\mathbf{m}^{Eucl}$ est donné par l'équation 6.7 (page 218), et $\mathbf{R}^{CW}_t = \left(\mathbf{R}^{WC}_t\right)^T$ ($\mathbf{R}^{WC}_t$ désigne la matrice de rotation associée au quaternion $\mathbf{q}_t$). On a donc :

$$\mathbf{m}^C = \mathbf{R}^{CW}_t \left(\begin{bmatrix} x^0_{(i)} \\ y^0_{(i)} \\ z^0_{(i)} \end{bmatrix} + \frac{1}{\rho_{(i)}} \mathbf{d}(\theta_{(i)}, \phi_{(i)}) - \mathbf{r}_t \right) = \frac{1}{\rho_{(i)}} \mathbf{R}^{CW}_t \left(\rho_{(i)} \left(\begin{bmatrix} x^0_{(i)} \\ y^0_{(i)} \\ z^0_{(i)} \end{bmatrix} - \mathbf{r}_t \right) + \mathbf{d}(\theta_{(i)}, \phi_{(i)}) \right) \tag{6.10}$$

Soit $\mathbf{h}_1$ la fonction qui permet de passer des coordonnées euclidiennes dans le repère caméra aux coordonnées en pixels dans l'image (suivant les 3 étapes décrites dans le paragraphe 6.1.3 page 213) :

$$\begin{array}{rccc} \mathbf{h}_1 : & \mathbb{R}^3 & \rightarrow & \mathbb{R}^2 \\ & \mathbf{m}^C & \mapsto & \mathbf{p} = \mathbf{h}_1\left(\mathbf{m}^C\right) \end{array} \tag{6.11}$$

4. $\mathbf{y}^1_{(i)}$ peut alors suivre les mises à jour de la trajectoire tout en conservant son rôle de point de fonctionnement.

La première étape de la fonction $\mathbf{h}_1$ consiste à normaliser les coordonnées sur le sphère unité. En conséquence, la fonction $\mathbf{h}_1$ est invariante à un changement d'échelle positif :

$$\forall \lambda \in \mathbb{R}^+,\ \forall \mathbf{m}^C \in \mathbb{R}^3, \quad \mathbf{h}_1(\lambda \mathbf{m}^C) = \mathbf{h}_1(\mathbf{m}^C) \tag{6.12}$$

Soit désormais $\mathbf{h}_2$ la fonction **paramétrée par** $x^0_{(i)}$, $y^0_{(i)}$ et $z^0_{(i)}$, de $\mathbb{R}^3 \times \mathbb{R}^6$ dans $\mathbb{R}^3$ définie par :

$$\begin{array}{rccl} \mathbf{h}_2 : & \mathbb{R}^3 \times \mathbb{R}^6 & \rightarrow & \mathbb{R}^3 \\ & (\mathbf{m}_{(i)}, \mathbf{x}_t) & \mapsto & \mathbf{R}_t^{CW} \left(\rho_{(i)} \left(\begin{bmatrix} x^0_{(i)} \\ y^0_{(i)} \\ z^0_{(i)} \end{bmatrix} - \mathbf{r}_t \right) + \mathbf{d}(\theta_{(i)}, \phi_{(i)}) \right) \end{array} \tag{6.13}$$

En prenant en compte l'invariance de la fonction $\mathbf{h}_1$ par changement d'échelle, la fonction d'observation finale est donnée par :

$$\begin{array}{rccl} \mathbf{h} : & \mathbb{R}^3 \times \mathbb{R}^6 & \rightarrow & \mathbb{R}^2 \\ & (\mathbf{m}_{(i)}, \mathbf{x}_t) & \mapsto & \mathbf{h}_1 \circ \mathbf{h}_2(\mathbf{m}_{(i)}, \mathbf{x}_t) \end{array} \tag{6.14}$$

Il s'agit d'une fonction paramétrée par $x^0_{(i)}$, $y^0_{(i)}$ et $z^0_{(i)}$. Ainsi, ces 3 composantes sont considérées comme déterministes. [5] La matrice jacobienne $\mathbf{H}$ associée à $\mathbf{h}$ ne comprendra pas de terme lié à la dérivation par rapport à ces 3 paramètres.

6.3.1.3 Initialisation des amers

L'initialisation des amers que nous proposons diffère quelque peu de celle présentée dans [Civera *et al.*, 2008; Montiel *et al.*, 2006]. Dans ces publications, les auteurs proposent de calculer les valeurs intiales des angles $\phi_{(i)}$ et $\theta_{(i)}$ grâce à l'orientation courante estimée et à la mesure effectuée. Ensuite, l'état est augmenté avec les valeurs calculées (pour $\phi_{(i)}$ et $\theta_{(i)}$), par l'estimation courante de $\mathbf{r}_t$ et avec la valeur prédéfinie pour $\rho_{(i)}$. De plus, la matrice de variances-covariances est augmentée en propageant les incertitudes liées à la mesure initiale de l'amer ainsi que les incertitudes de la position courante et l'incertitude prédéfinie pour l'inverse de profondeur.

Dans le cas du SAM, l'initialisation se fait plus simplement. Nous calculons dans un premier temps des valeurs initiales pour $\phi_{(i)}$ et $\theta_{(i)}$ comme dans le cas de l'EKF-SLAM. Ces valeurs seront utilisées pour définir le point de fonctionnement associé au amer. En revanche, nous n'avons pas besoin d'ajouter de terme dans la matrice et le vecteur d'information concernant ces composantes : les équations du SAM prendront directement en compte la première mesure de l'amer lors du calcul des

5. Ces composantes servent de point de fonctionnement : elles sont donc bien déterministes. Le fait qu'on choisisse de les fixer aux valeurs estimées de la position de la caméra où l'amer est vu pour la première fois constitue un choix " stratégique " ; ce choix nous permet d'initialiser les amers dès la première itération. Ceci aurait été impossible si on avait fixé ces 3 composantes à zéro (représentation sphérique).

paramètres d'information. Ajouter des termes supplémentaites au moment de l'initialisation reviendrait donc à prendre en compte deux fois la mesure initiale, ce qui tendrait à rendre le filtre inconsistant. L'initialisation des valeurs de $\phi_{(i)}$ et $\theta_{(i)}$ se fait en deux étapes :

1. Calcul de la direction de l'amer dans le repère global. Elle est obtenue par :

$$\mathbf{u} = \begin{bmatrix} u_{x(i)} \\ u_{y(i)} \\ u_{z(i)} \end{bmatrix} = \mathbf{R}_t^{WC} \begin{bmatrix} x_{(i)}^S \\ y_{(i)}^S \\ z_{(i)}^S \end{bmatrix} \tag{6.15}$$

   où $x_{(i)}^S$, $y_{(i)}^S$ et $z_{(i)}^S$ sont les coordonnées de la direction de l'amer dans le repère de la caméra (déduites directement de l'image). $\mathbf{R}_t^{WC}$ est l'estimation courante de la matrice de rotation.

2. Déduction de $\phi_{(i)}$ et $\theta_{(i)}$ par :

$$\begin{cases} \phi_{(i)} & = & \arctan2(u_{y(i)}, u_{x(i)}) \\ \theta_{(i)} & = & \arctan\left(\frac{u_z}{\sqrt{u_{x(i)}^2 + u_{y(i)}^2}}\right) \end{cases} \tag{6.16}$$

En ce qui concerne l'initialisation de $\rho_{(i)}$, nous utilisons la valeur ρ_0 préconisée dans [Civera *et al.*, 2008] pour le calcul du point de fonctionnement. Ceci n'est pas suffisant : lors de la première itération, le vecteur de mesure n'est que de dimension 2 (alors que le vecteur à initialiser est de dimension 3) : aucune information sur la profondeur n'est donnée. Pour combler ce manque d'information, nous ajoutons des termes aux composantes de la matrice et du vecteur d'information associés aux $\rho_{(i)}$:

$$\begin{cases} \sigma_\rho^{-2} & \text{sur la diagonale de la matrice } \mathbf{\Omega} \text{ aux indices correspondant aux } \rho_{(i)} \\ \sigma_\rho^{-2}\rho_0 & \text{sur le vecteur } \xi \text{ aux indices correspondant aux } \rho_{(i)} \end{cases} \tag{6.17}$$

Cette dernière opération permet de fixer la distribution *a priori* pour la valeur des $\rho_{(i)}$. Elle permet par ailleurs d'assurer que la matrice d'information soit constamment de rang plein (sinon, le fait d'utiliser un état de dimension 3 mais une mesure initiale de dimension 2 impliquerait des chutes de rang sur la matrice d'information).

6.3.1.4 Cas des amers à profondeur négative

Dans cet algorithme, rien ne garantit que l'inverse de la profondeur ne devienne pas négative. Lorsque de tels cas se présentent, nous avons choisi, par sécurité, de les considérer comme des points aberrants. Pour ne pas perturber le reste de l'estimation, les points concernés sont retirés du filtre.

Cette approche peut être améliorée mais n'a pas été approfondie dans ces travaux. Des débuts de piste peuvent être trouvés dans [Parsley et Julier, 2008] pour le cas de l'EKF.

6.3.2 Comparaison avec l'algorithme initial

Nous proposons dans ce paragraphe de comparer les propriétés de notre algorithme avec celles de l'algorithme proposé dans les papiers originaux. Outre les bonnes propriétés du SAM en matière de consistance, nous allons voir que la représentation minimale permet de garantir le respect d'hypothèses nécessaires à l'utilisation des filtres.

Nous avons vu qu'intuitivement, l'utilisation des 6 paramètres dans le filtre implique des problèmes d'observabilité. En effet, quelles que soient les valeurs utilisées pour définir les 3 premières composantes, on pourra toujours trouver un triplet $(\rho_{(i)}, \theta_{(i)}, \phi_{(i)})$ qui permet de définir correctement l'amer. **Toutes ces solutions sont donc indiscernables**. Il n'existe donc pas une seule solution pour le filtre, mais une infinité, ce qui peut évidemment poser des problèmes de convergence pour le filtre Kalman.

Décrivons par ailleurs les propriétés que l'algorithme de SAM aurait eues **si nous avions conservé les 6 paramètres**. Supposons que l'on doive initialiser l'amer (i) à l'instant k. L'initialisation des 3 premières composantes de $\mathbf{m}_{(i)}$ est par construction **complètement corrélée** avec $\mathbf{r}_k$. La conséquence directe est que la matrice de variances-covariances associée à la concaténation de $\mathbf{r}_k$ et des 3 premières composantes de $\mathbf{m}_{(i)}$ n'est que de rang 3 (alors qu'on a 6 paramètres). On constate alors qu'on a une chute de rang naturelle dans la matrice d'information totale, ce qui peut conduire à des problèmes numériques. Examinons désormais ce qui se passe lorsqu'on observe l'amer (i) pour $t > k$:

1. Dans un premier temps, on peut remarquer que la mesure à l'instant t permet " naturellement " d'améliorer la localisation de l'amer ainsi que celle du robot. Par inférence dans le réseau bayesien, l'estimation de $\mathbf{r}_k$ est également améliorée,

2. Dans un second temps, on peut remarquer que les trois premières composantes de $\mathbf{m}_{(i)}$ sont complètement correlées avec $\mathbf{r}_k$. Ainsi, l'ajout d'information sur les 3 premières composantes de $\mathbf{m}_{(i)}$ grâce à la matrice jacobienne $\mathbf{H}$ est propagée **de manière directe** sur l'estimation de $\mathbf{r}_k$.

La deuxième source d'information est inconsistante. Elle équivaut en effet à dire que l'observation à l'instant t de l'amer (i) apporte directement de l'information sur la position de l'amer, la position du robot à l'instant t et la position du robot à l'instant k. Autrement dit, on crée un lien dans le réseau bayésien qui n'a pas lieu d'être (figure 6.6). Ce dernier lien contredit une des hypothèses utilisées dans l'écriture des équations du SAM : **l'hypothèse stipulant que la mesure à l'instant t ne dépend que de l'état du robot à l'instant t et de l'amer considéré n'est plus vérifiée**. Enfin, on a pu remarquer que l'implémentation d'un tel algorithme conduit systématiquement à des divergences

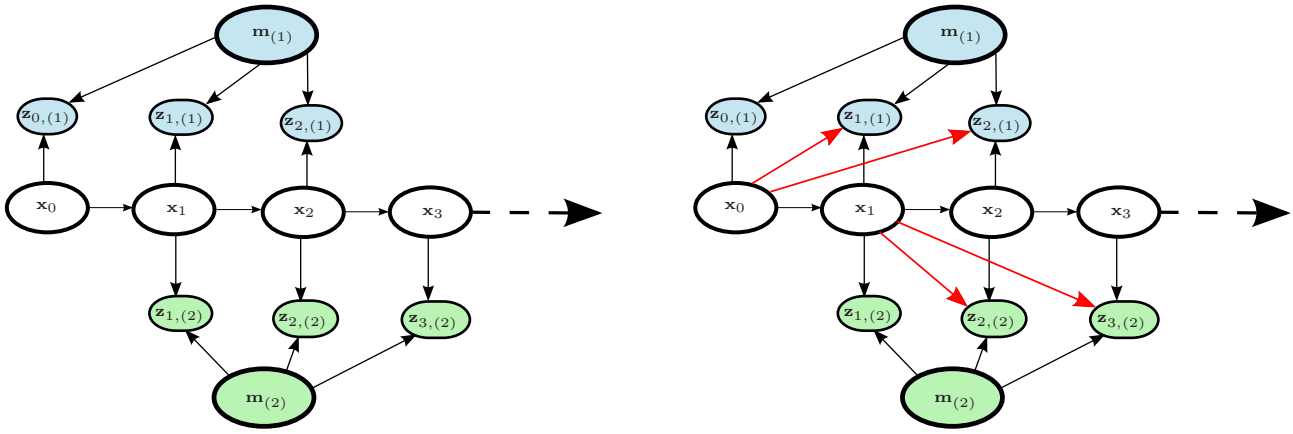


FIGURE 6.6 – Réseau bayésien associé dans le cas de 2 amers. Le premier amer est vu aux instants 0, 1 et 2. Le second est vu aux instants 1, 2 et 3 — (a) Réseau associé à notre algorithme – (b) Réseau associé à la paramétrisation initiale. Les flèches rouges représentent des liens inconsistants liés à la surparamétrisation.

(la matrice d'information étant en fait impossible à inverser).

Dans l'algorithme présenté dans [Montiel *et al.*, 2006], les auteurs utilisent la paramétrisation à 6 coordonnées dans le cadre d'un filtre de Kalman étendu et non dans le cas du SAM. Néanmoins, la problématique est la même. Cette représentation aboutit également à l'ajout de lien dans le réseau bayésien, rendant alors l'algorithme de filtrage de Kalman non adapté (une des hypothèses du filtre de Kalman étant que la mesure à l'instant t ne dépend que de l'état à l'instant t). Le fait que le filtre de Kalman travaille uniquement sur la densité de probabilité marginale limite les risques de divergence et " d'explosion " du résultat qui ont été observés dans le cadre du SAM. Ceci n'empêche pas le filtre d'être inapplicable d'un point de vue théorique. Enfin, on pourra remarquer qu'une méthode permettant de limiter ce problème serait d'initialiser les amers sans prendre en compte la corrélation avec l'instant initial lors de l'initialisation de la matrice de variances-covariances (cette méthode n'a pas été retenue dans les publications de référence). Le risque est alors de ne pas contraindre suffisamment les 3 premières composantes de l'amer : le risque d'avoir une solution oscillante est alors plus grand.

6.3.3 Synthèse

Au final, nous avons présenté dans cette section une adaptation de la méthode de représentation inverse au cas du SAM. Nous avons montré que notre algorithme possède de meilleures propriétés théoriques du fait que la représentation des amers est désormais **minimale**. Ainsi, la différence d'ap-

proche réside dans la définition de la fonction de mesures $\mathbf{h}$:

- Dans le cas de l'EKF original, on a $\mathbf{z}_{t,(i)} = \mathbf{h}(\mathbf{x}_t, \mathbf{r}_k, \phi_{(i)}, \theta_{(i)}\rho_{(i)})$,
- Dans le cas de notre algorithme, on a $\mathbf{z}_{t,(i)} = \mathbf{h}_{\mu_k^{\mathbf{r}}}(\mathbf{x}_t, \phi_{(i)}, \theta_{(i)}\rho_{(i)})$ (où $\mu_k^{\mathbf{r}}$ désigne le point de fonctionnement courant pour $\mathbf{r}_k$),

(où on a supposé que l'amer (i) a été vu pour la première fois à l'instant k).

Notre approche est possible grâce à l'utilisation de l'algorithme de SAM : le fait d'estimer constamment la trajectoire permet d'affiner la valeur de $\mu_k^{\mathbf{r}}$ de sorte à garantir une linéarisation correcte des équations.

6.4 Résultats

Nous présentons dans cette section des résultats issus de deux manipulations réelles :

1. La première expérimentation correspond à l'expérimentation dans le bâtiment Kahn que nous avons présenté au chapitre précédent. Elle est réalisée sur **sol plan**, ce qui constitue une vérité terrain : la trajectoire trouvée pour la caméra doit appartenir au plan horizontal,
2. La seconde expérimentation conduit à un déplacement de la plate-forme dans l'espace : le robot évolue sur deux plans reliés entre eux par un plan incliné. Cette trajectoire nous permettra de vérifier que l'algorithme est capable de retrouver cette trajectoire plus compliquée.

6.4.1 Séquence planaire

Dans un premier temps, nous avons essayé d'appliquer notre algorithme sur la séquence planaire du bâtiment Kahn présentée au chapitre précédent. Le but original était de vérifier si le résultat obtenu était bien cohérent par rapport aux résultats du chapitre précédent (à un facteur d'échelle près). Par ailleurs, on s'attend à ce que l'altitude ne varie pratiquement pas.

Nous n'avons malheureusement pas réussi à traiter l'intégralité de la séquence. L'algorithme était systématiquement soumis à des divergences sur le long terme. Plusieurs raisons peuvent expliquer cette " contre performance " :

- La séquence initiale est très longue. Nous avons à traiter au total environ 1400 itérations. La dimension du vecteur d'état est **au moins** incrémentée de 13 unités à chaque itération. Ainsi, si on avait réussi à traiter l'intégralité de la séquence, on aurait $1400 \times 13 = 18200$ composantes **pour décrire uniquement la trajectoire du robot**.[6] Même si la matrice d'information est

6. Dans le cas à 3 degrés de liberté aidé par l'odométrie, on avait environ $1400 \times 3 = 4200$ composantes, ce qui

éparse, un vecteur d'état de cette taille peut entraîner l'apparition de problèmes numériques. Par ailleurs, le nombre d'amers à traiter sur l'intégralité de cette séquence est conséquent, ce qui rend l'état encore plus grand.

– Certaines images ne possèdent pas assez de point d'intérêt. Ceci tend à ne pas contraindre suffisamment la solution. On retrouve alors des " glissements " de la solution (comme dans le cas présenté sur la figure 5.10 page 196). Un effet de bord de cette instabilité est une mauvaise détections des outliers lors de la reprojection des points dans l'image. En effet, les oscillations ou glissements constants de la solution font que la prédiction du mouvement se trouve légèrement trop erronée. La conséquence directe est que certains points corrects sont classés comme des *outliers*. Ceci contribue à faire diminuer le nombre de points acceptés et tend à dégrader la solution. Lorsque cette situation dure trop longtemps, l'algorithme rentre dans un cercle vicieux dont l'issue est la divergence et le rejet systématique de tous les points acceptés.

Ainsi, la combinaison de ces deux effets fait que l'algorithme ne retourne pas de solution cohérente sur le long terme, les problèmes numériques amplifiant les problèmes sur les données. Par ailleurs, les temps de calculs grandissent considérablement avec le nombre d'amers.

En conséquence, nous n'essayons pas de traiter l'ensemble de la trajectoire en une seule fois. Nous nous contentons de traiter 3 fenêtres indépendamment les unes des autres. Celles-ci ont été dimensionnées en fonction des capacités de l'algorithme à fournir une solution valable :

Fenêtre 1 : itérations 1 à 400

Fenêtre 2 : itérations 350 à 900

Fenêtre 3 : itérations 850 à 1272 (fin)

Les résultats sont présentés sur les figures 6.7 et 6.8. On peut tout d'abord constater que l'on retrouve bien les trajectoires obtenues dans le chapitre précédent. De plus, la visualisation des trajectoires 3D montre que l'algorithme détecte bien que la caméra évolue dans le plan $z = 0$, ce qui est confirmé par la visualisation séparée des coordonnées sur la figure 6.8. Nous avons enfin converti les quaternions obtenus en angles d'eulers (roulis, tangage et lacet). On retrouve bien des angles très faibles en valeur absolue, systématiquement inférieur à 2 degrés. Les mouvements de rotation de la caméra se font donc exclusivement autour de l'axe **z** : on retrouve bien le résultat attendu.

En conséquence, les résultats obtenus par l'algorithme sont donc très encourageants. Malgré les 6 degrés de liberté " déverrouillés " par l'algorithme, on retrouve bien toutes les caractéristiques de la

constitue une différence très importante.

trajectoire plane : les angles de roulis et tangages sont quasi-nuls, tout comme la coordonnée z.

6.4.2 Séquence avec plan incliné

6.4.2.1 Présentation de la séquence

Dans cette seconde expérimentation, nous testons les capacités de l'algorithme à retrouver une trajectoire complexe dans l'espace. Pour cela, nous avons construit une plate-forme surélevée reliée au sol par une rampe (figure 6.9). Les caractéristiques de la plate-forme sont :

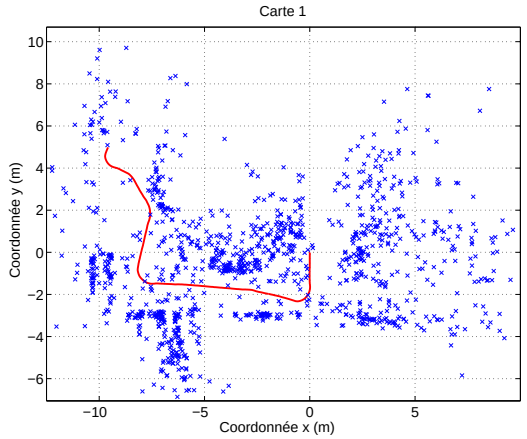
- Hauteur de la plate-forme : 35cm
- Longueur de la rampe (ie. longueur de l'hypoténuse) : 180cm
- Angle déduit de la rampe : $\gamma = \arcsin\left(\frac{35}{180}\right) \approx 11.2\,\text{deg}$

La trajectoire réalisée peut être découpée en 6 phases :

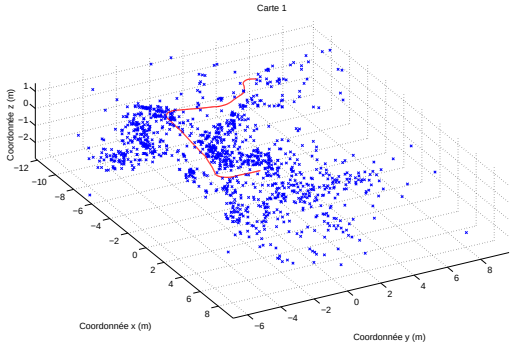
1. Avancée sur le plan du sol à vitesse réduite puis virage serré pour monter sur la rampe (figure 6.9a) ,
2. Montée de la rampe en ligne droite à puissance maximale (figure 6.9b). La vitesse réelle diminue progressivement par manque de motricité du robot,
3. Avancée en ligne droite sur le plateau (figure 6.9c),
4. Demi-tour sur place (figure 6.9d). Ce demi-tour se traduit par une trajectoire en demi-cercle de la caméra (la caméra n'étant pas montée sur le centre de l'essieu moteur),
5. Avancée en ligne droite vers la rampe (figure 6.9e),
6. Descente de la rampe à vitesse modérée constante puis virage serré pour relonger le mur,
7. Avancée en ligne droite le long du mur (figure 6.9f)

On a donc ici une trajectoire complexe avec des changements d'orientation brutaux. Dans la suite, nous nous intéresserons à voir si on retrouve bien les propriétés de la trajectoire. Nous avons de plus introduit une fermeture de boucle pendant la descente de la rampe figure 6.10. Ce type de fermeture de boucle est rendu possible par la caméra omnidirectionnelle (l'angle entre la montée et la descente étant d'envrion 180 degrés).

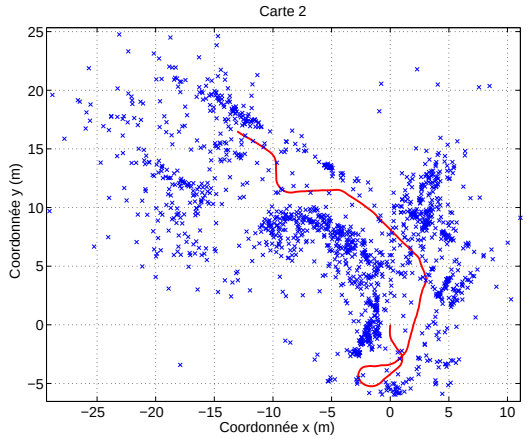
Nous allons utiliser cette trajectoire originale pour montrer les qualités de notre algorithme de SAM par rapport à l'algorithme initial basé sur la représentation inverse de la profondeur. Les données issues du traitement d'images ainsi que la fermeture de boucle sont effectuées à l'avance et sont rigoureusement les même pour les deux algorithmes. Les résultats qui suivent ont été obtenus avec exactement le même jeu de données.



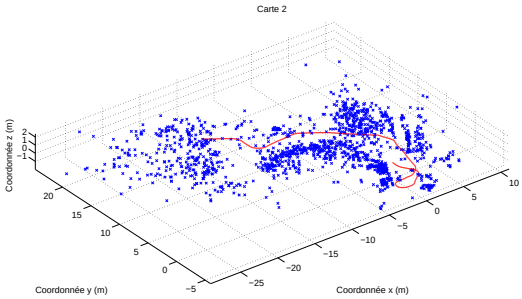
(a)



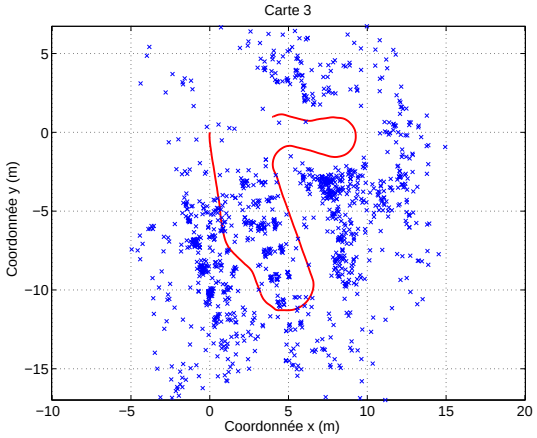
(b)



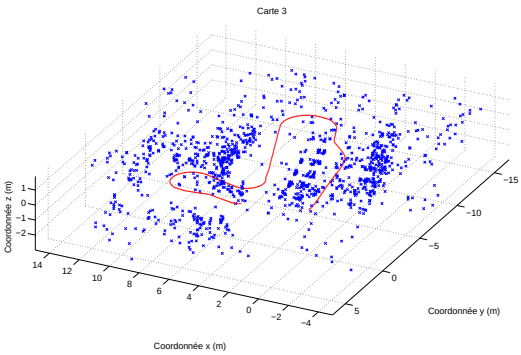
(c)



(d)

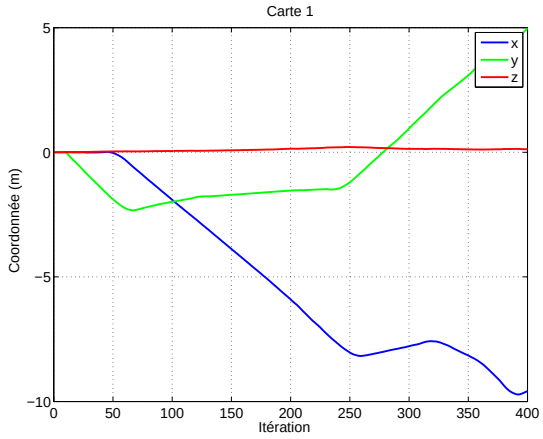


(e)

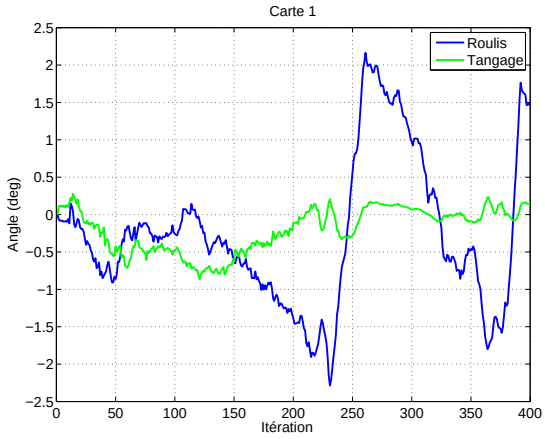


(f)

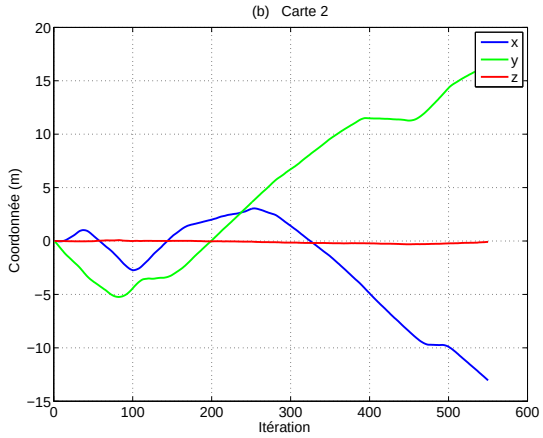
FIGURE 6.7 – Cartes et trajectoires obtenues par le SLAM à 6 degrés de liberté dans le cas de la séquence plane



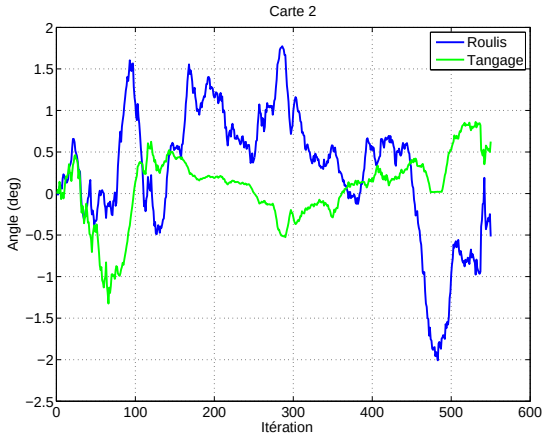
(a)



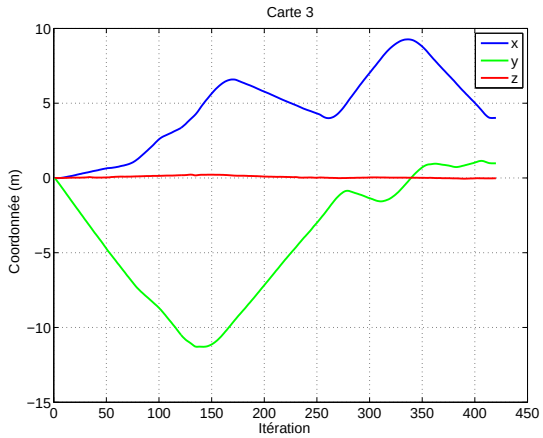
(b)



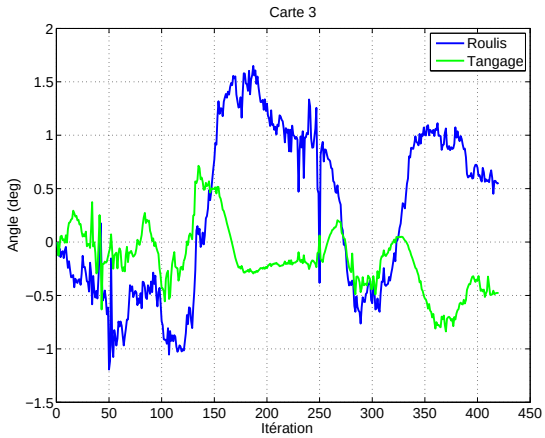
(c)



(d)

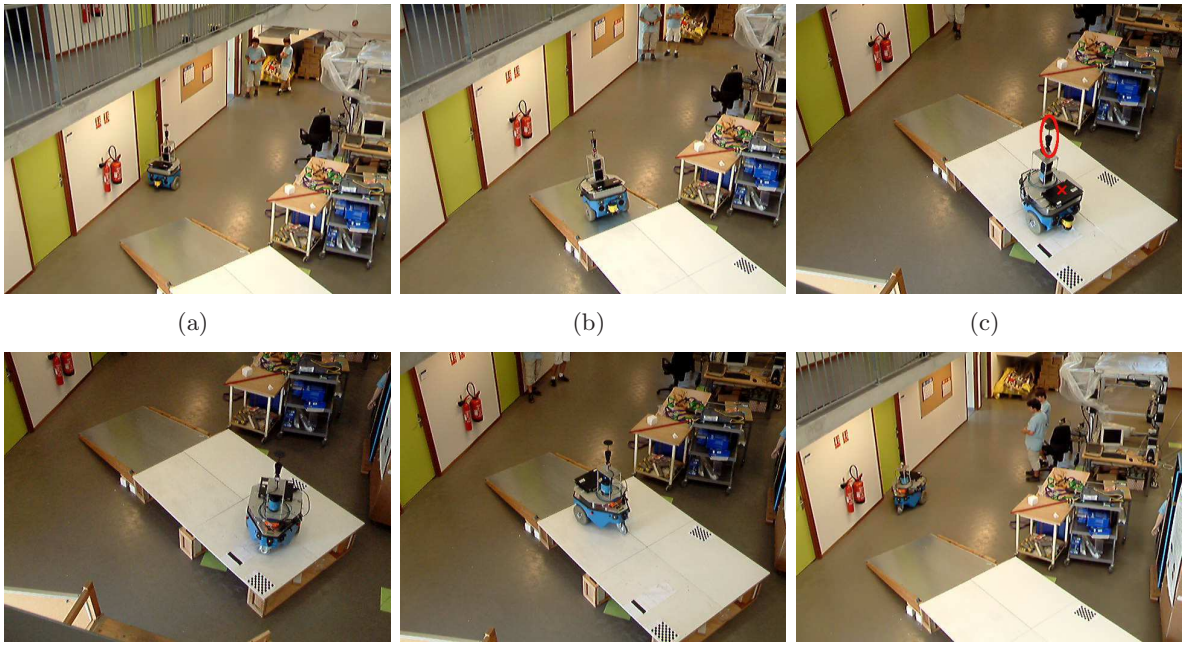


(e)



(f)

FIGURE 6.8 – Coordonnées et orientations du robot obtenues par le SLAM à 6 degrés de liberté dans le cas de la séquence plane



(a) (b) (c)

(d) (e) (f)

FIGURE 6.9 – Photos décrivant l'expérimentation avec le plan incliné.

6.4.2.2 Résultats

Filtre de Kalman étendu

Les résultats concernant le filtre de Kalman étendu sont présentés sur la figure 6.11. Même si l'on retrouve l'allure globale de la trajectoire, on constate que la solution obtenue présente des irrégularités, :

- le début demi-tour est détecté avec un peu de retard : la fonction d'évolution suppose que la vitesse de rotation est constante, on ne peut donc pas prévoir le changement brutal d'orientation en l'absence d'odométrie. Ceci se voit sur la figure 6.11a : il y a un léger retour en arrière au début du demi-tour,
- la fermeture de boucle provoque un saut dans la trajectoire.

Ainsi, les résultats de l'EKF sont globalement assez irréguliers ; leur exploitation peut donc être difficile.

SAM

Les résultats concernant l'algorithme de SAM sont quant à eux présentés sur les figure 6.12 à 6.14. Contrairement au cas de l'EKF, les résultats obtenus ici sont très lisses. On retrouve bien les différentes phases de la trajectoire, avec notamment l'évolution sur le plateau à une altitude z constante (figure 6.13). Par ailleurs, nous avons comparé les ellipses de confiances obtenues à la fin de la trajectoire

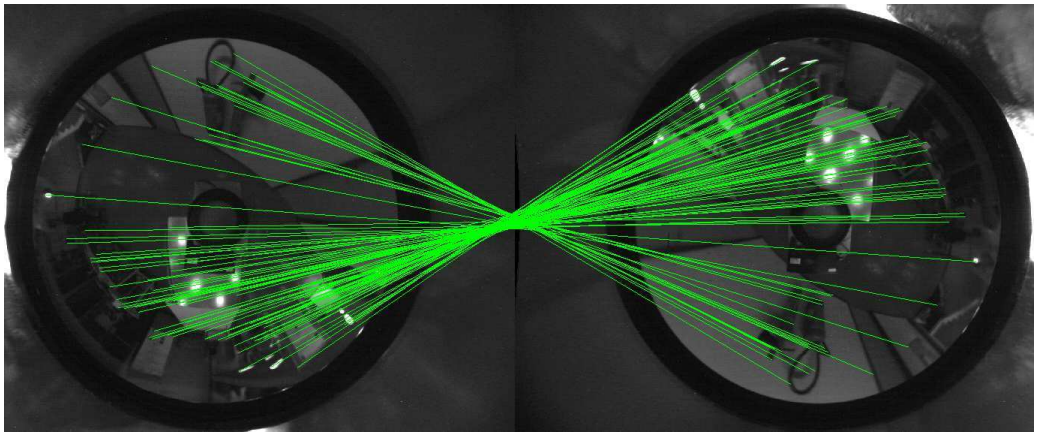


FIGURE 6.10 – Fermeture de boucle pendant la phase de descente de la rampe

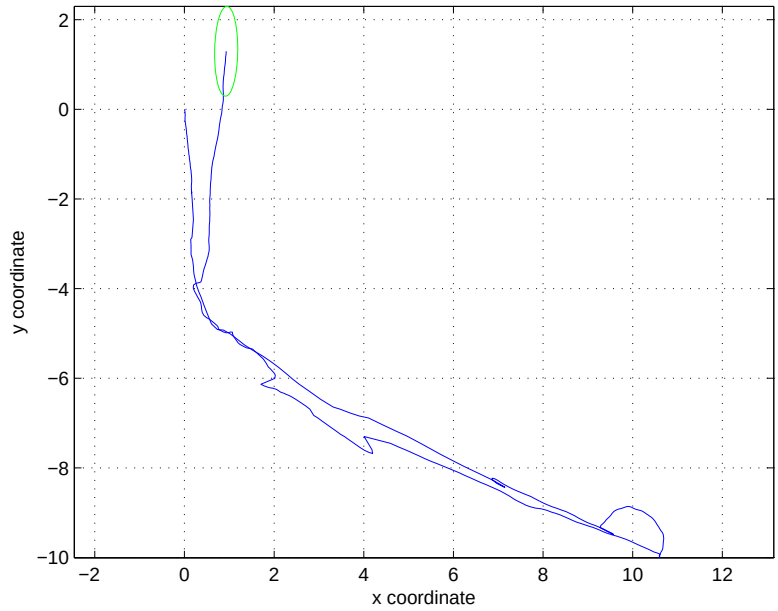
dans le cas des deux algorithmes (figures 6.11a et 6.12a). On se rend alors compte que l'ellipse associée au SAM est plus grande que celle associée à l'EKF (alors que les données utilisées par les deux algorithmes sont exactement les même). Il s'agit certainement d'une manifestation de l'inconsistance de l'EKF : les ellipses de confiances deviennent trop petites dans le cas de l'EKF.

Nous avons enfin été capables, dans le cas du SAM, de calculer précisément l'orientation de la rampe en utilisant les coordonnées relevées sur la solution au début et à la fin de la montée. Nous avons obtenu 10.1 degrés soit une erreur de 1.1 degrés (la vérité terrain étant de 11.2 degrés). Ceci confirme la qualité des résultats obtenus par l'algorithme de SAM.

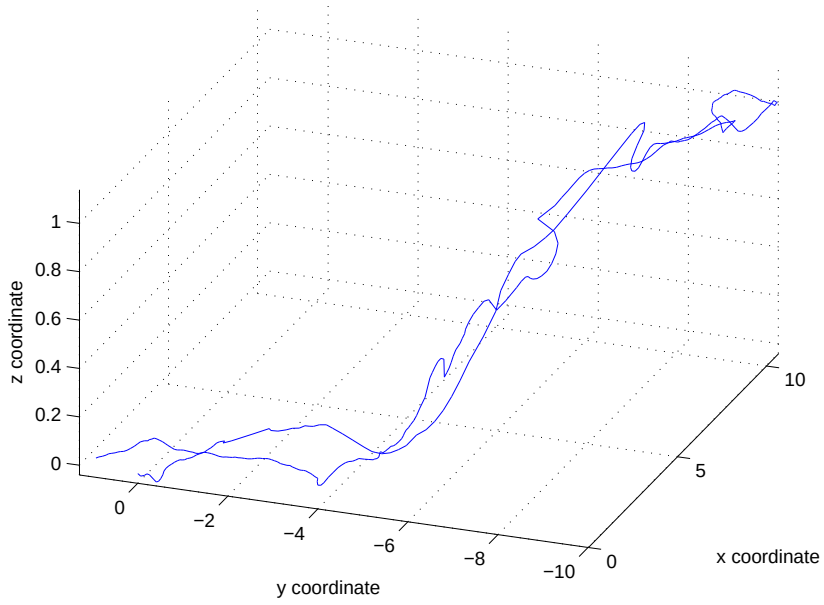
6.4.3 Conclusion quant aux résultats

Au final, l'ensemble des résultats obtenus par notre algorithme de SAM utilisant la représentation inverse de la profondeur sont très encourageants. Dans les deux expérimentations présentées, nous avons réussi à obtenir des trajectoires et des cartes tout à fait cohérentes. Nous avons en effet réussi à retrouver toutes les caractéristiques d'une trajectoire plane dans le premier cas, et nous avons retrouvé la plupart des caractéristiques de la rampe dans le second cas. Il s'agit d'excellent résultats étant donné que le seul capteur utilisé ici est la caméra.

Néanmoins, ces bons résultats nécessitent d'utiliser un état de taille très importante. Ceci a conduit à diverses instabilités et à une augmentation très conséquente des temps de calculs par rapport à l'algorithme décrit au chapitre précédent. Ainsi, nous avons dû calculer la solution sur 3 fenêtres dans le cas de la longue séquence plane. Cet algorithme n'est donc utilisable que sur le court terme.

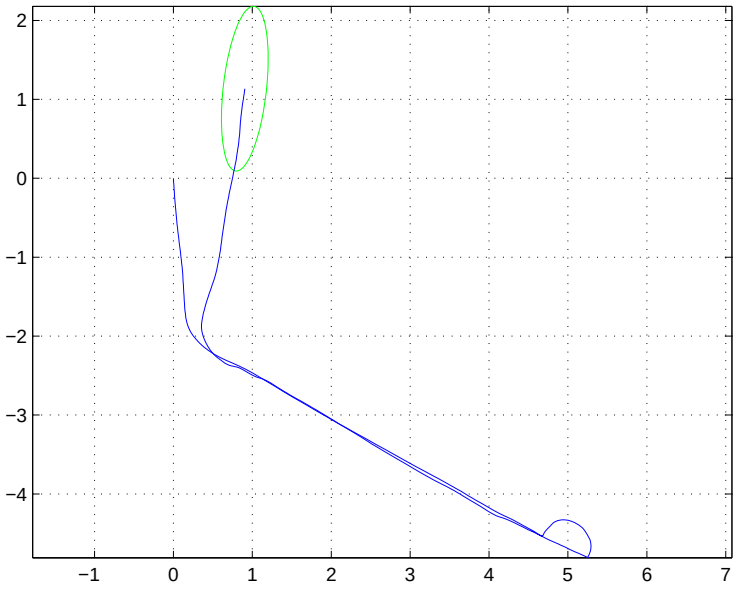


(a)

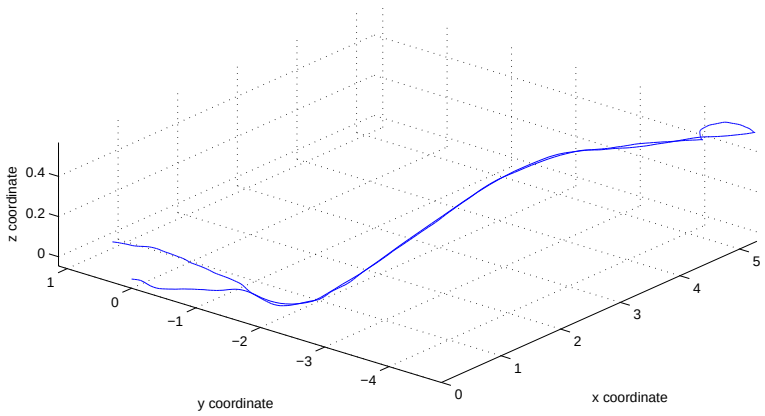


(b)

FIGURE 6.11 – Resultats du filtre de Kalman étendu sur la trajectoire à 6 degrés de liberté — (a) Projection sur le plan 2D – (b) Trajectoire 3D

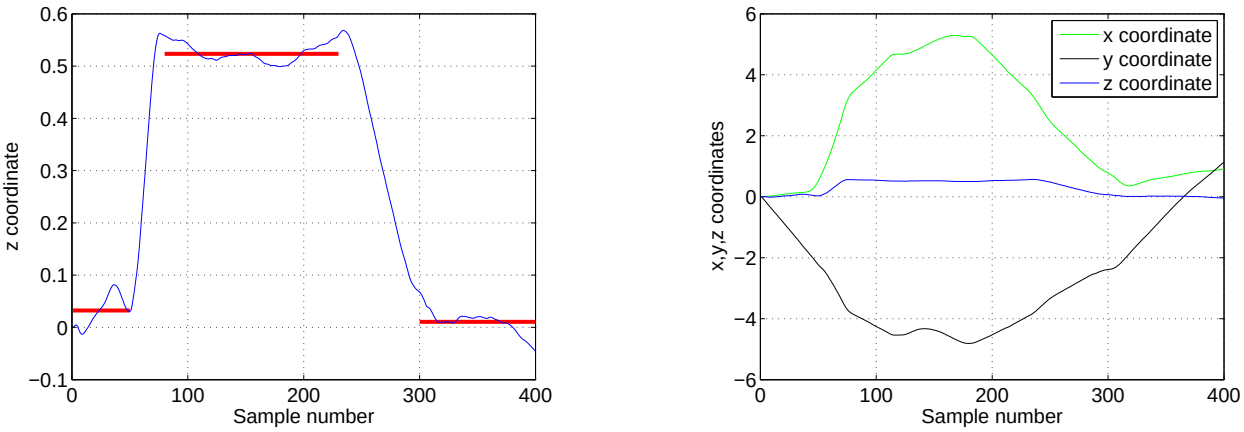


(a)



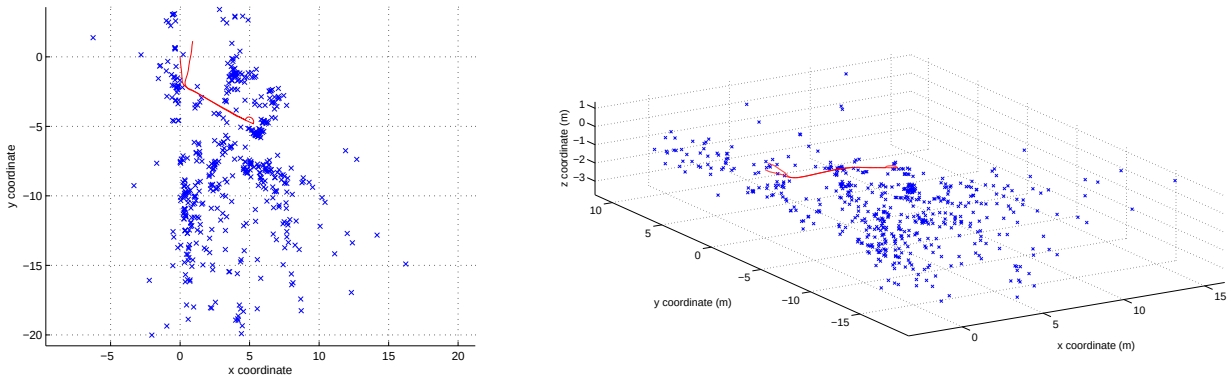
(b)

FIGURE 6.12 – Resultats du SAM sur la trajectoire à 6 degrés de liberté — (a) Projection sur le plan 2D – (b) Trajectoire 3D



(a) (b)

FIGURE 6.13 – Qualité de la coordonnée z dans le cas du SAM — (a) Valeurs de z en fonction du temps – (b) Valeurs de x, y and z en fonction du temps : on se rend bien compte de la qualité de z par rapport à l'échelle globale (x et y)



(a) (b)

FIGURE 6.14 – Carte et trajectoire obtenues par le SAM

6.5 Conclusion de la seconde partie

Nous avons proposé dans ce chapitre une méthode originale pour résoudre le problème du SLAM visuel à 6 degrés de liberté sans odométrie. Nous avons pour cela adapté au cas du SAM la représentation des amers avec l'inverse de la profondeur. Nous avons alors obtenu un algorithme permettant de **représenter les amers de façon minimale dans le filtre**, alors que dans la version initiale de l'algorithme, les amers sont sur-paramétrés. Les résultats obtenus se sont avérés très cohérents. Néanmoins, nous n'avons pas réussi à utiliser cet algorithme sur de longues trajectoires. Il paraît nécessaire de subdiviser la carte globale en plus petites cartes locales pour obtenir de bons résultats.

Par ailleurs, nous avons vu au chapitre précédent que l'algorithme de SAM aidé par l'odométrie pouvait également présenter des instabilités lorsque certaines zones ne contiennent pas suffisamment d'amers. Il paraît alors naturel de limiter l'application de cet algorithme sur des zones locales dans lesquelles on est capable de garantir que l'ensemble de la solution sera suffisamment corrélée.

Ainsi, ces deux défauts nous ont amené à définir une **stratégie locale** pour l'utilisation des algorithmes. La description des méthodes utilisées fait l'objet de la troisième partie de ce manuscrit (chapitres 7 et 8).

Troisième partie

Représentations locales et SLAM

Chapitre 7

Méthodes existantes pour la construction de représentations locales

Sommaire

Nous avons présenté dans les chapitres 5 et 6 deux solutions permettant de résoudre le problème du SLAM visuel. La première est basée sur une fusion entre les informations issues de l'odométrie et de la caméra. La seconde méthode est basée uniquement sur les données de la caméra et utilise une extension de l'état du robot. Dans les deux cas, nous avons utilisé une méthode basée sur un algorithme de lissage de la trajectoire : le SAM. Malgré les bons résultats obtenus, nous avons noté 2 limitations à nos algorithmes :

1. L'incertitude augmente au fur et à mesure que l'on s'éloigne de la position initiale. En l'absence de fermeture de boucle, la taille des ellipses de confiance n'**est pas bornée**, ce qui peut devenir problématique. En effet, l'écriture du résultat dans le repère initial du robot perd alors de son sens. Dans le cas où le résultat du SLAM doit être utilisé par un autre algorithme ou par un utilisateur humain, la carte retournée n'a pas de cohérence globale.

2. La taille du vecteur d'état augmente avec le temps. Dans le cas du SAM, l'ajout d'amer ainsi que l'augmentation de la trajectoire font grandir le vecteur d'état. Dans le cas du SAM à 3 degrés de liberté, chaque nouvelle position ajoute 3 nouvelles composantes au vecteur d'état (l'augmentation reste " assez contenue "). Dans le cas du SAM à 6 degrés de liberté, chaque nouvelle position ajoute 13 nouvelles composantes : la taille du vecteur d'état devient alors rapidement très importante. Ceci nous a posé des problèmes lors de l'obtention de certains résultats du chapitre 6 (nous avons été obligé de séparer le jeu de données en 3 cartes).

Nous proposons dans cette troisième partie du manuscrit de thèse de prendre en compte ces deux points et d'y apporter des solutions.

Dans un premier temps, nous proposons d'analyser quelques solutions issues de la littérature. Nous verrons dans la section 7.1 des solutions différentes de paramétrisation des amers. Celles-ci ne font plus appel à la représentation euclidienne dans un repère fixe. Nous verrons dans la section 7.2 des méthodes basées sur la gestion de **cartes locales** pour réduire la charge computationnelle des algorithmes. Il s'agit ici de découper astucieusement le problème global difficile à résoudre en un temps raisonnable en plusieurs problèmes locaux tractables que l'on peut traiter " presque " indépendamment. Nous concluons ce chapitre par une synthèse permettant d'introduire les grandes lignes de la solution que nous présentons au chapitre suivant.

7.1 Représentations des amers

Nous présentons dans cette section deux solutions alternatives pour la représentation des amers. Dans les deux cas, on va essayer de n'utiliser que des **propriétés locales** des mesures. La première méthode que nous proposons est l'utilisation d'une représentation robocentrée (paragraphe 7.1.1). La seconde méthode, quant à elle, est basée sur une représentation purement locale des liaisons entre les différentes poses de la trajectoire (paragraphe 7.1.2).

7.1.1 Représentation robocentrée

La méthode de représentation robocentrée consiste à exprimer l'état des amers dans le repère du robot. Dans cette représentation, l'état du robot n'est plus à estimer car il est fixé à zéro. En revanche, les amers ne sont plus statiques. L'équation d'évolution classique est donc transposée aux amers. Les principales modifications à employer pour utiliser ce type d'algorithme sont données dans [Martinez-Cantin et Castellanos, 2006; Piniés *et al.*, 2006; Castellanos *et al.*, 2007].

Ce type d'approche a plusieurs avantages :

- Les erreurs de linéarisation sont plus faibles. En effet, l'état ne contient dans ce cas que des informations relatives entre la position des amers et du robot. Ainsi, lorsqu'un nouvel amer est entré dans le filtre, l'erreur commise n'est due qu'aux erreurs de mesures. Dans le cas du SLAM classique avec une seule carte globale, l'erreur d'initialisation d'un nouvel amer prend en compte l'erreur sur la position du robot. Ainsi, les points de fonctionnements obtenus pour le filtre de Kalman sont *a priori* plus fiables avec cette représentation.
- L'application finale peut nécessiter une représentation robocentrée. Un tel exemple est proposé dans [Piniés *et al.*, 2006] pour l'application du SLAM à la recherche de victimes d'avalanches. Dans cette application, le " robot " est dans ce cas le secouriste (récupérant dans un boîtier personnel le résultat du SLAM), et les amers sont les victimes d'une avalanche. Dans ce cas, la lecture des informations par le secouriste sera largement facilitée si toutes les indications sont données relativement à son emplacement. Ainsi, le secouriste peut facilement se déplacer dans la direction des victimes. Par ailleurs, le secouriste dispose également d'une estimation de la zone de recherche pour chaque victime par le biais des ellipses d'incertitudes déduites de la matrice de variances-covariances.

Néanmoins, l'approche robocentrée a également quelques inconvénients :

- L'aspect " qualité de la linéarisation " est à moduler. En effet, même si les erreurs absolues

concernant les points de fonctionnement des amers sont plus importantes dans le cas d'une carte globale, on peut se rendre compte qu'elles sont fortement corrélées avec l'erreur sur l'estimation de la position du robot. Or, les équations de mesures utilisées font systématiquement appel à la différence entre la position des amers et la pose courante du robot. Il en est de même des matrices jacobiennes associées. Ces différences restent bien estimée même dans le cas d'une carte globale grâce aux corrélations qui existent (l'estimation des états du robot et des amers est localement entachée par la même erreur de positionnement globale). Ainsi, même si la qualité des estimations globale n'est pas idéale, l'effet sur les linéarisations doit rester contenu.

- Dans le cas de la représentation robocentrée, seuls les amers proches du robot disposent de bonnes propriétés d'estimation et de linéarisation. En effet, lorsqu'un amer est proche du robot, son estimation est naturellement bonne. Ensuite, si le robot s'éloigne de l'amer, l'estimation relative va naturellement se dégrader, rendant alors la représention locale moins fiable. Cet aspect est pris en compte dans [Martinez-Cantin et Castellanos, 2006] par l'utilisation de cartes locales : seuls les derniers amers sont pris en compte dans le filtre.
- L'approche robocentrée est délicate à mettre en œuvre dans le cas du SAM. En effet, elle impose de définir **une trajectoire pour chaque amer**. La taille de l'état serait alors bien trop importante pour pouvoir traiter rapidement les données. Ainsi, les approches robocentrées sont en général basées sur le filtre de Kalman étendu. Cette solution n'est pas idéale en raison des problèmes d'inconsistances posés par ce filtre.

En conséquence, l'approche robocentrée ne nous paraît pas idéale pour la résolution du problème du SLAM. Le fait d'utiliser un filtre de Kalman étendu peut poser problème. Par ailleurs, les gains obtenus quant à la linéarisation restent au final assez faibles, sachant que dans le cas du SAM, l'actualisation de toute la trajectoire permet d'optimiser l'intégralité des mesures (au lieu de se focaliser sur les dernières mesures vues).

Au final, nous n'avons pas approfondi cette approche. Le lecteur intéressé par une étude plus poussée des représentations robocentrées pourra se reporter aux publications fournies ainsi qu'aux références associées.

7.1.2 Représentation par graphe local ([Mei *et al.*, 2009, 2010])

La seconde représentation que nous proposons consiste à représenter la trajectoire sous forme de graphe et à **n'estimer que les transformations élémentaires entre les différentes poses** ([Mei

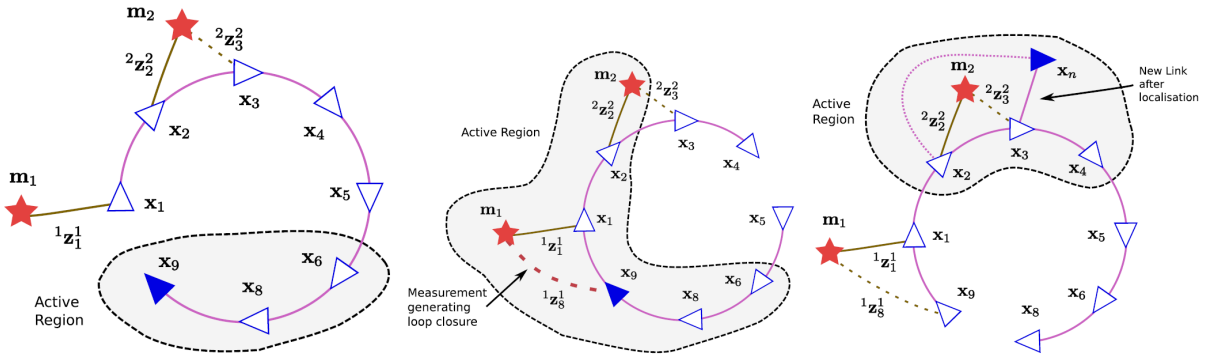


FIGURE 7.1 – Représentation par graphe ([Mei *et al.*, 2009]) — Gauche : Avant une fermeture de boucle. La région active est constituée de $\mathbf{x}_6$, $\mathbf{x}_8$ et $\mathbf{x}_9$ ($K = 2$). ${}^k\mathbf{z}_i^j$ désigne la mesure à l'instant i de l'amer (j) représenté par rapport à la position k – Centre : L'observation de $\mathbf{m}_1$ permet d'effectuer une estimation de la position relative de $\mathbf{x}_9$ et $\mathbf{x}_1$. Ceci permet une fermeture de boucle qui fait entrer $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{m}_1$ et $\mathbf{m}_2$ dans la zone active. Le lien entre les positions 4 et 5 existe toujours mais n'est pas représenté pour montrer que l'on ne force pas la composition de transformations a être égale à l'identité le long de la boucle – Droite : trajectoire après localisation quelques étapes plus tard. Alors que la robot $(\mathbf{x}_n)$ était connectée à $\mathbf{x}_2$, l'estimation montre une plus grande proximité avec $\mathbf{x}_3$. Le lien original entre $\mathbf{x}_n$ et $\mathbf{x}_2$ est donc annulé (représenté en pointillés sur la figure) pour être remplacé par le nouveau (trait plein).

et al., 2009, 2010]). Les amers, quant à eux, sont paramétrés par rapport à un repère de référence (pris en général à la première itération où ils sont observés). Cette paramétrisation permet donc de définir un graphe dans lequel seules les liaisons directes sont évaluées. Par ailleurs, les auteurs utilisent une approche locale pour la mise à jour du graphe. Seuls les nœuds nécessitant moins de K déplacements dans le graphe sont pris en compte dans l'optimisation. En général, cela signifie que l'on traite une fenêtre allant de l'instant courant (t) jusqu'à l'instant $N-t$. Néanmoins, en cas de fermeture de boucle, des liens sont ajoutés entre le nœud courant et les anciens nœuds. Dans ce cas, la zone d'optimisation s'étend autour de la fermeture de boucle entre les nœuds $N+t$ et $N-t$ (figure 7.1).

De plus, lors des fermetures de boucle, le lien le plus " éloigné " (d'un point de vue topologique) est supprimé. Ceci indique qu'il n'y a pas de procédure pour forcer la composition des déplacements le long de la boucle a être égale à l'identité. Ainsi, on n'effectue pas d'optimisation de type " relaxation " lors des fermetures de boucle. Ceci permet par exemple de traiter le problème du " robot kidnappé " (lorsque le robot est déposé dans un ascenseur ou sur un autre véhicule en mouvement).

Cette représentation permet d'obtenir d'excellents résultats de SLAM. Par ailleurs, le fait de n'estimer que des paramètres locaux dans des zones de tailles réduites permet d'effectuer une mise à jour en **temps constant**.

7.1.3 Conclusion sur les représentations alternatives

Nous avons présenté dans cette section les deux représentations alternatives les plus prometteuses pour pallier les défauts de la représentation euclidienne classique. La représentation robocentrée permet d'améliorer la consistance de l'EKF-SLAM et d'assurer que les amers vus dans la position courante ont de bonnes propriétés de linéarisation. Cette représentation ne permet malheureusement que d'utiliser le filtre de Kalman étendu, ce qui rend son intérêt limité pour notre application.

La seconde méthode, quant à elle, est complètement différente et basée sur une représentation graphique mêlant topologie et métrique locale. Elle permet de ne travailler qu'avec un nombre limité de poses, et ce de manière consistante. Malgré ses très bons résultats, cette approche n'a pas été plus étudiée en raison de sa très récente publication. Nous avions développé notre solution avant la publication de ces travaux.

7.2 Utilisation de cartes locales

Nous avons finalement choisi de résoudre notre problème d'augmentation de l'incertitude et de taille du vecteur d'état par l'utilisation de cartes locales. L'idée est de restreindre l'utilisation de l'algorithme de filtrage à de " petites " zones. Ceci permet à la fois de borner les temps de calculs et de limiter les risques d'inconsistance dus à l'apparition d'une dérive (phénomène de marche aléatoire) sur le long terme. L'idée naïve pour introduire une carte locale consiste à réinitialiser une toute nouvelle carte après un nombre d'itérations donné ; la nouvelle carte étant créée de manière complètement indépendante de la première.

Si l'approche naïve a l'avantage de la simplicité, elle ne permet pas d'assurer une cohésion entre les différentes cartes. Pour cela, il est nécessaire d'employer des algorithmes plus élaborés. La plupart des approches de la littérature effectuent des approximations, de sorte que la solution finale n'est pas optimale en termes d'utilisation de l'information disponible ([Leonard et Feder, 2001; Tardos *et al.*, 2002; Folkesson *et al.*, 2005; Estrada *et al.*, 2005]). Nous proposons dans cette section de présenter trois solutions pertinentes utilisant la notion de carte locale sans qu'il y ait de perte de performances.

7.2.1 *Atlas SLAM* ([Bosse *et al.*, 2003, 2004])

L'algorithme *Atlas SLAM* ([Bosse *et al.*, 2003, 2004]) utilise une approche topologique pour la gestion de cartes locales. Dans cette approche, les cartes locales sont des cartes métriques euclidiennes classiques de **taille bornée** (afin de borner les temps de calculs). Pour les calculs de SLAM locaux,

les auteurs indiquent que n'importe quelle méthode de SLAM peut être utilisée, du moment qu'elle permet de donner une estimation consistante. Dans les papiers cités, les auteurs insistent surtout sur la gestion des nœuds du graphe topologique.

Ainsi, un nœud dans un graphe est associé à chaque carte. Chaque arête du graphe définit la distance entre les origines des repères associés en termes de probabilité : il y a donc une notion d'incertitude associée à ces arêtes. A chaque instant, l'algorithme teste si la carte courante est compatible avec un nœud " proche ". Pour cela, on effectue une projection des nœuds considérés dans la carte courante. On pourra noter que cette projection n'est pas forcément unique. La figure 7.2 montre que dans le cas où il y a des cycles, il existe plusieurs façons de parcourir le graphe pour effectuer la projection. Pour rendre cette projection unique, le graphe topologique intial est transformé en arbre grâce à la projection de Dijkstra (figure 7.3). Seul le plus court chemin est conservé dans l'arbre final (la métrique choisie est basée sur l'incertitude de la projection, mesurée par le déterminant de la matrice de variances-covariances associée). Ainsi, cette projection permet d'avoir une idée des candidats potentiels pour une fermeture de boucle. Des méthodes de mise en correspondance et d'alignement sont enfin utilisées pour définir la meilleure hypothèse (qui sera la sortie finale de l'algorithme) et élaguer les hypothèses les moins probables. Si toutes les hypothèses sont considérées comme mauvaises, un nouveau nœud est ajouté dans le graphe.

Lorsqu'une nouvelle carte est créée, les opérations suivantes sont effectuées :

- L'origine de la nouvelle carte est définie par la position courante du robot,
- La position du robot est réinitialisée à zéro,
- L'incertitude de la transformation entre la nouvelle carte et la précédente est donnée par l'incertitude de la position courante juste avant de changer de carte (il s'agit de l'incertitude associée à la nouvelle arête dans le graphe),
- La matrice de variances-covariances du robot est réinitialisée à zéro.

Le processus de l'algorithme permet également de réajuster les liens du graphe lorsque d'anciens nœuds sont réutilisés (fermeture de boucle). Une carte globale peut également être déduite du graphe. Il s'agit en fait d'un ajustement des arêtes dans le graphe qui peut être calculé grâce à la projection de Dijkstra en prenant le nœud 0 comme racine. Lorsqu'il y a des fermetures de boucle (et donc plusieurs chemins possibles pour certains nœuds), une minimisation des erreurs de projection est effectuée (cette étape peut être vue comme une opération de relaxation du graphe). Cette dernière étape est une étape de post-traitements et n'est pas réalisée immédiatement dans le cas d'application en temps réel.

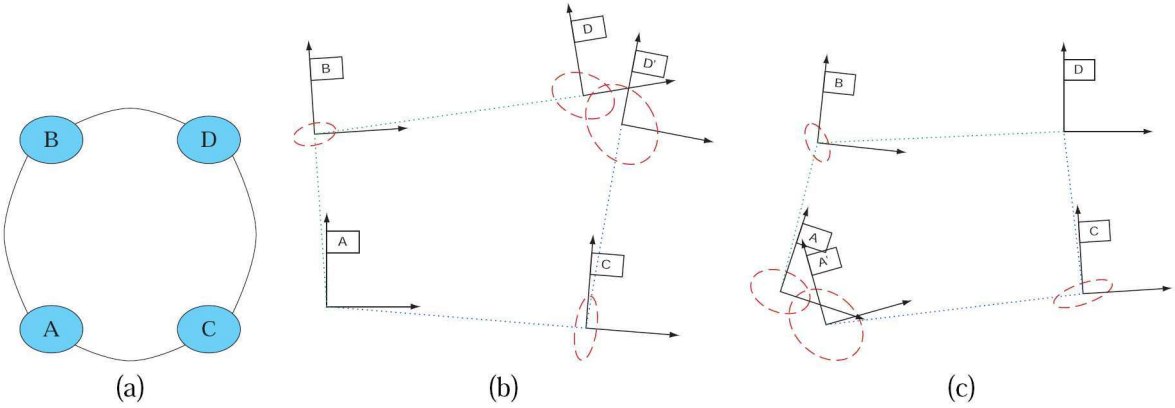


FIGURE 7.2 – Projection dans le graphe ([Bosse *et al.*, 2004]) — (a) Graphe topologique initial comportant 4 nœuds. On cherche à savoir quel chemin utiliser pour projeter le nœud D dans le repère du nœud A et inversement. On peut soit passer par B, soit par C – (b) Résultat obtenu lorsque le nœud A sert de référence. La projection D (utilisant le chemin ABD) est de meilleure qualité que la projection D' (utilisant le chemin ACD) – (c) Résultat obtenu lorsque le nœud D sert de référence. Là aussi, il est plus avantageux de composer les transformations en passant par B.

Finalement, on remarquera que cet algorithme permet une gestion intéressante des différentes sous-cartes à travers un graphe topologique. Néanmoins, on remarquera que la gestion interne des cartes locales est réalisée de manière très classique : l'état est remis à zéro et on redémarre sans information *a priori*.

7.2.2 *Tectonic SAM* ([Ni *et al.*, 2007])

L'algorithme de *Tectonic SAM* ([Ni *et al.*, 2007]) est une variante du SAM classique consistant à résoudre le problème du SAM par une approche *diviser pour régner*. Cette approche est basée sur le fait que l'une des étapes les plus couteuses en temps de calculs est l'évaluation de l'ensemble des matrices jacobiennes associées aux linéarisations. Par ailleurs, cette étape est d'autant plus longue que **l'intégralité de la trajectoire ainsi que l'ensemble des amers sont remis à jour** : il faut donc tout recalculer.

L'idée de cet algorithme est de diviser la carte totale en un certain nombre de sous-cartes dans lesquelles on calcule une solution locale. Une fois que chaque sous-carte a convergé, il devient inutile de recalculer les matrices jacobiennes associées aux linéarisations. Les éléments communs à plusieurs cartes sont traités dans un processus à part et constituent le " séparateur " (figure 7.4a). Une explication empirique simple de cet algorithme est la cartographie de bureaux :

– Chaque bureau peut être cartographié indépendamment des autres. Il s'agit des sous-cartes,

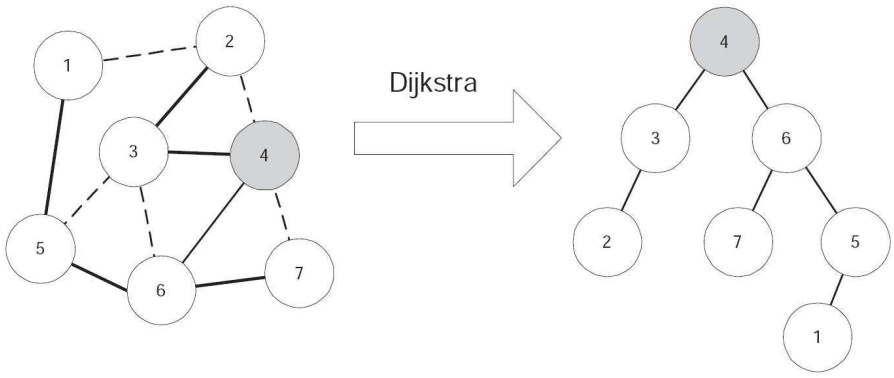


FIGURE 7.3 – Transformation du graphe en arbre avec la projection de Dijkstra ([Bosse *et al.*, 2004]) — Ici, le nœud 4 est le nœud de base (la racine de l'arbre). A gauche, on a représenté en pointillés les arêtes qui n'ont pas été retenues dans l'arbre. On pourra remarquer qu'il n'existe pas de lien direct entre les nœud 4 et 7. Ceci signifie que la transformation pour passer directement du nœud 4 au nœud 7 est moins précise que la composition de la transformation entre les nœuds 4 et 6 suivie de la transformation entre les nœuds 6 et 7.

– Les couloirs entre les bureaux permettent de voir une partie de plusieurs bureaux simultanément : il s'agit des séparateurs.

Néanmoins, lors de la mise à jour totale, l'ensemble des positions est utilisé, mais seuls les états liés aux séparateurs sont mis à jour. Le problème de ce type d'approche est que si l'estimation des séparateurs évolue fortement, la position globale des points dans les sous-cartes locales varie fortement également. Dans ce cas, ne pas mettre à jour les linéarisations des sous-cartes ne serait pas correct et pourrait introduire des problèmes d'inconsistances. Pour contourner ce problème, on introduit de nouveaux nœuds dans la représentation (b_1 et b_2 sur la figure 7.4b) qui codent la position globale de la carte. Tous les éléments (trajectoire du robot et amers) qui ne font pas partie des séparateurs sont représentés par rapport à ces nœuds de base. Ainsi, les valeurs qui seront figées ne sont plus des valeurs globales, mais des valeurs locales qui auront encore tout leur sens lors de l'optimisation finale. Les nœuds b_i sont alors inclus dans les séparateurs. En conséquence, si le mouvement relatif entre deux sous-cartes est incertain, cela n'influencera que le recalage globale. Les propriétés locales resteront conservées.

Cette approche par cartes locales est une évolution astucieuse de l'algorithme de $\sqrt{\text{SAM}}$ qui consistait à ordonner les colonnes de la matrice d'information de sorte à pouvoir effectuer une décomposition de Cholesky efficace ([Dellaert et Kaess, 2006]). Un exemple du résultat obtenu par le *Tectonic SAM* est donné sur la figure 7.5. Néanmoins, même si cet algorithme permet d'effectuer des calculs efficaces pour le SAM, il ne permet pas la construction de " véritables " cartes locales. En effet, l'évaluation

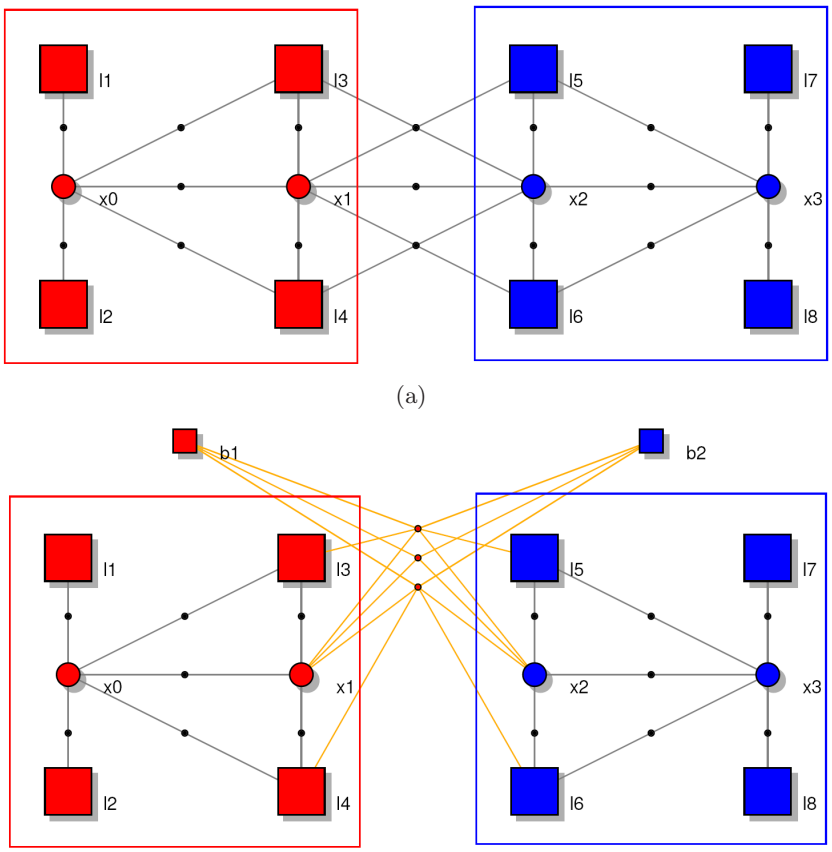


FIGURE 7.4 – Partionnement du graphe en cartes locales ([Ni *et al.*, 2007]) — (a) Graphe initial : x_0, x_1, l_1, l_2, l_3 et l_4 constituent les variables de la première carte locale, mais seuls x_0, x_1, l_1 et l_2 ne sont pas partagé avec la seconde carte (bleue). x_1, l_3 et l_4 constituent les séparateurs. La seconde carte locale est en bleue, et dispose également de variables internes et de variables frontières – (b) Modification du graphe pour y intégrer les nœuds de base b_1 et b_2. Ainsi, toutes les variables de la carte rouge sont exprimées dans le repère lié à b_1 alors que les variables de la carte bleue sont exprimées par rapport au repère lié à b_2.

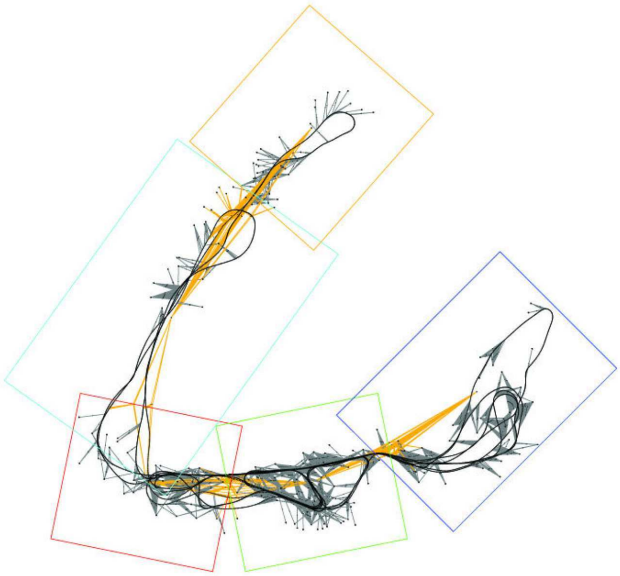


FIGURE 7.5 – Résultats du *Tectonic SAM* sur le jeu de données du Victoria Park ([Ni *et al.*, 2007]) — Chaque sous-carte est calculée localement. L'ajustement global ne concerne que les paramètres globaux associés aux sous-cartes (les rectangles sur la figure).

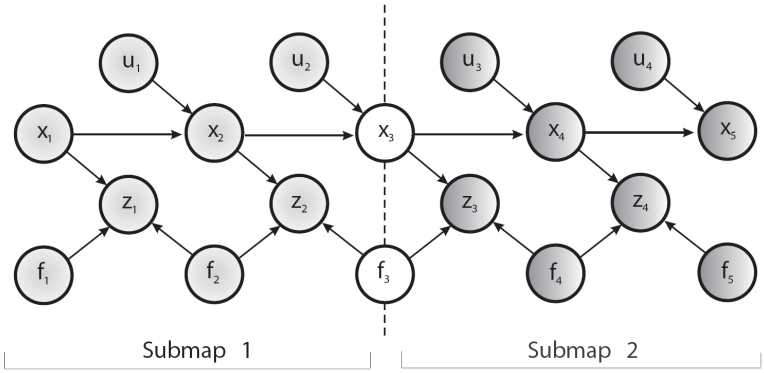
des cartes locales est effectué **sans prendre en compte les mesures appartenant aux nœuds séparateurs**. Ainsi, le résultat de l'algorithme n'aura de l'intérêt qu'après l'ajustement global. Avant, il ne s'agit que de sous-cartes indépendantes privées d'une partie de leurs mesures (celles partagées avec d'autres sous-cartes), ce qui représente l'aspect le plus basique des cartes locales. Par ailleurs, la convergence des sous-cartes ne sera satisfaisante que si le nombre d'amers à l'intérieur de celle-ci (ie. hors frontières) est suffisant pour contraindre la solution.

Enfin, la nécessité d'effectuer un recalage global fait que l'algorithme ne pourra pas être utilisé constamment en temps réel. Même si les temps de calculs sont largement améliorés, les temps de calculs liés à l'ajustement global continueront d'augmenter avec la taille de la carte globale. Si cette dernière étape n'est effectuée qu'en post-traitement car la charge de calcul est trop lourde, la " pauvreté " relative des sous-cartes créées rend l'algorithme de *Tectonic SAM* nettement moins intéressant.

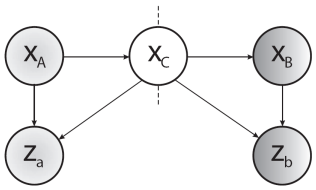
7.2.3 Cartes conditionnellement indépendantes ([Piniés et Tardós, 2007, 2008])

La dernière méthode que nous présentons consiste à définir un ensemble de sous-cartes locales indépendantes les unes des autres mais pouvant néanmoins partager de l'information [Piniés et Tardós, 2007, 2008].

Dans cette approche, on va chercher à découper l'environnement ainsi que la trajectoire du robot



(a)



(b)

FIGURE 7.6 – Exemple de réseau bayésien que l'on veut séparer en deux cartes locales ([Piniés et Tardós, 2008]) — (a) Dans ce réseau, l'amer $\mathbf{f}_3$ est partagé par les deux cartes, ce qui rend la création des sous-cartes non triviale – (b) Simplification du réseau où on a regroupé l'ensemble des états (poses du robot et amers) de la première sous-carte dans $\mathbf{x}_A$, l'ensemble des mesures associées (incluant les vecteur de commande) dans $\mathbf{z}_a$. Les états et mesures de la seconde sous-carte sont regroupés dans $\mathbf{x}_B$ et $\mathbf{z}_b$. Enfin, l'état $\mathbf{x}_3$ et $\mathbf{f}_3$ sont regroupés dans $\mathbf{x}_C$.

en sous-cartes. Cette approche se différencie des autres car dès que l'on commence une nouvelle carte, on va arrêter de traiter les amers et la trajectoire de la précédente carte (dans les deux autres approches présentées, l'aspect global était toujours pris en compte dans le graphe), sauf dans le cas d'une fermeture de boucle. Si l'on veut pouvoir effectuer cette opération de façon consistante, il est nécessaire que les sous-cartes soient conditionnellement indépendantes. **Autrement dit, elles ne doivent pas partager d'information**.[1] Ceci est naturellement le cas lorsque les sous-cartes ne partagent aucun amer. Mais ce cas particulier n'est en général pas vérifié (on retrouve plutôt un réseau bayésien comme sur la figure 7.6a). Pour garantir l'indépendance, la méthode la plus simple consiste à réinitialiser l'ensemble des amers qui étaient vus dans la carte précédente. Dans le cas du SLAM monoculaire, cela implique la perte de l'estimation courante du facteur d'échelle. Par ailleurs, on doit également réinitialiser certains paramètres comme les vitesses ou encore les estimations de biais dans le cas des filtres utilisant une centrale inertielle. C'est pour cette raison que les cartes locales traitées indépendemment dans l'algorithme de *Tectonic SAM* exculent les amers appartenant aux séparateurs (ie. les amers visibles dans plusieurs cartes).

Dans [Piniés et Tardós, 2007, 2008], les auteurs contournent ce problème en utilisant des propriétés particulières du réseau bayésien associé au SLAM. Considérons désormais la simplification du réseau bayésien de la figure 7.6a en regroupant ensembles les états des sous-cartes et des nœuds frontières

1. Sans cette précaution, toute tentative de fusion pour retrouver la carte globale sera inconsistante.

(figure 7.6b). L'état est alors limité à $\mathbf{x}_A$, $\mathbf{x}_B$ et $\mathbf{x}_C$. On montre alors que conditionnellement à l'état $\mathbf{x}_C$, les états $\mathbf{x}_A$ et $\mathbf{x}_B$ sont indépendants : les cartes sont donc conditionnellement indépendantes. Cette propriété est issue de la *d-séparation* des deux sous-cartes conditonnellement à $\mathbf{x}_C$ dans le graphe de la figure 7.6b (voir annexe A page 321 pour plus de détails sur l'interprétation des réseaux bayésiens). On a alors :

$$p\left(\mathbf{x}_A|\mathbf{x}_B,\mathbf{x}_C,\mathbf{z}_a,\mathbf{z}_b\right) = p\left(\mathbf{x}_A|\mathbf{x}_C,\mathbf{z}_a\right) \tag{7.1}$$

Cette propriété permet de ré-écrire la densité de probabilité associée à la carte complète :

$$\begin{array}{rcl} p\left(\mathbf{x}_A,\mathbf{x}_B,\mathbf{x}_C|\mathbf{z}_a,\mathbf{z}_c\right) & = & p\left(\mathbf{x}_A|\mathbf{x}_B,\mathbf{x}_C,\mathbf{z}_a,\mathbf{z}_c\right) p\left(\mathbf{x}_B,\mathbf{x}_C|\mathbf{z}_a,\mathbf{z}_c\right) \\ & = & p\left(\mathbf{x}_A|\mathbf{x}_C,\mathbf{z}_a\right) p\left(\mathbf{x}_B,\mathbf{x}_C|\mathbf{z}_a,\mathbf{z}_c\right) \end{array} \tag{7.2}$$

où on a utilisé la règle de Bayes dans la première étape. L'application de cette règle permet également d'écrire :

$$p\left(\mathbf{x}_A,\mathbf{x}_C|\mathbf{z}_A\right) = p\left(\mathbf{x}_A|\mathbf{x}_C,\mathbf{z}_A\right) p\left(\mathbf{x}_C|\mathbf{z}_A\right) \tag{7.3}$$

En combinant les résultats des équations 7.2 et 7.3, on obtient :

$$p\left(\mathbf{x}_A,\mathbf{x}_B,\mathbf{x}_C|\mathbf{z}_a,\mathbf{z}_c\right) = \frac{p\left(\mathbf{x}_A,\mathbf{x}_C|\mathbf{z}_A\right)}{p\left(\mathbf{x}_C|\mathbf{z}_A\right)} p\left(\mathbf{x}_B,\mathbf{x}_C|\mathbf{z}_a,\mathbf{z}_c\right) \tag{7.4}$$

L'équation 7.4 nous permet donc de retrouver facilement la densité de probabilité de la carte complète lorsque l'on calcule indépendemment deux sous-cartes. Les 3 facteurs nécessaires sont en effet connus :

- $p\left(\mathbf{x}_A,\mathbf{x}_C|\mathbf{z}_A\right)$: il s'agit de la densité de probabilité obtenue **juste à la fin de la première carte locale**,
- $p\left(\mathbf{x}_C|\mathbf{z}_A\right)$: il s'agit de la densité de probabilité marginale obtenue **juste à la fin de la première carte locale**. Elle peut être déduite du terme précédent après intégration, [2]
- $p\left(\mathbf{x}_B,\mathbf{x}_C|\mathbf{z}_a,\mathbf{z}_c\right)$: il s'agit du résultat du SLAM limité à la seconde carte locale. Ce terme s'obtient en utilisant l'algorithme de SLAM classique sur les données $\mathbf{z}_c$, avec l'utilisation de la densité *a priori* $p\left(\mathbf{x}_C|\mathbf{z}_A\right)$ pour la variables $\mathbf{x}_C$. Autrement dit, on applique l'algorithme de SLAM sur la seconde carte locale après avoir initialisé $\mathbf{x}_C$ avec le résulat marginal de la précédente carte.

Il devient alors facile, dans le cas gaussien, de remonter aux paramètres de la densité de probabilité de la carte globale (ie. l'espérance mathématique et la matrice de variances-covariances). Ceux-ci s'obtiennent par identification. Le détail des calculs est donné dans [Piniés et Tardós, 2007, 2008]. Ainsi, lorsque l'on a créé plusieurs cartes locales, on peut revenir à la carte globale par rétro-propagation des

2. Dans le cas gaussien, il suffit d'extraire les blocs associés à $\mathbf{x}_C$ dans l'espérance et la matrice de variances-covariances associée à $p\left(\mathbf{x}_A,\mathbf{x}_C|\mathbf{z}_A\right)$.

différentes cartes locales. Les auteurs montrent également comment traiter le cas de la fermeture de boucle.

L'approche présentée ici utilise directement les coordonnées dans le repère global. Ainsi, la valeur utilisée pour initialiser $\mathbf{x}_C$ est associée à une matrice de variances-covariances non nulle (y compris pour la position de départ du robot). Les sous-cartes obtenues risquent donc de montrer des problèmes d'inconsistance due à l'accumulation des incertitudes. Pour éviter ce phénomène, les auteurs de [Piniés et Tardós, 2007, 2008] proposent une façon plus consistante de traiter ce problème en introduisant la notion de repère local. **Les états du nœud $\mathbf{x}_C$ sont alors transformés dans le repère lié à la dernière position du robot**. En reprenant l'exemple de la figure 7.6, ceci implique de :

1. Transformer l'état $\mathbf{x}_3$ en un état $\mathbf{x}'_3$ dont la valeur est égale à zéro de façon certaine (matrice de variances-covariances nulle),

2. Transformer $\mathbf{f}_3$ en $\mathbf{f}'_3$ via :

$$\mathbf{f}'_3 = \ominus \mathbf{x}_3 \oplus f_3 \tag{7.5}$$

où les opérateurs $\ominus$ et $\oplus$ désignent les opérations permettant de transformer les coordonnées de $\mathbf{f}_3$ (initialement dans le repère global) dans le repère lié à $\mathbf{x}_3$ (incluant le mouvement de translation ainsi que le mouvement de rotation du repère). [3]

Ceci permet de transformer le réseau bayésien original de sorte à ce que la partie commune aux deux sous-cartes ne soit consitutée que de $\mathbf{f}'_3$. Il n'est pas nécessaire d'y introduire le nouvel $\mathbf{x}'_3$ car il est naturellement fixé à zéro, et ce **de manière indépendante à l'ensemble des mesures effectuées dans la première carte**. Ceci nous amène donc au réseau bayésien présenté sur la figure 7.7. On peut alors utiliser les principes énoncés précédemment pour fusionner les cartes locales et remonter à la carte globale.

Au final, les auteurs de [Piniés et Tardós, 2007, 2008] expérimentent leurs algorithmes avec un filtre de Kalman étendu classique dans le cas du SLAM monoculaire. Ils montrent que la variante avec représentation locale donne de bien meilleurs résultats que la représentation globale classique. Ceci était assez prévisible car :

– La " variante globale " de l'algorithme revient finalement à tronquer l'état à chaque création d'une nouvelle sous-carte. Ainsi, on s'attend à retrouver le même résultat qu'avec le filtre de

3. Dans le cas d'un mouvement de translation pure, ces opérateurs auraient simplement été une addition et une soustraction.

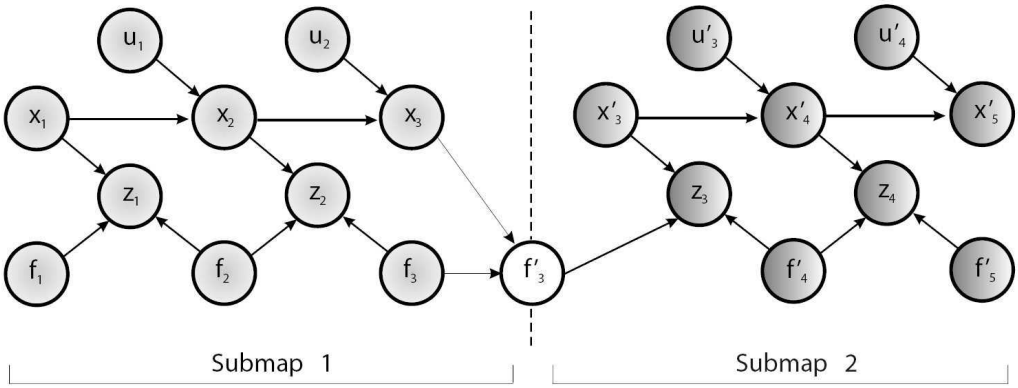


FIGURE 7.7 – Transformation du réseau bayésien pour la création de sous-cartes avec repère local ([Piniés et Tardós, 2008])

Kalman étendu classique, dont les estimations des incertitudes arrêtent d'augmenter sur le long terme,

– La " variante locale ", en remettant à zéro la matrice de variances-covariances de la position du robot, permet de maintenir des valeurs de variances suffisamment faibles pour rester dans une plage d'utilisation où les inconsistances du filtre de Kalman étendu sont beaucoup moins visibles.

7.2.4 Conclusion quant à l'utilisation de cartes locales

Au final, nous avons présenté dans cette section trois méthodes originales permettant de réaliser des cartes locales. Les deux premières méthodes (*Atlas SLAM* et *Tectonic SAM*) utilisent plutôt une approche topologique pour résoudre le problème. Dans ces deux méthodes, les sous-cartes créées ne sont pas optimales, dans le sens où elles n'utilisent pas les informations partagées avec les autres cartes. Ainsi, la méthode de *Tectonic SAM* ne sera efficace que si les mesures " purement locales " sont suffisamment contraignantes pour faire converger la sous-carte vers le minimum global. Il s'agit d'une hypothèse finalement assez forte impliquant que l'ensemble des sous-cartes n'aient pas trop de zones de recouvrement. Néanmoins, cette dernière méthode permet au final d'obtenir la solution complète une fois le processus global achevé.

La dernière méthode proposée est différente car elle permet aux sous-cartes de partager de l'information de manière consistante. Néanmoins, l'approche finale retenue avec un filtre de Kalman étendu ne nous paraît pas idéale. Nous verrons au chapitre suivant que la méthode que nous avons développé se rapproche beaucoup de la méthode locale proposée dans [Piniés et Tardós, 2007, 2008].

7.3 Conclusion

Au final, nous avons présenté dans ce chapitre un panel assez large des méthodes de représentation locales existantes, donnant ainsi un juste aperçu des principales catégories de méthodes permettant de modifier la représentation du SLAM. Certaines méthodes, comme le SLAM robocentré modifient directement le vecteur d'état du système afin de le rendre plus local. La méthode de représentation par graphe local où seules certaines arêtes sont évaluées nous paraît être la méthode alternative de SLAM la plus prometteuse actuellement. Néanmoins, sa relative jeunesse et l'avancée de nos travaux ont fait que nous n'avons pas approfondi plus en détail ce type d'approche.

Nous avons finalement préféré la méthode plus classique consistant à diviser le problème en cartes locales. Nous avons remarqué que la plupart des méthodes traitant de cartes locales isolent les nœuds communs à plusieurs cartes. Il en résulte que le traitement " purement local " ne prend pas en compte toutes les mesures effectuées au sein des cartes locales. Si les mesures propres à une carte ne sont pas assez contraignantes, cela peut poser des soucis de convergence, même si l'algorithme final prend en compte les états frontières dans un processus séparé. Ce défaut est par exemple commun aux méthodes d'*Atlas SLAM* et de *Tectonic SAM*. La dernière méthode de SLAM proposée est basée sur le partage d'information au sein de cartes conditionnellement indépendantes. Cette approche nous semble la plus aboutie et est très proche de l'algorithme que nous proposons au chapitre suivant. [4]

Enfin, nous pouvons remarquer que toutes les approches basées sur la constructions de cartes locales ont pour but principal de borner le nombre d'amers à prendre en compte par l'algorithme de filtrage principal (SAM ou EKF). Ceci a deux conséquences :

1. Améliorer voire borner les temps de calcul (qui augmentent normalement avec le carré du nombre d'amers),
2. Améliorer la consistance en maintenant des valeurs de variances relativement faibles (ce qui est surtout valable pour l'EKF).

La plupart des approches proposées ne proposent pas de solution fine pour choisir quand commencer une nouvelle carte locale. En général, la solution est de fixer une certaine taille d'état à ne pas dépasser en fonction de la puissance de calcul disponible. La taille maximale de l'état est donc fixée à l'avance. Ceci ne nous semble pas idéal, notamment pour prendre en compte les effets dus à la consistance. Nous étudierons plus particulièrement ce problème au chapitre suivant.

4. L'algorithme proposé au chapitre suivant a été développé en parallèle des travaux présentés dans [Piniés et Tardós, 2007, 2008], alors que nous n'avions pas connaissance de ces travaux.

Chapitre 8

Algorithme pour la création et la gestion de cartes locales : application au cas du SAM

Sommaire

Nous avons présenté au chapitre précédent plusieurs méthodes afin de contourner les problèmes liés à la taille de l'état et aux grands déplacements dans les algorithmes de SLAM. Parmi l'ensemble des familles de méthodes étudiées, nous avons choisi d'utiliser une méthode basée sur l'utilisation de **cartes locales**. Ce chapitre est consacré à l'étude en détail de la solution choisie. Les travaux qui y sont décrits ont été présentés lors du Workshop OMNIVIS'09 ([Joly et Rives, 2009]).

Dans un premier temps, nous nous intéressons à définir une stratégie afin de piloter le changement de sous-carte (section 8.1). En effet, dans toutes les approches présentées au chapitre précédent, le choix de commencer une nouvelle carte est régi par un critère absolu sur le nombre maximal de points autorisés dans la carte courante. Ceci est légitime dans l'optique de maintenir des temps de calculs bornés. Néanmoins, nous pensons que ce critère n'est pas le seul à prendre en compte. Nous étudierons une méthode basée sur l'étude du résultat de la carte courante pour savoir s'il peut être intéressant de changer de carte, et ce du point de vue de la consistance. Nous explicitons ensuite notre algorithme utilisé pour créer de nouvelles cartes (section 8.2). Nous verrons que celui-ci est basé sur le fait qu'il est possible de faire en sorte que chaque carte locale courante puisse prendre en compte l'intégralité des informations acquises depuis l'instant initial. La troisième section, quant à elle, est consacrée à l'exposition des résultats. Nous jugerons sur les deux séquences introduites au chapitre 5 la qualité des deux algorithmes présentés dans le cas du SLAM à 3 degrés de liberté aidé par l'odométrie. Enfin, nous effectuons une synthèse globale quant aux cartes locales dans la section 8.4.

8.1 Critère pour le changement de carte locale

8.1.1 Problématique et solutions existantes

Dans la bibliographie concernant l'utilisation de cartes locales, la partie concernant le choix du changement de carte locale n'est que rarement abordé. Cet aspect est généralement caché derrière l'objectif de borner les temps de calculs. Néanmoins, toutes les publications s'accordent à dire qu'utiliser des cartes locales permet d'améliorer la consistance de l'algorithme de filtrage, ce qui est notamment très visible dans le cas de l'EKF-SLAM ([Piniés et Tardós, 2008]). Il existe certaines approches pour tenter de trouver le nombre d'amer optimum par carte local ([Paz et Neira, 2006]). Néanmoins, les auteurs de [Paz et Neira, 2006] ne prennent en compte que le nombre d'opérations réalisées par les étapes de l'EKF, afin de rendre l'algorithme utilisable en temps réel. Ainsi, la notion de consistance lors du changement de carte locale est inexistante.

Il est pourtant clair qu'en l'absence de fermeture de boucle, les incertitudes associées à la position

du robot et aux amers suivent un phénomène de marche aléatoire : les incertitudes associées aux estimations augmentent de façon non bornée. Nous pensons qu'il est nécessaire de garantir que les incertitudes associées aux amers ainsi qu'à la position du robot reste " contenue ". La notion de " contenue " reste à définir. En effet, les incertitudes peuvent augmenter plus ou moins rapidement en fonction du nombre d'amers considérés, de la géométrie des lieux, etc. Ainsi, définir un critère absolu en termes d'incertitude ne paraît pas la solution idéale. Ceci est d'autant plus vrai que le GraphSLAM est moins sensible aux erreurs globales de linéarisation que l'EKF-SLAM. [1] Par ailleurs, dans le cas du SLAM visuel pur, la trajectoire n'est connue qu'à un facteur d'échelle près. Ainsi, définir par exemple une taille absolue sur les ellipses de confiances associées aux amers ne serait pas cohérent.

Nous chercherons dans la suite à définir un critère relatif facile à paramétrer et dont " l'interprétation phyisique " pourra se faire facilement. Dans un premier temps, nous étudions deux exemples permettant de mettre en évidence la nécessité de créer une nouvelle carte indépendamment des capacités en termes de calculs de la platerforme lançant le SLAM (paragraphe 8.1.2). Nous détaillons enfin le critère utilisé dans le paragraphe 8.1.3.

8.1.2 Etude d'exemples

Un défaut que nous avions constaté au chapitre 5 lors de l'utilisation de la méthode de SAM était sa divergence lorsque le nombre de mesures devient trop important et que certains liens du réseau bayésien sont trop faibles (ie. zones avec peu de mesures). Cet aspect avait été illustré sur la figure 5.10, rappelée ici en 8.1. Intuitivement, cet exemple montre que la fin de la trajectoire n'a plus rien à voir avec le début. En revanche, les propriétés locales étaient conservées : on avait constaté un mouvement rigide entre deux morceaux de la carte. Ainsi, cet exemple nous montre qu'il aurait été judicieux de couper la carte globale en au moins deux sous-cartes. Ceci aurait permis d'avoir des sous-cartes dans lesquelles apparaît une certaine cohésion dans les mesures. Dans le cas présenté en exemple, le début de la trajectoire et la fin sont **pratiquement décorrélées**, et ce en partie à cause du manque de mesures lors d'une partie de la trajectoire.

Pour illustrer encore plus notre propos, considérons désormais la figure 8.2. Il s'agit du résultat d'une simulation d'un robot évoluant parfaitement en ligne droite et observant 50 amers disposés aléatoirement dans une bande autour de la trajectoire. Les données simulées sont l'odométrie et un cap-

1. Pour rappel, il est montré expérimentalement dans [Bailey *et al.*, 2006a] que l'incertitude associée à l'orientation du robot plafone autour d'un écart-type à 5 degrés lors de l'utilisation du filtre de Kalman étendu. Ceci peut donner un premier critère de consistance dans le cas de l'EKF-SLAM. Il n'est néanmoins pas représentatif des approches de type SAM ; nous ne prendrons donc pas en compte ce critère absolu.

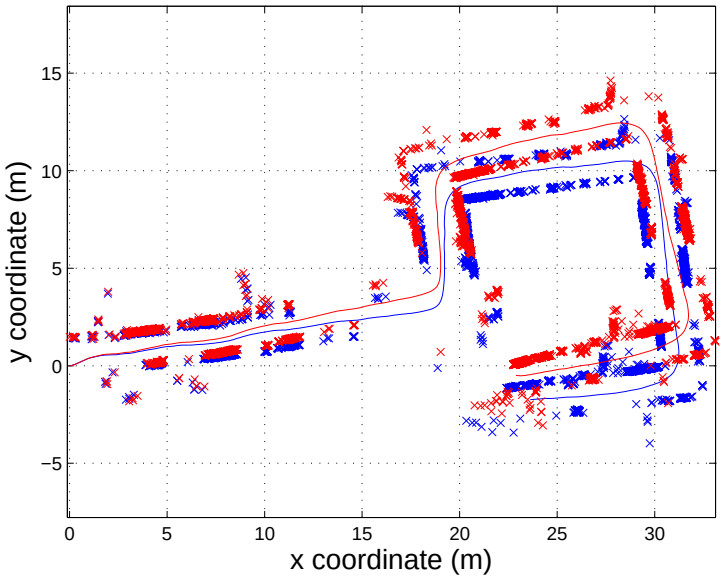


FIGURE 8.1 – Illustration du décalage dans la solution en raison d'une zone très pauvre en informations (reprise de la figure 5.10 page 196)

teur donnant directement l'angle entre l'axe du robot et l'amer considéré (avec une visibilité limitée). Les paramètres concernant les bruits sont donnés dans le tableau 8.1. Nous avons tracé le résultat obtenu dans deux configurations de l'algorithme de SAM :

1. Solution classique, ie. conditionnellement à $\mathbf{x}_0$ (figure 8.2a). Dans ce cas, on suppose que le vecteur position de la toute première itération est égal à zéro. On remarque alors que la taille des ellipses de confiances augmente continuellement tout au long de la trajectoire (vers la droite). Les amers les plus proches de l'origine ont des enveloppes d'incertitudes très petites par rapport aux amers plus éloignés. [2] Ainsi, les incertitudes associées à la fin de la trajectoire sont beaucoup plus grandes que les incertitudes associées au début de la trajectoire. Néanmoins, ce graphique ne nous dit pas la part de corrélation qui existe autour de la position finale.

2. Solution reconditionnée par rapport à la **dernière position du robot** (figure 8.2b). Ici, on a supposé que la **position finale de la trajectoire est égale au vecteur nul, et ce avec une incertitude nulle**. Cette solution reconditionnée peut être déduite de la solution principale du SAM (nous discutons de cet aspect dans la section suivante ainsi qu'en annexe D). Dans ce cas,

2. Ceci est d'une part dû à la visibilité limitée, mais pas seulement. En effet, même si la visibilité avaient été infinie, on aurait constaté le même phénomène (mais légèrement moins rapidement). Ceci est dû au fait que les matrices jacobiennes associées aux mesures sont pratiquement nulles lorsqu'on est loin de l'amer. Ainsi, les amers éloignés de la position initiale ne sont associés à des matrices jacobiennes significatives que lorsque le robot a déjà accumulé de l'incertitude.

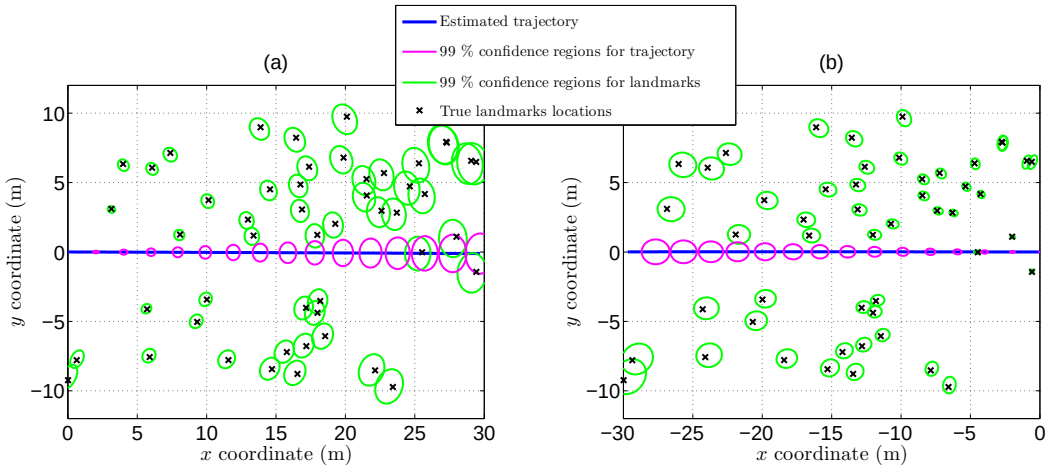


FIGURE 8.2 – Résultats du SAM lors d'une simulation 2D d'un robot avançant en ligne droite et observant 50 amers dans le cas *bearing-only* — (a) Cas classique : la solution est conditionnée par rapport à $\mathbf{x}_0 = \mathbf{0}$ – (b) Reconditionnement : la solution est reconditionnée par rapport à $\mathbf{x}_t = \mathbf{0}$.

on remarque que les amers proches de la position finale du robot ont une incertitude très faible. Au contraire, ceux proches de la position initiale sont cette fois-ci associés à une incertitude élevée. La faible incertitude associée aux derniers amers montre que le déplacement relatif entre les états proche de la position finale est très bien connu. Ceci n'était pas visible sur la figure 8.2a (où les grandes ellipses ne montraient pas l'importante corrélation entre les amers proches de la solution finale).

Ainsi, la figure 8.2 met en évidence le fait que les amers sont localement fortement corrélés. On voit effectivement deux ordres de grandeur différents entre le positionnement relatif de deux amers, suivant qu'ils soient éloignés ou non. Ceci peut justifier certaines instabilités de la carte lorsque le nombre d'amers devient trop important (figure 8.1). Même si ces corrélations peuvent être déduites du résultat du SAM, il peut paraître inutile de conserver l'intégralité de la carte lorsque l'on sait que le début de la carte et la fin ne partagent pratiquement plus d'information.

Remarque 8.1 (Interprétation de la figure 8.2)

Les résultats des figures 8.2a et 8.2b ne peuvent pas être fusionnés. Il convient de remarquer qu'il s'agit bien de deux solutions différentes correspondant à deux façon différentes de poser le problème du SLAM (ie. choix de la position de base). Le raccourci visant à fusionner ces deux cartes de sorte à avoir de faibles incertitudes au début et à la fin de la trajectoire conduirait en réalité à admettre que la

	Description	Erreur associée
Entrées : $\mathbf{u}_{1:t}$	Odométrie : vitesses de rotation ($\omega = 0$) et linéaire ($V = 1\text{m.s}^{-1}$).	$\sigma_\omega = 0.1\text{rad.s}^{-1}$ et $\sigma_V = 0.2\text{m.s}^{-1}$
Modèle	Modèle d'intégration d'odométrie classique (3 degrés de liberté). On suppose ce modèle entaché d'erreurs concernant la position.	$\mathbf{Q}_{\text{mod}} = \text{diag}((0.001m)^2, (0.001m)^2, 0)$
Mesures : $\mathbf{z}_{1:t}$	Mesures angulaires entres les amers et le robot : $\mathbf{z}_k^{(i)} = \varphi_k^{(i)}$.	$\sigma_\varphi = \frac{\pi}{180}\text{rad}$ Visibilité à 360 degrés mais limitée à une distance de 10m.

TABLE 8.1 – Paramètres de la simulation présentée sur la figure 8.2.

position initiale et la position finale sont toutes les deux connues. Même si le calcul d'une telle fusion est possible mathématiquement, il ne répond pas à une réalité physique.

8.1.3 Définition du critère

Pour prendre en compte les réflexions effectuées au paragraphe précédent, nous proposons d'utiliser un critère de changement de carte vérifiant **quand les amers couramment observés ne sont pratiquement plus liés aux premiers amers**. Les raisons qui ont motivées ce choix sont :

- si la carte doit être fournie à un utilisateur humain, celui-ci n'aura à sa disposition que le résultat de l'estimation voire également les enveloppes d'incertitudes associées. Il est impossible d'informer " facilement " l'utilisateur final du fait que certaines zones de la carte sont fortement corrélées et quel est le niveau de ces corrélations. Dans le cas d'une application militaire où la carte correspond à une zone de combat (et devant être interprétée par des humains), il paraît plus sûr de donner aux utilisateurs un ensemble de sous-cartes très consistantes et bénéficiant d'une forte cohésion plutôt que de donner une grande carte globale dont on n'est pas suffisamment certain des positionnements relatifs sur le long terme. Par ailleurs, même si l'utilisateur final n'est pas un être humain mais un autre algorithme, il peut être préférable de limiter la quantité d'information à lui envoyer. Segmenter les cartes locales de cette façon permet à l'algorithme en aval de ne pas avoir besoin de calculer l'ensemble des corrélations qui peuvent être couteuses à calculer (elles sont alors supposées suffisamment fortes),
- si l'on veut assurer la consistance de la sous-carte traitée, conserver des zones peu corrélées peut entraîner des instabilités et des zones de convergence élargies (cf. figure 8.1), [3]

3. En pratique, on n'observe des instabilités que lorsque le nombre d'amers devient très important et provoque des

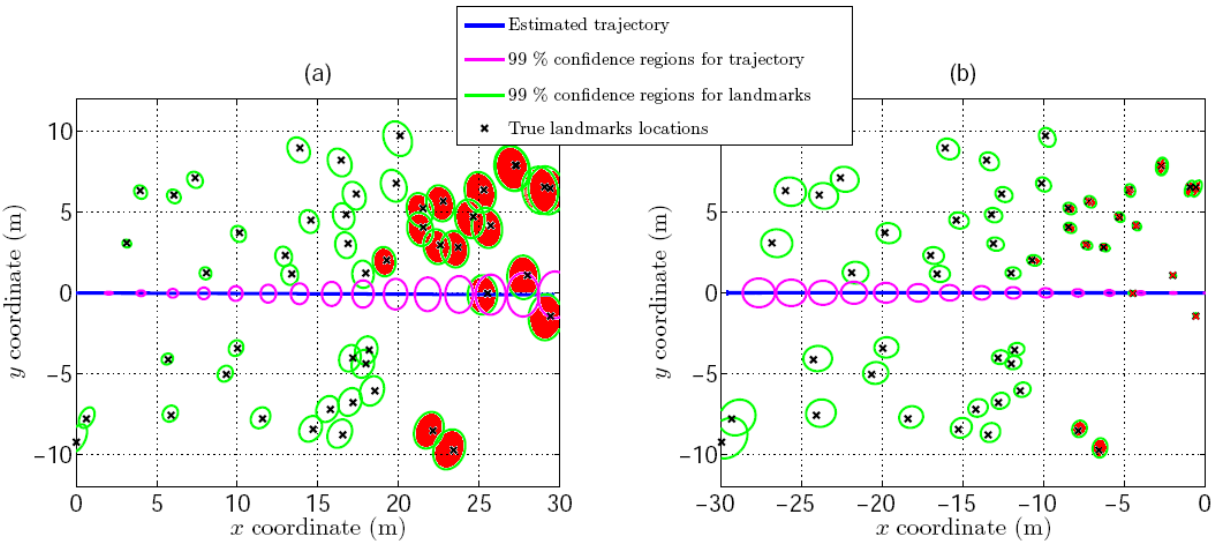


FIGURE 8.3 – Résultats de la figure 8.2 où on a mis en évidence les amers en vue à la dernière itération (zone rouge)

– la notion de corrélation est indépendante du facteur d'échelle, de la taille de la zone ou du nombre d'amers. Il s'agit d'une information **relative** qui correspond bien à ce que l'on recherche. Avec une telle approche, on peut très bien avoir des sous-cartes très denses dont les ellipses de confiance sont de l'ordre de quelques centimètres (typiquement une zone en environnement d'intérieur) et des sous-cartes beaucoup moins denses dans lesquelles la taille minimale des ellipses est de l'ordre du décimètre voire du mètre (typique d'une zone en extérieur dans laquelle les amers sont tous assez éloignés du robot).

Pour résoudre ce problème de segmentation, nous proposons de définir un critère inspiré de l'explication de la figure 8.2. Soit $\mathcal{M}_t$ l'ensemble des amers visibles à la dernière itération (instant t) et représenté en rouge sur la figure 8.3. L'idée du critère est de comparer la taille des ellipses de $\mathcal{M}_t$ dans les deux cas de conditionnement. Si les ellipses sont " beaucoup plus petites " lorsque l'on conditionne par rapport à la position courante (figure 8.3b) que par rapport à la position initiale (figure 8.3a), alors on peut estimer que les derniers amers vus ne partagent plus beaucoup d'information avec la position intiale, et donc avec le début de la carte. Il paraît alors justifié de commencer une nouvelle carte. Cette information va être calculée par le biais des matrices de variances-covariances.

Définissons ainsi pour chacun des amers $k \in \mathcal{M}_t$ les deux matrices suivantes :

1. $\mathbf{\Sigma}^0_{(k)}$: la matrice de variances-covariances associée à $p\left(\mathbf{m}_{(k)}|\mathbf{x}_0, \mathbf{u}_{0:t}, \mathbf{z}_{0:t}\right)$. Il s'agit de la matrice

instabilités numériques. Lorsque cela se produit, on a en général dépassé la taille critique de la carte nécéssaire à un traitement en temps réel.

de variances-covariances associée à l'amer $\mathbf{m}_{(k)}$ dans le cas où il est exprimé par rapport à la position initiale du robot,

2. $\mathbf{\Sigma}^t_{(k)}$: la matrice de variances-covariances associée à $p\left(\mathbf{m}_{(k)}|\mathbf{x}_t, \mathbf{u}_{0:t}, \mathbf{z}_{0:t}\right)$. Il s'agit de la matrice de variances-covariances associée à l'amer $\mathbf{m}_{(k)}$ dans le cas où il est exprimé par rapport à la position du robot à l'instant t.

Sachant que la racine carré du déterminant de la matrice de variances-covariances est proportionnelle au volume (resp. aire) de l'ellipsoïde (resp. ellipse) de confiance pour un amer ponctuel 3D (resp. 2D), nous proposons de calculer les coefficients $c_{(k)}$ définis par :

$$c_{(k)} = \sqrt{\frac{\det \mathbf{\Sigma}^t_{(k)}}{\det \mathbf{\Sigma}^0_{(k)}}} \tag{8.1}$$

Ainsi, le rapport $c_{(k)}$ nous indique clairement dans quel repère il vaut mieux exprimer l'amer (k). Finalement, le test que nous choisissons consiste à calculer la moyenne des $c_{(k)}$ est à la comparer à un seuil s (inférieur à 1) :

$$\frac{1}{\text{card}(\mathcal{M}_t)} \sum_{k \in \mathcal{M}_t} c_{(k)} \overset{?}{\leq} s \quad \text{avec } s < 1 \tag{8.2}$$

Le seuil s est le seul paramètre de réglage à définir, et son interprétation est relativement aisée. Il permet de configurer quel rapport moyen on tolère sur le volume des ellipsoïdes (entre celle calculée dans le repère attaché à $\mathbf{x}_0$ et celle calculée dans le repère attaché à $\mathbf{x}_t$). Ce critère ne fait donc intervenir aucun réglage absolu : la seule grandeur à régler est adimensionnelle, tout en ayant une interprétation physique simple.

8.1.4 Conclusion

Nous avons présenté dans cette section un critère relatif permettant de gérer les changements de cartes locales. Celui-ci permet de détecter lorsque la corrélation devient trop faible entre les amers couramment observés et les amers du début de la carte. Ceci permet de garantir que les sous-cartes que l'on construit ne sont pas affectées par des " degrés de liberté " risquant de faire diverger l'algorithme de SAM. On a ainsi des cartes dont l'homogénéité des incertitudes est controlée. Cette dernière propriété est de première importance si la carte doit servir pour un algorithme critique (comme le contrôle du robot.

Néanmoins, le critère que nous proposons ne doit pas être utilisé seul. La solution classique consistant à limiter le nombre d'amers afin de borner les temps de calculs doit également subsister en tant

que garde-fou. **Une implémentation dans des conditions réelles conduirait donc à utiliser deux critères : le changement de carte s'opère alors dès qu'une des deux conditions est réalisée**.

On pourra enfin remarquer que le calcul de ce critère nécessite le reconditionnement des variables aléatoires suivi d'une marginalisation. Même si on ne cherche à retrouver que les derniers amers vus, cette opération peut s'avérer longue si le nombre d'amers à marginaliser est important. Ceci n'est pas rédhibitoire. En effet, il n'est pas nécessaire d'effectuer le calcul du critère à toutes les itérations. Dans le cadre d'une résolution en temps réel, on peut très bien envisager une programmation parallèle sur plusieurs CPU dont un serait dédié au calcul du critère. Même si le calcul du critère prend K itérations, si on reçoit à $t + K$ l'information que le critère était inférieur au seuil, il sera *a fortiori* encore souhaitable de changer de carte à $t + K$.

8.2 Création des cartes locales

Dans la section précédente, nous nous sommes intéressés à chercher le moment adéquat pour commencer une nouvelle carte locale. Nous présentons dans cette section l'intégralité de l'algorithme utilisé pour créer les cartes locales ainsi que les développements mathématiques nécessaires.

Le principe de l'algorithme que nous avons développé est **de faire en sorte que la carte courante exploite l'intégralité des informations acquises même si celles-ci ont été acquises avant le changement de carte**.

Dans un premier temps, nous présentons les nouvelles notations liées à la gestion des cartes locales (paragraphe 8.2.1). Nous définirons alors clairement notre objectif d'un point de vue mathématique. Nous présentons ensuite un théorème permettant de répondre sans approximation à la question posée. Nous montrons au paragraphe 8.2.3 comment appliquer ce théorème à notre cas, à savoir la création de cartes locales avec un algorithme de SAM. Enfin, une synthèse des développements présentés, ainsi que des éléments de comparaison avec la bibliographie sont présentés au paragraphe 8.2.4.

8.2.1 Notations, hypothèses et formulation du problème

Les nouvelles notations que nous introduisons font directement référence au réseau bayésien de la figure 8.4. Dans la suite, nous notons :

- t_1 le temps choisi pour commencer la nouvelle carte locale,
- t_2 le temps final,

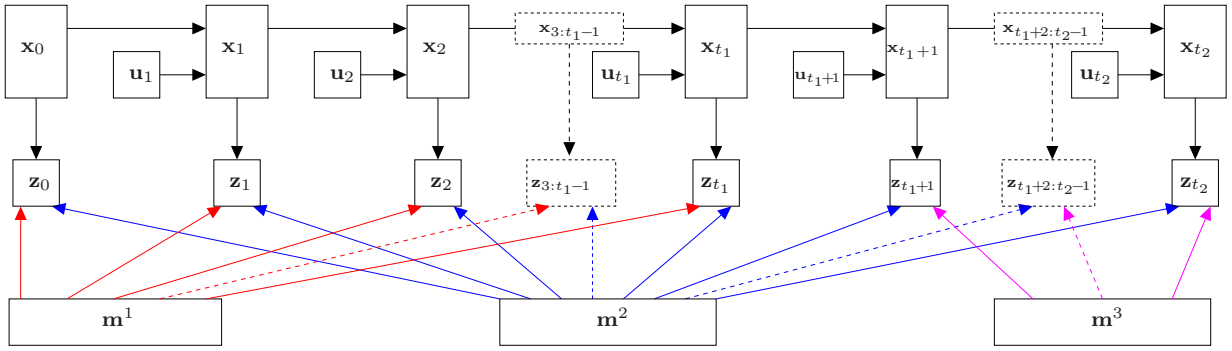


FIGURE 8.4 – Réseau bayésien associé au SLAM

- $\mathbf{m}^1$ l'ensemble des amers visibles **uniquement** dans la première carte, ie. entre les instants 0 et t_1 (compris),
- $\mathbf{m}^2$ l'ensemble des amers visibles à la fois dans la première carte et dans la seconde carte,
- $\mathbf{m}^3$ l'ensemble des amers visibles **uniquement** dans la seconde carte, ie. entre les instants t_1+1 et t_2.

Dans la suite, nous allons chercher à calculer la densité de probabilité associée à la trajectoire $\mathbf{x}_{t_1+1:t_2}$, ainsi qu'aux amers $\mathbf{m}^2$ et $\mathbf{m}^3$, conditionnellement à $\mathbf{x}_{t_1}$ en prenant en compte l'intégralité des mesures acquises depuis l'instant initial. La densité recherchée est donc :

$$\boxed{p\left(\mathbf{x}_{t_1+1:t_2}, \mathbf{m}^2, \mathbf{m}^3 | \mathbf{x}_{t_1}, \mathbf{z}_{0:t_2}, \mathbf{u}_{0:t_2}\right)}$$

Cette approche est différente de la plupart des approches classiques dont les calculs dans les cartes locales ne concernent en général que $p\left(\mathbf{x}_{t_1+1:t_2}, \mathbf{m}^2, \mathbf{m}^3 | \mathbf{x}_{t_1}, \mathbf{z}_{t_1+1:t_2}, \mathbf{u}_{t_1+1:t_2}\right)$ (densité de probabilité dont on a tronqué les mesures dans la première carte) voire $p\left(\mathbf{x}_{t_1+1:t_2}, \mathbf{m}^3 | \mathbf{x}_{t_1}, \mathbf{z}_{t_1+1:t_2}, \mathbf{u}_{t_1+1:t_2}\right)$ (densité de probabilité sans prendre en compte les amers présents dans les deux cartes).

Pour arriver à notre résultat, nous supposons connaître $p\left(\mathbf{x}_{0:t_1}, \mathbf{m}^1, \mathbf{m}^2 | \mathbf{x}_0, \mathbf{z}_{0:t_1}, \mathbf{u}_{0:t_1}\right)$: il s'agit de la densité de probabilité associée à la solution de la première carte. Ceci implique que la densité de probabilité $p\left(\mathbf{m}^2 | \mathbf{x}_0, \mathbf{z}_{0:t_1}, \mathbf{u}_{0:t_1}\right)$ est également connue (on peut la déduire facilement par marginalisation).

8.2.2 Théorème permettant la création de cartes locales

Théorème 8.1 (Construction d'une carte locale)

En prenant en compte les hypothèses et notations introduites au paragraphe précédent, on a :

$$\boxed{p\left(\mathbf{x}_{t_1+1:t_2}, \mathbf{m}^2, \mathbf{m}^3 | \mathbf{x}_{t_1}, \mathbf{z}_{0:t_2}, \mathbf{u}_{0:t_2}\right) \propto p\left(\mathbf{x}_{t_1+1:t_2}, \mathbf{m}^2, \mathbf{m}^3 | \mathbf{x}_{t_1}, \mathbf{z}_{t_1+1:t_2} \mathbf{u}_{t_1+1:t_2}\right) \times p\left(\mathbf{m}^2 | \mathbf{z}_{0:t_1} \mathbf{u}_{0:t_1}, \mathbf{x}_{t_1}\right)} \quad (8.3)$$

Le théorème 8.1 permet de construire de façon immédiate une nouvelle carte locale. En effet, il montre que la densité de probabilité cherchée est proportionnelle à $p\left(\mathbf{x}_{t_1+1:t_2}, \mathbf{m}^2, \mathbf{m}^3 | \mathbf{x}_{t_1}, \mathbf{z}_{t_1+1:t_2} \mathbf{u}_{t_1+1:t_2}\right)$ (densité de probabilité que l'on peut calculer grâce à un procédé classique de SAM) multipliée par un terme déjà connu. Nous verrons au paragraphe suivant comment utiliser ce résultat. La suite de ce paragraphe est consacrée à la démonstration du théorème précédent.

Démonstration du théorème 8.1 :

▲ La démonstration qui suit fait appel à des notions d'interprétations des réseaux bayésiens. Un rappel de ces notions est donné en annexe A (page 321).

Rappelons tout d'abord les deux formulations de la règle de Bayes (énoncées au chapitre 1) :

- $p(a,b) = p(a|b)\, p(b)$
- $p(a|b) = p(b|a)\, \frac{p(a)}{p(b)}$

En applicant la première forme de la règle de Bayes sur $p\left(\mathbf{x}_{t_1+1:t_2}, \mathbf{m}^2, \mathbf{m}^3 | \mathbf{x}_{t_1}, \mathbf{z}_{0:t_2}, \mathbf{u}_{0:t_2}\right)$ avec $a = \mathbf{x}_{t_1+1:t_2}$ et $b = \mathbf{m}^2$, on obtient :

$$p\left(\mathbf{x}_{t_1+1:t_2}, \mathbf{m}^2, \mathbf{m}^3 | \mathbf{x}_{t_1}, \mathbf{z}_{0:t_2}, \mathbf{u}_{0:t_2}\right) = p\left(\mathbf{x}_{t_1+1:t_2}, \mathbf{m}^3 | \mathbf{m}^2, \mathbf{z}_{0:t_1}, \mathbf{u}_{0:t_1}, \mathbf{z}_{t_1+1:t_2} \mathbf{u}_{t_1+1:t_2}, \mathbf{x}_{t_1}\right) \times p\left(\mathbf{m}^2 | \mathbf{z}_{0:t_1}, \mathbf{u}_{0:t_1}, \mathbf{z}_{t_1+1:t_2}, \mathbf{u}_{t_1+1:t_2}, \mathbf{x}_{t_1}\right) \tag{8.4}$$

Nous allons désormais simplifier les deux facteurs de l'équation 8.4 afin d'arriver au résultat.

Premier facteur de (8.4)

Conditionné à $\mathbf{m}^2$ et $\mathbf{x}_{t_1}$, les variables aléatoires $[\mathbf{x}^T_{t_1+1:t_2}, \mathbf{m}^{3T}]^T$ et $[\mathbf{z}^T_{0:t_1}, \mathbf{u}^T_{0:t_1}]^T$ sont indépendantes. On peut en effet voir sur la figure 8.4 que tous les chemins qui commencent à un nœud appartenant à l'ensemble $\{\mathbf{x}_{t_1+1}, \ldots, \mathbf{x}_{t_2}, \mathbf{m}^3\}$ et qui se terminent à un nœud dans $\{\mathbf{z}_0, \ldots, \mathbf{z}_{t_1}, \mathbf{u}_0, \ldots, \mathbf{u}_{t_1}\}$ contiennent au moins :

- la connexion en série $i \rightarrow \mathbf{x}_{t_1} \rightarrow k$ si le chemin utilise directement la chaîne de Markov pour " traverser " le nœud $\mathbf{x}_{t_1}$,
- la connexion divergente $i \leftarrow \mathbf{m}^2 \rightarrow k$ si le chemin utilise le nœud $\mathbf{m}^2$,
- la connexion divergente $\mathbf{z}_{t_1} \leftarrow \mathbf{x}_{t_1} \rightarrow \mathbf{x}_{t_1+1}$ sinon.

Les ensembles de nœuds $\{\mathbf{x}_{t_1+1}, \ldots, \mathbf{x}_{t_2}, \mathbf{m}^3\}$ et $\{\mathbf{z}_0, \ldots, \mathbf{z}_{t_1}, \mathbf{u}_0 \ldots \mathbf{u}_{t_1}\}$ sont donc d-séparés conditionnellement à l'ensemble de nœuds $\{\mathbf{x}_{t_1}, \mathbf{m}^2\}$: il y a indépendance des vecteurs aléatoires considérés. En conséquence, le premier facteur de l'équation 8.4 peut être simplifié :

$$p\left(\mathbf{x}_{t_1+1:t_2}, \mathbf{m}^3 | \mathbf{m}^2, \mathbf{z}_{0:t_1}, \mathbf{u}_{0:t_1}, \mathbf{z}_{t_1+1:t_2} \mathbf{u}_{t_1+1:t_2}, \mathbf{x}_{t_1}\right) = p\left(\mathbf{x}_{t_1+1:t_2}, \mathbf{m}^3 | \mathbf{m}^2, \mathbf{z}_{t_1+1:t_2} \mathbf{u}_{t_1+1:t_2}, \mathbf{x}_{t_1}\right) \tag{8.5}$$

Second facteur de (8.4)

Appliquons la seconde formulation de la règle de Bayes avec :

- $a = \mathbf{m}^2$,
- $b = \mathbf{z}_{0:t_1}, \mathbf{u}_{0:t_1}, \mathbf{z}_{t_1+1:t_2}, \mathbf{u}_{t_1+1:t_2}$.

On obtient alors :

$$p\left(\mathbf{m}^2|\mathbf{z}_{0:t_1}, \mathbf{u}_{0:t_1}, \mathbf{z}_{t_1+1:t_2}, \mathbf{u}_{t_1+1:t_2}, \mathbf{x}_{t_1}\right) = \frac{p\left(\mathbf{z}_{0:t_1}, \mathbf{u}_{0:t_1}, \mathbf{z}_{t_1+1:t_2}, \mathbf{u}_{t_1+1:t_2}|\mathbf{m}^2, \mathbf{x}_{t_1}\right) \times p\left(\mathbf{m}^2|\mathbf{x}_{t_1}\right)}{p\left(\mathbf{z}_{0:t_1}, \mathbf{u}_{0:t_1}, \mathbf{z}_{t_1+1:t_2}, \mathbf{u}_{t_1+1:t_2}|\mathbf{x}_{t_1}\right)} \quad (8.6)$$

Une analyse du réseau bayésien de la figure 8.4 similaire à l'analyse précédente permet de montrer que les vecteurs aléatoires $[{\mathbf{z}_{0:t_1}}^T, {\mathbf{u}_{0:t_1}}^T]^T$ et $[{\mathbf{z}_{t_1+1:t_2}}^T, {\mathbf{u}_{t_1+1:t_2}}^T]^T$ sont indépendants conditionnellement à $\mathbf{m}^2$ et $\mathbf{x}_{t_1}$. On a donc :

$$p\left(\mathbf{z}_{0:t_1}, \mathbf{u}_{0:t_1}, \mathbf{z}_{t_1+1:t_2}, \mathbf{u}_{t_1+1:t_2}|\mathbf{m}^2, \mathbf{x}_{t_1}\right) = p\left(\mathbf{z}_{0:t_1}, \mathbf{u}_{0:t_1}|\mathbf{m}^2, \mathbf{x}_{t_1}\right) \times p\left(\mathbf{z}_{t_1+1:t_2}, \mathbf{u}_{t_1+1:t_2}|\mathbf{m}^2, \mathbf{x}_{t_1}\right) \quad (8.7)$$

En appliquant la règle de Bayes sur chacun des facteurs de l'équation 8.7, on obtient :

$$\begin{aligned} p\left(\mathbf{z}_{0:t_1}, \mathbf{u}_{0:t_1}, \mathbf{z}_{t_1+1:t_2}, \mathbf{u}_{t_1+1:t_2}|\mathbf{m}^2, \mathbf{x}_{t_1}\right) = p\left(\mathbf{m}^2|\mathbf{z}_{0:t_1}, \mathbf{u}_{0:t_1}, \mathbf{x}_{t_1}\right) \frac{p\left(\mathbf{z}_{0:t_1}, \mathbf{u}_{0:t_1}|\mathbf{x}_{t_1}\right)}{p\left(\mathbf{m}^2|\mathbf{x}_{t_1}\right)} \times \\ p\left(\mathbf{m}^2|\mathbf{z}_{t_1+1:t_2,\mathbf{x}_{t_1}}, \mathbf{u}_{t_1+1:t_2}\right) \frac{p\left(\mathbf{z}_{t_1+1:t_2}, \mathbf{u}_{t_1+1:t_2}|\mathbf{x}_{t_1}\right)}{p\left(\mathbf{m}^2|\mathbf{x}_{t_1}\right)} \end{aligned} \quad (8.8)$$

En substituant l'équation 8.8 dans l'équation 8.6, on obtient :

$$p\left(\mathbf{m}^2|\mathbf{z}_{0:t_1}, \mathbf{u}_{0:t_1}, \mathbf{z}_{t_1+1:t_2}, \mathbf{u}_{t_1+1:t_2}, \mathbf{x}_{t_1}\right) = \quad \alpha \frac{p\left(\mathbf{m}^2|\mathbf{z}_{0:t_1}, \mathbf{u}_{0:t_1}, \mathbf{x}_{t_1}\right) \times p\left(\mathbf{m}^2|\mathbf{z}_{t_1+1:t_2}, \mathbf{u}_{t_1+1:t_2}, \mathbf{x}_{t_1}\right)}{p\left(\mathbf{m}^2|\mathbf{x}_{t_1}\right)} \quad (8.9)$$

où $\alpha = \frac{p\left(\mathbf{z}_{0:t_1}, \mathbf{u}_{0:t_1}|\mathbf{x}_{t_1}\right) p\left(\mathbf{z}_{t_1+1:t_2}, \mathbf{u}_{t_1+1:t_2}|\mathbf{x}_{t_1}\right)}{p\left(\mathbf{z}_{0:t_1}, \mathbf{u}_{0:t_1}, \mathbf{z}_{t_1+1:t_2}, \mathbf{u}_{t_1+1:t_2}|\mathbf{x}_{t_1}\right)}$ est indépendant de $\mathbf{x}_{t_1+1:t_2}$, $\mathbf{m}^2$ et $\mathbf{m}^3$. En conséquence, α est une constante de normalisation. De plus, la figure 8.4 montre que $\mathbf{x}_{t_1}$ et $\mathbf{m}^2$ sont inconditionnellement independants. Ceci implique $p\left(\mathbf{m}^2|\mathbf{x}_{t_1}\right) = p\left(\mathbf{m}^2\right)$.[4] Par ailleurs, nous supposons que nous n'avons pas de connaissance *a priori* sur $\mathbf{m}^2$; $p\left(\mathbf{m}^2\right)$ peut donc être inclu dans la constante de normalisation. L'équation 8.9 devient alors :

$$p\left(\mathbf{m}^2|\mathbf{z}_{0:t_1}, \mathbf{u}_{0:t_1}, \mathbf{z}_{t_1+1:t_2}, \mathbf{u}_{t_1+1:t_2}, \mathbf{x}_{t_1}\right) \propto p\left(\mathbf{m}^2|\mathbf{z}_{0:t_1}, \mathbf{u}_{0:t_1}, \mathbf{x}_{t_1}\right) \times p\left(\mathbf{m}^2|\mathbf{z}_{t_1+1:t_2}, \mathbf{u}_{t_1+1:t_2}, \mathbf{x}_{t_1}\right) \quad (8.10)$$

Retour à l'équation principale

En utilisant les équations 8.5 et 8.10, l'équation 8.4 devient :

$$\begin{aligned} p\left(\mathbf{x}_{t_1+1:t_2}, \mathbf{m}^2, \mathbf{m}^3|\mathbf{x}_{t_1}, \mathbf{z}_{0:t_2}, \mathbf{u}_{0:t_2}\right) \propto p\left(\mathbf{x}_{t_1+1:t_2}, \mathbf{m}^3|\mathbf{m}^2, \mathbf{z}_{t_1+1:t_2}\mathbf{u}_{t_1+1:t_2}, \mathbf{x}_{t_1}\right) \times \\ p\left(\mathbf{m}^2|\mathbf{z}_{0:t_1}, \mathbf{u}_{0:t_1}, \mathbf{x}_{t_1}\right) \times p\left(\mathbf{m}^2|\mathbf{z}_{t_1+1:t_2}, \mathbf{u}_{t_1+1:t_2}, \mathbf{x}_{t_1}\right) \end{aligned} \quad (8.11)$$

4. Etant donné qu'elle n'est conditionnée à aucune mesure, la densité de probabilité $p\left(\mathbf{m}^2\right)$ représente l'*a priori* que l'on a sur la variable $\mathbf{m}^2$.

Les premier et troisième facteurs de l'équation 8.11 peuvent être regroupés et simplifiés grâce à la règle de Bayes :

$$p\left(\mathbf{x}_{t_1+1:t_2}, \mathbf{m}^3 | \mathbf{m}^2, \mathbf{z}_{t_1+1:t_2}\mathbf{u}_{t_1+1:t_2}, \mathbf{x}_{t_1}\right) \times p\left(\mathbf{m}^2 | \mathbf{z}_{t_1+1:t_2}, \mathbf{u}_{t_1+1:t_2}, \mathbf{x}_{t_1}\right) = p\left(\mathbf{x}_{t_1+1:t_2}, \mathbf{m}^3, \mathbf{m}^2 | \mathbf{z}_{t_1+1:t_2}\mathbf{u}_{t_1+1:t_2}, \mathbf{x}_{t_1}\right) \tag{8.12}$$

On obtient le résultat final en substituant le résultat de l'équation 8.12 dans l'équation 8.11 :

$$p\left(\mathbf{x}_{t_1+1:t_2}, \mathbf{m}^2, \mathbf{m}^3 | \mathbf{x}_{t_1}, \mathbf{z}_{0:t_2}, \mathbf{u}_{0:t_2}\right) \propto p\left(\mathbf{x}_{t_1+1:t_2}, \mathbf{m}^3, \mathbf{m}^2 | \mathbf{z}_{t_1+1:t_2}\mathbf{u}_{t_1+1:t_2}, \mathbf{x}_{t_1}\right) \times p\left(\mathbf{m}^2 | \mathbf{z}_{0:t_1}, \mathbf{u}_{0:t_1}, \mathbf{x}_{t_1}\right) \tag{8.13}$$

■

8.2.3 Application du théorème

8.2.3.1 Utilisation des termes

Le théorème 8.1 nous permet de créer une nouvelle carte locale très facilement. Analysons dans un premier temps les deux facteurs intervenant dans l'équation 8.3 : [5]

- $p\left(\mathbf{x}_{t_1+1:t_2}, \mathbf{m}^3, \mathbf{m}^2 | \mathbf{z}_{t_1+1:t_2}\mathbf{u}_{t_1+1:t_2}, \mathbf{x}_{t_1}\right)$ désigne la densité de probabilité associée à la trajectoire et aux amers de la carte courante, **en ne prenant en compte que les mesures acquises dans la carte courante**. Avec les hypothèses effectuées dans le cadre du SAM, il s'agit d'une densité gaussienne dont les paramètres d'information $\mathbf{\Omega}$ et ξ peuvent être calculés directement avec l'algorithme de SAM classique en supposant connue la valeur de $\mathbf{x}_{t_1}$ [6] et en utilisant l'ensemble des mesures de $t_1 + 1$ à t_2.
- $p\left(\mathbf{m}^2 | \mathbf{z}_{0:t_1}, \mathbf{u}_{0:t_1}, \mathbf{x}_{t_1}\right)$ désigne la densité de probablité des amers communs à la carte courante et à la carte précédente, conditionnellement à $\mathbf{x}_{t_1}$ et **en utilisant uniquement les mesures effectuées depuis le début jusqu'à la fin de la carte précédente**. Ce terme n'est pas connu directement. Néanmoins, nous supposons connaître $p\left(\mathbf{m}^2 | \mathbf{z}_{0:t_1}, \mathbf{u}_{0:t_1}, \mathbf{x}_{t_0}\right)$ (cf. hypothèse page 266). Cette densité est supposée gaussienne. La différence avec la densité de probabilité cherchée est le point de conditionnement. Nous avons besoin de la densité de probabilité associée à $\mathbf{m}^2$ lorsque $\mathbf{x}_{t_1}$ est connu et fixé à zéro alors que l'on dispose de la densité de probabilité

5. Dans ce qui suit, nous désignons par " carte courante " la nouvelle carte, correspondant aux mesures de $t_1 + 1$ à t_2.

6. Nous supposons que cette valeur est fixée à zéro, ceci permet de visualiser facilement et rapidement les coordonnées courantes par rapport à la position initiale. Ce choix est arbitraire et nous aurions très bien pu choisir n'importe quelle autre valeur. Ceci aurait eu pour effet d'affecter globalement la solution d'une translation et d'une rotation.

de $\mathbf{m}^2$ lorsque $\mathbf{x}_0$ est fixé à zéro. Le passage de l'un à l'autre est possible grâce au processus de reconditionnement (cf. paragraphe 8.2.3.3). On peut donc supposer connus les paramètres d'information associés à $p\left(\mathbf{m}^2|\mathbf{z}_{0:t_1}, \mathbf{u}_{0:t_1}, \mathbf{x}_{t_1}\right)$; on les note dans la suite $\mathbf{\Omega}_{22}^{ini}$ et ξ_2^{ini}.

Finalement, les deux facteurs de l'équation 8.3 décrivent tous les deux des densités de probabilités gaussiennes dont les paramètres d'information sont connus. La densité de probabilité finale est donc gaussienne. Considérons désormais la décompostion des paramètres de $\mathbf{\Omega}$ et de ξ associée à $p\left(\mathbf{x}_{t_1+1:t_2}, \mathbf{m}^3, \mathbf{m}^2|\mathbf{z}_{t_1+1:t_2}\mathbf{u}_{t_1+1:t_2}, \mathbf{x}_{t_1}\right)$ selon les variables associées à la trajectoire, $\mathbf{m^2}$ et $\mathbf{m}^3$:

$$\mathbf{\Omega} = \begin{bmatrix} \mathbf{\Omega_{xx}} & \mathbf{\Omega}_{\mathbf{x}2} & \mathbf{\Omega}_{\mathbf{x}3} \\ \mathbf{\Omega}_{2\mathbf{x}} & \mathbf{\Omega}_{22} & \mathbf{\Omega}_{23} \\ \mathbf{\Omega}_{3\mathbf{x}} & \mathbf{\Omega}_{32} & \mathbf{\Omega}_{33} \end{bmatrix} \quad \text{et} \quad \xi = \begin{bmatrix} \xi_{\mathbf{x}} \\ \xi_2 \\ \xi_3 \end{bmatrix} \tag{8.14}$$

En prenant en compte les notations de l'équation 8.14 ainsi que l'équation 8.3, les paramètres d'information de $p\left(\mathbf{x}_{t_1+1:t_2}, \mathbf{m}^2, \mathbf{m}^3|\mathbf{x}_{t_1}, \mathbf{z}_{0:t_2}, \mathbf{u}_{0:t_2}\right)$ sont donnés par :

$$\boxed{\mathbf{\Omega}^{fin} = \begin{bmatrix} \mathbf{\Omega_{xx}} & \mathbf{\Omega}_{\mathbf{x}2} & \mathbf{\Omega}_{\mathbf{x}3} \\ \mathbf{\Omega}_{2\mathbf{x}} & \mathbf{\Omega}_{22} + \mathbf{\Omega}_{22}^{ini} & \mathbf{\Omega}_{23} \\ \mathbf{\Omega}_{3\mathbf{x}} & \mathbf{\Omega}_{32} & \mathbf{\Omega}_{33} \end{bmatrix} \quad \text{et} \quad \xi^{fin} = \begin{bmatrix} \xi_{\mathbf{x}} \\ \xi_2 + \xi_2^{ini} \\ \xi_3 \end{bmatrix}} \tag{8.15}$$

8.2.3.2 Implémentation de l'algorithme

Le processus de création de cartes locales est décrit en détail dans l'algorithme 8.1. Il consiste finalement à appliquer de façon classique l'algorithme de SAM dans la carte locale. A chaque étape, on ajoute aux paramètres d'information les paramètres liés à la carte précédente (équation 8.15). Il est à noter que **les paramètres à ajouter ne dépendent que de la carte précédente**. Ils sont donc déterminés à chaque fin de carte (par le biais d'un processus de reconditionnement des variables aléatoires) et n'ont plus à être calculés dans la carte courante. Ainsi, la complexité de l'algorithme est pratiquement inchangée : à chaque itération, on a juste une addition de matrices et une addition de vecteurs à effectuer. Néanmoins, cette dernière opération a pour effet de densifier un peu la matrice d'information (initialement, il n'y a pas de termes croisés entre amers dans la matrice d'information totale). La conséquence directe de cette densification est une efficacité moindre de l'algorithme d'inverison permettant de retrouver une estimation du vecteur d'état connaissant les paramètres d'information. Nous n'avons néanmoins pas constaté de différence significative dans notre implémentation.

8.2.3.3 Remarque quant au reconditionnement

Le reconditionnement par rapport à la dernière position n'est pas complètement trivial. Nous distinguons deux configurations :

Fixer s, N_{max} # Seuils pour le critère et nombre d'amers
maximum par carte

$\mathbf{\Omega}^{ini}, \xi^{ini}, \mu^{\mathbf{m}} \leftarrow [\,]$ # Paramètres initiaux pour la carte locale
$\mu, \mu^{\mathbf{x}} \leftarrow \mathbf{0}$
$t_1 \leftarrow 0$ # Temps à partir duquel commence
la carte courante

Pour tout $i = 2 \dots i_{FIN}$ **faire**
| $\widetilde{\mu_i^{\mathbf{x}}} \leftarrow \mathbf{f}(\mu_{i-1}^{\mathbf{x}}, \mathbf{u}_i)$ # Utilisation de la fonction de prédiction
| # pour étendre la trajectoire de linéarisation
| $\mu^{\mathbf{x}} \leftarrow [\mu^{\mathbf{x}T}\ \widetilde{\mu_i^{\mathbf{x}}}^T]^T$
| $\widetilde{\mu_{new}^{\mathbf{m}}} \leftarrow$ **initialiser_nouveaux_amers**$(\mu^{\mathbf{x}}, \mu^{\mathbf{m}}, \mathbf{z})$ # Introduire des points de fonctionnements
| # pour les nouveaux amers dont on accepte
| # l'entrée dans le filtre
| $\mu^{\mathbf{m}} \leftarrow [\mu^{\mathbf{m}T}\ \widetilde{\mu_{new}^{\mathbf{m}}}^T]^T$
| $[\mathbf{\Omega}, \xi] \leftarrow$ **resolution_SAM**$(\mu^{\mathbf{x}}, \mu^{\mathbf{m}}, \mathbf{u}_{t_1+1:i}, \mathbf{z}_{t_1+1:i})$
| $[\mathbf{\Omega}, \xi] \leftarrow [\mathbf{\Omega}, \xi] \oplus [\mathbf{\Omega}^{ini}, \xi^{ini}]$ # " $\oplus$ " désigne ici l'opération effectuée à
| # l'équation 8.15
| $\mu = \mathbf{\Omega}^{-1}\xi$
| $[\mu^{\mathbf{x}}, \mu^{\mathbf{m}}] \leftarrow$ **separer_mu**(μ)
| $c \leftarrow$ **calculer_critère**$(\mathbf{\Omega}, \xi)$ # Eventuellement, étape à réaliser toutes
| # les K itérations
| **Si** $c < s$ **ou taille**$(\mu^{\mathbf{m}}) > N_{max}$ **alors** # Création d'une nouvelle carte
| | $[\mathbf{\Omega}^{ini}, \xi^{ini}] \leftarrow$ **reconditionnement**$(\mathbf{\Omega}, \xi)$
| | $\mu^{\mathbf{m}} \leftarrow$ **selection_amers**$(\mu^{\mathbf{m}})$ # On ne garde que les derniers amers vus
| | $\mu^{\mathbf{m}} \leftarrow$ **recond_amers**$(\mu^{\mathbf{m}})$ # Expression de $\mu^{\mathbf{m}}$ dans la nouvelle
| | # base locale
| | $t_1 \leftarrow i$
| | $\mu^{\mathbf{x}} \leftarrow \mathbf{0}$
| | $\mu \leftarrow [\mu^{\mathbf{x}T}\mu^{\mathbf{m}T}]^T$
| **Fin si**
Fin pour

Algorithme 8.1 – Gestion des cartes locales

1. Nous sommes dans la **première carte locale**. Dans ce cas, il est possible de calculer directement $p\left(\mathbf{x}_{0:t_1}, \mathbf{m}^1, \mathbf{m}^2 | \mathbf{z}_{0:t_1}, \mathbf{u}_{0:t_1}\right)$, sans conditionner par rapport à $\mathbf{x}_0$ (voir chapitre 1). [7] Le conditionnement habituel peut s'effectuer directement avec le théorème de conditionnement (théorème 1.1 page 17). On peut alors facilement conditionner par rapport à $\mathbf{x}_{t_1}$ au lieu de $\mathbf{x}_0$
2. Nous ne sommes plus dans la première carte locale et avons déjà utilisé l'équation 8.3 du théorème 8.1. Le théorème nous donne directement un résultat conditionnel à la première position de la carte locale. Nous n'avons plus accès à la densité de probabilité jointe comme au cas précédent. En conséquence, **on ne peut plus faire un conditionnement immédiat par suppression des termes dans la matrice d'information car les termes liés à la position initiale n'existent déjà plus**. On peut néanmoins toujours effectuer le conditionnement en transformant la matrice et le vecteur d'information avec les matrices jacobiennes appropriées (voir annexe D).

Enfin, même s'il est théoriquement possible d'utiliser n'importe quelle valeur pour la valeur associée à $\mathbf{x}_{t_1}$ du conditionnement, on ne peut pas, en pratique, choisir n'importe quelle valeur. En effet, les non-linéarités rendent les paramètres d'information non consistants lorsqu'on s'éloigne trop des points de fonctionnement. Ainsi, il est fortement recommandé d'utiliser dans un premier temps le point de linéarisation courant pour reconditionner. Ensuite, si l'on veut effectivement fixer la valeur de $\mathbf{x}_{t_1}$ à zéro, on peut effectuer une transformation rigide déterministe de la solution de sorte à ramener $\mathbf{x}_{t_1}$ à zéro (cette transformation sera quant à elle consistante et exacte). Tous les détails de cette opération sont donnés dans l'annexe D.

8.2.4 Conclusion

Nous avons présenté dans cette section un algorithme permettant de créer successivement des cartes locales, et ce de façon simple et consistante. Nous avons montré rigoureusement comment utiliser les informations issues de la carte précédente. Ainsi, même si l'on réinitialise la localisation du robot à chaque nouvelle carte locale, nous sommes capables de tirer parti **de l'intégralité des mesures qui ont été effectuées depuis le début**. Par ailleurs, la factorisation présentée dans le théorème 8.1 permet de rendre l'implémentation de l'algorithme simple et efficace (on a juste à effectuer deux additions supplémentaires).

Notre algorithme est très proche de l'approche par cartes locales décrite dans [Piniés et Tardós,

7. La matrice d'information déduite est alors non inversible.

2008]. Nous utilisons tout comme eux une propriété d'indépendance conditionnelle des cartes locales par rapport au nœud frontière, ce qui permet de construire des cartes locales partageant de l'information. Néanmoins, nous proposons une démonstration complète permettant d'arriver au résultat. Enfin, l'algorithme final que nous utilisons est basé sur le SAM, ce qui permet de garantir de meilleures propriétés de consistance.

8.3 Résultats

Nous présentons dans cette section les résultats obtenus quant à la création de cartes locales. Nous y verrons à la fois l'intérêt du critère décrit à la section 8.1 ainsi que la qualité des cartes locales calculées avec notre algorithme. Les résultats obtenus sont issus des expérimentations présentées au chapire 5 (il s'agit des deux expérimentations en intérieur dans les bâtiments Borel et Kahn de l'INRIA Sophia Antipolis-Méditerranée) ; il s'agit de SLAM à 3 degrés de liberté. [8]

Cette section s'articule en deux parties. Nous présentons d'abord les résultats concernant l'utilisation du critère de changement de carte locale (8.3.1), puis nous montrons l'intérêt de l'utilisation de notre théorème dans le paragraphe 8.3.2.

8.3.1 Qualité du critère

Nous nous intéressons dans ce paragraphe à la façon dont les cartes locales ont été segmentées grâce à l'utilisation du critère défini dans la section 8.1. Même si l'algorithme 8.1 stipule d'utiliser deux critères de changement de carte (notre critère original ainsi qu'un critère basé sur les temps de calculs), nous avons inhibé le second critère afin de ne tester que notre algorithme. Ceci est rendu possible car tous nos tests sont effectués avec le logiciel Matlab sans contrainte d'exécution en temps réel.

La figure 8.5 présente l'évolution du critère de corrélation lorsque l'on ne conserve qu'une seule carte globale. Quelque soit l'expérimentation, on constate que le critère tend à décroître au fur et à mesure que l'on s'éloigne de la position initiale. On remarque par ailleurs que la décroissance est plus rapide au début. Par la suite, la valeur du critère tend à diminuer moins rapidement voire à se stabiliser à une faible valeur. Ceci prouve donc l'intérêt de changer de carte afin de conserver une cohésion suffisamment élevée au sein de la carte.

8. Nous n'avons pas encore appliqué l'algorithme de cartes locales sur le cas du SLAM à 6 degrés présenté au chapitre 6. Néanmoins, le principe reste identique.

La figure 8.6, quant à elle, présente les projections de diverses cartes-locales. [9] Chaque nouvelle carte commence lorsque le critère atteint un certain seuil. [10]

Dans le cas de l'expérimentation du bâtiment Borel (voir la figure 8.6a ainsi que la première partie du tableau 8.2), on constate que les cartes locales couvrent des distances assez proches, [11]. Ceci traduit une baisse progressive à " vitesse constante " de la cohésion avec le début d'une carte. On remarque toutefois que les troisième et huitième cartes ont des caractéristiques particulières :

- la troisième carte comporte peu d'amers. Ceci est dû au fait que la carte commence sur la partie de la trajectoire où il n'y avait pratiquement pas d'amers (seulement 2 amers sont partagés avec la carte précédente). En conséquence, la trajectoire du robot s'est trouvée rapidement décorrélée. Ainsi, dès qu'un nombre significatif d'amers a été observé (juste après le virage), l'algorithme a détecté que le nouveau groupe d'amers avait une forte corrélation interne, mais était décorrélé du début de la carte,
- la huitième carte, quant à elle, est très courte. Néanmoins, elle possède un nombre conséquent d'amers (567). Le changement rapide de carte vient ici du fait que l'on a observé dans les dernières images de cette carte une quantité très importante d'amers très rapprochés (due à un poster très texturé). Ceci a contribué à donner de très fortes corrélations sur cette zone, conduisant finalement à introduire les 157 derniers amers dans une nouvelle carte (sur le début de l'expérimentation, le nombre d'amers vus simultanément n'est jamais aussi élevé).

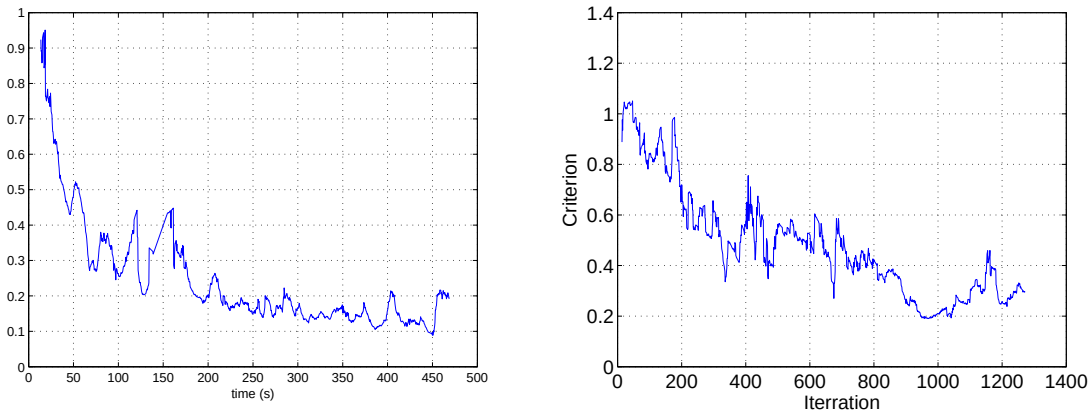
Ainsi, on constate des changements de carte locales dus le plus souvent à une baisse progressive de la corrélation entre le début et la fin de la carte. Dans d'autres cas, les baisses de corrélations sont plus brutales et proviennent d'une variation importante du nombre d'amers observés. Ceci montre la bonne adaptation de notre critère.

On retrouve des résultats similaires dans le cas de l'expérimentation du bâtiment Kahn (figure 8.6b et seconde partie du tableau 8.2). On constate que la troisième carte est plus petite que les autres, à la fois en longueur de trajectoire et en nombre d'amers. Ceci est dû au fait que cette carte commence dans un secteur très sombre (où seulement 5 amers sont observés). A la sortie de cette zone, lorsque le nombre d'amers visibles augmente, le critère a considérablement diminué. Au contraire, la dernière

9. La projection des cartes côte à côte est donnée à titre informatif. Pour rappel, toutes les cartes sont conditionnées par rapport à la première position associée. La juxtaposition des cartes n'a plus de sens mathématique.

10. Le seuil choisi est différent dans chaque expérimentation. 0.3 pour l'expérimentation dans le bâtiment Borel et 0.6 pour celle dans le bâtiment Kahn. Ceci n'a pas d'importance et ne change pas l'interprétation des résultats. Le plus important est de bien conserver un même seuil sur la totalité d'une expérimentation donnée.

11. En termes de longueur de trajectoire (ici équivalent au nombre d'itérations car le robot évolue à vitesse constante).



(a) (b)

FIGURE 8.5 – Evolution du critère testant la corrélation dans le cas où on ne fait pas de cartes locales — (a) Expérimentation du bâtiment Borel – (b) Expérimentation du bâtiment Kahn.

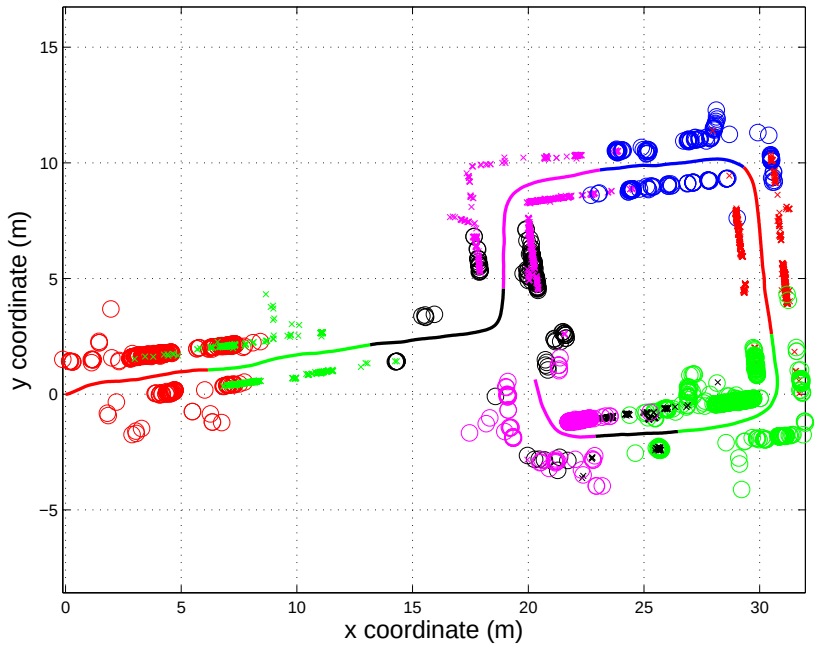
carte est plus importante que les autres (elle atteint plus de 100 itérations de plus que la précédente alors que le critère n'a même pas franchi le seuil). On remarque alors que cette carte commence avec beaucoup d'amers et garde constamment beaucoup d'amers visibles (carte rouge sur la figure 8.6b). Ainsi, l'ensemble de la carte est alors très corrélé.

Au final, les résultats obtenus quant à la segmentation des cartes sont tout à fait cohérent. L'algorithme est en effet capable de détecter lorsque la corrélation au sein d'une carte diminue rapidement (cas des passages où on observait peu de points d'intérêts). Au contraire, les zones où de nombreux points sont observés continûment restent corrélées sur de plus grandes distances, ce qui a été mis en évidence par l'algorithme.

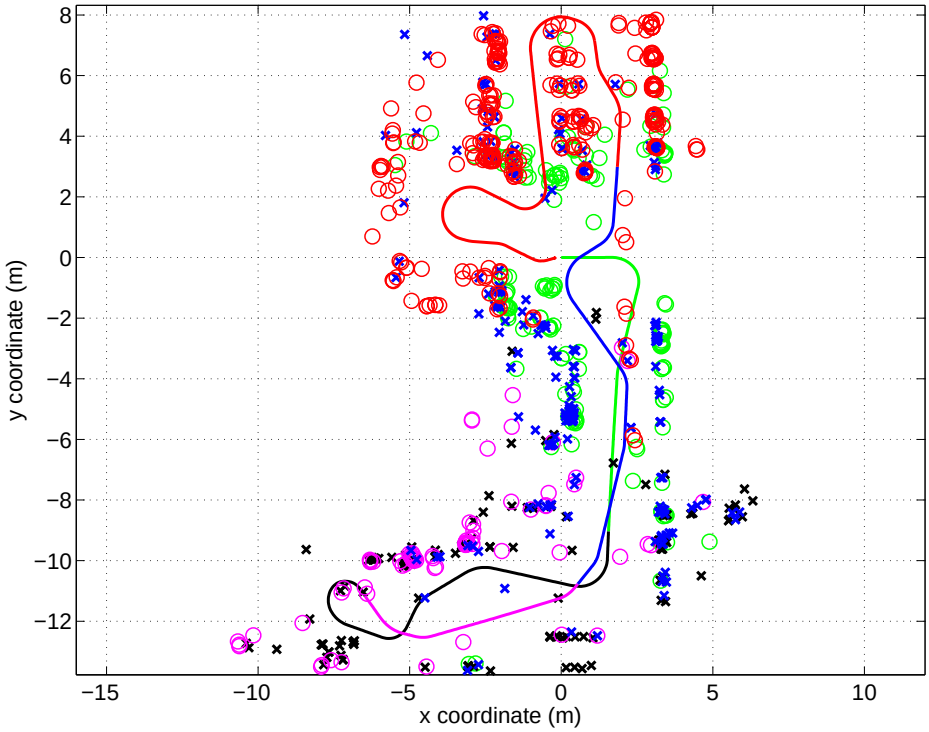
8.3.2 Application du théorème

Nous présentons dans ce paragraphe les résultats obtenus concernant l'application de la factorisation présentée dans le théorème 8.1. La figure 8.7 présente une comparaison entre deux implémentations de l'algorithme de cartes locales :

1. en rouge sont représentées les ellipses de confiance associées à l'algorithme de cartes locales utilisant le théorème,
2. en noir sont représentées les ellipses de confiance associées au calcul de la solution en ne prenant en compte **que les mesures dans la carte courante**. Ce résultat est celui obtenu lorsqu'on ne garde que le second facteur de l'équation 8.3.



(a)



(b)

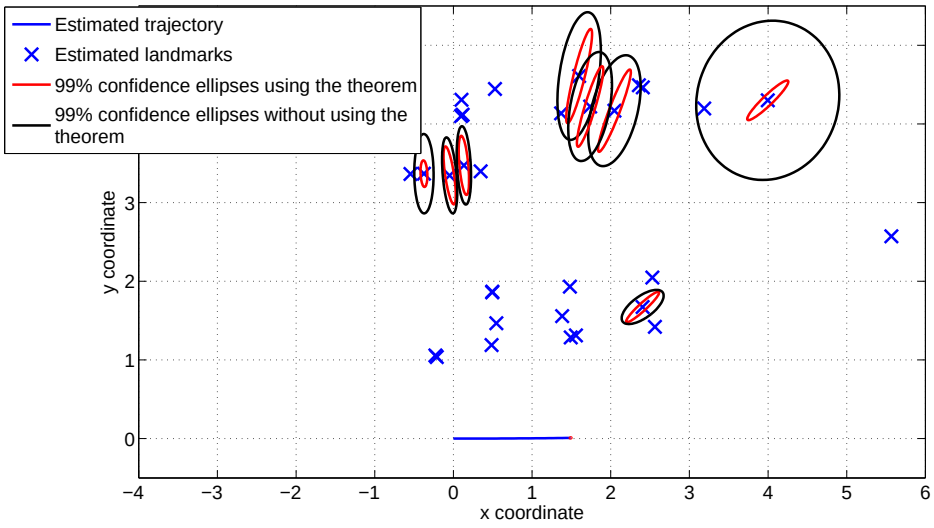
FIGURE 8.6 – Projection des cartes locales. Chaque changement de couleur et de symbole implique un changement de carte locale (les cercles ne représentent pas des ellipses d'incertitudes mais sont utilisés en alternance avec les croix pour pouvoir superposer de façon lisible les marqueurs communs à deux cartes successives) — (a) Expérimentation du bâtiment Borel – (b) Expérimentation du bâtiment Kahn.

	Borel		**Kahn**	
	Nombre d'amers	Nombre d'itérations	Nombre d'amers	Nombre d'itérations
Carte 1	490 — (0/61)	158	168 — (0/6)	216
Carte 2	579 — (61/2)	149	152 — (6/5)	254
Carte 3	147 — (2/69)	144	83 — (5/13)	142
Carte 4	748 — (69/8)	165	230 — (13/27)	278
Carte 5	202 — (8/17)	126	371 — (27/∅)	373
Carte 6	812 — (17/8)	147	∅	∅
Carte 7	668 — (8/25)	151	∅	∅
Carte 8	567 — (25/157)	75	∅	∅
Carte 9	632 — (157/∅)	67	∅	∅

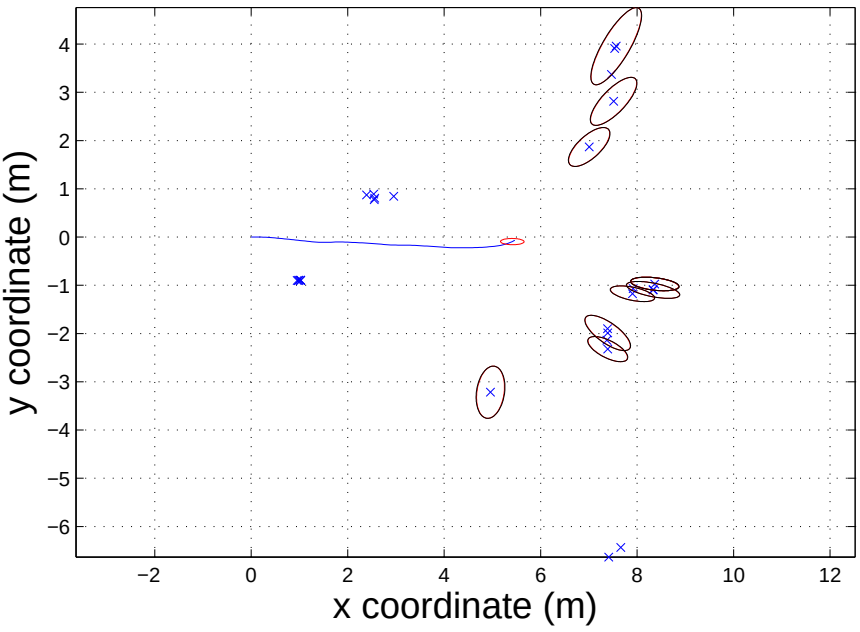
TABLE 8.2 – Caractéristiques des cartes locales en termes de nombre d'amers et de longueur de la trajectoire — En ce qui concerne le nombre d'amers, le premier chiffre désigne le nombre d'amers total de la carte. Le premier chiffre dans le couple entre paranthèses désigne le nombre d'amer hérités de la carte précédente. Le second chiffre entre parenthèses désigne le nombre d'amers transmis à la carte suivante.

La figure 8.7a présente le cas typique. Les ellipses rouges sont beaucoup plus petites que les ellipses noires : on voit alors nettement le gain d'information apporté par l'application du théorème 8.1. La figure 8.7b, quant à elle, est plus difficile à expliquer. Elle montre en effet que les ellipses noires et rouges sont pratiquement confondues : l'application du théorème n'apporte donc rien. Il s'agit de la fin de la troisième carte de l'expérimentation Borel. Cette carte était assez délicate car le manque de mesures a fait que les amers observés vers la fin de la carte ne partageaient pratiquement plus aucune information avec les quelques amers observés au début. Ainsi, l'information partagée entre les seconde et troisième cartes ne s'est finalement pas propagée jusqu'à la fin de la troisième carte : on se retrouve donc dans le cas limite où l'information capitalisée dans les autres cartes locales a été perdue.

Les figures 8.8 présente quant à elle des résultats issues d'une simulation simple. Il s'agit d'une trajectoire circulaire de 60s au cours de laquelle une carte locale a été créée après 35s. On peut voir sur la figure 8.8 que toutes les étapes de l'algorithme sont consistantes. Sur la figure 8.8a, la carte globale est consistante. Ceci constitue une indication sur la qualité des réglages de l'algorithme. On s'attend alors à ce que les versions locales de l'algorithme soient également consistantes. C'est ce que l'on vérifie dans un premier temps sur la figure 8.8b : les ellipses de confiance dans la première carte locale contiennent les vraies valeurs. Enfin, pour visualiser la qualité de la seconde carte locale (figure 8.8c), nous avons exprimé les vérités terrain concernant les coordonnées des amers dans le repère associé à la position du robot à $t = 35s$; ceci nous permet d'obtenir une vérité terrain compatible avec la seconde carte locale. **Nous constatons que la seconde carte locale est bien consistante après**



(a)



(b)

FIGURE 8.7 – Comparaison des résultats avec et sans utilisation du théorème — (a) Cas où l'apport est flagrant – (b) Cas limite où l'application du théorème n'apporte rien

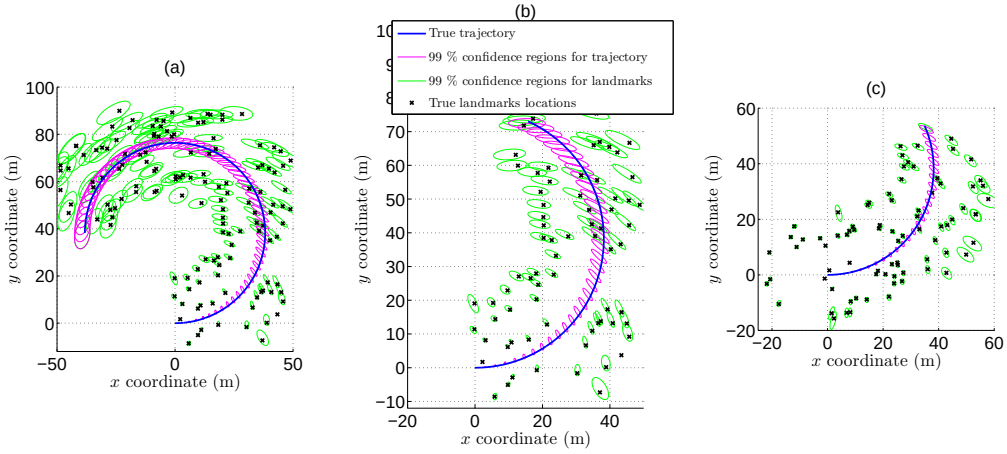


FIGURE 8.8 – Simulation pour tester la consistance du théorème 8.1 — (a) Carte globale obtenue – (b) Première carte locale (durant 35s) – (c) Seconde carte locale : on a tout exprimé dans le repère lié à la position du robot à $t = 35s$. Pour pouvoir reprojeter cette carte locale et la visualiser conjointement avec la première carte (b), il faudrait tourner la carte d'un peu moins de 180 degrés.

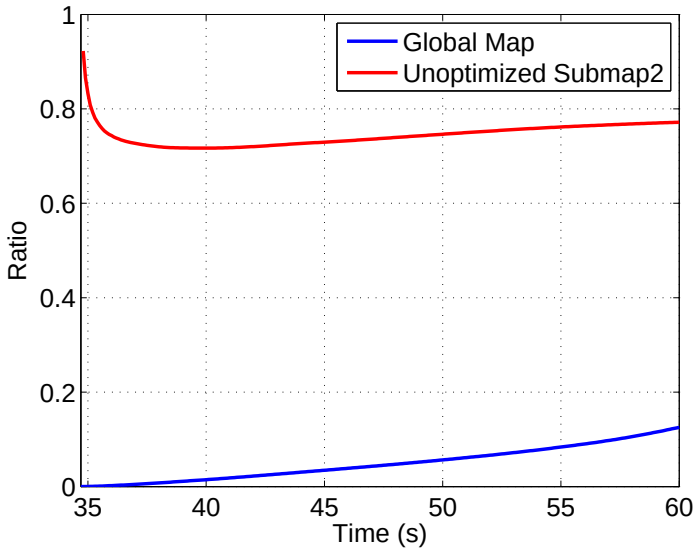


FIGURE 8.9 – Rapport entre les aires des zones d'incertitudes du robot dans deux configurations — Rouge : Résultat avec théorème / Résultat avec carte locale sans utiliser le théorème – Droite : Résultat avec théorème / Résultat avec une unique carte globale.

application du théorème.

Nous avons enfin comparé les aires des ellipses d'incertitudes associées aux positions du robot obtenues dans la seconde carte locale de la simulation précédente avec les aires des ellipses obtenues par deux autres algorithmes (figure 8.9) :

1. l'implémentation globale. Il s'agit de la courbe bleue sur la figure 8.9, représentant le rapport entre les aires des ellipses (des positions du robot) obtenues dans la carte locale en utilisant le théorème et les aires obtenues avec l'algorithme global. L'avantage revient nettement à l'algorithme local : les ellipses d'incertitudes ont une aire au moins 7.5 fois plus petite que dans le cas global,
2. l'implémentation locale n'utilisant que les mesures locales. [12] Il s'agit de la courbe rouge sur la figure 8.9, représentant le rapport entre les aires des ellipses obtenues dans la carte locale en utilisant le théorème et les aires obtenues avec l'algorithme local suboptimal. On constate que le rapport reste constamment favorable à l'algorithme exploitant l'intégralité des informations. De plus, on constate deux comportements distincts sur la courbe rouge :
   (a) Au $t = 35s$, le rapport est égal à 1 car les deux cartes locales sont initialisées avec une incertitude nulle sur la position du robot. Ensuite, le rapport diminue rapidement grâce à l'exploitation des mesures de la carte précédente par l'algorithme tirant parti de l'équation 8.3.
   (b) Dans la seconde partie, le rapport augmente. Ceci est dû au fait que les amers communs aux deux cartes ne sont plus observés. Ainsi, la position du robot tend à se décorréler des conditions initiales de la carte locale. A l'extrême limite, on devrait revenir à un rapport de 1 (correspondant au cas dégénéré que l'on a observé sur la figure 8.7b).

8.3.3 Synthèse des résultats

Au final, les résultats obtenus se sont avérés convaincants. Nous avons en effet pu montrer l'intérêt du critère permettant de construire des cartes locales en détectant avec succès des zones où la quantité d'information n'est pas suffisante pour maintenir une bonne corrélation de la carte.

Par ailleurs, l'algorithme de construction de cartes locales exploitant l'intégralité des informations depuis l'instant initial a montré de bons résultats. Dans tous les cas, la solution est améliorée par rapport à un algorithme qui n'exploiterait que les mesures à l'intérieur de la carte courante. Néanmoins, les limites se font sentir lorsque les amers visibles ne partagent pas assez d'information avec le début

12. Il s'agit du même algorithme qui a fourni les ellipses noires de la figure 8.7.

de la carte courante : l'apport dû au théorème 8.1 devient alors négligeable. Ceci est logique et ne peut être vu comme une faiblesse de l'algorithme. Au contraire, le critère de séparation de carte conduit à détecter ce type de situations.

8.4 Conclusion de la troisième partie

En conclusion, nous avons proposé dans ce chapitre une solution permettant de segmenter l'environnement en différentes cartes locales. A chaque itération, la carte locale calculée est optimale (dans le sens où elle prend en compte l'intégralité des mesures effectuées depuis l'instant initial). Ceci répondait à deux problématiques posées à la fin de la seconde partie du manuscrit :

- Réduire les temps de calculs,
- Assurer une certaine cohésion au sein de chaque carte locale. Ceci se traduit par le fait que le déplacement relatif des amers doit être le plus contraint possible.

Dans la littérature, la plupart des algorithmes ne proposaient que des solutions visant à limiter le nombre d'amers. Nous avons ici défini un critère qui permet d'assurer que chaque carte locale possède bien une forte cohésion interne. Par ailleurs, on a pu voir que la plupart des approches ne proposaient que des cartes locales suboptimales (les cartes locales n'intégrant en général pas les nœuds frontière, voir chapitre 7) ; notre approche permet quant à elle d'intégrer la totalité des mesures passées dans chaque carte locale.

Néanmoins, l'approche décrite dans le paragraphe 7.2.3 page 251 ([Piniés et Tardós, 2008]) semble très proche de notre façon de construire les cartes locales. Les deux algorithmes utilisent en effet les propriétés déduites du conditionnement par rapport à la dernière pose de la précédente carte. Un examen attentif des deux méthodes permettrait de montrer que les calculs réalisés sont en fait pratiquement identiques, la différence se situant dans l'algorithme de filtrage utilisé (nous avions développé notre solution indépendamment des travaux de [Piniés et Tardós, 2008]). Néanmoins, les démonstrations présentées dans [Piniés et Tardós, 2008] ne sont pas complètes quant à la justfication de la création des cartes locales. Les développements sont surtout focalisés sur le traitement des fermetures de boucle et l'algorithme de rétropropagation. Au contraire, nous proposons dans ce manuscrit une démonstration formelle et complète du résultat permettant de construire une nouvelle carte. La factorisation obtenue montre que si la précédente carte est consistante et que l'on est capable d'effectuer une estimation purement locale consistante, [13] alors le résultat final associé à la carte courante

13. ie. une estimation de $p\left(\mathbf{x}_{t_1+1:t_2}, \mathbf{m}^2, \mathbf{m}^3 | \mathbf{x}_{t_1}, \mathbf{z}_{t_1+1:t_2} \mathbf{u}_{t_1+1:t_2}\right)$ consistante.

sera assurément consistant (l'équation 8.3 ne fait appel à aucune approximation ni aucune hypothèse sur les densités de probabilité). L'utilisation de l'algorithme de SAM permet d'obtenir les conditions de consistances nécessaires.

Le principal défaut de notre approche est qu'elle ne prend pour l'instant pas en compte la rétropropagation. Ainsi, si la carte courante est toujours optimale, il n'en est pas de même pour les cartes précédentes dont les calculs ont été arrêtés et ne prennent pas en compte les dernières mesures. Sauf en cas de fermeture de boucle, cette rétropropagation n'est intéressante qu'en post-traitements. Une première continuité de ces travaux serait de formaliser l'algorithme de rétropropagation des informations. On s'attend néanmoins à trouver des résultats identiques à ceux présentés dans [Piniés et Tardós, 2008], après adaptation au cas du SAM (au lieu de l'EKF).

On pourra enfin remarquer que le résultat de la segmentation n'est pas uniquement intéressant pour faciliter l'algorithme de filtrage et la résolution du SAM. En effet, l'utilisation du critère permet de garantir que chaque carte locale est " suffisamment " contrainte. Ainsi, le résultat peut être utilisé par un opérateur humain ou un autre algorithme qui pourra effectuer des calculs critiques consistants à l'échelle de la carte locale (on pourra par exemple définir une trajectoire de planification de manière sûre, ce qui ne serait pas possible à l'échelle de la carte globale). Dans la quatrième et dernière partie du manuscrit, nous proposons d'utiliser les propriétés de cette segmentation pour détecter les plans dans les cartes locales. La segmentation de nuages de points en primitives de plus haut niveau nécessite en effet que les points considérés bénéficient de bonnes estimations relatives. Notre critère a précisemment été défini dans ce but. Ainsi, il paraît cohérent de ne chercher à segmenter que carte locale par carte locale.

Quatrième partie

Utilisation du SLAM : recherche d'information haut niveau

Chapitre 9

Détection automatique des plans

Sommaire

Nous présentons dans la quatrième et dernière partie de ce manuscrit deux applications de l'algorithme de SLAM permettant de calculer en post-traitements des caractéristiques haut-niveau de l'environement. Dans les deux cas, l'idée est d'utiliser le résultat du SLAM afin d'effectuer un traitement le plus **automatique** possible. Nous présentons dans le présent chapitre une première application, à savoir la segmentation de plans. Nous présenterons dans le chapitre 10 une méthode permettant d'estimer l'espace libre pour le robot.

Dans ce chapitre, nous traitons le problème de la détection de plans dans le nuage de points constituant la carte du SLAM. L'idée est de trouver automatiquement les plans présents dans la scène afin de les reconstruire *a posteriori*, en les augmentant de textures. Nous décrivons dans un premier temps la méthode utilisée pour segmenter un nuage de points déterministe en différents plans. Ensuite, nous montrons dans la section 9.2 l'intérêt de l'utilisation des cartes locales vues au chapitre précédent. La section 9.3 est quant à elle consacrée à l'étude des résultats de segmentation dans le cas de la trajectoire du bâtiment Borel (introduite au chapitre 5) ; il s'agit d'une expérimentation contenant beaucoup de posters plans très texturés et qui se prête bien à la segmentation. Ces résultats seront améliorés par un plaquage de texture exploitant l'intégralité des positions depuis lesquelles le plan a été vu (section 9.4). Nous concluons enfin ce chapitre par une synthèse des résultats obtenus et quelques perspectives.

9.1 Détection de plans à partir d'un nuage de points

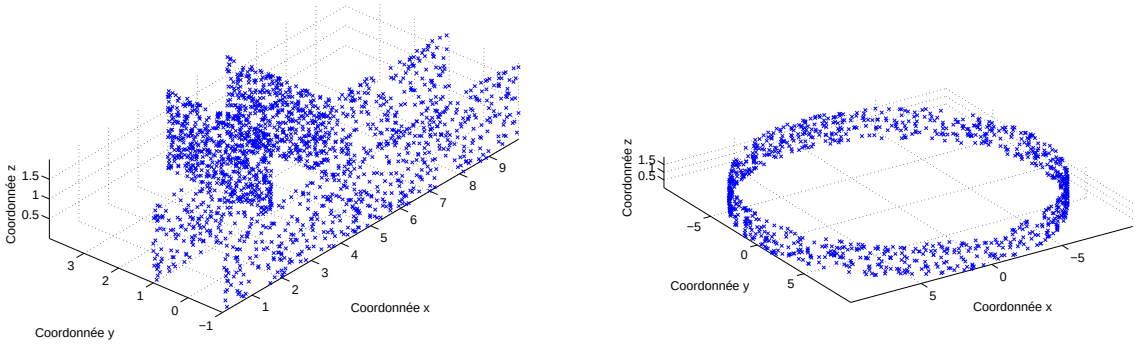
9.1.1 Introduction

Nous présentons dans cette section l'algorithme utilisé en vue de segmenter les plans dans un nuage de points supposé déterministe. La validité de cette hypothèse sera discutée dans la section suivante.

L'approche utilisée se fait en 3 étapes. Dans un premier temps, une triangulation de Delaunay est effectuée (paragraphe 9.1.2). Le résultat de cette triangulation permet de définir des " plans élémentaires " (3 points) dont on peut calculer la normale. Ces plans élémentaires sont regroupés ou non avec leur voisins en fonction de leur normale (paragraphe 9.1.3). Nous verrons alors que le résultat obtenu est beaucoup trop segmenté. Un processus de fusion sera alors utilisé (paragraphe 9.1.4).

Dans toute cette section, nous allons utiliser deux exemples simulés afin d'illustrer les 3 étapes de segmentation (figure 9.1), à savoir :

1. un couloir en " T " dans lequel ne figurent que des plans (figure 9.1a). Il s'agit ici de voir si on est capable de détecter effectivement les 5 plans qui constituent la scène,



(a) (b)

FIGURE 9.1 – Nuages de points simulés pour la détection de plans

2. un cylindre (figure 9.1b). Ici, il n'y a pas de plan à détecter en réalité. Néanmoins, de la même façon qu'il est possible d'approcher un cercle par un polygone régulier avec suffisament d'arrêtes, il est possible d'approcher un cylindre par une série de plans. L'intérêt de cet exemple est de vérifier la capacité de l'algorithme à accepter de petites déformations des zones qui ne sont pas tout à fait planes tout en empêchant l'excès inverse (qui serait de considérer comme un plan un demi-cylindre complet). Nous verrons particulièrement l'intérêt de cet exemple au paragraphe 9.1.4.

Pour ces deux exemples, les points considérés ne respectent pas exactement les primitives décrites : nous avons introduit un bruit afin de tester la robustesse de notre approche.

9.1.2 Triangulation de Delaunay

La première étape de l'algorithme consiste à définir un maillage du nuage de points. Le but est d'obtenir des facettes élémentaires pour lesquelles il est possible de calculer un vecteur normal. La comparaison entre facettes adjacentes de ce vecteur normal permettra ensuite de choisir si l'on regroupe ou non les facettes dans un même plan. La triangulation de Delaunay permet de créer de telles facettes.

Dans le cas 2D, la triangulation de Delaunay permet de définir des **triangles** dont **aucun autre point n'est à l'intérieur de leur cercle circonscrit**. C'est ce que l'on voit sur la figure 9.2 (les points à trianguler sont en bleu, la triangulation est présentée en noir et le cercle circonscrit des triangles obtenus sont en rouge). Par ailleurs, si il n'y a pas de points cocycliques ou alignés, la triangulation est unique.

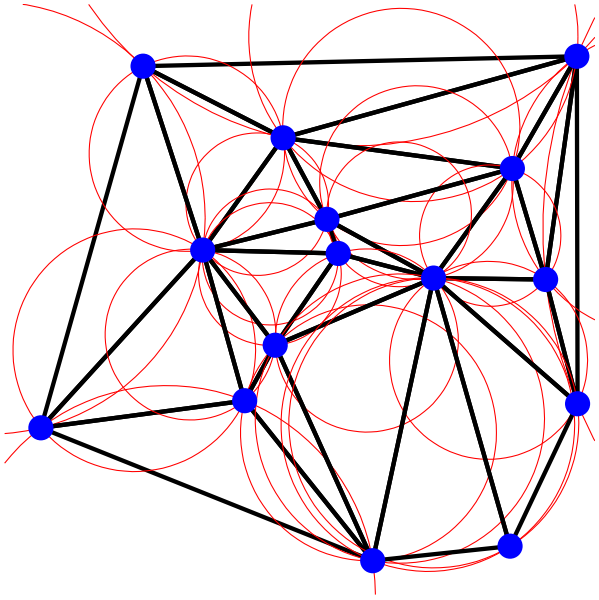


FIGURE 9.2 – Exemple de triangulation de Delaunay dans le cas 2D

Dans le cas 3D, la triangulation de Delaynay permet de définir des **tétraèdres** dont **aucun autre point n'est à l'intérieur de la sphère circonscrite**. La triangulation sera unique si il n'y a pas des groupes de points coplanaires ou cocycliques. La propriété de la sphère circonscrite permet d'assurer qu'il n'y a aucun croisement de tétraèdres. Elle permet donc de définir un maillage cohérent pour des études locales. Nous avons néanmoins modifié légèrement ce maillage afin de supprimer les triangles **trop grands** par rapport à la taille du robot. Le but de cette opération est d'éviter de fusionner des plans alignés entre lesquels le robot pourrait passer (dans le cas du couloir en T, cela évite de prolonger le couloir principal).

La première étape de notre algorithme de segmentation consiste donc à calculer la triangulation de Delaunay 3D du nuage de points. Le résultat obtenu est une série de tétraèdres. Nous allons traiter dans la suite chacune des faces de chaque tétraèdre comme un plan élémentaire dont on testera la cohérence avec ses voisins.

9.1.3 Première segmentation

La seconde étape de l'algorithme consiste à effectuer une première segmentation très sélective. L'idée est d'agglomérer une par une les facettes adjacentes lorsqu'elles semblent appartenir à un même plan. Pour cela, nous comparons le produit scalaire de leur normale. Le principal défaut de ce type

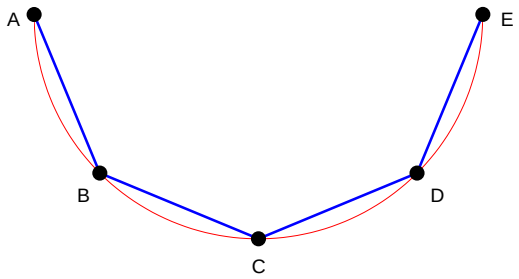
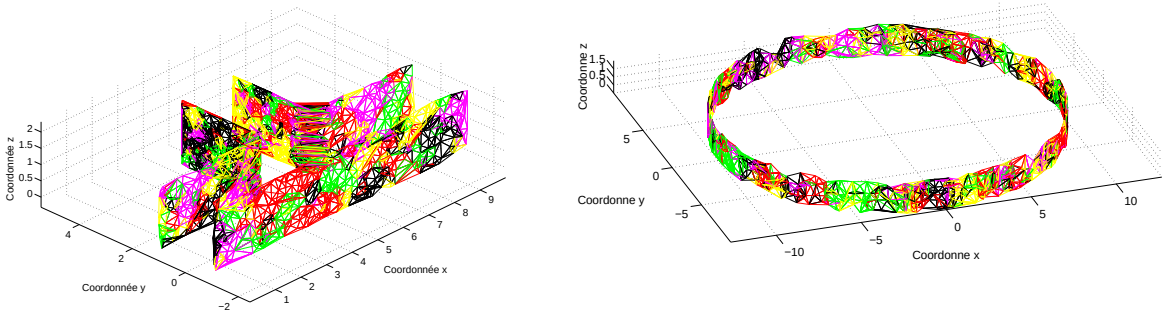


FIGURE 9.3 – Exemple de points sur un demi-cercle

d'algorithme est que les facettes sont en général très petites, et que des facettes qui peuvent sembler cohérentes de proches en proches peuvent ne pas l'être d'un point de vue global. C'est ce que montre la figure 9.3 dans le cas 2D. Imaginons en effet que l'on souhaite fusionner les points alignés. Admettons que les segments $[AB]$ et $[BC]$ satisfont une condition de normale et que les 3 points A, B et C peuvent être considérés comme alignés. Il en sera alors de même pour les segments $[BC]$ et $[CD]$, puis pour $[CD]$ et $[DE]$. Néanmoins, on ne pourra pas considérer pour autant que les vecteurs $\overrightarrow{AB}$ et $\overrightarrow{DE}$ sont colinéaires. Ainsi, il serait dangereux d'effectuer des comparaisons de proches en proches par propagation.

Pour éviter le phénomène que nous venons de décrire, nous allons systématiquement comparer les normales des facettes avec la même facette de référence. L'algorithme de sélection consiste donc à sélectionner une facette de référence, puis à comparer son vecteur normal avec toutes les autres facettes adjacentes. Celles dont le produit scalaire est supérieur à un seuil donné sont agglomérées. Ensuite, on fait entrer dans le test les facettes adjacentes aux dernière facettes acceptées, puis on refait le test. Pour reprendre l'exemple de la figure 9.3, cela signifie que l'on conserve comme référence l'orientation du vecteur $\overrightarrow{AB}$. Le test serait alors valide avec $\overrightarrow{BC}$, mais pas avec $\overrightarrow{CD}$ ni $\overrightarrow{DE}$.

Le fait de ne comparer la normale qu'avec la même facette de référence permet alors de ne conserver que les facettes très cohérentes avec la facette initiale. En contrepartie, cette méthode empêche tout effet de filtrage ou de moyennage de la solution lors de l'ajout de facettes. La conséquence immédiate est une segmentation excessive du nuage de points (figure 9.4).



(a) (b)

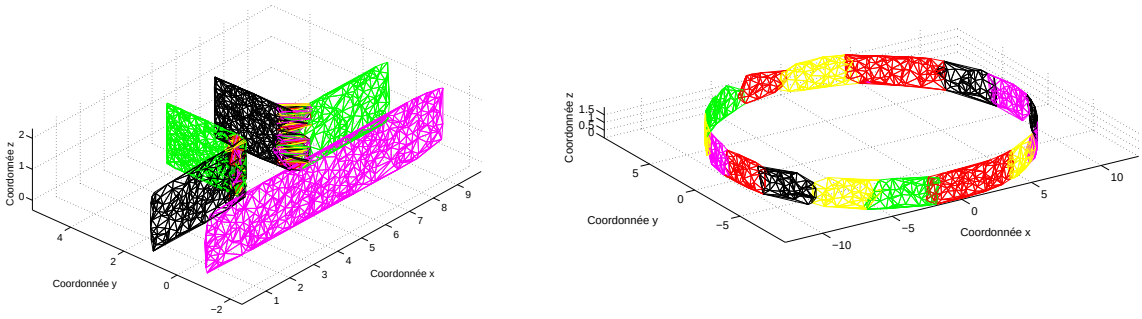
FIGURE 9.4 – Résultats issus de la première segmentation. Les changement de couleur indiquent des plans différents.

9.1.4 Fusion des plans

Le résultat obtenu dans le paragraphe précédent n'est pas satisfaisant, et ce en raison de la segmentation excessive de certains plans pourtant évidents. Nous avons donc ajouté à l'algorithme une phase de fusion des plans. L'idée est de comparer les normales des plans partageant des points en commun. Si leur direction est suffisament proche, les plans sont fusionnés. Ensuite, on introduit le nouveau plan issu de la fusion dans la liste des plans et on enlève les anciens. On réitère le processus de fusion sur l'ensemble des plans (y compris le nouveau plan), et ce jusqu'à ce qu'il n'y ait plus de fusion possible.

En comparaison avec la phase précédente, la " brique de base " n'est plus un triangle élémentaire, mais un morceau de plan dont la cohérence est forte (le test de la phase précédente étant très contraignant). Cette fusion de plan permet de lisser le résultat obtenu avec les triangles élémentaires. En effet, le calcul de la normale est plus stable lorsque l'on a déjà un certain nombre de points dans le plan (alors que l'on a que 3 points dans les triangles élémentaires). Par ailleurs, le fait d'utiliser des plans dont le nombre de points est déjà conséquent permet de limiter le risque de dérive présenté au paragraphe précédent (cf figure 9.3) (lorsque l'on fusionne deux plans, la normale résultante ne varie pratiquement pas).

Les résultats obtenus après cette étape sont visibles sur la figure 9.5. On voit bien dans le cas du couloir en T que les plans principaux ont été intégralement fusionnés. Il subsiste quelques plans parasites dans les coins. Ceux-ci peuvent éventuellement être automatiquement filtré si l'on ajoute un critère ne demandant de conserver que les plans issus de la première segmentation contenant un



(a) (b)

FIGURE 9.5 – Résultats après fusion des plans issus de la première segmentation. Les changements de couleur indiquent des plans différents.

certain nombre de points. Dans le cas du cylindre, on obtient une segmentation cohérente : les plans obtenus conservent de bonnes propriétés de planéité (le phénomène de dérive qui pouvait avoir lieu a bien été évité).

9.1.5 Conclusion

En conclusion, nous avons proposé dans cette section un algorithme permettant de détecter automatiquement les sous-ensembles de points correspondant à des zones planaires au sein d'un nuage de points. Celui-ci est résumé dans l'algorithme 9.1. Notre méthode a donné de bons résultats sur des données simulées. Nous n'avons pas approfondi davantage le développement de cet algorithme. Les résultats présentés dans ce chapitre étant des résultats préliminaires, nous n'avons pas cherché à obtenir l'outil de segmentation le plus robuste et optimal possible, mais un outil pouvant être implémenté rapidement et donnant des résultats acceptables. Une étude approfondie de la bibliographie sur ce sujet permettrait certainement d'améliorer les résultats précédents.

9.2 Utilisation des propriétés de corrélation

Nous avons présenté dans la section précédente un algorithme permettant de segmenter les régions planes dans un nuage de points. Nous allons utiliser directement cet algorithme sur le résultat du SLAM. Nénanmoins, l'algorithme utilisé fait l'hypothèse d'un nuage de points déterministe et ne prend pas en compte la nature probabiliste de la carte du SLAM. Ceci étant, il est nécessaire de s'assurer que les nuages de points considérés sont bien consistants.

Données : $Points$ # Nuage de points initial
Fixer $d, seuil$ # d : distance maxi pour les arrêtes
dans la triangulatrion de Delaunay
$seuil$: seuil pour acceptation des plans
$Triangles \leftarrow$ **delaunay**$(Points)$ # Paramètres initiaux pour la carte locale
$Triangles \leftarrow$ **elimineGrandTriangles**$(Triangles, d)$ # Elimination des triangles élémentaires
trop grands
$Triangles\_seuls \leftarrow Triangles$ # $Triangles\_seuls$ contient les triangles qui
n'ont pas été agglomérés
$plans \leftarrow [\,]$ # Ensemble des plans détectés
Pour tout $tri \in Triangles\_seuls$ **faire**
| $[p_n, tri_u] \leftarrow$ **agglo_triangles**$(tri, Triangles\_seuls)$ # p_n : nouveau plan
| # tri_u : triangles utilisés pour p_n
| $plans \leftarrow plans \bigcup \{p_n\}$ # On ajoute p_n aux plans
| $Triangles\_seuls \leftarrow Triangles\_seuls \setminus tri_u$ # On retire tri_u de $Triangles\_seuls$
Fin pour # Fin de la première segmentation
$fin \leftarrow 0$
Tant que $fin = 0$ **faire**
| $plans_{nouveaux} \leftarrow [\,]$
| **Pour tout** $p \in plans$ **faire**
| | $[p_n, p_u] \leftarrow$ **fusion_plans**$(p, plans)$ # p_n : nouveau plan (vide si pas de fusion)
| | # p_u : plans utilisés pour créer p_n
| | $plans_{nouveaux} \leftarrow plans_{nouveaux} \bigcup \{p_n\}$ # On ajoute p_n aux plans nouveaux
| | $plans \leftarrow plans \setminus p_u$ # On retire p_u de $plans$
| **Fin pour**
| $plans \leftarrow plans \bigcup plans_{nouveaux}$ # On garde les anciens plans qui n'ont pas été
modifié et on ajoute les nouveaux
| **Si** $plans_{nouveaux} = [\,]$ **alors** # Pas de nouvelle fusion $\Longrightarrow$ convergence
| | $fin \leftarrow 1$
| **Fin si**
Fin faire # Fin de la fusion des plans
Retourner $plans$

Algorithme 9.1 – Extraction des plans dans un nuage de points

L'approche par cartes locales présentée au chapitre 8 est parfaitement adaptée dans ce but. Le critère que nous avons présenté permet en effet de segmenter la totalité de la carte en zones corrélées. Ainsi, le déplacement relatif entre les points d'une carte locale est bien connu, ce qui n'est pas le cas lorsqu'on garde la carte globale. Ceci permet d'éviter de fusionner des plans dont le positionnement relatif n'est pas suffisament bien estimé.

Enfin, l'utilisation de cartes locales permet de réduire la charge de calcul, notamment pour le calcul de la triangulation de Delaunay.

9.3 Résultats issus de la séquence Borel

Nous présentons ici les résultats de segmentation obtenus dans le cas de la séquence Borel (introduite au chapitre 5 page 173). Pour rappel, il s'agit d'une séquence obtenue en déplaçant le robot dans des couloirs sur lesquels étaient accrochés divers posters et affiches. Ceux-ci permettent l'extraction de nombreux points d'intérêts : on s'attend donc à trouver de nombreuses zones planes dans cette séquence.

La segmentation a été effectuée **par carte locale** (en utilisant les résultats obtenus au chapitre 8). Les résultats de segmentation sont visibles sur la figure 9.6 : il s'agit des résultats obtenus dans la sixième carte locale. On peut remarquer que l'on détecte bien une partie des plans de la carte. Néanmoins, on détecte également des plans indésirables en plus des plans principaux. Ceci est dû à la présence de quelques points de mauvaise qualité dans les données. C'est ce que l'on voit dans la zone entourée sur la figure 9.6d : un des points est en retrait du mur et provoque la création de nombreux plans. On constate également dans la zone inférieure gauche de la figure 9.6d une série de plans due ici à la présence d'une porte.

Néanmoins, on constate que les plans " parasites " contiennent un nombre de points très inférieur à celui des plans principaux (figure 9.7). Ainsi, de nombreux plans détectés ont un nombre de points inférieur à 5 (ce qui est très faible). Il s'agit dans ce cas de petits regroupements de points qu'il convient de ne pas considérer. Par ailleurs, peu de plans ont un nombre de points réellement significatif. En ne gardant que les plans dont le nombre de points est supérieur à 30, on obtient le résultat de segmentation présenté sur la figure 9.8. Le résultat est alors nettement amélioré, sans toutefois être parfait (le grand plan rouge aurait pu être fusionné avec un plan plus petit représenté en jaune).

En conséquence, malgré les quelques défauts constatés, ces premiers résultats sont encourageants et montrent la pertinence de l'approche. On est en effet capable de retrouver les principaux plans

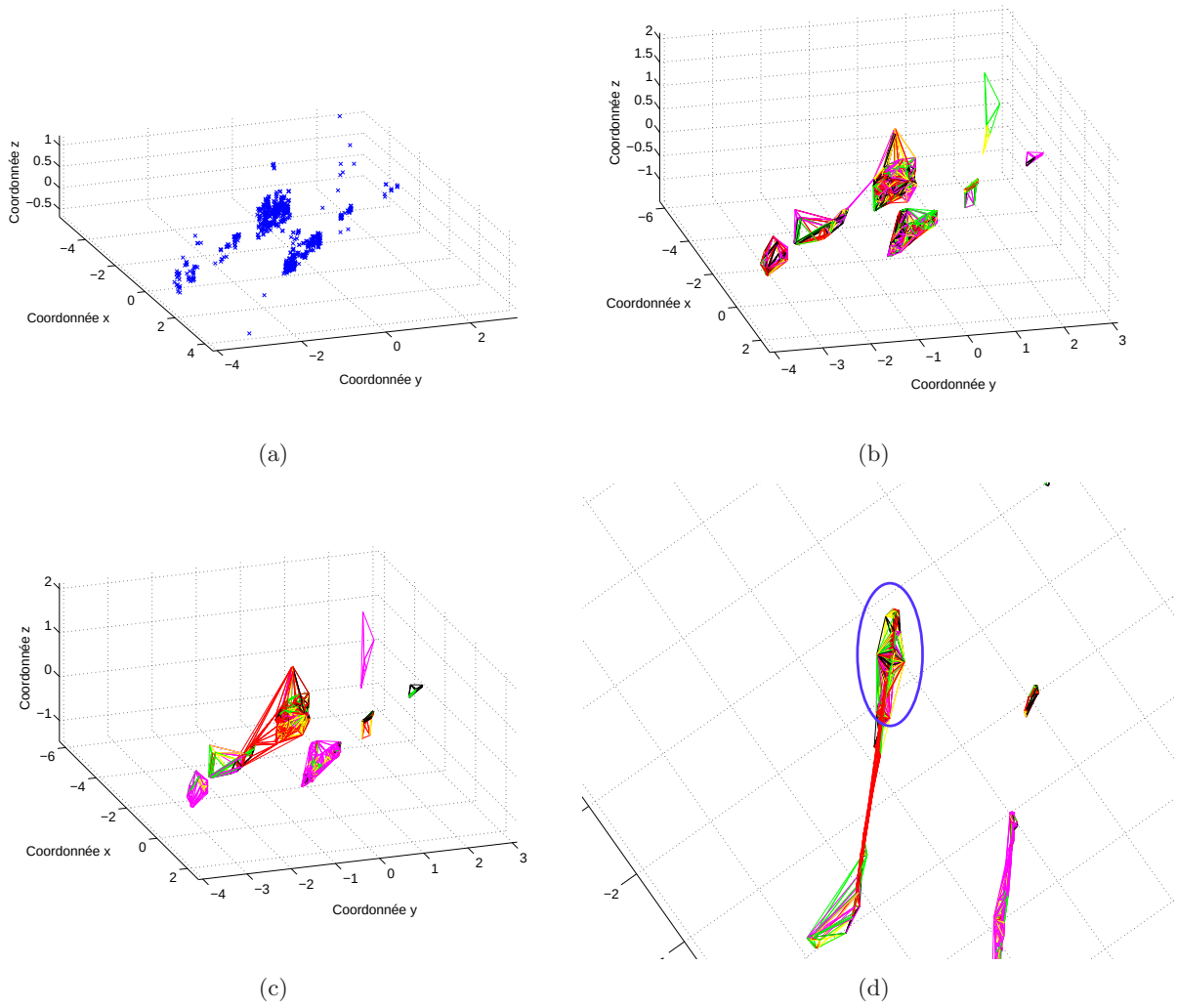


FIGURE 9.6 – Résultat de segmentation sur la sixième carte locale — (a) Nuage de points initial – (b) Première segmentation – (c) Résultat après fusion des plans – (d) Visualisation d'un des défauts

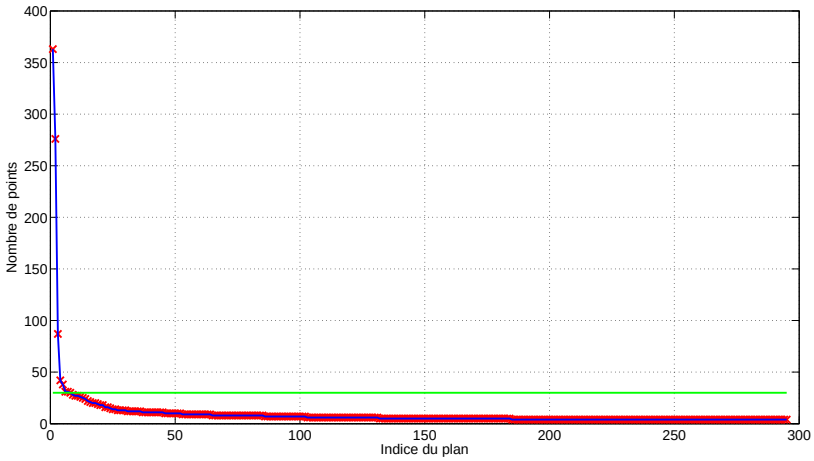


FIGURE 9.7 – Nombre de points dans chaque plan — On constate que seuls peu de plans sont réellement significatifs.

des nuages de points issus du SLAM (après élagage des plans non significatifs). L'approche mérite néanmoins d'être améliorée, et ce afin de rejeter directement les quelques points aberrants provoquant la détection de plans parasites.

9.4 Fusion des images pour la projection de textures

Les résultats obtenus dans la section précédente ne sont qu'une étape dans l'optique de la création d'une représentation haut niveau des plans. Nous proposons dans cette section d'utiliser les précédents résultats de sorte à obtenir une **représentation texturée des plans**. Pour obtenir ces résultats, nous feront également appel aux images originales ainsi qu'au résultat complet du SLAM (carte locale par carte locale).

Cette section s'articule en deux parties. Nous présentons d'abord la méthodologie employée pour texturer les plans. Nous présentons ensuites les résultats obtenus.

9.4.1 Description de la méthode

9.4.1.1 Choix de la région à texturer

Supposons que nous avons un nuage de points pratiquement coplanaires. Pour obtenir la zone à texturer, nous calculons dans un premier temps l'équation du plan associé aux points. L'ensemble du nuage est ensuite projeté sur le plan. La zone à texturer est finalement déterminée par l'enveloppe convexe du nuage de points projeté.

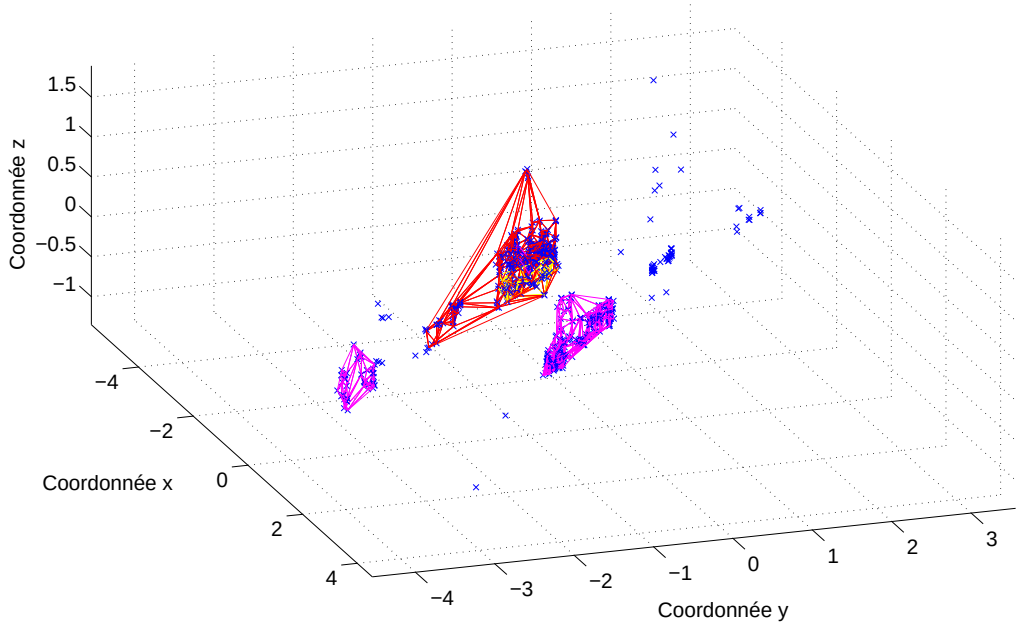


FIGURE 9.8 – Segmentation de la sixième carte locale après élagage des plans contenant moins de 30 points

La zone à texturer est par définition une zone continue. Or, nous souhaitons définir une texture, c'est à dire une image. Il est donc nécessaire de discrétiser la zone à texturer. La finesse de la discrétisation déterminera la précision de la texture. Néanmoins, il est inutile d'utiliser un pas de discrétisation trop petit car la texture est créée à partir d'une image (qui est elle-même constituée d'un nombre fini de pixels). Ainsi, même si on peut effectuer une interpolation subpixelique de l'image originale, la qualité de la texture finale atteindra un seuil à partir d'un pas de discrétisation.

Le plan à texturer est vu depuis plusieurs vues. Il est possible que seule une partie du plan soit visible à un instant donné. Il serait donc inaproprié d'essayer de texturer l'intégralité du plan à chaque instant. [1] La stratégie que nous avons décidé d'employer consiste à ne traiter que l'enveloppe convexe associée aux **points d'intérêts vus à l'instant traité**.

9.4.1.2 Création d'une texture à partir d'une seule vue

Supposons que l'on connaisse la zone à texturer à l'instant t. La position de l'ensemble des " pixels " de cette zone est connue dans l'espace (par l'équation du plan). Par ailleurs, la position ainsi que

1. Dans le cas où une partie du plan est occultée par un objet mobile, on risquerait d'ajouter des textures incohérentes en tentant de texturer tout le plan.

l'orientation de la caméra sont également connues (résultat du SLAM). En conséquence, il est possible d'exprimer les coordonnées de l'ensemble des pixels à texturer dans le repère de la caméra. Nous n'avons alors qu'à appliquer le modèle de projection associé à notre caméra pour convertir les coordonnées dans le repère de la caméra en coordonnées dans l'image. [2] Finalement, il n'y a plus qu'à " lire " la valeur de niveau de gris du pixel dans l'image. [3]

9.4.1.3 Fusion de toutes les vues

Nous avons présenté au paragraphe précédent comment résoudre la problème de la texturation pour une position donnée (et donc avec une seule image). Dans notre cas, nous utilisons plusieurs vues du plan. Il est donc nécessaire de définir une stratégie pour fusionner les différentes solutions apportées par les différentes images.

A titre d'exemple, considérons le cas de la figure 9.9. Supposons que l'on a détecté le plan associé à l'ensemble des affiches visibles sur la droite des images et que l'on souhaite texturer ce plan à partir des deux images présentées sur la figure. On constate que l'intégralité des affiches est visible dans les deux images. Néanmoins, il apparaît clair qu'une partie des affiches sera mieux reconstruite avec la première image alors que la seconde partie sera mieux reconstruite grâce à la seconde image. Pour déterminer clairement quelle image utiliser dans quelle zone du plan, nous proposons de calculer un critère pour chaque pixel du plan à texturer. Soit p^i le $i^{\text{ème}}$ pixel du plan à texturer. [4] Ce pixel se projette en p_{j_1} dans la vue 1 et en p_{j_2} dans la vue 2. Soient désormais $n(p_{j_1})$ (resp. $n(p_{j_2})$) le nombre de points p^k du plan à texturer qui se projettent en (p_{j_1}) (resp. (p_{j_2})). La valeur de $n(p_{j_1})$ reflète la qualité d'information que le pixel p_{j_1} apporte sur p^j. Au mieux, ce nombre vaut 1 : le pixel p_{j_1} n'est associé qu'à p^j, la correspondance est parfaite. Au pire, $n(p_{j_1})$ est très grand : cela signifie que le pixel p_{j_1} intercepte une zone très diffuse du plan à texturer. Dans ce dernier cas, le niveau de gris du pixel (p_{j_1}) de la vue 1 n'est pas très représentatif du niveau de gris réel de p^j (c'est ce qui se passe lorsque la zone à texturer est loin de la caméra et/ou vue avec un angle très fermé). On peut donc finalement définir une stratégie pour choisir quelle vue utiliser :

1. si $n(p_{j_1}) < n(p_{j_2})$, alors on associe à p^j le niveau de gris de p_{j_1} dans la vue 1,

2. Voir chapitre 4 page 164 pour la description du modèle unifié pour la caméra omnidirectionnelle.

3. Il existe plusieurs stratégies de " lecture ". La projection dans l'image ne donne jamais une valeur entière en pixels (alors que l'on a des valeurs de niveau de gris que sur des valeurs entières). On peut par exemple effectuer une interpolation linéaire voire bilinéaire de l'image. Dans notre cas, nous avons directement arrondi avec la partie entière. En pratique, cela ne change pas beaucoup le rendu.

4. Dans la suite, nous indexons avec un exposant les pixels du plan à texturer, et avec un indice les pixels issus de l'image originale.

2. si $n(p_{j_1}) > n(p_{j_2})$, alors on associe à p^j le niveau de gris de p_{j_2} dans la vue 2.

Dans le cas général, lorsqu'un pixel p^j du plan à texturer est vu depuis plusieurs positions de la caméra, le résultat dépend des $n(p_{j_k})$. Nous avons retenu deux solutions de fusion :

- Prendre en compte l'intégralité des vues en effectuant une moyenne pondérée par les $n(p_{j_k})$. Cette solution permet de lisser le résultat. En pratique, nous n'avons pas trouvé de façon satisfaisante de pondérer. Le rendu final était systématiquement trop flou,
- Ne conserver que la vue associée à la plus petite valeur des $n(p_{j_k})$. Cette méthode peut être vue comme un filtrage avec un noyau discontinu (0 ou 1). En pratique, nous avons constaté que le rendu global est plus précis que dans le cas précédent, mais il est en contrepartie plus bruité. Cette méthode nous a donné des résultats plus satisfaisants que la précédente. **Nous la conservons pour la suite**.

9.4.2 Résultats obtenus

Les résultats obtenus sont présentés sur les figures 9.10 et 9.11. Dans tous les cas, nous avons utilisé la segmentation obtenue avec la méthode décrite à la section précédente, en ne conservant que les plans les plus significatifs. La figure 9.10 présente quelques exemples de textures obtenues. On constate que toutes les images obtenues sont nettes et relativement bien détaillées : les résultats sont très encourageants. La figure 9.10c correspond à la partie de la trajectoire montrée sur la figure 9.9. On voit alors que le résultat est de bonne qualité sur toute la texture, alors qu'on a très bien vu que les affiches ne peuvent pas être vues nettes toutes simultanément. Ceci montre la pertinence de notre algorithme de fusion.

On peut néanmoins noter quelques défauts dans les résultats obtenus. Le défaut le plus récurrent est la présence d'un léger décalage qui survient dans certaines projections. Ceci est bien visible sur la figure 9.10c : la partie inférieure du tableau panneau contenant les affiches possède une discontinuité flagrante. Ceci est dû au fait que le résultat du SLAM n'est pas parfait, ce qui peut fausser légèrement la projection. Néanmoins, nous pensons que cet aspect peut facilement être corrigé en effectuant une phase de minimisation dense de type ESM [5] ([Benhimane et Malis, 2004; Malis, 2004; Benhimane et Malis, 2006; Benhimane, 2007]). En effet, les décalages constatés étant très faibles, ils sont compatibles avec les contraintes d'initialisation et de linéarisation des méthodes de suivi dense. Bien que nous n'avons pas implémenté cette fusion de méthodes, nous estimons qu'elle permettra d'améliorer la qualité des

5. Efficient Second Order Minimisation : il s'agit d'un algorithme de suivi de plan dans les images.

(a)

(b)

FIGURE 9.9 – Images omnidirectionnelles dans lesquelles figurent des affiches (entourées en rouge). L'ensemble des affiches est visible sur les deux images, mais certains sont plus détaillées sur la première, et d'autre sont plus détaillées sur la seconde.

résultats pour une surchage minime de calculs.

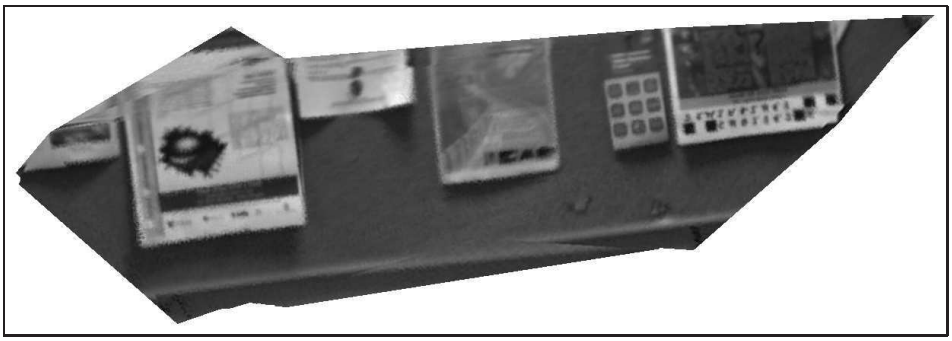
Enfin, la figure 9.11 présente à titre d'illustration les plans texturés en 3D et dans les cartes locales originales. L'ajout de plans texturés permet d'améliorer la lecture des différentes cartes locales. Néanmoins, les résultats actuels ne sont pas suffisant pour une exploitation concrète des résultats : la représentation reste trop éparse.

9.5 Conclusion

Nous avons présenté dans ce chapitre les premier résultats obtenus quant à la recherche et représentation de plans à partir des résultats du SLAM. Nous avons montré qu'il était possible de retrouver les principaux plans présents dans les nuages de points issus du SLAM. Nous avons également mis en évidence l'intérêt des cartes locales : on ne traite que des données qui sont très cohérentes entre elles (et dont les poses relatives peuvent être considérées comme déterministes en première approximation).

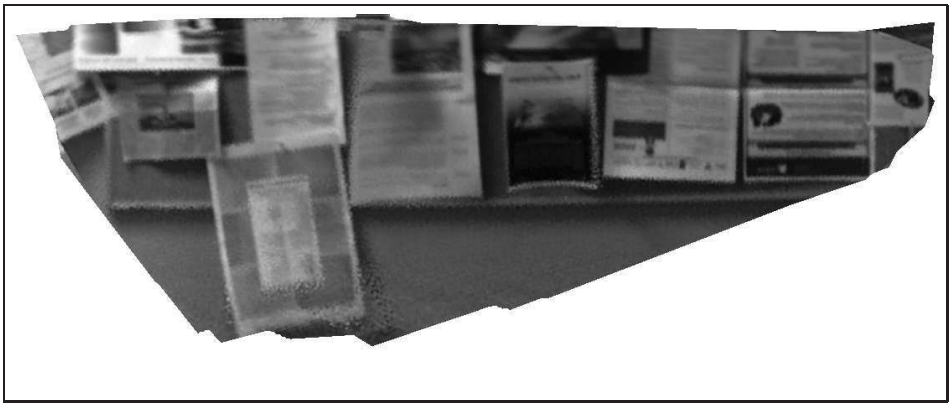
L'intégralité des traitements effectués ici peut être faite de manière automatique. La force de cette méthode est qu'aucun opérateur n'a à intervenir pour initialiser un plan.

Les résultats obtenus sont tous à fait satisfaisants, alors qu'aucun travail d'optimisation n'a été effectué. Les résultats pourraient être encore largement améliorés. On pourrait notamment supprimer plus efficacement l'ensemble des plans parasites au moment de la segmentation. Enfin, la qualité des textures peut encore être améliorée. Dans un premier temps, la fusion avec une méthode d'optimisation directe telle l'algorithme ESM paraît inévitable. Ensuite, on pourrait affiner la méthode de fusion en recherchant des pondérations plus fines et plus optimales que le choix " brutal " effectué dans la version actuelle de l'algorithme.



(a)

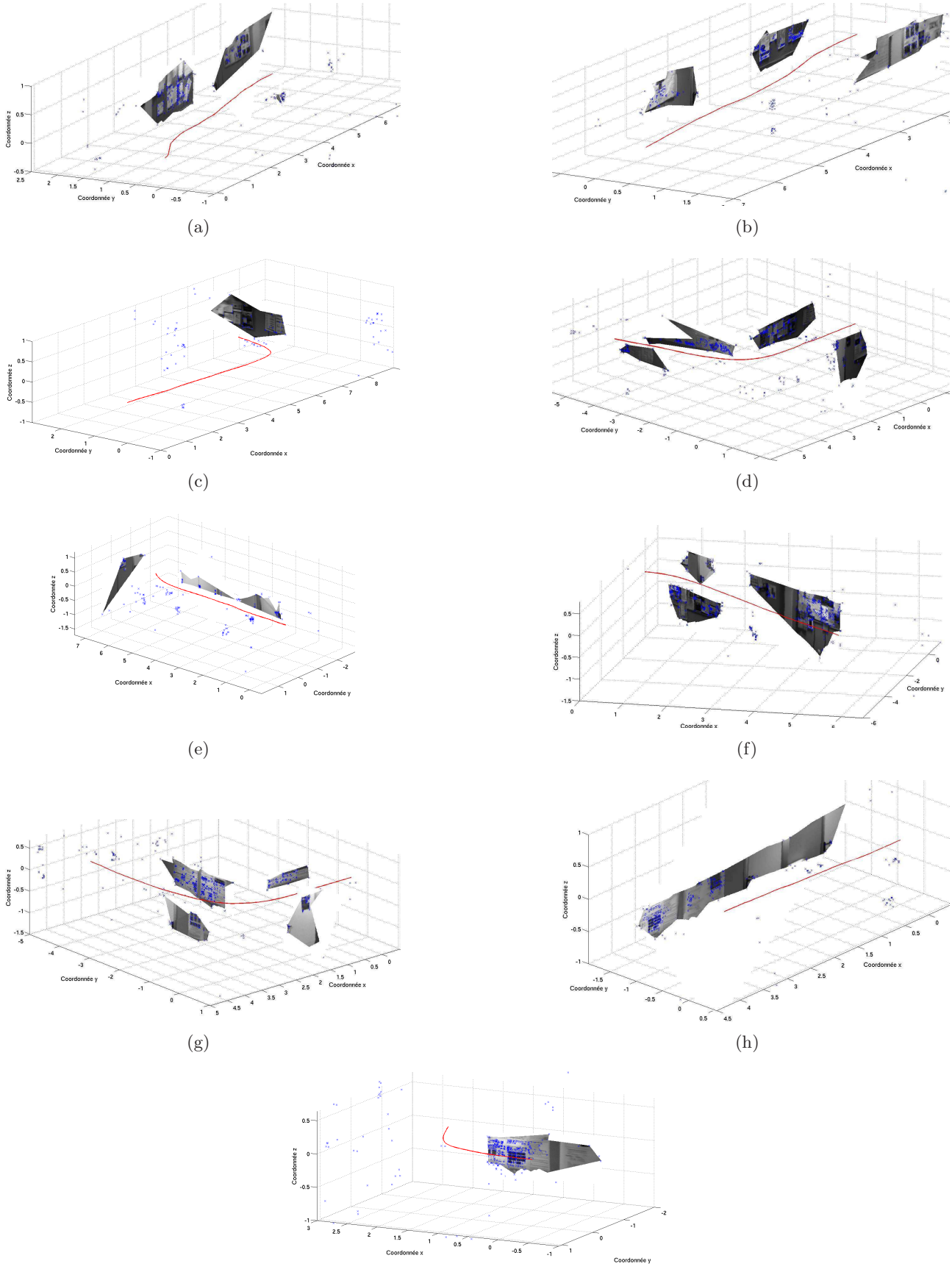
(b)



(c)

(d)

FIGURE 9.10 – Textures tirées des plans dans les cartes locales 2, 4, 6 et 7



(a) (b) (c) (d) (e) (f) (g) (h) (i)

FIGURE 9.11 – Projection des plans texturés dans l'espace — (a–i) Cartes locales 1 à 9

Chapitre 10

Détermination de l'espace libre du robot

Sommaire

Nous présentons dans ce dernier chapitre une deuxième utilisation des résultats du SLAM. Nous allons ici essayer de déduire l'espace accessible par le robot de l'ensemble de la carte. Déterminer l'espace libre de manière consistante est une étape importante dans une optique de contrôle automatique du robot : savoir où sont les obstacles et où le robot peut se déplacer est en effet une étape indispensable pour les algorithmes de planification de trajectoires. Pour cela, nous allons une nouvelle fois utiliser la triangulation de Delaunay sur le résultat du SLAM (trajectoire et amers). L'idée sera ici de " creuser " l'ensemble de la triangulation en fonction des élements observés. Les méthodes proposées ici ne sont qu'un début de travail exploratoire et mériteraient d'êtres approfondies. Néanmoins, nous montrerons qu'il est possible d'obtenir de bons résultats préliminaires avec le principe de base décrit dans ce chapitre.

Ce chapitre s'articule en trois parties. Nous décrivons dans la première section les principes utilisés par cet algorithme. La section 10.2 est quant à elle dédiée à la présentation des résultats obtenus, dans le cadre des expérimentations dans les bâtiments Borel et Kahn. Nous concluons enfin l'ensemble de la quatrième partie du manuscrit dans la section 10.3.

10.1 Algorithme de détection de l'espace libre

L'algorithme de détection de l'espace libre a été développé pour le cas où le robot se déplace dans un plan 2D. En général, les informations dont nous disposons sont trop éparses pour obtenir une information 3D fiable. Elles sont néanmoins suffisantes pour obtenir une projection 2D de qualité acceptable.

Nous cherchons donc ici à superposer sur la projection 2D de la carte les zones qui sont potentiellement accessibles par le robot. L'algorithme se déroule en 3 étapes :

1. dans un premier temps, nous calculons la triangulation de Delaunay complète associée à la carte,
2. ensuite, nous " creusons " la triangulation obtenue en fonction des mesures effectuées,
3. enfin, le résultat creusé est projeté sur le plan du sol.

10.1.1 Triangulation de Delaunay

Pour déterminer l'espace libre disponible pour le robot, nous allons délimiter l'espace en zones discrètes. L'idée sera alors de voir à chaque instant quels amers sont vus. A chaque amer vu sera associé un rayon de visibilité : les zones traversées seront considérées comme libre.

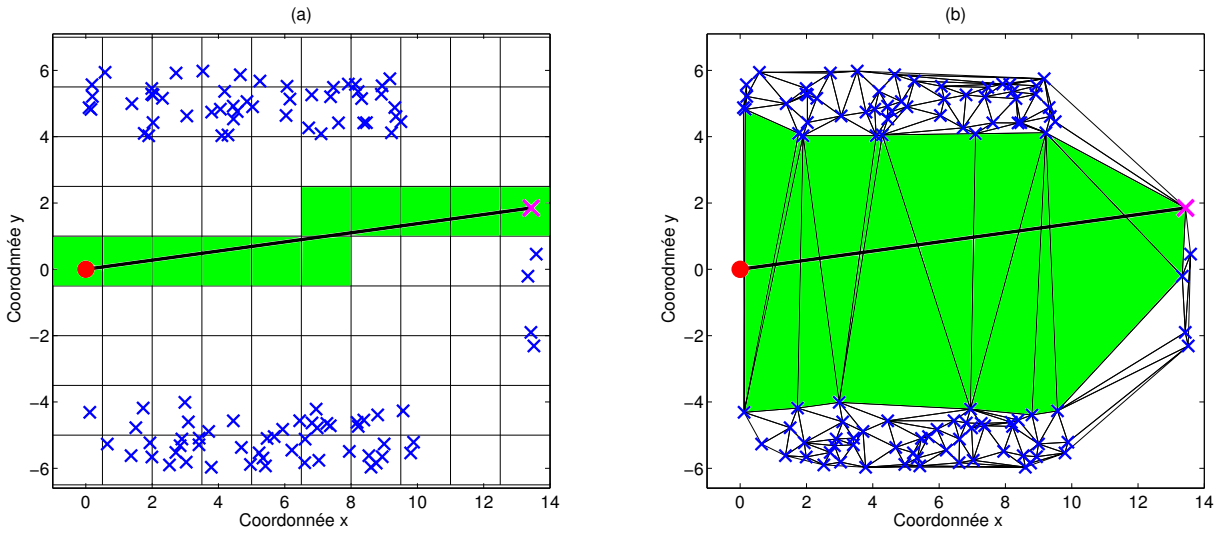


FIGURE 10.1 – Comparaison entre maillage constant (a) et maillage obtenu avec la triangulation de Delaunay dans le cas 2D (b) — Le point rouge représente la position du robot, le croix bleues l'ensemble des amers. Le trait noir épais représente le fait qu'à la position donnée du robot, l'amer magenta a été observé. L'observation de cet amer est censé informer sur la présence d'un long couloir. C'est ce que l'on obtient lorsque l'on " creuse " le long du rayon dans les cellules issues du Delaunay. Dans le cas constant, on ne " creuse " qu'un petit couloir limité.

Le choix du maillage de l'espace n'est pas trivial. Une première solution consisterait à effectuer un maillage régulier de l'espace délimité par les amers. Cette solution n'est pas idéale car le réglage de la taille des cellules conditionne directement la solution. Si le maillage est trop grossier, on perdra en précision. Si au contraire le maillage est trop fin, l'opération de creusage aura tendance à ne considérer que des " tunnels " très étroits comme espace libre si le nombre d'amers vus dans la région est trop bas. En conséquence, l'approche que nous avons choisie est d'adapter la taille des cellules en fonction de la densité des amers. Pour cela, nous avons utiliser le principe de la **triangulation de Delaunay 3D**. L'intérêt de cette triangulation est clairement mis en évidence sur la figure 10.1.

10.1.2 " Creusage " de l'espace libre

Nous avons déjà introduit l'idée de " creusage " de l'espace libre dans le paragraphe précédent. Il s'agit donc, à partir de la triangulation de Delaunay, d'éliminer un à un les tétraèdres qui correspondent à de l'espace libre. Pour cela, nous allons considérer chaque position du robot. Pour chacune de ces positions, nous allons analyser l'ensemble des mesures réalisées. Etant donné que la carte des amers est supposée connue, nous pouvons définir un rayon reliant le robot à l'amer pour chaque observation. Tous les tétraèdres traversés par au moins un rayon sont considérés comme de l'espace libre. Par

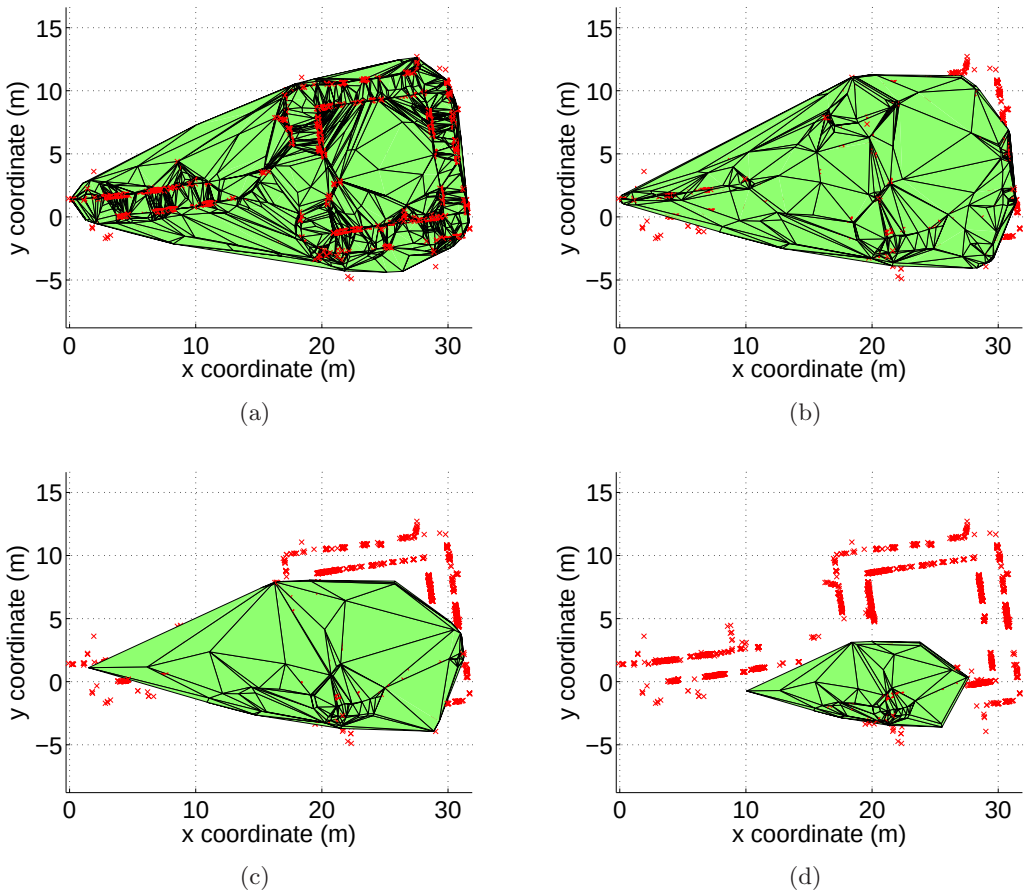


FIGURE 10.2 – Différentes coupes de la triangulation de Delaunay des points issus de la séquence Borel — (a) Avec le plan $z = 0$ – (b) Avec le plan $z = 0.5m$ – (c) Avec le plan $z = 0.7m$ – (d) Avec le plan $z = 1m$

ailleurs, l'ensemble des tétraèdres contenant les positions du robot est considéré comme de l'espace libre.

10.1.3 Déduction de l'espace libre en deux dimensions

Le résultat obtenu après avoir creuser la triangulation de Delaunay n'est en général pas exploitable directement. Le résultat que l'on a choisi de conserver est la coupe de l'espace vide avec le plan horizontal $z = 0$. Cette stratégie nous permet d'obtenir une bonne idée de l'espace accessible par le robot. Le choix de la coupe en $z = 0$ vient du fait que les points des cartes sont en moyenne répartis équitablement autour de ce plan. Ceci signifie que l'intersection entre la triangulation de Delaunay initiale et ce plan conduit à une surface conséquente (figure 10.2). Ceci justifie l'utilisation la coupe avec le plan $z = 0$.

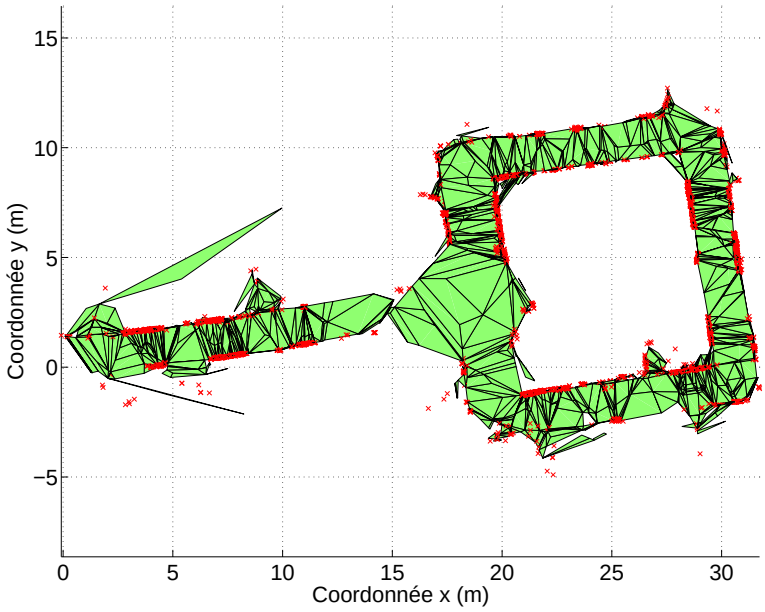


FIGURE 10.3 – Espace libre calculé sur la carte globale de l'expérimentation Borel (représenté en vert)

10.1.3.1 Conclusion

Au final, l'algorithme de détection de l'espace libre consiste dans un premier temps à calculer la triangulation de Delaunay associée à l'ensemble des amers issus du SLAM. Cette triangulation constitue une discrétisation de l'espace 3D. On utilise ensuite les mesures effectuées pour savoir quels sont les amers vus à chaque instant. On en déduit donc des rayons qui permettent de déduire des tétraèdres libres. Le résultat final est obtenu après intersection avec le plan $z = 0$.

10.2 Résultats obtenus

Nous présentons dans cette section les résultats obtenus dans le cadre des expérimentations Kahn et Borel décrites au chapitre 6.

10.2.1 Expérimentation Borel

Les premiers résultats obtenus sur la séquence Borel sont présentés sur la figure 10.3. Il s'agit des résultats obtenus en utilisant la **carte globale**. On remarque que la solution est cohérente dans son ensemble : on retrouve bien la structure de couloir présentée dans la scène réelle. Néanmoins, on constate quelques défauts :

- A l'abscisse $x = 15$m, la zone d'espace libre se resserre, ce qui donne l'impression que le couloir a une forme particulière. En réalité, il n'en est rien. Ce défaut est dû au très faible nombre d'amers présents dans ce secteur. Le faible nombre d'observations n'a pas permis de creuser suffisament le graphe de Delaunay,
- Vers le début de la carte, on trouve des triangles qui peuvent être assimiliés à des artefacts (le plus visible étant celui juste au dessus du couloir). Ce type d'artefact est dû au fait que la triangulation de Delaunay est globale et peut donc relier entre eux des points très éloignés (typiquement, des points appartenant au tout début du couloir et des points de l'angle supérieur gauche de la boucle [1]). Les tétraèdres associés à ce type de liaisons sont particulièrement allongés. Ainsi, il est possible que ces tétraèdres soient traversés par quelques rayons mais que seule une toute petite partie d'entre eux soit accessible.

Le second défaut pourraît être corrigé facilement en n'effectuant la triangulation de Delaunay que sur une petite partie de la carte. L'utilisation de l'algorithme sur les cartes locales pourraît alors être intéressante :

- on bénéficierait d'une part de la forte cohésion des cartes locales garantie par l'utilisation du critère (tout comme pour le cas de la segmentation des plans),
- on éviterait par ailleurs l'apparition des artefacts décrits précédemment.

Les résultats obtenus avec l'utilisation des cartes locales sont présentés sur la figure 10.4. Les résultats sont globalement bons : les plus gros artefacts présents sur la carte globale ont disparu. Néanmoins, on continue d'observer quelques défauts. Un autre défaut visible est celui de zones déclarées libres mais isolées de la partie principale (particulièrement visible sur la figure 10.4d). Ceci est dû au fait que certains tétraèdres sont accessibles en 3D, mais pas en conservant une trajectoire dans le plan $z = 0$.

10.2.2 Expérimentation Kahn

Les résultats concernant l'expérimentation Kahn sont présentés sur la figure 10.5 dans le cas de la carte globale. Tout comme pour le cas précédent, les résultats sont cohérents. On retrouve bien toute la zone libre correspondant à la trajectoire du robot. Tout comme pour le cas précédent, on retrouve quelques artefacts dus au fait que l'on calcule l'intersection entre un ensemble de tétraèdes et un plan horizontal. Les résultats concernant les cartes locales auraient conduit à une analyse très similaire au cas précédent et ne sont pas présentés ici.

1. On peut voir ce type de liaison sur la figure 10.2a

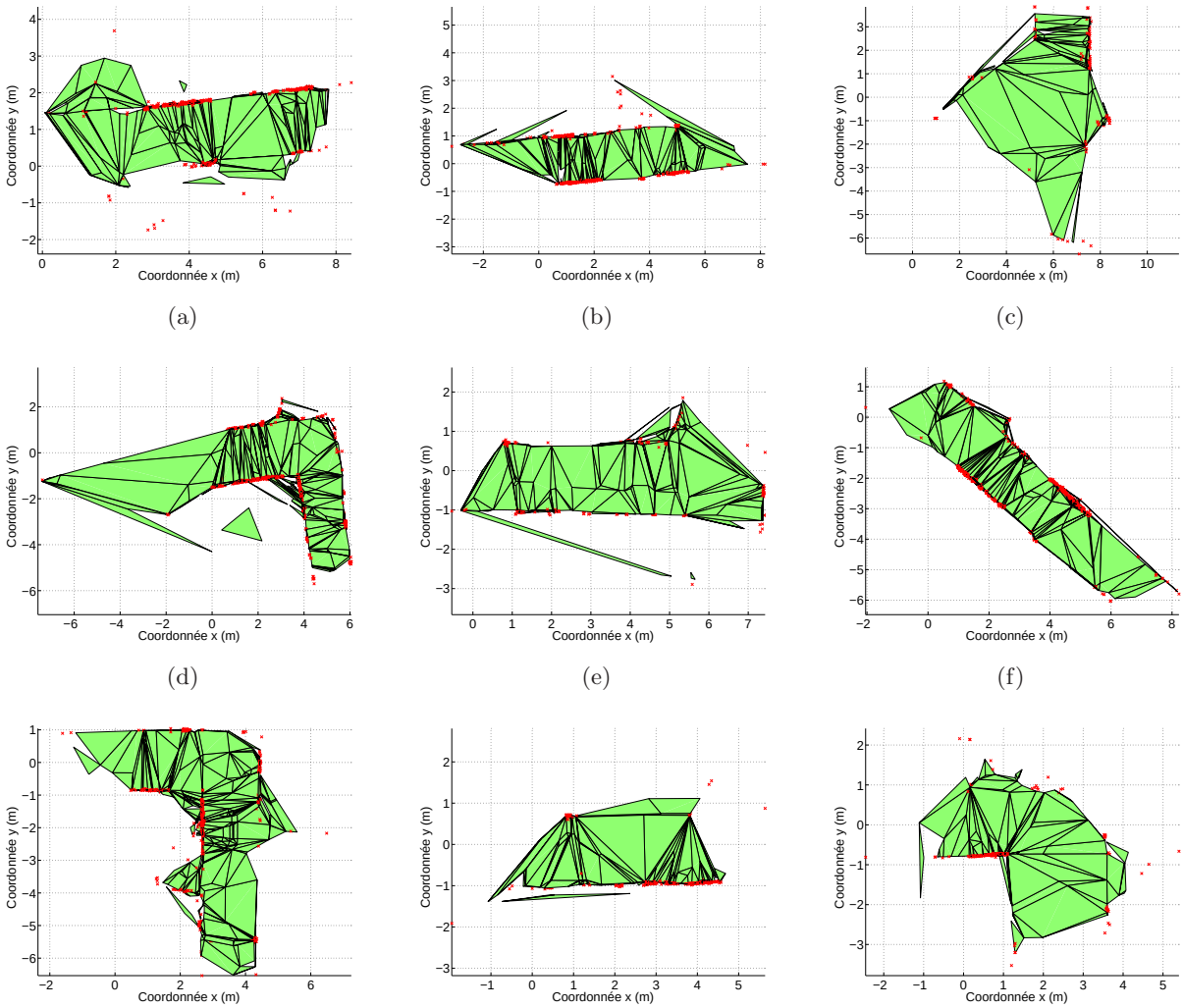


FIGURE 10.4 – Espace libre calculé sur les différentes cartes locales de l'expérimentation Borel

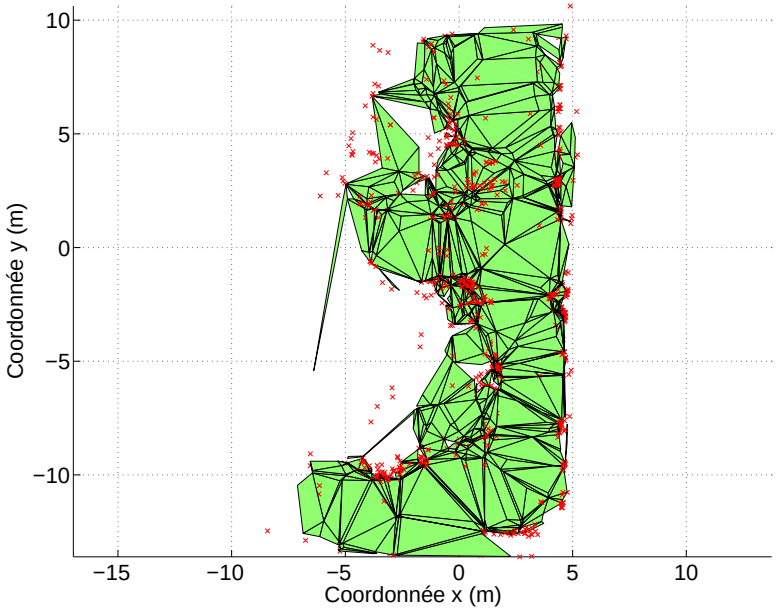


FIGURE 10.5 – Espace libre calculé sur la carte globale de l'expérimentation Kahn

10.2.3 Conclusion quant aux résultats

Les résultats présentés ici sont assez encourageants. Dans tous les cas, nous avons réussi à obtenir une représentation assez fidèle de l'espace accessible par le robot alors que l'on partait d'un résultat très épars fourni par l'algorithme de SLAM. Le résultat est alors parfaitement lisible et exploitable pour un utilisateur humain.

Il s'agit néanmoins de résultats préliminaires qui peuvent être nettement améliorés. Par exemple, il serait envisageable de filtrer automatiquement la majorité des artefacts qui ont été mis en évidence. Enfin, cette méthode dépend clairement de la qualité de la représentation de l'environnement. Ainsi, même si la représentation est assez fidèle à la réalité, il n'est pas garanti qu'elle soit dépourvue d'erreurs. Ceci rend son exploitation directe par le robot délicate.

Enfin, l'approche proposée n'est qu'une approche conçue pour le post-traitement. Tout est effectué lorsque le résultat final est connu (ce qui est nécessaire pour la triangulation de Delaunay). Il serait intéressant d'étendre ces résultats à une approche incrémentale pour pouvoir avoir en ligne une approximation de l'espace accessible.

10.3 Conclusion de la quatrième partie

Au final, nous avons présenté dans cette quatrième partie deux utilisations en post-traitements du résultat fourni par l'algorithme de SLAM. Les deux approches permettent d'apporter une information de plus haut niveau que le résultat " classique " de notre algorithme de SAM (ie. un vecteur d'état et sa matrice de variances-covariances – ou d'information – associée). Il s'agissait ici de donner un résultat facilement lisible par un opérateur extérieur, mais qui peut également être exploitable par un algorithme avancé.

Dans ce but, nous avons présenté au chapitre 9 une méthode permettant de détecter automatiquement les plans présents dans la scène, sans l'utilisation du moindre *a priori*. Les résultats obtenus étaient encourageants : la plupart des principaux plans ont été détectés. Néanmoins, la taille et le nombre des plans détectés est directement dépendante de la qualité du nuage de point obtenus par l'algorithme de SLAM.

Dans le présent chapitre, nous avons présenté une méthode permettant de fournir automatiquement l'espace accessible de la scène. Les résultats obtenus sont parfaitement lisibles par un utilisateur humain et permettent de donner une bonne idée de la topologie des lieux. Néanmoins, les résultats actuels doivent encore être améliorés pour pouvoir être utilisés de manière robuste par un autre algorithme.

Au final, les approches proposées dans la dernière partie de ce manuscrit doivent encore être approfondies. Il semble que la marge d'amélioration soit encore importante. Néanmoins, les résultats obtenus laissent penser que les méthodes générales employées sont tout à fait pertinentes.

Conclusion et perspectives

1 Conclusion et résumé des contributions

L'objectif principal de cette thèse était de proposer une méthode SLAM visuel qui donne des résultats à la fois précis et consistants. La notion de précision implique que l'erreur d'estimation doit être la plus faible possible ; la notion de consistante signifie quant à elle que l'on doit être capable d'estimer de façon juste l'erreur commise dans le filtre. Nous avons d'abord proposé une étude des propriétés des principaux algorithmes probabilistes. Nous avons rappelé et montré que deux des algorithmes les plus utilisés, à savoir le filtre de Kalman étendu et le filtre particulaire présentent de réels soucis de consistance. Ceci nous a amené à considérer les méthodes dites de SAM qui travaillent sur la totalité de la trajectoire.

Dans l'optique de la consistance, nous avons également considéré une méthode ensembliste de SLAM par intervalles. L'intérêt de cette méthode est qu'elle est intrinsèquement consistante dès lors que les hypothèse de départ sont respectées (l'intégralité des erreurs sur le système est borné, et une valeurs pessimiste des bornes est connue). L'arithmétique des intervalles permet alors de donner une solution au SLAM sous forme de boîtes dans lesquelles la probabilité d'appartenance de la solution est de 100%. Malheureusement, le prix à payer pour l'obtention de zones déterministe est une augmentation drastique de la taille des zones d'incertitudes si l'environnement n'est pas perçu de manière très redondante. Cette condition n'est pas vérifiée dans le cadre du SLAM visuel, les amers n'étant jamais entièrement observés. Ainsi, la méthode de SLAM par intervalles ne vérifiait pas le critère de précision requis pour la validation d'une méthode (et ce même si le centre des boîtes était très proche de la solution réelle). Par ailleurs, la nature des opérations effectuées dans le cas des traitements d'images fait qu'il est difficile de garantir la bornitude des erreurs de mesures (notamment en raison des algorithmes de mises en correspondances qui ont un taux d'*outliers* non nul). En conséquence, l'algorithme de SLAM par intervalles n'a pas été retenu pour le SLAM visuel malgré des intérêts théoriques évidents.

Nous avons ensuite proposé trois implémentations `Matlab` de l'algorithme de SAM :

1. La première implémentation est une simple application des principes de fusion entre odométrie et vision dans le cadre d'un déplacement plan du robot et d'amers ponctuels. Nous avons mis en évidence les bonnes propriétés des solutions pour des cas réels,
2. La seconde implémentation, quant à elle, suppose que la position de la caméra dans le repère du robot est inconnue. Nous nous situons toujours dans le cas d'un déplacement plan du robot. Nous avons été capables de montrer qu'il est théoriquement possible de déduire ces paramètres de position en plus de la position du robot et des amers, à condition que la trajectoire du robot vérifie une propriété précise : son rayon de courbure doit subir des variations. Nous avons vérifié ces aspects théoriques sur les séquences réelles, et avons montré que l'on dispose d'une solution finale quasiment équivalente à la précédente bien que l'on dispose de moins d'hypothèses,
3. La dernière implémentation, quant à elle, ne fait plus aucune hypothèse sur le mouvement du robot (on se place dans le cas à 6 degrés de liberté) et ne nécessite plus la présence de l'odométrie. Nous avons pour cela adapté au cas du SAM une méthode originale de paramétrisation des amers. Nous avons montré que notre implémentation respecte les hypothèses nécessaires à l'implémentation d'une méthode de lissage probabiliste alors que l'implémentation originale basée sur l'EKF introduit des liens non consistants dans le réseaux bayésien associé au SLAM. La méthode que nous avons proposée est donc bien définie théoriquement, et a obtenu de bons résultats.

Les implémentations que nous avons proposées possèdent néanmoins quelques instabilités liées à l'augmentation de la taille de la carte et à la perte de corrélation entre le début et la fin de la trajectoire (en l'absence de fermeture de boucle). Nous avons développé au cours de cette thèse une méthode de création de cartes locales avec un critère défini pour limiter et contrôler la perte de corrélation. Le critère à vérifier pour changer de carte locale utilise un paramètre local et scalaire correspondant à un rapport d'incertitudes. Il n'est alors plus nécessaire de définir de critère absolu (qui pourrait dépendre du type d'environnement considéré) pour déterminer quand créer une nouvelle carte. Nous avons par ailleurs apporté une contribution théorique au calcul des cartes locales en montrant qu'il est possible de prendre en compte de façon consistante les anciennes mesures dans les nouvelles cartes locales.

Enfin, la dernière contribution apportée concerne l'utilisation des résultats fournis par l'algorithme de SAM. Nous avons en effet montré qu'il est possible d'augmenter la représentation classique fournie par la projection des plans texturés et de l'espace libre. Pour cela, nous avons utilisé des méthodes déterministes sur les nuages de points obtenus. Nous avons justifié l'utilisation de telles méthodes

car notre critère de segmentation en cartes locales permet de garantir une forte cohésion au sein de chacune des cartes.

En résumé, les contributions apportées sur ces travaux sont essentiellement méthodologiques. Nous avons testé diverses méthodes de SLAM et les avons validé ou non dans le cadre de tests avec le logiciel `Matlab`. Dans chacune des nos contributions, nous nous sommes attachés à conserver l'objectif de généralité et de facilité de réglage (et donc d'implémentation des solutions). Nous avons ainsi limité au maximum les hypothèses sur l'environnement, notamment dans le cadre de notre algorithme de cartes locales.

2 Perspectives

2.1 A court terme...

Les solutions proposées dans cette thèse ont toutes été validées dans le cadre d'implémentations **hors ligne** avec le logicel `Matlab`. Nous avons montré ainsi que l'ensemble de nos contributions a un intérêt méthodologique certain. Néanmoins, nous n'avons pas implémenté nos algorithmes dans des conditions réelles. Ainsi, la première suite à donner à ces travaux serait d'implémenter les résultats de façon optimisée afin de les valider en ligne.

Par ailleurs, les solutions que nous avons proposées répondent chacune à des objectifs bien précis mais ne couvrent pas tous les aspects du SLAM. Ainsi, l'aspect fermeture de boucle n'a pas été abordé dans ces travaux. Il faudrait dans un premier temps implémenter une méthode permettant de détecter lorsqu'on l'on retrouve une zone précédemment explorée. Il existe aujourd'hui une littérature abondante sur le sujet : le premier travail serait d'analyser les différentes méthodes afin de selectionner celle la plus adaptée au cas du SLAM omnidirectionnelle. Il sera ensuite nécessaire de prolonger les travaux effectués sur les cartes locales afin de propager les informations issues de la fermeture de boucle dans les cartes locales.

Enfin, les travaux concernant la détection de plan et d'espace libre sont encore très incomplets. Ils constituent essentiellement les prémisses d'algorithmes plus fins à implémenter. Une extension immédiate serait par exemple d'introduire une méthode de suivi dense de type ESM (*Efficient Second order Minimization*) afin d'effectuer un suivi amélioré des textures projetées.

2.2 A moyen terme...

Sur le moyen terme, certaines de nos approches peuvent être améliorées au prix d'un travail théorique conséquent. La première amélioration préssentie concerne l'extension des résultats obtenus dans le cas où la position de la caméra par rapport à l'odométrie n'est pas connue. Les bons résultats obtenus en relâchant la contrainte sur ces nouveaux paramètres amènent directement à se poser la question de l'estimation des paramètres intrinsèques du modèle de caméra omnidirectionnelle (ie. les distances focales, centre de la caméra...). Une première approche consisterait à augmenter " brutalement " l'état du SAM et de constater ou non la convergence des paramètres. Nous avions mené quelques tests dans ce sens et pressentons qu'il semble possible d'améliorer l'estimation des paramètres intrinsèques de la caméra. Néanmoins, il est nécessaire de vérifier qu'une telle implémentation a un sens en vérifiant les **propriétés d'observabilité** du nouveau système. Les calculs à mener seront plus complexes que ceux présentés dans ce manuscrit, et nécessitent donc un travail de recherche supplémentaire. La vraie contribution n'est pas l'augmentation du vecteur d'état et l'implémentation de nouvelles matrices jacobiennes, mais bel et bien la preuve du bien fondé d'une telle augmentation.

Un second objectif à moyen terme serait d'utiliser les résultats de la détection de plan pour améliorer la représentation ponctuelle initiale et éventuellement densifier le nuage de points (en post-traitements dans un premier temps). En effet, le fait de pouvoir texturer complètement certaines parties de la solution permet de reconstruire des informations qui n'avaient pas été prises en compte par l'algorithme de SAM original. L'intérêt serait d'effectuer une fusion des résultats qui soit consistante, dans le sens où on est capable de garantir que le résultat final n'utilise pas plusieurs fois certaines informations (ce qui tendrait à rendre le résultat trop optimiste).

2.3 A long terme...

Enfin, l'objectif à plus long terme est l'**autonomie des robots**. Si les résultats du SLAM peuvent dans un premier temps servir à apporter une information aux opérateurs sur le terrain, il faut garder à l'esprit que leur objectif principal reste de permettre d'augmenter l'autonomie des robots.

Cet aspect, était initialement prévu dans le cadre de cette thèse. Nous espérions dans un premier temps pouvoir étendre facilement les résultats de commande référencée capteurs obtenus dans le cas du SLAM avec capteur laser. Dans le cas du SLAM visuel, la nature éparse des points d'intérêts fait qu'il est difficile de définir des stratégies de commandes simples. En conséquence, l'**aspect commande** n'a pas été abordé dans ces travaux de thèse. Etre capable de fusionner efficacement une méthode de

SLAM visuel et un algorithme de contrôle et de planification automatique de trajectoire constituerait une avancée majeure dans le domaine du SLAM.

Dans ce but, les premiers résultats obtenus quant à la visualisation de l'espace libre entrouvent quelques portes pour la planification de trajectoires. Ce problème semble très difficile à résoudre et constitue une stratégie de recherche à part entière sur le long terme.

Annexes

Annexe A

Notions d'analyse des graphes des réseaux Bayésiens

Nous présentons dans cette annexe les notions de base concernant l'analyse et l'interprétation des réseaux bayésiens. Dans tout ce qui suit, les variables aléatoires peuvent être scalaires ou vectorielles, sans contrainte de cohérence de dimension entre les différentes variables. Un réseau bayésien constitue une représentation graphique des modèles de probabilités. Ce type de représentation est particulièrement adapté lorsque l'on dispose d'informations causales. Ceci est le cas du SLAM, où l'évolution du robot se fait sous la forme d'une chaîne de Markov (l'état à l'instant t dépend de l'état à l'instant $t-1$) et les vecteurs de mesures sont directement des fonctions de l'état.

Les réseaux bayésiens font partie d'une catégorie bien particulière de graphes : il s'agit des graphes acycliques orientés, ou *directed acyclic graph* en anglais ([Smyth *et al.*, 1996; Wikipedia]). Il existe d'autres types de graphes, comme les graphes non orientés. Sous certaines conditions, il est possible de trouver un équivalent non orienté à un graphe orienté et vice-versa. Nous ne discutons pas de ces possibilités ; seuls les aspects généraux des réseaux bayésiens sont abordés. Pour des détails techniques plus précis sur le sujet, le lecteur intéressé pourra par exemple se reporter à [Smyth *et al.*, 1996; Yedidia *et al.*, 2002; Smail, 2004].

Cette annexe s'articule en 5 sections. Nous présentons dans un premier temps les briques de bases des réseaux bayésiens. La section A.2, quant à elle, est consacrée à l'étude de la transmission de l'information sur 3 nœuds consécutifs. Nous présentons ensuite le concept de séparation dirigée (*d-séparation*) dans la section A.3. Le réseau bayésien associé au SLAM est décrit dans la section A.4. Nous concluons enfin cette annexe par une synthèse des notions présentées.

FIGURE A.1 – Deux nœuds dans un réseau bayesien. Le sens de la flèche indique qu'il existe une relation de causalité de X vers Y.

A.1 Principes de base

Nous présentons dans ce paragraphe les principes de base permettant d'interpréter localement un réseau bayésien. Nous présentons dans un premier temps la liaison de base. Ensuite, nous présentons les 3 types de connexions possibles dans le cas des réseaux à 3 évènements.

A.1.1 Liaisons et densité de probabilité conditionnelle

Dans les réseaux bayésiens, on représente un lien de causalité entre deux nœuds par une flèche (figure A.1). On aura le type de liaison représenté dans la figure A.1 si la probablité (ou densité de probabilité) de Y sachant X est connue. Ceci signifie que toute information que l'on a sur X modifie l'information que l'on a sur Y, et, réciproquement, toute information sur Y peut modifier la connaissance que l'on a de X.

Lorsqu'un nœud posséde plusieurs ascendants, cela signifie que nous supposons connue la densité de probabilité de la variable associée au nœud conditionnée à tous ses ascendants. Par ailleurs, **conditionnellement à tous ses ascendants directs, un nœud donné est indépendant de tous les nœuds n'appartenant pas à ses descendants**. Cette dernière propriété justifie que les réseaux bayésiens soient des graphes acycliques (on ne doit trouver aucun circuit respectant le sens des liaisons). En effet, s'il existe un circuit respectant le sens des liaisons, alors un descendant direct N^e d'un nœud N peut aussi faire partie de ses ancêtres (et pourra donc être interprété comme un " non descendants "). Or, conditionnellement à ses ascendants, N doit être indépendant de ses " non descendants " et donc de N^e. N^e étant un descendant direct de N, ceci est impossible.

A.1.2 Feuilles du réseau

Les nœuds n'ayant pas de ascendants sont appelés des feuilles. Il n'y a pas de densité de probabilité conditionnelle associée à ces nœuds. Eventuellement, on peut disposer de la densité de probabilité *a priori*.

Toutes les feuilles sont indépendantes les unes des autres. Plus généralement, les variables aléatoires

associées à deux nœuds sont inconditionnellement indépendantes *si et seulement si* les nœuds associés n'ont pas d'ancêtre commun.

A.1.3 Résumé des caractéristiques d'un réseau bayésien

En résumé, un réseau bayésien est un graphe acyclique orienté auquel est attachée une variable aléatoire à chaque nœud. La notion d'acyclique signifie qu'il ne peut y avoir de circuit respectant le sens des liaisons. Les liaisons ou absence de liaison indiquent la nature des informations partagées entre un nœud et ses voisins :

- Les feuilles du réseau ne dépendent directement d'aucun autre nœud du graphe. On ne dispose pas de densité de probabilité conditionnement à un autre nœud,
- Les autres nœuds, quant à eux, disposent d'une densité de probabilité connue conditionnellement à leurs ascendants directs.

Enfin, les algorithmes de filtrage ou de lissage reviennent à effectuer une inférence sur le réseau, en supposant connus les nœuds correspondant à des mesures.

A.2 Circulation de l'information dans le cas de trois évènements

Nous présentons dans ce paragraphe comment circule l'information dans le cas où 3 éléments sont reliés à la suite. Les notations qui suivent font référence au tableau A.1.

A.2.1 Connexion convergente

Dans le cas d'une connexion convergente, l'information ne peut circuler entre les nœuds X et Y que si la valeur du nœud Z est connue. Intuitivement, ceci signifie que Z peut s'écrire comme une fonction de X et de Y. Pour que la connaissance de X puisse permettre de déduire Y, il faut que Z soit connu (sinon il manque une équation).

Plus rigoureusement, nous allons montrer que la densité de probabilité conditionnelle $p(X|Y,Z)$ est différente de la densité de probabilité conditionnelle $p(X|Z)$. Ceci prouvera que conditionné à Z, X et Y ne sont pas indépendants : il y a donc partage d'information. Pour cela, nous allons calculer le rapport entre ces deux densités :

$$\frac{p(X|Y,Z)}{p(X|Y)} = \frac{p(Z|X,Y)\,p(X|Y)}{p(Z|Y)}\frac{1}{p(X|Y)} \qquad \text{(règle de Bayes sur } p(X|Y,Z)) \tag{A.1}$$

$$= \frac{p(Z|X,Y)}{p(Z|Y)} \tag{A.2}$$

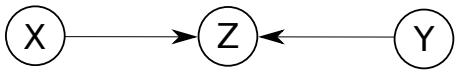
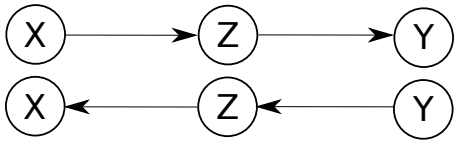
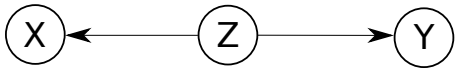
Nom	Graphe	Propriété
Connexion convergente		L'information ne circule de X vers Y (et vice-versa) que si Z **est connu**
Connexion en série		L'information ne circule de X vers Y (et vice-versa) que si Z **est inconnu**
Connexion divergente		L'information ne circule de X vers Y (et vice-versa) que si Z **est inconnu**

TABLE A.1 – Résumé des résultats concernant la transmissions de l'information entre deux nœuds séparés par un troisième

On peut distinguer deux situations avec l'équation A.2 :

1. $p(Z|X,Y) = p(Z|Y)$, ce qui implique $p(X|Y,Z) = p(X|Y)$: il n'y a pas de partage d'information entre X et Y. L'égalité $p(Z|X,Y) = p(Z|Y)$ signifie que conditionnellement à X et Y, Z ne dépend **que de** Y. Autrement dit, le lien représenté entre Z et X n'existe pas en réalité, seul le lien entre Z et Y existe. Dans ce cas, il était évident que X et Y ne pouvaient pas partager d'information,

2. $p(Z|X,Y) \neq p(Z|Y)$, ce qui implique $p(X|Y,Z) \neq p(X|Y)$: X et Y partagent de l'information conditionnellement à Z. Il s'agit du cas général.

Si l'on garde pour hypothèse qu'il n'y a pas de liaison " inutile ", on vient de montrer que les nœuds X et Y partagent de l'information si le nœud Z est connu.

On va enfin montrer que les nœuds X et Y ne partagent pas d'information lorsque Z est inconnu. Pour cela, il suffit d'avoir $p(X|Y) = p(X)$. Ceci est immédiat car X et Y sont inconditionnellement indépendants (ils n'ont aucun ancêtre en commun).

Ainsi, on vient de montrer que **dans le cas d'une connexion convergente, les nœuds X et Y ne partagent de l'information que si le nœud Z est connu**.

A.2.2 Connexion en série

Dans le cas d'une connexion en série, l'information ne peut circuler entre les nœuds X et Y que si la valeur du nœud Z est inconnue. Sans perte de généralité, nous considérons la connexion $X \longrightarrow Z \longrightarrow Y$. Si l'on reprend l'approche intuitive introduite au paragraphe précédent, Y peut être vu comme une fonction de Z et Z comme une fonction de X. Ainsi, même si Z est inconnu, Y peut être vu comme une fonction de X. Dans le cas où Z est connu, on ne peut plus propager l'information de X vers Y. En effet, fixer une valeur pour X ne changerait rien à la qualité de Z (qui est également fixé). La conséquence est que l'estimation de Y sera alors directement fonction de la valeur de Z (et donc indépendante de X).

D'un point de vue des densités de probabilité, nous devons montrer que :

1. $p(X|Y,Z) = p(X|Z)$ (lorsque Z est connu, X et Y sont indépendants)
2. $p(X|Y) \neq p(X)$ (lorsque Z n'est pas connu, X et Y ne sont pas indépendants).

La première propriété à montrer est équivalente à $p(Y|X,Z) = p(Y|Z)$. Cette dernière égalité est vraie par construction car le nœud Y est indépendant de tous ses " non descendants " (ie. X) conditionnellement à ses ascendants directs (ie. Z). D'où le résultat.

Concernant la seconde propriété, appliquons d'abord la règle de Bayes sur $p(X|Y)$:

$$p(X|Y) = p(Y|X)\frac{p(X)}{p(Y)} \tag{A.3}$$

Faisons désormais apparaître la variable Z par application de la règle de Bayes sur $p(Y|X)$ et $p(Y)$:

$$\begin{cases} p(Y|X) &= \frac{p(Y,Z|X)}{p(Y|Z,X)} \\ p(Y) &= \frac{p(Y,Z)}{p(Y|Z)} \end{cases} \tag{A.4}$$

En prenant en compte le fait que $p(Y|Z,X) = p(Y|Z)$, la substitution du système A.4 dans l'équation A.3 donne :

$$p(X|Y) = \frac{p(Y,Z|X)}{p(Y,Z)}p(X) \tag{A.5}$$

Pour que la propriété à démontrer soit vraie, il faut prouver que la fonction $p(Y,Z|X)/p(Y,Z)$ est différente de la fonction unité. Nous prouvons que cela est impossible par l'absurde. Supposons en effet que les deux densité de probabilité (qui sont des fonctions de Y et de Z) soient égales. On peut alors intégrer par rapport à Y sur l'ensemble des valeurs que peut prendre Y, ce qui donne :

$$\forall Z, X \quad \underbrace{\int p(Y,Z|X)\mathrm{d}Y}_{p(Z|X)} = \underbrace{\int p(Y,Z)\mathrm{d}Y}_{p(Z)} \tag{A.6}$$

On aurait alors $p(Z|X) = p(Z)$, ce qui signifie que Z est indépendant de X. Il s'agit d'un cas dégénéré dans lequel la liaison entre Z et X n'existe pas.[1] En supposant que cette liaison existe effectivement, ce cas ne peut pas se présenter. Ceci nous permet donc d'écrire $p(Y, Z|X) \neq p(Y, Z)$, et donc, d'après l'équation A.5 : $p(X|Y) \neq p(X)$.

A.2.3 Connexion divergente

Dans le cas d'une connexion divergente, l'information ne peut circuler entre les nœuds X et Y que si la valeur du nœud Z est inconnue. En reprenant l'approche intuitive introduite aux précédents paragraphes, X et Y peuvent tous deux être vus comme une fonction de Z. Lorsque X est connu et Z est inconnu, la valeur de X nous donne une indication concernant Z, que l'on peut alors utiliser pour Y. En revanche, lorsque Z est connu, l'information que l'on a sur X ne peut être utilisée pour améliorer l'information sur Z (Z étant déjà déterminé). Il n'y a donc pas de propagation possible de l'information jusqu'au nœud Y.

D'un point de vue des densités de probabilité, nous devons montrer que :

1. $p(X|Y, Z) = p(X|Z)$ (lorsque Z est connu, X et Y sont indépendants)
2. $p(X|Y) \neq p(X)$ (lorsque Z n'est pas connu, X et Y ne sont pas indépendants).

La première propriété à montrer est triviale. En effet, conditionnellement à Z (l'unique ascendant de X), X est indépendant de tous ses " non descendants " (ie. Y). On a donc bien $p(X|Y, Z) = p(X|Z)$.

La seconde propriété, quant à elle, peut être démontrée en utilisant point par point le raisonnement effectué pour démontrer la même propriété pour le cas de la connexion série. Toutes les étapes sont rigoureusement applicables au cas de la connexion divergente.

A.3 Séparation dirigée (*d-séparation*) et indépendance conditionnelle

A.3.1 Enoncé des conditions

Le concept de séparation dirigée permet de déterminer si deux nœuds sont indépendants étant donné un ensemble de nœuds V donné. Soit $\mathcal{P}$ un chemin **non orienté** ($\mathcal{P}$ peut aussi bien contenir des liaisons utilisées dans le sens direct que des liaisons utilisées en sens inverse) permettant de relier de nœuds X et Y d'un réseau bayésien. On dira que le chemin $\mathcal{P}$ *d-sépare* les nœuds X et Y conditionnellement à V *si et seulement si* au moins une des conditions suivantes est réalisée :

1. Dans ce cas, il est évident que X et Y ne partagent pas d'information.

- $\mathcal{P}$ contient une connexion en série de la forme $i \to m \to j$ telle que $m \in V$,
- $\mathcal{P}$ contient une connexion divergente de la forme $i \leftarrow m \to j$ telle que $m \in V$,
- $\mathcal{P}$ contient une connexion convergente de la forme $i \to m \leftarrow j$ telle que $m \notin V$ et qu'aucun descendant de m n'appartienne à V.

Remarque A.1 (Interprétation des conditions)
Les 2 premières conditions énoncées sont des conséquences directes de l'étude de la circulation de l'information menée à la section précédente. La troisième condition, quant à elle, est légèrement plus compliquée à justifier en raison de la condition supplémentaire sur les descendants de m. Cet aspect est justifié au paragraphe A.3.2.

On dira que des ensembles de nœuds $\mathcal{X}$ et $\mathcal{Y}$ sont *d-séparés* conditionnellement à l'ensemble de nœuds V *si et seulement si* tous les chemins commençant d'un nœud de $\mathcal{X}$ et se terminant en un nœud de $\mathcal{Y}$ *d-séparent* le nœud initial et le nœud final conditionnellement à V. **Dans ce cas, les variables aléatoires associées aux nœuds de $\mathcal{X}$ sont indépendantes des variables aléatoires associées aux nœuds de $\mathcal{Y}$ conditionnellement aux variables aléatoires associées aux nœuds de V.**

A.3.2 Etude du cas particulier

Nous avons vu dans la section précédente que dans le cas d'une connexion convergente de 3 éléments, les nœuds à l'extrémité (X et Y) ne partagent pas d'information lorsque le nœud central est inconnu. Nous proposons de montrer que si l'élement central a un descendant connu, alors les nœuds X et Y ne sont plus indépendants. Ceci permettra de justifier la formulation de la troisème condition concernant la *d-séparation*. Considérons sans perte de généralité le réseau présenté sur la figure A.2. [2] On va chercher à calculer $p(X|Y,Z)$. En appliquant la règle de Bayes, on obtient :

$$p(X|Y,Z) = \frac{p(Z|X,Y)}{p(Z|Y)} p(X|Y) \tag{A.7}$$

En appliquant la règle de Bayes sur $p(Z|X,Y)$ afin de faire apparaître W, on a :

$$p(Z|X,Y) = \frac{p(Z,W|X,Y)}{p(Z|W,X,Y)} \tag{A.8}$$

Conditionnellement à W, le nœud Z est indépendant de X et Y. On a donc :

$$p(Z|X,Y) = \frac{p(Z,W|X,Y)}{p(Z|W)} \tag{A.9}$$

2. Nous avons ici considéré que le descendant est un descendant direct. Pour le cas plus général, on pourrait utiliser récursivement le raisonnement introduit ici.

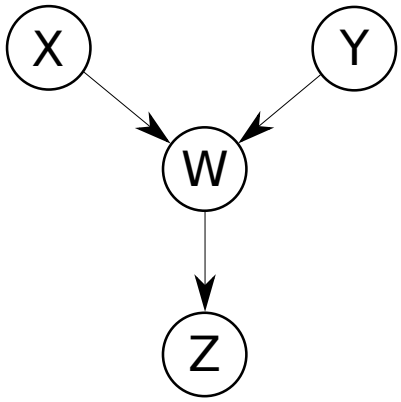


FIGURE A.2 – Connexion convergente de X et Y sur W, dans le cas particulier où W possède un descendant

De la même façon, on montre que :

$$p\left(Z|Y\right) = \frac{p\left(Z,W|Y\right)}{p\left(Z|W\right)} \tag{A.10}$$

En substituant les équations A.9 et A.10 dans l'équation A.7, on obtient :

$$p\left(X|Y,Z\right) = \frac{p\left(Z,W|X,Y\right)}{p\left(Z,W|Y\right)} p\left(X|Y\right) \tag{A.11}$$

Admettons désormais que $p\left(Z,W|X,Y\right) = p\left(Z,W|Y\right)$ quelque soit la valeur de Z considérée. Ceci implique, après intégration sur Z, que $p\left(W|X,Y\right) = p\left(W|Y\right)$. Ainsi, le lien entre X et W devrait être inexistant pour que l'hypothèse $p\left(Z,W|X,Y\right) = p\left(Z,W|Y\right)$ soit valide (ce qui est faux par construction).

On a donc en conséquence au moins un quadruplet de valeurs (W,X,Y,Z) tel que $p\left(Z,W|X,Y\right) \neq p\left(Z,W|Y\right)$. Les densités de probabilité $p\left(X|Y,Z\right)$ et $p\left(X|Y\right)$ sont donc bien différentes : X et Y partagent de l'information conditionnellement à Z, et ce même si le nœud Z est supposé inconnu.

Ceci explique finalement la précision supplémentaire sur la troisième condition pour la *d-séparation*.

A.4 Réseau bayésien associé au SLAM

Nous présentons dans cette section le réseau bayésien associé au cas du SLAM. Sa construction est quasi-immédiate. Elle se fait en utilisant les deux densités de probabilité conditionnelles (associées aux équations d'évolution du robot et de mesures) :

- La trajectoire du robot est représentée de manière linéaire par la chaîne de Markov $p\left(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{u}_t\right)$,
- Les amers sont ensuite disposés et reliés à la chaîne principale par l'intermédiaire des nœuds associés aux vecteurs de mesures grâce à la connaissance de $p\left(\mathbf{z}_t|\mathbf{x}_t,\mathbf{m}\right)$.

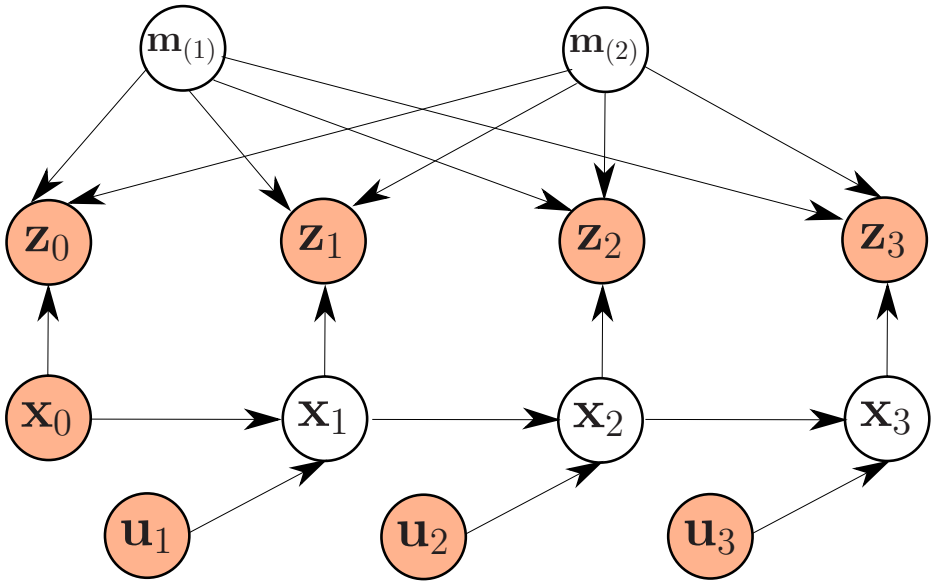


FIGURE A.3 – Réseau bayésien associé au SLAM. Les nœuds supposés connus sont représentés en couleur.

Le graphe typique obtenu est représenté sur la figure A.3, dans le cas où on a 4 positions et 2 amers. On vérifie alors bien que le graphe obtenu est acyclique. Dans ce graphe, les quantités connues ont été représentées en rouge : il s'agit des mesures des amers, des vecteurs de commandes ainsi que de la position initiale du robot.

A.5 Conclusion

Nous avons présenté dans cette annexe les éléments indispensables à la bonne utilisation des réseaux bayésiens. Nous avons vu rappelé l'interprétation basique des liasons, ce qui permet la construction du réseau lorsqu'on connaît les densités de probabilités conditionnelles.

L'interprétation graphique des réseaux bayésiens permet d'obtenir de forts résultats d'indépendance conditionnelle grâce à la notion de *d-séparation*. Nous avons finalement montré que ces résultats graphiques ne sont pas complètement triviaux. Néanmoins, ces justifications ne sont que rarement abordées dans la littérature du SLAM.

Annexe B

Observabilité du SLAM

Nous présentons dans cette annexe des résultats d'observabilité liés au SLAM. Le principe de l'observabilité est de vérifier si le problème d'estimation que l'on cherche à résoudre est bien posé ou non. Autrement dit, on cherche à savoir s'il existe ou non une solution au problème du SLAM.

Nous présentons dans la section B.1 des rappels concernant les notions d'observabilité des systèmes non-linéaires. Nous y introduisons notamment la notion des dérivées de Lie. La section B.2 est quant à elle vouée à la description des principales hypothèses utilisées dans ce chapire ; nous y introduisons notamment les fonctions de base servant à l'ensemble des études menées. Nous présentons dans la section B.3 une étude d'observabilité associée au SLAM visuel à 3 degrés de liberté. La section B.4 fournit des détails quant à l'observabilité d'un modèle d'état incluant certains paramètres de calibration de la caméra. Enfin, une synthèse et analyse des résultats est donnée dans la section B.5.

B.1 Observabilité non-linéaire

Nous présentons ici les principales notions concernant l'analyse d'observabilité appliquée aux systèmes non-linéaires. Considérons un système dont la représentation continue sous forme d'état est donnée par :

$$\left\{ \begin{array}{rcl} \dot{\mathbf{x}} & = & \mathbf{f}(\mathbf{x}, \mathbf{u}) \\ \mathbf{z} & = & \mathbf{h}(\mathbf{x}) \end{array} \right. \tag{B.1}$$

où $\mathbf{x}$ désigne l'état du système, $\mathbf{u}$ le vecteur de commande et $\mathbf{z}$ le vecteur de mesures. $\mathbf{f}$ et $\mathbf{h}$ sont des fonctions qui peuvent être non linéaires. La notion d'observabilité présentée dans ce paragraphe s'appuie sur les travaux d'Herman et Krener ([Hermann et Krener, 1977]). Ces derniers se basent sur le concept d'*indiscernabilité* locale de l'état initial.

Les définitions suivantes sont extraites de la thèse de doctorat de Philippe Bonnifait ([Bonnifait, 1997]). Nous présentons les éléments indispensables à la compréhension des développements suivants.

Le lecteur intéressé par plus de détails pourra se rapporter à [Bonnifait, 1997] et aux références qui y sont proposées.

Définition B.1 (Indiscernabilité)

Deux états initiaux $\mathbf{x}(t_0) = \mathbf{x}_0$ *et* $\mathbf{x}(t_0) = \mathbf{x}_1$ *sont dits* ***indiscernables*** *si,* $\forall t > t_0$ *les sorties correspondantes* $\mathbf{z}_0(t)$ *et* $\mathbf{z}_1(t)$ *sont identiques quelle que soit l'entrée* $\mathbf{u}[t_0, t]$ *appliquée au système.*

Définition B.2 (Observabilité)

Un état $\mathbf{x}_0$ *est dit* ***observable*** *s'il existe une commande qui permet de discerner* $\mathbf{x}_0$ *de tous les autres états.*

La définition d'observabilité donnée est globale et n'est pas valable pour toute entrée $\mathbf{u}$. Ainsi, un état sera considéré comme observable même si certaines séquences d'entrées ($\mathbf{u}$) ne permettent pas de distinguer $\mathbf{x}_0$. Par ailleurs, pour une séquence d'entrées $\mathbf{u}$ permettant de discerner $\mathbf{x}_0$, il est possible que les différences " souhaitées " dans le comportement entrée/sortie du système n'apparaissent au bout d'un temps très long (voire infini à la limite). Autrement dit, $\mathbf{x}_0$ peut rester indiscernable de ses voisins " pendant très longtemps ". Cette constatation nous amène à définir la notion d'observabilité locale qui est beaucoup plus intéressante :

Définition B.3 (Observabilité locale)

S'il existe T *tel que l'on puisse distinguer* $\mathbf{x}_0$ *sur l'intervalle de temps* $[t_0\ T]$*, alors* $\mathbf{x}_0$ *est* ***localement observable.***

On remarquera que l'observabilité locale implique l'observabilité. Le critère de rang que nous présentons dans la suite ne nous permettra pas de démontrer l'observabilité locale ni l'observabilité. En pratique, on n'est capable d'obtenir des résultats que dans un voisinage donné. Ceci nous amène à la définition de l'**observabilité faible**.

Définition B.4 (Observabilité faible)

Si on arrive seulement à distinguer $\mathbf{x}_0$ *de ses voisins, alors* $\mathbf{x}_0$ *est* ***faiblement observable.***

Au final, la notion d'**observabilité faible locale** découle de la combinaison des définitions B.3 et B.4. Dans la suite, nous chercherons à démontrer si notre système vérifie cette propriété à l'aide d'une condition de rang sur la matrice d'observabilité (définie dans la suite). Le fait que l'observabilité que l'on obtient soit faible n'est pas problématique : les algorithmes non linéaires utilisés travaillent

autour de points de fonctionnement connus. L'aspect le plus important est donc de savoir si l'état est observable autour de ces points de fonctionnement. Pour la suite, nous avons besoin de définir la notion de dérivée de Lie.

Définition B.5 (Dérivée de Lie)

Soit $h : \mathbb{R}^n \to \mathbb{R}$ *une fonction réelle dérivable et* $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$ *une fonction réelle vectorielle. La dérivée de Lie par rapport au champ de vecteur* $\mathbf{f}$ *est définie par :*

$$L_{\mathbf{f}}h = \frac{\partial h}{\partial \mathbf{x}} \cdot \mathbf{f} \tag{B.2}$$

L'intérêt des dérivées de Lie est qu'elles permettent de prendre en compte le modèle d'évolution du robot dans le calcul des dérivées temporelles du vecteur de mesures (on dérive par rapport au champ de vecteurs défini par la fonction $\mathbf{f}$ donnant l'évolution du robot). On définit par ailleurs les dérivées de Lie d'ordre k par :

$$L^k_{\mathbf{f}}h = \frac{\partial L^{k-1}_{\mathbf{f}}h}{\partial \mathbf{x}} \cdot \mathbf{f} \tag{B.3}$$

La condition de rang utilisée dans les développements suivants pour montrer l'observabilité est donnée dans le théorème B.1.

Dans la suite, nous écrivons le gradient de h par rapport à l'état $\mathbf{x}$ $\mathrm{d}h$ (au lieu de $\frac{\partial h}{\partial \mathbf{x}}$). Il en est de même pour les gradients des dérivées de Lie successives.

Théorème B.1 (Matrice d'observabilité et condition de rang)

Considérons désormais la fonction $\mathbf{h} : \mathbf{R}^n \to \mathbf{R}^p$ *dont les composantes s'écrivent :*

$$\mathbf{h} = [h^1 \dots \ h^p]^T \tag{B.4}$$

Soit $\mathbf{O}$ *la matrice d'observabilité définie par :*

$$\mathbf{O} = \begin{bmatrix} \mathrm{d}h^1 \\ \mathrm{d}L_{\mathbf{f}}h^1 \\ \vdots \\ \mathrm{d}L^{n-1}_{\mathbf{f}}h^1 \\ \vdots \\ \vdots \\ \mathrm{d}h^p \\ \mathrm{d}L_{\mathbf{f}}h^p \\ \vdots \\ \mathrm{d}L^{n-1}_{\mathbf{f}}h^p \end{bmatrix} \begin{matrix} \left.\begin{matrix} \\ \\ \\ \\ \end{matrix}\right\} \mathbf{O}^1 \text{ : matrice d'observabilité associée à } h^1 \\ \left.\begin{matrix} \\ \\ \\ \\ \end{matrix}\right\} \text{matrices d'observabilité associée à } h^2 \dots h^{p-1} \\ \left.\begin{matrix} \\ \\ \\ \\ \end{matrix}\right\} \mathbf{O}^p \text{ : matrice d'observabilité associée à } h^p \end{matrix} \tag{B.5}$$

Si la matrice $\mathbf{O}$ *est de rang plein, alors le système est* ***localement faiblement*** *observable. Par implication, il est faiblement observable.*

Remarque B.1 (Relation d'implication uniquement)

Le théorème B.1 ne donne qu'une implication. Ainsi, il est possible que la matrice d'observabilité définie dans l'équation B.5 ne soit pas de rang plein et que le système soit tout de même localement faiblement observable. Ceci est dû au fait que l'on peut peut-être trouver une matrice d'observabilité de rang plein en continuant les dérivations de Lie pour les ordres supérieurs à $n-1$.[1] *On admettra qu'un sous-espace est inobservable si les colonnes qui le représentent dans la matrice d'observabilité sont liées aux autres colonnes quel que soit le degré de dérivation considéré.*

Remarque B.2 (Analogie avec le cas linéaire)

Le théorème B.1 peut être utilisé dans le cas d'un système linéaire. Dans ce cas, la dérivée de Lie d'ordre k *est égale à* $\mathbf{CA}^k$*, où* $\mathbf{A}$ *désigne la matrice associée à l'évolution du système (*$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu}$*) et* $\mathbf{C}$ *la matrice associée aux observations (*$\mathbf{z} = \mathbf{Cx}$*). On retrouve alors la matrice d'observabilité classique du cas linéaire. Néanmoins, le théorème spécifique au cas linéaire est plus fort car il fournit une relation d'équivalence.*

Dans la suite de cette annexe, nous parlerons systématiquement d'observabilité ou de non observabilité (on ne précisera plus qu'il ne s'agit en fait que de faible observabilité).

B.2 Hypothèses générales

Nous présentons dans cette section une étude d'observabilité utilisant un modèle 2D. Le robot évolue dans le plan horizontal avec 3 degrés de liberté (2 en translation et 1 en rotation). Les amers sont des points du plan paramétrés par leurs deux coordonnées.

Remarque B.3 (Cas purement 2D)

Nous ne traitons pas ici le cas où les amers sont paramétrés en 3D, comme cela a été fait dans la partie principale du document. D'un point de vue théorique, l'ajout de la $3^{\text{ème}}$ *composante sur les amers n'apporte pas beaucoup. En effet, elle est associée à la mesure d'élévation qui permet de lever toute ambiguité dès lors que le positionnement en* (x, y) *de l'amer est réalisé. Les principaux problèmes liés à l'observabiltié apparaissent sur la partie de l'état que l'on considère ici.*

1. Ce qui n'est pas possible pour le cas linéaire.

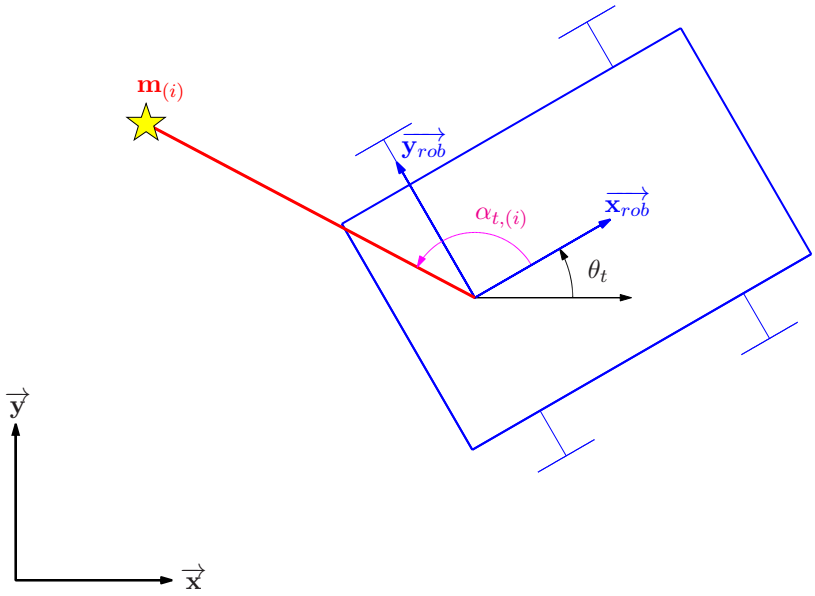


FIGURE B.1 – Définition des repères

B.2.1 Modèle d'état

L'état du robot est représenté par le vecteur $\mathbf{x} = \begin{bmatrix} x & y & \theta \end{bmatrix}^T$. x et y sont les coordonnées du robot dans le repère absolu R_0. θ est l'orientation du repère du robot par rapport au repère R_0 (cf. figure B.1). Nous utilisons un modèle " classique " d'évolution de voiture de type tricycle :[2]

$$\dot{\mathbf{x}} = \begin{bmatrix} V\cos\theta \\ V\sin\theta \\ \omega \end{bmatrix} \tag{B.6}$$

où V et ω sont les entrées du système.

Nous supposons que nous observons N amers simultanément. Chaque amer (i) est représenté par ses coordonnées dans le repère global : $\mathbf{m}_{(i)} = \begin{bmatrix} x_{(i)} & y_{(i)} \end{bmatrix}^T$.[3] Les amers sont supposés immobiles, l'équation d'évolution associée est donc triviale.

2. Nous nous plaçons pour cette étude dans le cas simplifié où la composante transversale (dans le repère du robot) de la vitesse est nulle.

3. Tout comme la partie principale du document, nous utilisons des parenthèses autour des indices désignant les amers pour éviter toute confusion.

Au final, le vecteur d'état vaut :

$$\mathbf{X} = \begin{bmatrix} \mathbf{x} \\ \mathbf{m}_{(1)} \\ \vdots \\ \mathbf{m}_{(N)} \end{bmatrix} \tag{B.7}$$

et l'équation d'état s'écrit :

$$\dot{\mathbf{X}} = f(\mathbf{X}) = \begin{bmatrix} \dot{\mathbf{x}} \\ \mathbf{0}_{2N\times 1} \end{bmatrix} \tag{B.8}$$

B.2.2 Modèle d'observation

Nous décrivons ici le modèle d'observation utilisé dans la suite. Nous supposons que nous avons un capteur de type *bearing-only* (une caméra peut-être considérée comme un tel capteur). En conséquence, nous supposons que le robot est capable de mesurer le gisement de chaque amer dans le repère local (angle $\alpha_{(i)}$ sur la figure B.1). Le vecteur d'observation vaut donc $\mathbf{z} = \begin{bmatrix} \alpha_{(1)} \cdots \alpha_{(N)} \end{bmatrix}^T$.

Chaque élément du vecteur $\mathbf{z}$ peut s'écrire en fonction de l'état $\mathbf{X}$:

$$\alpha_{(i)} = \arctan 2 \left(y_{(i)} - y, x_{(i)} - x \right) - \theta \tag{B.9}$$

B.3 Modèle classique 2D

Nous présentons dans cette section l'étude d'observabilité menée en utilisant le modèle classique défini dans la section précédente. Nous présentons d'abord le cas avec une seule balise observée, puis le cas où N balises sont suivies.

B.3.1 Etude pour un seul amer

B.3.1.1 Calcul de $d\alpha_{(1)}$ et $dL_f\alpha_{(1)}$

La première ligne de la matrice d'observabilité est constituée de $d\alpha_{(1)} = \frac{d\alpha_{(1)}}{d\mathbf{X}}$. Un calcul immédiat montre que :

$$d\alpha_{(1)} = \begin{bmatrix} \frac{\Delta y_{(i)}}{\Delta x_{(i)}^2 + \Delta y_{(i)}^2} & -\frac{\Delta x_{(i)}}{\Delta x_{(i)}^2 + \Delta y_{(i)}^2} & -1 & -\frac{\Delta y_{(i)}}{\Delta x_{(i)}^2 + \Delta y_{(i)}^2} & \frac{\Delta x_{(i)}}{\Delta x_{(i)}^2 + \Delta y_{(i)}^2} \end{bmatrix} \tag{B.10}$$

avec $\Delta x_{(i)} = x_{(i)} - x$ et $\Delta y_{(i)} = y_{(i)} - y$.

Par définition, $L_f\alpha_{(1)}$ vaut :

$$\begin{aligned} L_{\mathbf{f}}\alpha_{(1)} &= \frac{\mathrm{d}\alpha_{(1)}}{\mathrm{d}\mathbf{X}} \cdot f(\mathbf{X}) \\ &= \frac{-\Delta x_{(1)} V \sin\theta}{\Delta x_{(1)}^2 + \Delta y_{(1)}^2} + \frac{\Delta y_{(1)} V \cos\theta}{\Delta x_{(1)}^2 + \Delta y_{(1)}^2} - \omega \end{aligned} \tag{B.11}$$

La deuxième ligne de la matrice d'observabilité s'obtient en calculant la matrice jacobienne de l'expression B.11 par rapport à $\mathbf{X}$. On obtient :

$$\mathrm{d}L_{\mathbf{f}}\alpha_{(1)} = \frac{\mathrm{d}L_{\mathbf{f}}\alpha_{(1)}}{\mathrm{d}\mathbf{X}} = \begin{bmatrix} \frac{-V\sin\theta\left(\Delta x_{(1)}^2-\Delta y_{(1)}^2\right)+2V\cos\theta\Delta x_{(1)}\Delta y_{(1)}}{\left(\Delta x_{(1)}^2+\Delta_{(1)}^2\right)^2} \\ \frac{-V\cos\theta\left(\Delta x_{(1)}^2-\Delta y_{(1)}^2\right)-2V\sin\theta\Delta x_{(1)}\Delta y_{(1)}}{\left(\Delta x_{(1)}^2+\Delta_{(1)}^2\right)^2} \\ -\frac{\Delta x V\cos\theta+\Delta y V\sin\theta}{\Delta x^2+\Delta y^2} \\ \frac{V\sin\theta\left(\Delta x_{(1)}^2-\Delta y_{(1)}^2\right)-2V\cos\theta\Delta x_{(1)}\Delta y_{(1)}}{\left(\Delta x_{(1)}^2+\Delta_{(1)}^2\right)^2} \\ \frac{V\cos\theta\left(\Delta x_{(1)}^2-\Delta y_{(1)}^2\right)+2V\sin\theta\Delta x_{(1)}\Delta y_{(1)}}{\left(\Delta x_{(1)}^2+\Delta_{(1)}^2\right)^2} \end{bmatrix}^T \tag{B.12}$$

On remarque dès à présent une particularité sur les deux premières lignes de la matrice d'observabilité : la première colonne est égale à l'opposé de la quatrième, et la deuxième colonne est égale à l'opposé de la cinquième. On remarque par ailleurs que ces deux premières lignes vérifient la relation suivante :

$$\forall k \in \{0,1\} \quad \mathrm{d}L_{\mathbf{f}}^k\alpha_{(1)}[3] = y.\mathrm{d}L_{\mathbf{f}}^k\alpha_{(1)}[1] - x.\mathrm{d}L_{\mathbf{f}}^k\alpha_{(1)}[2] + y^{(1)}.\mathrm{d}L_{\mathbf{f}}^k\alpha_{(1)}[4] - x^{(1)}.\mathrm{d}L_{\mathbf{f}}^k\alpha_{(1)}[5] \tag{B.13}$$

où $\mathrm{d}L_{\mathbf{f}}^k\alpha_{(1)}[i]$ désigne la $i^{\text{ème}}$ colonne de $\mathrm{d}L_{\mathbf{f}}^k\alpha_{(1)}$.

B.3.1.2 Propriétés de $dL_{\mathbf{f}}^k\alpha_{(1)}$

Nous avons vu dans le paragraphe précédent que certaines colonnes de la matrice d'observabilité semblent trivialement liées (ce lien est vérifié sur les deux premières lignes). Nous allons à présent montrer que ceci reste vrai pour la matrice d'observabilité complète (sans calculer toutes les dérivées de Lie, ce qui deviendrait vite très lourd).

Proposition B.1

En notant $\mathrm{d}L_f^k\alpha_{(1)}[i]$ *la* $i^{ème}$ *colonne de* $\frac{dL_f^k\alpha_{(1)}}{d\mathbf{X}}$*, on a :*

$$\boxed{\forall k \in \mathbb{N} \begin{cases} \mathrm{d}L_{\mathbf{f}}^k\alpha_{(1)}[1] &=& -\mathrm{d}L_{\mathbf{f}}^k\alpha_{(1)}[4] \\ \mathrm{d}L_{\mathbf{f}}^k\alpha_{(1)}[2] &=& -\mathrm{d}L_{\mathbf{f}}^k\alpha_{(1)}[5] \\ \mathrm{d}L_{\mathbf{f}}^k\alpha_{(1)}[3] &=& \left(x_{(1)}-x\right)L_{\mathbf{f}}^k\alpha_{(1)}[2] - \left(y_{(1)}-y\right)L_{\mathbf{f}}^k\alpha_{(1)}[1] \end{cases}} \tag{B.14}$$

Démonstration de la proposition B.1

▲ Par récurrence sur le degré de dérivation.

L'initialisation de la récurrence a déjà été effectuée au paragraphe précédent (pour $k = 0$ et $k = 1$).

On suppose que la proposition à démontrer est vraie au rang $k-1$. On se propose dans un premier temps de calculer $L_{\mathbf{f}}^{k}\alpha_{(1)}$:

$$\begin{aligned} L_{\mathbf{f}}^{k}\alpha_{(1)} &= \frac{\mathrm{d}L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\mathrm{d}\mathbf{X}} \cdot \mathbf{f}(\mathbf{X}) \\ &= \frac{\partial L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial x} V\cos\theta + \frac{\partial L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial y} V\sin\theta + \frac{\partial L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial \theta}\omega \end{aligned} \tag{B.15}$$

Nous pouvons ensuite en déduire $\mathrm{d}L_{\mathbf{f}}^{k}\alpha_{(1)}[1]$:

$$\begin{aligned} \mathrm{d}L_{\mathbf{f}}^{k}\alpha_{(1)}[1] &= \frac{\partial L_{\mathbf{f}}^{k}\alpha_{(1)}}{\partial x} \\ &= \frac{\partial^2 L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial x^2} V\cos\theta + \frac{\partial^2 L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial x\partial y} V\sin\theta + \frac{\partial^2 L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial x\partial \theta}\omega \end{aligned} \tag{B.16}$$

Par ailleurs, nous pouvons également déduire $\mathrm{d}L_{\mathbf{f}}^{k}\alpha_{(1)}[4]$ de B.15 :

$$\begin{aligned} \mathrm{d}L_{\mathbf{f}}^{k}\alpha_{(1)}[4] &= \frac{\partial L_{\mathbf{f}}^{k}\alpha_{(1)}}{\partial x^{(1)}} \\ &= \frac{\partial^2 L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial x^{(1)}\partial x} V\cos\theta + \frac{\partial^2 L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial x^{(1)}\partial y} V\sin\theta + \frac{\partial^2 L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial x^{(1)}\partial \theta}\omega \\ &= \frac{\partial}{\partial x}\underbrace{\left(\frac{\partial L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial x^{(1)}}\right)}_{-\frac{\partial L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial x}} V\cos\theta + \frac{\partial}{\partial y}\underbrace{\left(\frac{\partial L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial x^{(1)}}\right)}_{-\frac{\partial L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial x}} V\sin\theta + \frac{\partial}{\partial \theta}\underbrace{\left(\frac{\partial L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial x^{(1)}}\right)}_{-\frac{\partial L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial x}}\omega \\ &= -\frac{\partial^2 L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{d\partial x^2} V\cos\theta - \frac{\partial^2 L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial x\partial y} V\sin\theta - \frac{\partial^2 L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial x\partial \theta}\omega \\ &= -L_{\mathbf{f}}^{k}\alpha_{(1)}[1] \end{aligned} \tag{B.17}$$

On montre avec un calcul très similaire que :

$$dL_{f}^{k}\alpha_{(1)}[2] = \frac{\partial^2 L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial y^2} V\sin\theta + \frac{\partial^2 L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial x\partial y} V\cos\theta + \frac{\partial^2 L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial y\partial \theta}\omega = -dL_{f}^{k}\alpha_{(1)}[5] \tag{B.18}$$

Enfin, nous obtenons la dernière partie du système B.14 par :

$$\begin{aligned} \mathrm{d}L_{\mathbf{f}}^{k}\alpha_{(1)}[3] &= \frac{\partial L_{\mathbf{f}}^{k}\alpha_{(1)}}{\partial \theta} \\ &= \frac{\partial^2 L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial x\partial \theta} V\cos\theta - V\sin\theta\frac{\partial L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial x} \\ &\quad + \frac{\partial^2 L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial y\partial \theta} V\sin\theta + V\cos\theta\frac{\partial L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial y} + \frac{\partial^2 L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial \theta^2}\omega \end{aligned} \tag{B.19}$$

En utilisant la troisième hypothèse de récurrence, on a :

$$\begin{aligned}
\mathrm{d}L_{\mathbf{f}}^{k}\alpha_{(1)}[3] &= V\cos\theta\frac{\partial}{\partial x}\left(\left(x_{(1)}-x\right)\frac{\partial L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial y}-\left(y_{(1)}-y\right)\frac{\partial L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial x}\right)\\
&\quad +V\sin\theta\frac{\partial}{\partial y}\left(\left(x_{(1)}-x\right)\frac{\partial L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial y}-\left(y_{(1)}-y\right)\frac{\partial L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial x}\right)\\
&\quad +\omega\frac{\partial}{\partial\theta}\left(\left(x_{(1)}-x\right)\frac{\partial L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial y}-\left(y_{(1)}-y\right)\frac{\partial L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial x}\right)\\
&\quad -V\sin\theta\frac{\partial L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial x}+V\cos\theta\frac{\partial L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial y}
\end{aligned}\tag{B.20}$$

En utilisant les règles de dérivation, on obtient :

$$\begin{aligned}
\mathrm{d}L_{\mathbf{f}}^{k}\alpha_{(1)}[3] &= V\cos\theta\left(x_{(1)}-x\right)\frac{\partial^2 L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial x\partial y}-V\cos\theta\frac{\partial L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial y}-V\cos\theta\left(y_{(1)-y}\right)\frac{\partial^2 L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial x^2}\\
&\quad -V\sin\theta\left(x_{(1)}-x\right)\frac{\partial^2 L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial y^2}-V\sin\theta\left(y^{(1)}-y\right)\frac{\partial^2 L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial x\partial y}+V\sin\theta\frac{\partial L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial x}\\
&\quad \omega\left(x_{(1)}-x\right)\frac{\partial^2 L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial y\partial\theta}-\omega\left(y_{(1)}-y\right)\frac{\partial^2 L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial x\partial\theta}\\
&\quad -V\sin\theta\frac{\partial L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial x}+V\cos\theta\frac{\partial L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial y}\\
&= \left(x_{(1)}-x\right)\underbrace{\left(V\cos\theta\frac{\partial^2 L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial x\partial y}V\sin\theta\frac{\partial^2 L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial y^2}\omega\frac{\partial^2 L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial y\partial\theta}\right)}_{(1)}\\
&\quad +\left(y_{(1)}-y\right)\underbrace{\left(-V\cos\theta\frac{\partial^2 L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial x^2}-V\sin\theta\frac{\partial^2 L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial x\partial y}-\omega\frac{\partial^2 L_{\mathbf{f}}^{k-1}\alpha_{(1)}}{\partial x\partial\theta}\right)}_{(2)}
\end{aligned}\tag{B.21}$$

Les équations B.16 et B.18 nous permettent de simplifier les facteurs (1) et (2) par $-\mathrm{d}L_{\mathbf{f}}^{k}\alpha_{(1)}[2]$ et $\mathrm{d}L_{\mathbf{f}}^{k}\alpha_{(1)}[1]$ respectivement. On obtient la dernière égalité à démontrer en utilisant ces simplifications dans l'équation B.21 :

$$\mathrm{d}L_{\mathbf{f}}^{k}\alpha_{(1)}[3]=\left(x_{(1)}-x\right)\mathrm{d}L_{\mathbf{f}}^{k}\alpha_{(1)}[2]-\left(y_{(1)}-x\right)\mathrm{d}L_{\mathbf{f}}^{k}\alpha_{(1)}[1]\tag{B.22}$$

Finalement, on vient de démontrer que la propriété reste vraie au rang de dérivation k. La récurrence nous dit qu'elle sera alors vraie pour tout $k\in\mathbb{N}$.

■

La proposition B.1 confirme l'intuition que l'on pouvait avoir à la vue des deux premières lignes de la matrice d'observabilité.

B.3.1.3 Rang de la matrice d'observabilité

Cas " normal "

En prenant en compte l'équation B.14 la matrice d'observabilité pour un système à un seul amer va s'écrire :

$$\mathbf{O} = \left[\begin{array}{c|c|c|c|c} \mathbf{C}^{(1)}_{5\times1} & \mathbf{C}^{(2)}_{5\times1} & \left(x_{(1)} - x\right)\mathbf{C}^{(1)}_{5\times1} - \left(y_{(1)} - y\right)\mathbf{C}^{(2)}_{5\times1} & -\mathbf{C}^{(1)}_{5\times1} & -\mathbf{C}^{(2)}_{5\times1} \end{array} \right] \tag{B.23}$$

où $C^{(i)}_{5\times1}$ désigne un vecteur colonne de dimension 5. L'équation B.23 montre que la matrice d'observabilité associée au système n'est pas de rang plein : seuls deux vecteurs sont réellement indépendants. Une base du noyau de $\mathbf{O}$ est donnée par :

$$\mathbf{n_1} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad , \quad \mathbf{n_2} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad , \quad \mathbf{n_3} = \begin{bmatrix} -y \\ x \\ 1 \\ -y_{(1)} \\ x_{(1)} \end{bmatrix} \tag{B.24}$$

Ainsi, le système associé au SLAM n'est pas observable. Ce résultat avait déjà été montré pour le cas *range and bearing* dans [Lee *et al.*, 2006]. Par ailleurs, [Huang *et al.*, 2008] ont montré que l'on obtient le même noyau pour tout système de SLAM dont l'équation de mesures est inversible localement. [4]

Cas " dégénérés "

Nous avons vu au paragraphe précédent que la matrice $\mathbf{O}$ est génériquement de rang 2. Nous cherchons à savoir dans ce paragraphe si ce rang peut descendre à 1, voire 0. D'après l'équation B.10, le rang de $\mathbf{O}$ ne peut jamais être nul (présence du -1 à la troisième colonne. Ainsi, nous cherchons à savoir sous quelle condition le rang est égal à 1.

L'équation B.11 laisse apparaître deux **conditions suffisantes** pour qu'il y ait une perte de rang :

1. $V = 0$. Dans ce cas, on a $L_{\mathbf{f}}\alpha_{(1)} = \omega$. Ainsi, toutes les lignes de la matrice d'observabilité sont nulles exceptée la première ligne. Ceci est logique : la quantité d'information obtenue sur le système est encore plus faible lorsque le véhicule est à l'arrêt,
2. $\cos\theta\Delta y_{(1)} = \sin\theta\Delta x_{(1)}$. Dans ce cas, on a également $L_{\mathbf{f}}\alpha_{(1)} = \omega$. Les dérivées de Lie suivante seront donc toutes nulles. On est dans le cas où le robot avance **dans la direction de l'amer** : il est donc impossible d'évaluer la distance entre l'amer et le robot tant que l'on reste sur cette

4. On est ici dans un cas légèrement différent, mais on retrouve le même résultat.

trajectoire. [5]

Cherchons désormais des **conditions nécessaires** pour qu'il y ait une perte de rang. On sait que les trois dernières colonnes de la matrice d'observabilité sont liées aux 2 premières. Une condition nécessaire pour que la première colonne soit une combinaison linéaire de la seconde est que le déterminant de la sous-matrice 2×2 " en haut à gauche " de la matrice d'observabilité soit égal à zéro. [6] On cherche donc à résoudre :

$$\begin{vmatrix} \frac{\Delta y_{(i)}}{\Delta x_{(i)}^2+\Delta y_{(i)}^2} & -\frac{\Delta x_{(i)}}{\Delta x_{(i)}^2+\Delta y_{(i)}^2} \\ \frac{-V\sin\theta\left(\Delta x_{(1)}^2-\Delta y_{(1)}^2\right)+2V\cos\theta\Delta x_{(1)}\Delta y_{(1)}}{\left(\Delta x_{(1)}^2+\Delta y_{(1)}^2\right)^2} & \frac{-V\cos\theta\left(\Delta x_{(1)}^2-\Delta y_{(1)}^2\right)-2V\sin\theta\Delta x_{(1)}\Delta y_{(1)}}{\left(\Delta x_{(1)}^2+\Delta y_{(1)}^2\right)^2} \end{vmatrix} = 0 \tag{B.25}$$

Tous calculs faits, l'équation B.25 est équivalente à :

$$\frac{V}{\Delta x_{(1)}^2+\Delta y_{(1)}^2} \cdot \left(\Delta y_{(1)}\cos\theta - \Delta x_{(1)}\sin\theta\right) = 0 \tag{B.26}$$

L'équation B.26 sera vérifiée *si et seulement si* $V = 0$ et/ou $\cos\theta\Delta y_{(1)} = \sin\theta\Delta x_{(1)}$. On retrouve les deux conditions suffisantes énoncées précedemment. Ainsi, l'équation B.26 prouve qu'il n'existe pas d'autre configuration dégénérée possible.

B.3.2 Etude pour N amers

Nous venons de prouver que le système à 1 amer est inobservable : quel que soit le degré de dérivation utilisé, la rang de la matrice d'observabilité sera égal à 2 (voire 1 dans le cas où on se déplace en direction de l'amer). Nous nous proposons à présent d'étudier la matrice d'observabilité du système lorsque celui-ci est constitué de N amers.

Cette dernière matrice s'écrit :

$$\mathbf{O}_{N(2N+3)\times 2N+3} = \begin{bmatrix} \mathbf{O}^{(1)}_{2n+3\times 2N+3} \\ \vdots \\ \mathbf{O}^{(i)}_{2N+3\times 2N+3} \\ \vdots \\ \mathbf{O}^{(N)}_{2N+3\times 2N+3} \end{bmatrix} \tag{B.27}$$

5. Si $\omega \neq 0$, on sortira immédiatement de cette condition singulière et l'estimation aura alors un sens. En revanche, tant que l'on est dans cette condition et que $\omega = 0$, on ne pourra jamais évaluer la position de l'amer par rapport au robot.

6. Cette condition permet d'assurer que la restriction de la matrice d'observabilité aux deux premières lignes est de rang 1. Elle n'est pas suffisante pour dire que la matrice complète est effictivement de rang 1.

où $\mathbf{O}^{(i)}_{2N+3\times 2N+3}$ désigne la "sous-matrice d'observabilité" associée à la mesure $\alpha_{(i)}$. Ces matrices sont composées de dérivées de Lie successives de chaque observation et sont de taille $(2N+3)\times(2N+3)$.[7]

En tenant compte de la proposition B.1, chaque matrice d'observalité s'obtient trivalement par :

$$\mathbf{O}^{(i)}_{2N+3\times 2N+3} = \left[\begin{array}{c|c|c|c|c|c|c} \mathbf{C}^{1(i)}_{2N+3\times 1} & \mathbf{C}^{2(i)}_{2N+3\times 1} & \Delta x_{(i)}\mathbf{C}^{2(i)}_{2N+3\times 1} - \Delta y_{(i)}\mathbf{C}^{2(i)}_{2N+3\times 1} & \mathbf{0}_{2N+3\times 2(i-1)} & -\mathbf{C}^{1(i)}_{2N+3\times 1} & -\mathbf{C}^{2(i)}_{2N+3\times 1} & \mathbf{0}_{2N+3\times 2(N-i)} \end{array}\right] \tag{B.28}$$

où $\mathbf{C}^{1(i)}_{2N+3\times 1}$ et $\mathbf{C}^{2(i)}_{2N+3\times 1}$ désignent les 2 colonnes indépendantes obtenues avec les dérivées de Lie successives de la mesure d'angle $\alpha_{(i)}$. Par ailleurs, on a posé $\Delta x_{(i)} = x_{(i)} - x$ et $\Delta y_{(i)} = y_{(i)} - y$.

Au final, la matrice d'observabilité complète du système s'écrit :

$$\mathbf{O}_{N(2N+3)\times 2N+3} = \left[\begin{array}{cccccccccc} \mathbf{C}^{1(1)} & \mathbf{C}^{2(1)} & \Delta x_{(1)}\mathbf{C}^{2(1)}_{2N+3\times 1} - \Delta y_{(1)}\mathbf{C}^{2(1)}_{2N+3\times 1} & -\mathbf{C}^{1(1)} & -\mathbf{C}^{2(1)} & \mathbf{0} & \cdots & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \mathbf{0} & \ddots & \ddots & \ddots & \ddots & \mathbf{0} \\ \mathbf{C}^{1(i)} & \mathbf{C}^{2(i)} & \Delta x_{(i)}\mathbf{C}^{2(i)}_{2N+3\times 1} - \Delta y_{(i)}\mathbf{C}^{2(i)}_{2N+3\times 1} & \mathbf{0} & \ddots & -\mathbf{C}^{1(i)} & -\mathbf{C}^{2(i)} & \ddots & \mathbf{0} \\ \vdots & \vdots & \vdots & \mathbf{0} & \ddots & \ddots & \ddots & \ddots & \mathbf{0} \\ \mathbf{C}^{1(N)} & \mathbf{C}^{2(N)} & \Delta x_{(N)}\mathbf{C}^{2(N)}_{2N+3\times 1} - \Delta y_{(N)}\mathbf{C}^{2(N)}_{2N+3\times 1} & \mathbf{0} & \cdots & \cdots & \mathbf{0} & -\mathbf{C}^{1(i)} & -\mathbf{C}^{2(i)} \end{array}\right] \tag{B.29}$$

La matrice $\mathbf{O}$ définie dans l'équation B.29 possède un noyau de dimension 3 dont une base est donnée par les 3 vecteurs suivants :

$$\mathbf{n_1} = \begin{bmatrix} 1\\0\\0\\1\\0\\\vdots\\1\\0\\\vdots\\1\\0 \end{bmatrix} \quad , \quad \mathbf{n_2} = \begin{bmatrix} 0\\1\\0\\0\\1\\\vdots\\0\\1\\\vdots\\1\\0 \end{bmatrix} \quad , \quad \mathbf{n_3} = \begin{bmatrix} -y\\x\\1\\-y_{(1)}\\x_{(1)}\\\vdots\\-y_{(i)}\\x_{(i)}\\\vdots\\-y_{(N)}\\x_{(N)} \end{bmatrix} \tag{B.30}$$

On retrouve donc une version " étendue "du noyau trouvé au paragraphe précédent. Ainsi, la matrice $\mathbf{O}$ est en général de rang $2N$. Tout comme le cas à un seul amer, on peut constater des chutes de rang marginales lorsque le robot se déplace dans la direction de certains amers (la dimension du noyau augmente d'une unité pour chaque amer dans la direction du robot.)

7. La taille de chaque sous-matrice augmente avec le nombre d'amers. Ceci est dû au fait que l'ajout d'un amer augmente le vecteur d'état, ce qui augmente le degré de dérivation de 1.

B.3.3 Explication de la non-observabilité

Nous venons de voir que le système du SLAM n'est pas observable tel qu'il est posé. La matrice d'observabilité ne sera en effet jamais de rang plein, et ce quel que soit le degré de dérivation utilisé pour la construire. Ceci est finalement assez logique. En effet, l'étude d'observabilité vise à savoir s'il est possible de distinguer l'état $\mathbf{x}_0$ de ses voisins. Or, l'ensemble des mesures disponibles n'est constitué que de mesures relatives entre l'état du robot et l'état des amers. Il n'y a à aucun moment la notion de mesure absolue. Ainsi, pour une solution donnée au problème du SLAM, la même solution décalée rigidement (ie. l'état du robot et les amers) en translation et en rotation conduit à des mesures identiques. La taille du noyau de dimension 3 traduit le fait que l'on ne peut pas distinguer la position initiale du robot (n'importe quelle condition initiale $[x_0, y_0, \theta_0]$ convient).

Dans [Huang *et al.*, 2008], les auteurs donnent une interprétation des vecteurs $\mathbf{n_1}$, $\mathbf{n_2}$ et $\mathbf{n_3}$ qui concorde parfaitement avec la remarque précédente. Pour cela, considérons $[\mathbf{x}^T\ \mathbf{m}^T]^T$ une solution du problème du SLAM (ici, $\mathbf{x}$ désigne la position initiale, le reste de la trajectoire peut être obtenu par intégration de l'équation d'évolution). L'analyse d'observabilité indique que les solutions $[\mathbf{x}^T, \mathbf{m}^T]^T + \lambda_1\mathbf{n_1} + \lambda_2\mathbf{n_2} + \lambda_3\mathbf{n_3}$ $((\lambda_1, \lambda_2, \lambda_3) \in \mathbb{R}^3)$ sont également solution.[8] On peut alors remarquer que :

- Ajouter $\lambda_1\mathbf{n_1} + \lambda_2\mathbf{n_2}$ revient à translater l'ensemble de la solution (trajectoire et amers) du vecteur $[\lambda_1\ \lambda_2]^T$ dans le plan (x, y). On retrouve bien la partie translation du mouvement rigide possible décrit au paragraphe précédent. Par ailleurs, ces deux vecteurs resteront cohérents quelles que soient les valeurs de λ_1 et λ_2 (qui ne doivent pas être nécessairement petites).
- Concernant l'ajout du vecteur $\lambda_3\mathbf{n_3}$, l'interprétation est plus compliquée. Pour simplifier, considérons le système à un seul amer. Posons par ailleurs $\mathbf{x} = [x\ y\ \theta]^T$ et $\mathbf{m} = [x_{(1)}\ y_{(1)}]^T$ Supposons que l'on désire appliquer une rotation d'un petit angle λ_3 sur l'ensemble de la solution. La nouvelle solution obtenue permet d'obtenir le même vecteur de mesures et vaut :

$$\begin{bmatrix}\tilde{\mathbf{x}}\\ \tilde{\mathbf{m}}\end{bmatrix} = \begin{bmatrix} x\cos\lambda_3 - y\sin\lambda_3 \\ x\sin\lambda_3 + y\cos\lambda_3 \\ \theta + \lambda_3 \\ x_{(1)}\cos\lambda_3 - y_{(1)}\sin\lambda_3 \\ x_{(1)}\sin\lambda_3 + y_{(1)}\cos\lambda_3 \end{bmatrix} \approx \begin{bmatrix} x - y\lambda_3 \\ x\lambda_3 + y \\ \theta + \lambda_3 \\ x_{(1)} - y_{(1)}\lambda_3 \\ x_{(1)}\lambda_3 + y_{(1)} \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \\ x_{(1)} \\ y_{(1)} \end{bmatrix} + \lambda_3 \begin{bmatrix} -y \\ x \\ 1 \\ -y_{(1)} \\ x_{(1)} \end{bmatrix} = \begin{bmatrix}\mathbf{x}\\ \mathbf{m}\end{bmatrix} + \lambda_3\mathbf{n_3} \tag{B.31}$$

Ainsi, une rotation d'un petit angle λ_3 est équivalent à l'ajout du vecteur $\lambda_3\mathbf{n_3}$ à la solution. On remarquera ici que l'interprétation géométrique du vecteur $\mathbf{n_3}$ n'est valable que pour le cas

8. L'étude d'observabilité étant locale (notion de faible observabilité), cette solution n'est valable que proche de la solution initiale considérée. En pratique, cela signifie que λ_1, λ_2 et λ_3 doivent être " suffisament petits ".

de petits angles.

Au final, on se rend compte que le problème lié à l'observabilité du SLAM possède une explication géométrique très intuitive. On a en effet montré que dans le cas de trajectoires " normales " (le robot ne se dirige pas en ligne droite dans la direction d'un amer), la matrice d'observabilité a un noyau de dimension 3. Celui-ci correspond à un mouvement rigide de l'intégralité de la solution. En pratique, ce problème est résolu dans le filtrage en **supposant que la position initiale du robot est connue**. Si l'on raisonne en termes de probabilités, ceci signifie que l'on cherche une solution **conditionnellement à $\mathbf{x}_0$ fixé**. Sous cette dernière hypothèse, le vecteurs $\mathbf{n_1}$, $\mathbf{n_2}$ et $\mathbf{n_3}$ ne permettent plus de calculer une nouvelle solution à partir d'une solution existante (l'hypothèse sur la condition initiale serait alors violée). Dans ce cas, on pourra dire que le problème du SLAM est observable, et ce quel que soit le nombre d'amers observés.

B.4 Observabilité du système augmenté par la position de la caméra dans le repère du robot

B.4.1 Introduction

Nous présentons dans cette section une analyse d'observabilité basée sur un modèle augmenté incluant la position de la caméra dans le repère du robot. Ainsi, l'équation d'évolution du robot est inchangée mais l'équation de mesures l'est. Nous restons dans le cas 2D, dans le cas où un seul amer est traité. Les nouvelles inconnues à considérer sont les paramètres du vecteur translation entre le repère robot et le repère caméra (t_x et t_y) et l'angle γ entre ces deux repères. Le vecteur de mesures est constitué de l'angle de gisement entre le repère caméra et l'amer. L'intégralité des notations est reportée sur la figure B.2. La nouvelle équation d'évolution est donnée par :

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{t_x} \\ \dot{t_y} \\ \dot{\gamma} \\ \dot{x_{(1)}} \\ \dot{y_{(1)}} \end{bmatrix} = \begin{bmatrix} V\cos\theta \\ V\sin\theta \\ \omega \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{B.32}$$

L'équation de mesure associée au système est donnée par :

$$\alpha_{t,(i)} = \arctan2\left(\; y_{(i)} - y - t_y\cos\theta - t_x\sin\theta \;\; , \; x_{(i)} - x - t_x\cos\theta + t_y\sin\theta \; \right) - \theta - \gamma \tag{B.33}$$

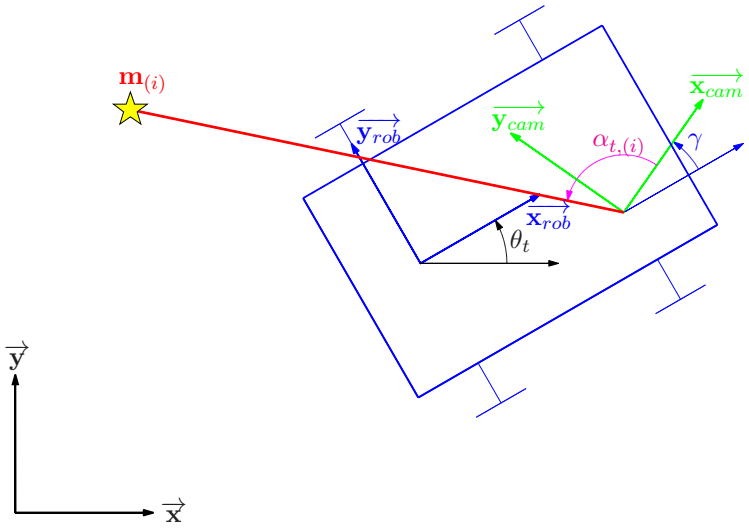


FIGURE B.2 – Définition des repères dans le cas augmenté

Nous présentons dans la suite de cette section les résultats de l'analyse d'observabilité menée pour ce système. Nous présentons dans un premier temps des éléments pour l'approche classique de l'observabilité avec l'utilisation des dérivées de Lie (paragraphe B.4.2). Enfin, nous présentons dans le paragraphe B.4.3 une analyse basée sur une discrétisation du système.

B.4.2 Analyse du cas continu

B.4.2.1 Introduction

Nous présentons dans les paragraphes suivants une étude permettant de vérifier si notre nouvelle formulation du SLAM conserve les propriétés d'observabilité que nous avons démontrées dans la section précédente. Pour cela, nous allons calculer la matrice d'observabilité associée au nouveau système (avec un seul amer). Le nouveau vecteur d'état considéré étant de dimension 8, l'analyse d'observabilité impliquerait de calculer jusqu'à la septième dérivée de Lie. Les calculs étant intensifs, les résultats présentés dans ce paragraphe ont été obtenus grâce au logiciel Maple.

B.4.2.2 Noyau obtenu

A l'aide de calculs effectués par Maple, nous avons trouvé deux vecteurs supplémentaires dans le noyau pour le cas où un seul amer est observé :

$$\mathbf{n_4} = \begin{bmatrix} \mathbf{0}_{3,1} \\ 0 \\ 1 \\ \frac{\omega^2 t_x}{(V-\omega t_y)^2+\omega^2 t_x^2} \\ \frac{-V^2\sin\theta-\omega V\left(x_{(1)}-x-t_x\cos\theta-t_y\sin\theta\right)-t_x\omega^2\left(y_{(1)}-y\right)+t_y\omega^2\left(x_{(1)}-x\right)}{(V-\omega t_y)^2+\omega^2 t_x^2} \\ \frac{V^2\cos\theta-\omega V\left(y_{(1)}-y-t_x\sin\theta+t_y\cos\theta\right)+t_x\omega^2\left(y_{(1)}-y\right)+t_y\omega^2\left(x_{(1)}-x\right)}{(V-\omega t_y)^2+\omega^2 t_x^2} \end{bmatrix} \tag{B.34}$$

et

$$\mathbf{n_5} = \begin{bmatrix} \mathbf{0}_{3,1} \\ 1 \\ 0 \\ \frac{-\omega^2 t_t+V\omega}{(V-\omega t_y)^2+\omega^2 t_x^2} \\ \frac{V^2\cos\theta-\omega V\left(y_{(1)}-y-t_x\sin\theta+t_y\cos\theta\right)+t_x\omega^2\left(x_{(1)}-x\right)+t_y\omega^2\left(y_{(1)}-y\right)}{(V-\omega t_y)^2+\omega^2 t_x^2} \\ \frac{V^2\sin\theta+\omega V\left(x_{(1)}-x-t_x\cos\theta-t_y\sin\theta\right)+t_x\omega^2\left(y_{(1)}-y\right)-t_y\omega^2\left(x_{(1)}-x\right)}{(V-\omega t_y)^2+\omega^2 t_x^2} \end{bmatrix} \tag{B.35}$$

Après plusieurs essais avec Maple, il semble que ces vecteurs fassent parti du noyau de la matrice d'observabilité quel que soit le degré de dérivation. Ceci pourrait être vérifié en faisant une récurrence similaire à celle présentée dans la démonstration de la proposition B.1 (les calculs étant ici sensiblement plus compliqués). Ainsi, le système à un seul amer n'est pas observable.

Par ailleurs, l'analyse des vecteurs $\mathbf{n_4}$ et $\mathbf{n_5}$ montre que cette propriété d'inobservabilité est conservée quel que soit le nombre d'amers. En effet, supposons désormais que l'on ajoute un second amer dans le système. Dans ce cas, la partie de la matrice d'observabilité associée à l'état des amers est bloc-diagonale. En conséquence, la concaténation du vecteur $\mathbf{n_4}$ avec une adaptation des deux dernières lignes de $\mathbf{n_4}$ à l'amer 2 est également un vecteur du noyau. Ceci est dû au fait que **les coefficients dans $\mathbf{n_4}$ correspondant aux paramètres du repère de la caméra sont indépendants de la position de l'amer considéré**. Il en est de même pour le vecteur $\mathbf{n_5}$.[9] A titre d'exemple, le vecteur

9. Dans le cas à N amers, $\mathbf{n_4}$ et $\mathbf{n_5}$ sont construits de façon semblable à $\mathbf{n_3}$.

$\mathbf{n_4}$ obtenu avec deux amers est :

$$\mathbf{n_4} = \begin{bmatrix} \mathbf{0}_{3,1} \\ 0 \\ 1 \\ \frac{\omega^2 t_x}{(V-\omega t_y)^2+\omega^2 t_x^2} \\ \frac{-V^2\sin\theta-\omega V\left(x_{(1)}-x-t_x\cos\theta-t_y\sin\theta\right)-t_x\omega^2\left(y_{(1)}-y\right)+t_y\omega^2\left(x_{(1)}-x\right)}{(V-\omega t_y)^2+\omega^2 t_x^2} \\ \frac{V^2\cos\theta-\omega V\left(y_{(1)}-y-t_x\sin\theta+t_y\cos\theta\right)+t_x\omega^2\left(y_{(1)}-y\right)+t_y\omega^2\left(x_{(1)}-x\right)}{(V-\omega t_y)^2+\omega^2 t_x^2} \\ \frac{-V^2\sin\theta-\omega V\left(x_{(2)}-x-t_x\cos\theta-t_y\sin\theta\right)-t_x\omega^2\left(y_{(2)}-y\right)+t_y\omega^2\left(x_{(2)}-x\right)}{(V-\omega t_y)^2+\omega^2 t_x^2} \\ \frac{V^2\cos\theta-\omega V\left(y_{(2)}-y-t_x\sin\theta+t_y\cos\theta\right)+t_x\omega^2\left(y_{(2)}-y\right)+t_y\omega^2\left(x_{(2)}-x\right)}{(V-\omega t_y)^2+\omega^2 t_x^2} \end{bmatrix} \tag{B.36}$$

En conséquence, la matrice d'observabilité du système augmenté possède une chute de 2 rangs qui traduit le fait que les paramètres de la position de la caméra ne " paraissent " pas observable. Nous allons voir que cette conclusion provient d'une limitation du modèle d'évolution utilisé. Pour cela, nous proposons dans un premier temps une interprétation géométrique directe de $\mathbf{n_4}$ et $\mathbf{n_5}$ dans le cas où $\omega = 0$. Nous proposons ensuite une explication pour le cas général.

B.4.2.3 Interprétation géométrique : $\omega = 0$

Le cas $\omega = 0$ correspond au cas où le robot évolue en ligne droite. Dans le cas à un seul amer, l'expression des vecteurs $\mathbf{n_4}$ et $\mathbf{n_5}$ est donnée par :

$$\begin{aligned} \mathbf{n_4} &= \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & -\sin\theta & \cos\theta \end{bmatrix}^T \\ \mathbf{n_5} &= \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & \cos\theta & \sin\theta \end{bmatrix}^T \end{aligned} \tag{B.37}$$

Dans ce cas, l'interprétation des vecteurs $\mathbf{n_4}$ et $\mathbf{n_5}$ est immédiate. En effet, si la trajectoire du robot est une ligne droite, alors celle de la caméra sera également une ligne droite parallèle à la trajectoire du robot (elle aura donc pour direction l'angle θ à cause de la contrainte de non-honolomie impliquant que la vitesse instantanée du robot reste parallèle à son axe x). Supposons que l'on ajoute λ à la valeur de t_x. Il est alors possible de retrouver les mêmes valeurs au niveau des mesures si on translate l'ensemble des amers de λ dans la direction du robot. Ceci correspond à une translation de $\lambda[\cos\theta, \sin\theta]$ dans le repère global. Au final, on retrouve bien le fait qu'ajouter $\lambda\mathbf{n_5}$ à une solution donnée ne change strictement rien aux mesures. Une analyse similaire concernant l'ajout de λ à la valeur t_y mettrait en évidence le rôle du vecteur $\mathbf{n_4}$. En revanche, il n'est pas possible d'ajouter un biais sur la valeur de γ sans introduire de déformation dans la carte. Ceci justifie la présence d'un noyau de dimension 2 et non de dimension 3.

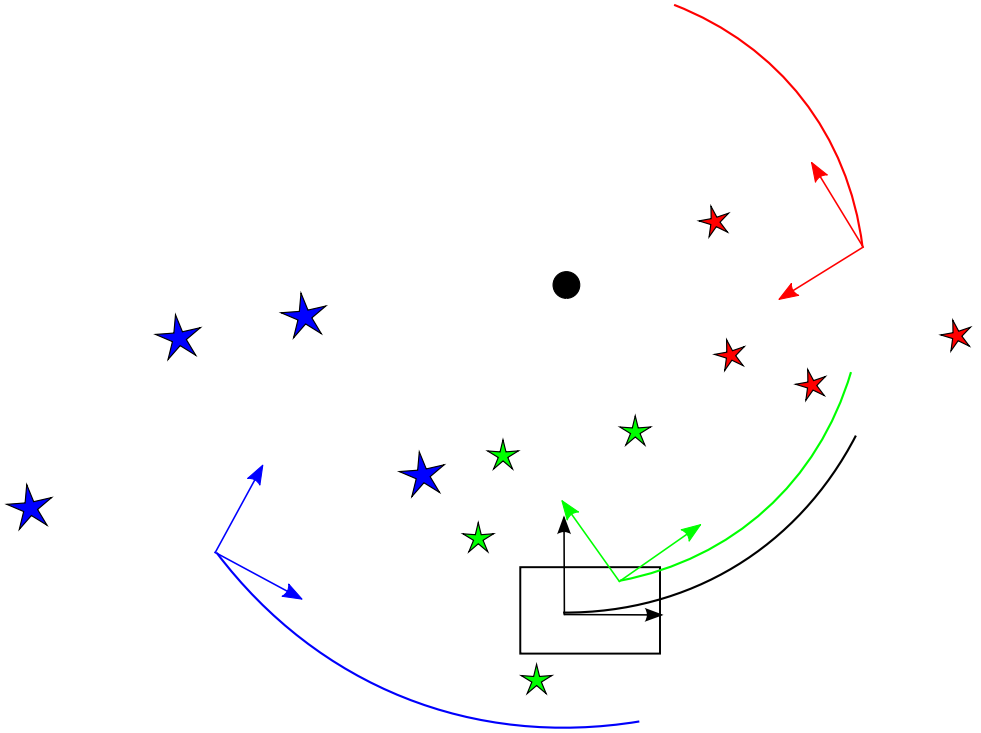


FIGURE B.3 – Non observabilité des paramètres de la caméra dans le cas d'une trajectoire circulaire

En conséquence, la non observabilité du système apparaît clairement dans le cas où le robot évolue avec une vitesse de rotation nulle.

B.4.2.4 Interprétation géométrique : cas général

Nous présentons dans ce paragraphe une interprétation concernant la non observabilité du système dans le cas où $\omega \neq 0$. Une approche naïve consisterait à admettre que le système est " plus excité " dans ce cas que dans le cas où $\omega = 0$ et que cela est suffisant pour rendre observable le système. La présence des vecteurs $\mathbf{n_4}$ et $\mathbf{n_5}$ nous dit le contraire. Nous proposons ici une approche géométrique permettant d'expliquer ces deux vecteurs.

Tout d'abord, il est nécessaire de se rendre compte que le type de trajectoire décrit par l'équation d'évolution (B.32) est un cercle. En conséquence, la trajectoire de la caméra est un cercle concentrique, mais de rayon différent. Une étude géométrique nous permet de nous rendre compte que cette contrainte sur l'évolution du robot **ne permet pas** de fixer la pose de la caméra dans le repère du robot.

Admettons que les entrées V et ω soient constantes pendant un intervalle de temps Δt donné. Le robot parcourt un arc de cercle (représenté en noir sur la figure B.3). Soient t_x, t_y et γ les paramètres réels définissant la pose de la caméra dans le repère du robot. La trajectoire associée à la caméra

pendant Δt ainsi que la carte d'amers sont représentés en vert sur la figure B.3. Géométriquement, il est facile de montrer que les trajectoires bleue et rouge permettent d'obtenir les mêmes vecteurs de mesures à chaque instant entre 0 et Δt :

- La trajectoire rouge correspond à une solution où l'origine du repère de la caméra a été translatée **sur le cercle** défini par la solution réelle. Il est alors possible d'avoir un vecteur de sortie identique si on prend le soin de tourner le nouveau repère de la caméra de telle façon que l'**angle avec la tangente au cercle** soit le même que dans le cas réel.
- La trajectoire bleue correspond à une solution où la trajectoire du repère de la caméra a été décalée sur un cercle dont le rayon de courbure est supérieur à celui du cercle réel. Dans ce cas, il est possible d'obtenir une carte compatible avec la nouvelle trajectoire en prenant le soin d'appliquer une homothétie dont le facteur est égal au rapport des rayons de courbure. Dans ce cas aussi, il est nécessaire d'orienter le repère de la caméra de sorte à conserver le même angle par rapport à la tangente au cercle que dans le cas réel.

Ainsi, on se rend compte que ce type de trajectoire ne permet pas de rendre observable la pose de la caméra dans le repère du robot. On peut définir t_x et t_y de manière arbitraire. Il en résultera alors une adaptation nécessaire de l'angle γ (ce qui justifie que l'on a seulement deux et non trois nouveaux vecteurs dans le noyau) et une mise à l'échelle de la carte. Nous proposons dans le paragraphe suivant de vérifier que les vecteurs $\mathbf{n_4}$ et $\mathbf{n_5}$ sont bien cohérents avec cette interprétation géométrique.

Dans la suite, nous notons les paramètres de la solution réelle comme des variables " normales " (t_x,t_y,...) et les paramètres d'une solution alternative avec des " primes " (t'_x, t'_y,...). Par ailleurs, on se place dans le cas où un seul amer est observé.

Considérons une solution alternative définie par :

- $x' = x$, $y' = y$, $\theta' = \theta$: la position de référence du robot est identique dans les deux cas (on ne cherche plus à mettre en évidence la non observabilité de la position du robot,
- $t'_x = t_x + \lambda$, $t'_y = t_y$: on translate la caméra de λ dans la direction x du repère du robot.

Les autre paramètres (amer et angle γ) sont à redéfinir en utilisant les remarques géométriques définies précédemment. Nous proposons de calculer la nouvelle solution en 4 étapes :

1. Calcul du centre de rotation
2. Définition des rayons de courbure réel et alternatif,
3. Calcul de l'angle γ
4. Définition de la nouvelle position de l'amer

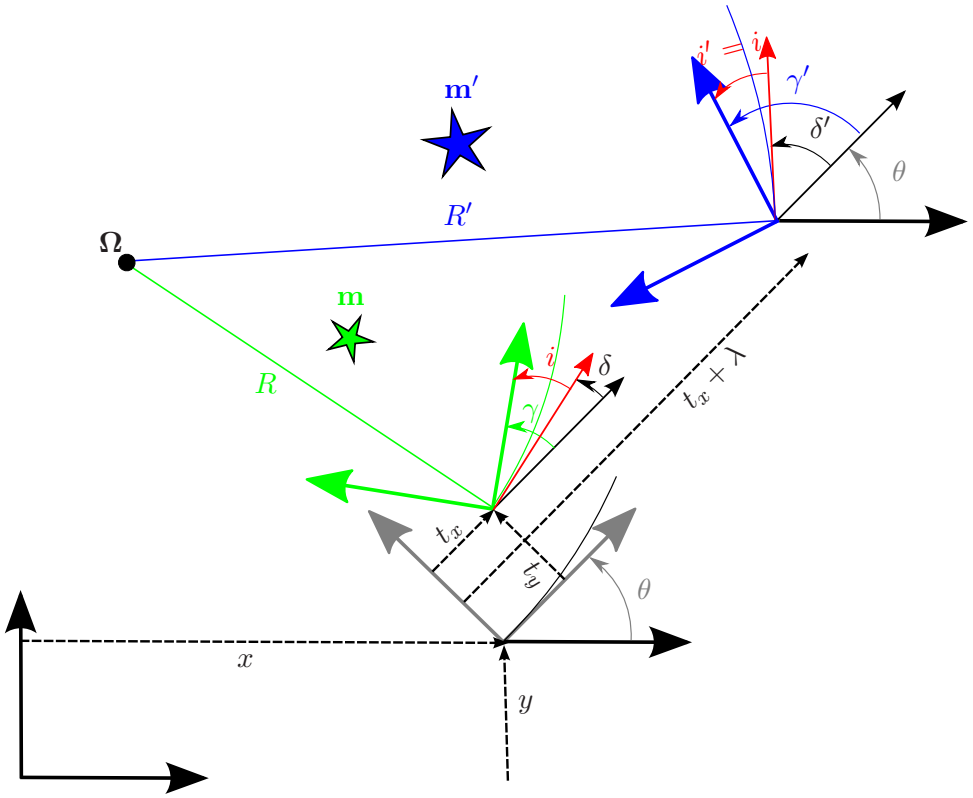


FIGURE B.4 – Différents paramètres pris en compte lors d'une translation du repère caméra de λ dans la direction x du repère robot — Le repère noir désigne le repère global. Le repère gris est le repère du robot (celui qui permet de définir le centre de rotation $\mathbf{\Omega}$. Le repère vert est le repère caméra original. Le repère bleu est le repère obtenu lorsqu'on translate le repère vert de λ dans la direction x du repère robot.

Calcul du centre de rotation

Le centre de rotation du robot (noté $\mathbf{\Omega}$) donné par sa position et son rayon de courbure V/ω (figure B.4).

$$\mathbf{\Omega} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} 0 \\ \frac{V}{\omega} \end{bmatrix} = \begin{bmatrix} x - \frac{V}{\omega}\sin\theta \\ y + \frac{V}{\omega}\cos\theta \end{bmatrix} \tag{B.38}$$

Définition des rayons de courbure

Les rayons de courbure sont donnés par la distance entre le centre de la caméra et le point $\mathbf{\Omega}$. Le point $\mathbf{\Omega}$ étant donné dans le repère global, il faut calculer les coordonnées de la caméra (x_c et y_c) dans le repère global :

$$\begin{cases} x_c &= x + t_x\cos\theta - t_y\sin\theta \\ y_c &= y + t_y\cos\theta + t_x\sin\theta \end{cases} \text{ et } \begin{cases} x'_c &= x + (t_x+\lambda)\cos\theta - t_y\sin\theta \\ y'_c &= y + t_y\cos\theta + (t_x+\lambda)\sin\theta \end{cases} \tag{B.39}$$

Les deux rayons de courbure R et R' sont donnés par :

$$\begin{cases} R &= \sqrt{\left(t_x\cos\theta - \left(t_y - \frac{V}{\omega}\right)\sin\theta\right)^2 + \left(t_x\sin\theta + \left(t_y - \frac{V}{\omega}\right)\cos\theta\right)^2} \\ R' &= \sqrt{\left((t_x+\lambda)\cos\theta - \left(t_y - \frac{V}{\omega}\right)\sin\theta\right)^2 + \left((t_x+\lambda)\sin\theta + \left(t_y - \frac{V}{\omega}\right)\cos\theta\right)^2} \end{cases} \tag{B.40}$$

Après simplification, on obtient :

$$\begin{cases} R &= \sqrt{t_x^2 + \left(t_y - \frac{V}{\omega}\right)^2} \\ R' &= \sqrt{(t_x+\lambda)^2 + \left(t_y - \frac{V}{\omega}\right)^2} \end{cases} \tag{B.41}$$

Calcul de γ'

L'orientation du repère caméra alternatif dans le repère du robot ne peut pas être la même que celle du repère caméra réel. Il faut en effet que l'orientatiosn du repère caméra alternatif respecte une contrainte sur l'angle avec la tangente du cercle. L'angle à conserver est la différence entre l'orientation absolue du repère caméra et l'orientation absolue de la tangente au cercle en ce point. En notant i cet angle, on a :

$$i = \gamma - \delta \tag{B.42}$$

où δ désigne l'angle de la tangente au cercle au niveau de la caméra (par rapport au repère du robot). D'après la figure B.4, l'angle δ est donné par :

$$\delta = \arctan2\left(t_x, \frac{V}{\omega} - t_y\right) \tag{B.43}$$

En combinant les équations B.42 et B.43, et en utilisant le fait que la valeur de i reste inchangée quelle que soit la position de la caméra, on peut calculer la valeur de γ' :

$$\gamma' = \gamma + \arctan2\left(t_x+\lambda, \frac{V}{\omega} - t_y\right) - \arctan2\left(t_x, \frac{V}{\omega} - t_y\right) \tag{B.44}$$

En supposant λ suffisament petit devant t_x, une linéarisation au premier ordre de l'équation B.44 donne :

$$\gamma' \approx \gamma + \lambda\frac{\frac{V}{\omega} - t_y}{t_x^2 + \left(\frac{V}{\omega} - t_y\right)^2} = \gamma + \lambda\frac{V\omega - \omega^2 t_y}{\omega^2 t_x^2 + (V - t_y\omega)^2} \tag{B.45}$$

L'équation B.45 nous permet de retrouver la composante de $\mathbf{n_5}$ associée à la composante γ.

Définition de la nouvelle position de l'amer

La définition de l'amer s'obtient par un changement d'échelle dans le repère de la caméra. Le facteur d'échelle est obtenu par le rapport des rayons de courbure. Soit ${}^c\mathbf{m}$ les coordonnées de l'amer dans le repère de la caméra. On a :

$$ {}^c\mathbf{m}' = \frac{R'}{R} \cdot {}^c\mathbf{m} \tag{B.46}$$

Par ailleurs, les coordonnées de ${}^c\mathbf{m}$ dans le repère global s'expriment en fonction de $\mathbf{m}$, des coordonnées du robot et des paramètres du repère caméra. On a donc :

$$ {}^c\mathbf{m} = \mathbf{R}(-\theta-\gamma) \cdot \left(\mathbf{m} - \begin{bmatrix} x + t_x\cos\theta - t_y\sin\theta \\ y + t_y\cos\theta + t_x\sin\theta \end{bmatrix} \right) \tag{B.47}$$

et

$$ {}^c\mathbf{m}' = \mathbf{R}(-\theta-\gamma') \cdot \left(\mathbf{m}' - \begin{bmatrix} x + (t_x+\lambda)\cos\theta - t_y\sin\theta \\ y + t_y\cos\theta + (t_x+\lambda)\sin\theta \end{bmatrix} \right) \tag{B.48}$$

où $\mathbf{R}(u)$ désigne la matrice de rotation d'angle u. En combinant les équations B.46 à B.48, on obtient l'expression de $\mathbf{m}'$:

$$ \mathbf{m}' = \frac{R'}{R}\mathbf{R}(\gamma'-\gamma) \cdot \begin{bmatrix} x_{(1)} - X \\ y_{(1)} - Y \end{bmatrix} + \begin{bmatrix} X + \lambda\cos\theta \\ Y + \lambda\sin\theta \end{bmatrix} \tag{B.49}$$

où on a posé :

$$ X = x + t_x\cos\theta - t_y\sin\theta \text{ et } Y = y + t_y\cos\theta + t_x\sin\theta \tag{B.50}$$

et où $x_{(1)}$ et $y_{(1)}$ désignent les coordonnées de $\mathbf{m}$. En supposant λ suffisament petit, les expressions de R'/R et $\mathbf{R}(\gamma'-\gamma)$ peuvent se simplifier :

$$ \begin{aligned} \frac{R'}{R} &= \frac{\sqrt{(t_x+\lambda)^2 + \left(t_y - \frac{V}{\omega}\right)^2}}{\sqrt{t_x^2 + \left(t_y - \frac{V}{\omega}\right)^2}} \\ &\approx 1 + \frac{\lambda t_x}{t_x^2 + \left(t_y - \frac{V}{\omega}\right)^2} = 1 + \lambda\frac{\omega^2 t_x}{\omega^2 t_x^2 + (V - \omega t_y)^2} \end{aligned} \tag{B.51}$$

$$ \mathbf{R}\left(\gamma'-\gamma\right) = \begin{bmatrix} \cos\left(\gamma'-\gamma\right) & -\sin\left(\gamma'-\gamma\right) \\ \sin\left(\gamma'-\gamma\right) & \cos\left(\gamma'-\gamma\right) \end{bmatrix} \approx \begin{bmatrix} 1 & -\lambda\frac{V\omega-\omega^2 t_y}{\omega^2 t_x^2 + (V-t_y\omega)^2} \\ \lambda\frac{V\omega-\omega^2 t_y}{\omega^2 t_x^2 + (V-t_y\omega)^2} & 1 \end{bmatrix} \tag{B.52}$$

Ainsi, une approximation au premier ordre de l'équation B.49 donne :

$$ \mathbf{m}' \approx \begin{bmatrix} 1 + \lambda\frac{\omega^2 t_x}{\omega^2 t_x^2 + (V-\omega t_y)^2} & -\lambda\frac{V\omega-\omega^2 t_y}{\omega^2 t_x^2 + (V-t_y\omega)^2} \\ \lambda\frac{V\omega-\omega^2 t_y}{\omega^2 t_x^2 + (V-t_y\omega)^2} & 1 + \lambda\frac{\omega^2 t_x}{\omega^2 t_x^2 + (V-\omega t_y)^2} \end{bmatrix} \cdot \begin{bmatrix} x_{(1)} - X \\ y_{(1)} - Y \end{bmatrix} + \begin{bmatrix} X + \lambda\cos\theta \\ Y + \lambda\sin\theta \end{bmatrix} \tag{B.53}$$

La première composante de $\mathbf{m}'$ est donc donnée par :

$$\begin{aligned}
x'_{(1)} &\approx x_{(1)} - X + \lambda\frac{\omega^2 t_x\left(x_{(1)}-X\right)-\left(V\omega-\omega^2 t_y\right)\left(y_{(1)}-Y\right)}{\left(V-\omega t_y\right)^2+\omega^2 t_x^2} + X + \lambda\cos\theta \\
&= x_{(1)} + \lambda\frac{\omega^2 t_x\left(x_{(1)}-X\right)-V\omega\left(y_{(1)}-Y\right)+\omega^2 t_y\left(y_{(1)}-Y\right)+\omega^2 t_x^2\cos\theta+\left(V-\omega t_y\right)^2\cos\theta}{\omega^2 t_x^2+\left(V-\omega t_y\right)^2} \\
&= x_{(1)} + \lambda\frac{V^2\cos\theta+t_x\omega^2\left(x_{(1)}-X+t_x\cos\theta\right)-\omega V\left(y_{(1)}-Y+2t_y\cos\theta\right)+t_y\omega^2\left(y_{(1)}-Y+t_y\cos\theta\right)}{\left(V-\omega t_y\right)^2+\omega^2 t_x^2} \\
&= x_{(1)} + \lambda\frac{\left\{\begin{array}{r} V^2\cos\theta+t_x\omega^2\left(x_{(1)}-x-t_x\cos\theta+t_y\sin\theta+t_x\cos\theta\right) \\ -\omega V\left(y_{(1)}-y-t_y\cos\theta-t_x\sin\theta+2t_y\cos\theta\right) \\ +t_y\omega^2\left(y_{(1)}-y-t_y\cos\theta-t_x\sin\theta+t_y\cos\theta\right)\end{array}\right\}}{\left(V-\omega t_y\right)^2+\omega^2 t_x^2} \\
&= x_{(1)} + \lambda\frac{V^2\cos\theta-\omega V\left(y_{(1)}-y-t_x\sin\theta+t_y\cos\theta\right)+t_x\omega^2\left(x_{(1)}-x\right)+t_y\omega^2\left(y_{(1)}-y\right)}{\left(V-\omega t_y\right)^2+\omega^2 t_x^2}
\end{aligned} \tag{B.54}$$

De même, la seconde composante de $\mathbf{m}'$ est donnée par :

$$\begin{aligned}
y'_{(1)} &\approx y_{(1)} - Y + \lambda\frac{\omega^2 t_x\left(y_{(1)}-Y\right)+\left(V\omega-\omega^2 t_y\right)\left(x_{(1)}-X\right)}{\left(V-\omega t_y\right)^2+\omega^2 t_x^2} + Y + \lambda\sin\theta \\
&= y_{(1)} + \lambda\frac{V^2\sin\theta+t_x\omega^2\left(y_{(1)}-Y+t_x\sin\theta\right)+\omega V\left(x_{(1)}-X-2t_y\sin\theta\right)-t_y\omega^2\left(x_{(1)}-X-t_y\sin\theta\right)}{\left(V-\omega t_y\right)^2+\omega^2 t_x^2} \\
&= x_{(1)} + \lambda\frac{\left\{\begin{array}{r} V^2\sin\theta+t_x\omega^2\left(y_{(1)}-y-t_y\cos\theta-t_x\sin\theta+t_x\sin\theta\right) \\ +\omega V\left(x_{(1)}-x-t_x\cos\theta+t_y\sin\theta-2t_y\sin\theta\right) \\ -t_y\omega^2\left(x_{(1)}-x-t_x\cos\theta+t_y\sin\theta-t_y\sin\theta\right)\end{array}\right\}}{\left(V-\omega t_y\right)^2+\omega^2 t_x^2} \\
&= x_{(1)} + \lambda\frac{V^2\sin\theta+\omega V\left(x_{(1)}-x-t_x\cos\theta-t_y\sin\theta\right)+t_x\omega^2\left(y_{(1)}-y\right)-t_y\omega^2\left(y_{(1)}-x\right)}{\left(V-\omega t_y\right)^2+\omega^2 t_x^2}
\end{aligned} \tag{B.55}$$

Les équations B.54 et B.55 nous permettent de retrouver les composantes de $\mathbf{n_5}$ associées à $x_{(1)}$ et $x_{(2)}$. On pourrait retrouver les coordonnées du vecteur $\mathbf{n_6}$ avec un calcul très similaire au calcul précédent, en supposant l'ajout sur t_y d'une petite valeur λ.

Au final, on vient de montrer que les modifications de la solution selon le modèle géométrique décrit précédemment permettent de retrouver les éléments du noyau de la matrice d'observabilité, à savoir $\lambda\mathbf{n_4}$ et $\lambda\mathbf{n_5}$ après linéarisation au premier ordre.

B.4.2.5 Limites de l'étude d'observabilité du système continu

Nous avons montré au paragraphe précédent que la matrice d'observabilité induite par ce système n'est jamais de rang plein. Une interprétation géométrique a confirmé ce résultat. Ceci nous dit que le système n'est pas **faiblement localement observable**. Ainsi, il n'est pas possible de discerner l'état

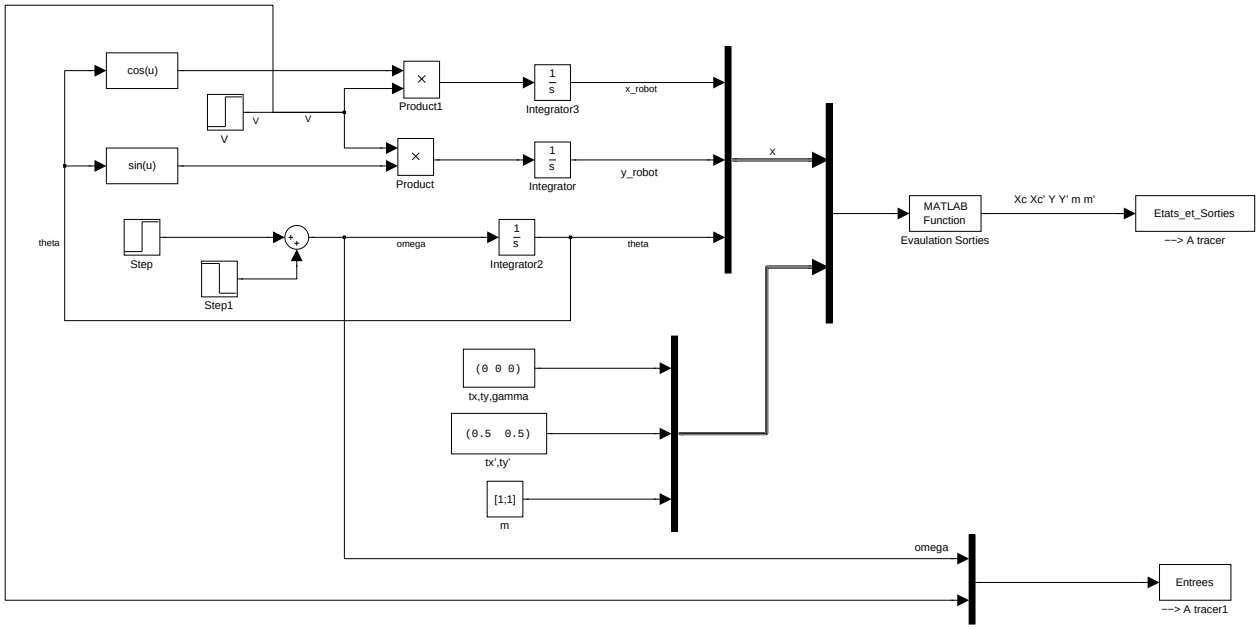


FIGURE B.5 – Schéma Simulink associé à l'illustration d'observabilité

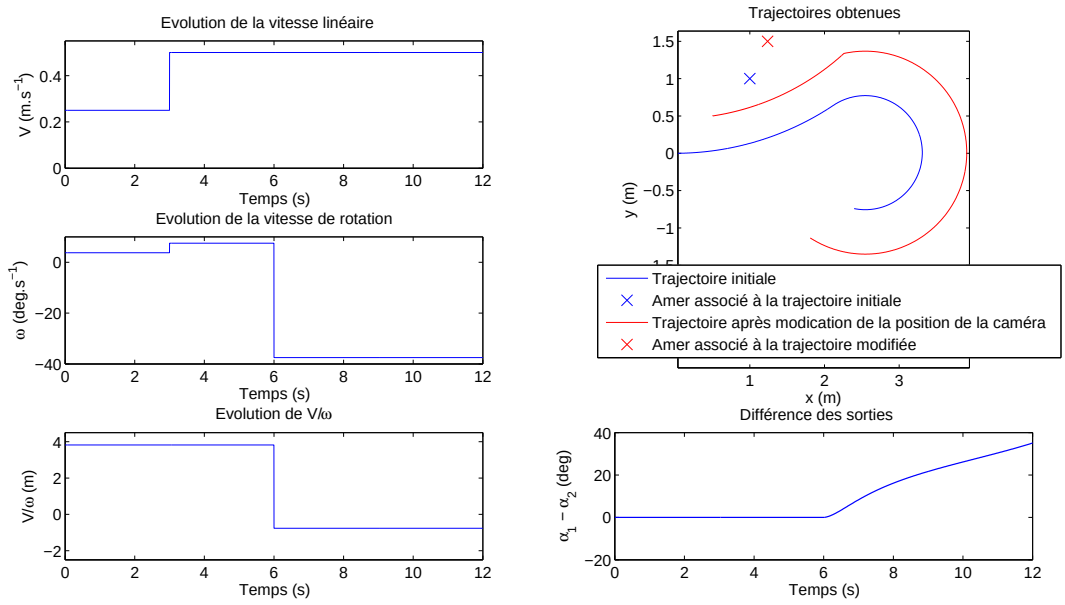


FIGURE B.6 – Résultats montrant l'observabilité du système après $t = 6s$. Nous avons défini une translation de [0.5 0.5] du repère de la caméra. Nous avons ensuite déduit les nouvelles valeurs pour γ et $\mathbf{m}$ à l'aide des considérations géométriques décrites précédemment. Nous avons obtenu $\gamma' \approx 8.52$deg et $\mathbf{m}' \approx [1.2382\ 1.5]$m (à l'origine, on avait $\gamma = 0$ et $\mathbf{m} = [1\ 1]$m).

initial instantanément : la dynamique du robot ne fournit pas assez d'information. Néanmoins, une étude attentive des conditions géométriques montre que pour que deux solutions restent indiscernables, une condition nécessaire est que le rapport des rayons de courbure reste constant (sinon le facteur d'échelle entre les deux cartes perd sa cohérence). Cette condition n'est plus réalisée dès que la vitesse de rotation du robot varie. C'est ce que nous avons mis en évidence à l'aide d'une simulation avec la boîte à outils Simulink de Matlab (figures B.5 et B.6). Nous avons simulé un début de trajectoire circulaire et avons calculé deux solutions différentes suivant la position de la caméra. Nous avons dans un second temps fait varier V et ω en maintenant V/ω (ie. le rayon de courbure) constant (entre les troisième et sixième seconde de la simulation). Dans ce cas, les mesures d'angle sont restées inchangées : il est alors impossible de distinguer les deux solutions. Enfin, nous avons fait varier le rayon de courbure de la trajectoire et avons alors constaté une différentiation des sorties : le système est donc bien observable.

Ainsi, on peut intuitivement distinguer deux phases dans l'observabilité du système :

1. Entre 0 et T (éventuellement, T peut tendre vers 0), le robot évolue avec une vitesse de rotation ω_0 : les différentes positions du repère de la caméra ne sont pas discernables
2. Après T, la vitesse de rotation change : le système est définitivement observable.

Ainsi, l'étude précédente conclut à une inobservabilité du système car elle est effectuée à ω fixé. Une étude prenant en compte le fait que le système est affine en la commande nous amènerait à calculer des dérivées de Lie dans les directions $\mathbf{f_1}$ et $\mathbf{f_2}$ définies par :

$$\begin{array}{l} \mathbf{f_1} = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \\ \mathbf{f_2} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \end{array} \tag{B.56}$$

permettrait de conclure à l'observabilité. Pour éviter cette étude qui impliquerait de coupler les dérivées de Lie " astucieusement" jusqu'à obtenir une matrice de rang 5 (ceci n'est pas trivial étant donné la forme de la fonction de mesures), nous proposons dans la suite une analyse d'observabilité basée sur une analyse discrète du système.

B.4.3 Observabilité du système discret

Pour conclure l'étude d'observabilité concernant le système augmenté, nous proposons une analyse discrète qui montre que l'ensemble des paramètres est *génériquement* observable. Nous présentons dans un premier temps la " méthodologie " du critère discret employé, puis son application.

B.4.3.1 Méthodologie du critère discret

Les études d'observabilité appliquées au cas des systèmes discrets ne sont pas nombreuses. On pourra notamment citer les travaux de [Martin *et al.*, 1991] ou [Moraal et Grizzle, 1995]. En général, l'idée est d'utiliser les équations d'évolution et de mesures et de calculer une matrice d'observabilité où la ligne k correspond à la dérivée des fonctions à l'instant k.

Dans notre cas, on ne s'intéresse qu'à l'observabilité des paramètres de la caméra. Le problème d'offset général sur la solution est déjà identifié et on ne cherche pas à le reproduire. En conséquence, si l'on suppose que $\mathbf{u}$ est connu et que la condition initiale sur la pose du robot est aussi connue, le modèle d'évolution du robot n'intervient plus dans notre étude d'observabilité. On supposera que les états du robot sont connus à chaque instant. Nous cherchons alors à savoir si on peut retrouver la position des amers et de la position de la caméra.

Dans le cas où nous avons un seul amer à observer (deux coordonnées x et y), chaque instant apporte une mesure. Le nombre d'inconnues est quant à lui égal à 5 (3 pour la position de la caméra et deux pour l'amer). Il faut donc au minimum 5 équations pour calculer ces paramètres. Nous allons donc chercher à savoir si la donnée des équations de mesures à 5 instants différents permet de déduire les paramètres voulus. Pour cela, nous cherchons si la matrice jacobienne associée à ces 5 équations est de rang 5. Si c'est le cas, alors la fonction considérée vérifie le théorème d'inversion locale. Ainsi, il existe une solution unique tant que l'on cherche les solutions dans un espace suffisament petit autour de la solution réelle. Dans ce cas, on dira que le système est observable.

B.4.3.2 Application

Définition et simplification de la matrice à étudier

Nous considérons 5 mesures d'un amer à 5 positions différentes du robot :

$$\alpha_{k,(1)} = \arctan2\left(y_{(1)} - y_k - t_y\cos\theta_k - t_x\cos\theta_k, x_{(1)} - x_k - t_x\cos\theta_k + t_y\sin\theta_k\right) - \theta_k - \gamma \;\forall k \in [\![1,5]\!] \tag{B.57}$$

La $k^{\text{ème}}$ ligne de la " matrice d'observabilité " $\mathbf{O}$ (telle que définie au paragraphe précédent) est donnée par :

$$\mathbf{L}[k] = \begin{bmatrix} -\sin\left(\theta_k+\gamma+\alpha_{k,(1)}\right) \\ \cos\left(\theta_k+\gamma+\alpha_{k,(1)}\right) \\ -\sin\theta_k\cdot\cos\left(\theta_k+\gamma+\alpha_{k,(1)}\right) + \cos\theta_k\cdot\sin\left(\theta_k+\gamma+\alpha_{k,(1)}\right) \\ -\cos\theta_k\cdot\cos\left(\theta_k+\gamma+\alpha_{k,(1)}\right) - \sin\theta_k\cdot\sin\left(\theta_k+\gamma+\alpha_{k,(1)}\right) \\ -(y_{(1)} - y_k - t_y\cos\theta_k - t_x\sin\theta_k)\sin\left(\theta_k+\gamma+\alpha_{k,(1)}\right) - (x_{(1)} - x_k - t_x\cos\theta_k + t_y\sin\theta_k)\cos\left(\theta_k+\gamma+\alpha_{k,(1)}\right) \end{bmatrix}^T \tag{B.58}$$

Après simplification, on obtient :

$$\mathbf{L}[k] = \begin{bmatrix} -\sin\left(\theta_k + \gamma + \alpha_{k,(1)}\right) \\ \cos\left(\theta_k + \gamma + \alpha_{k,(1)}\right) \\ \sin\left(\gamma + \alpha_{k,(1)}\right) \\ -\cos\left(\gamma + \alpha_{k,(1)}\right) \\ t_x \cos\left(\gamma + \alpha_{k,(1)}\right) + t_y \sin\left(\gamma + \alpha_{k,(1)}\right) - (y_{(1)} - y_k)\sin\left(\theta_k + \gamma + \alpha_{k,(1)}\right) - (x_{(1)} - x_k)\cos\left(\theta_k + \gamma + \alpha_{k,(1)}\right) \end{bmatrix}^T \tag{B.59}$$

D'après l'équation B.59, on a une relation triviale sur les colonnes de la matrice d'observabilité :

$$\mathbf{C}[5] = \mathbf{C}^* + y_{(1)}\mathbf{C}[1] - x_{(1)}\mathbf{C}[2] + t_y\mathbf{C}[4] - t_x\mathbf{C}[5] \tag{B.60}$$

où $\mathbf{C}[k]$ désigne la $k^{\text{ème}}$ colonne de la matrice d'observabilité et $\mathbf{C}^*$ est la matrice colonne définie par :

$$\mathbf{C}^* = \begin{bmatrix} y_1 \sin\left(\theta_1 + \gamma + \alpha_{1,(1)}\right) + x_1 \cos\left(\theta_1 + \gamma + \alpha_{1,(1)}\right) \\ \vdots \\ y_5 \sin\left(\theta_5 + \gamma + \alpha_{5,(1)}\right) + x_5 \cos\left(\theta_1 + \gamma + \alpha_{5,(1)}\right) \end{bmatrix} \tag{B.61}$$

En conséquence, le rang de la matrice **O est identique** à celui de la matrice $\mathbf{O}'$ dont les lignes sont données par :

$$\mathbf{L}'[k] = \begin{bmatrix} -\sin\left(\theta_k + \gamma + \alpha_{k,(1)}\right) \\ \cos\left(\theta_k + \gamma + \alpha_{k,(1)}\right) \\ \sin\left(\gamma + \alpha_{k,(1)}\right) \\ -\cos\left(\gamma + \alpha_{k,(1)}\right) \\ y_k \sin\left(\theta_k + \gamma + \alpha_{k,(1)}\right) + x_k \cos\left(\theta_k + \gamma + \alpha_{k,(1)}\right) \end{bmatrix}^T \tag{B.62}$$

On pourra remarquer que la condition de rang est indépendante de la position de la caméra dans le repère du robot ainsi que de la position de l'amer. [10]

Conditions sur les 4 premières colonnes

Nous cherchons dans un premier temps des conditions pour garantir que les 4 premières colonnes sont indépendantes. La matrice $\mathbf{O}'[:, 1:4]$ doit donc être de rang 4 :

$$\mathbf{O}'[:, 1:4] = \begin{bmatrix} -\sin\left(\theta_1 + \widetilde{\alpha_1}\right) & \cos\left(\theta_1 + \widetilde{\alpha_1}\right) & \sin\widetilde{\alpha_1} & -\cos\widetilde{\alpha_1} \\ -\sin\left(\theta_2 + \widetilde{\alpha_2}\right) & \cos\left(\theta_2 + \widetilde{\alpha_2}\right) & \sin\widetilde{\alpha_2} & -\cos\widetilde{\alpha_2} \\ -\sin\left(\theta_3 + \widetilde{\alpha_3}\right) & \cos\left(\theta_3 + \widetilde{\alpha_3}\right) & \sin\widetilde{\alpha_3} & -\cos\widetilde{\alpha_3} \\ -\sin\left(\theta_4 + \widetilde{\alpha_4}\right) & \cos\left(\theta_4 + \widetilde{\alpha_4}\right) & \sin\widetilde{\alpha_5} & -\cos\widetilde{\alpha_4} \\ -\sin\left(\theta_5 + \widetilde{\alpha_5}\right) & \cos\left(\theta_5 + \widetilde{\alpha_5}\right) & \sin\widetilde{\alpha_5} & -\cos\widetilde{\alpha_5} \end{bmatrix} \tag{B.63}$$

10. Ce résultat peut paraître surprenant dans un premier temps car on sait que le système sera inobservable si la caméra se déplace dans la direction de l'amer. Nous retrouverons néanmoins cette condition plus loin.

où on a posé $\widetilde{\alpha_k} = \gamma + \alpha_{k,(1)}\ \forall k \in [\![1\ 5]\!]$. $\widetilde{\alpha_k}$ représente l'angle de gisement par rapport à l'orientation du robot. On a donc :

$$\mathbf{O'}[:,1:4] = \left[\begin{array}{c|c|c|c} -\sin\theta_1 \cdot \cos\widetilde{\alpha_1} - \cos\theta_1 \cdot \sin\widetilde{\alpha_1} & \cos\theta_1 \cdot \cos\widetilde{\alpha_1} - \sin\theta_1 \cdot \sin\widetilde{\alpha_1} & \sin\widetilde{\alpha_1} & -\cos\widetilde{\alpha_1} \\ -\sin\theta_2 \cdot \cos\widetilde{\alpha_2} - \cos\theta_2 \cdot \sin\widetilde{\alpha_2} & \cos\theta_2 \cdot \cos\widetilde{\alpha_2} - \sin\theta_2 \cdot \sin\widetilde{\alpha_2} & \sin\widetilde{\alpha_2} & -\cos\widetilde{\alpha_2} \\ -\sin\theta_3 \cdot \cos\widetilde{\alpha_3} - \cos\theta_1 \cdot \sin\widetilde{\alpha_3} & \cos\theta_3 \cdot \cos\widetilde{\alpha_3} - \sin\theta_3 \cdot \sin\widetilde{\alpha_3} & \sin\widetilde{\alpha_3} & -\cos\widetilde{\alpha_3} \\ -\sin\theta_4 \cdot \cos\widetilde{\alpha_4} - \cos\theta_1 \cdot \sin\widetilde{\alpha_4} & \cos\theta_4 \cdot \cos\widetilde{\alpha_4} - \sin\theta_1 \cdot \sin\widetilde{\alpha_4} & \sin\widetilde{\alpha_4} & -\cos\widetilde{\alpha_4} \\ -\sin\theta_5 \cdot \cos\widetilde{\alpha_5} - \cos\theta_5 \cdot \sin\widetilde{\alpha_5} & \cos\theta_5 \cdot \cos\widetilde{\alpha_5} - \sin\theta_5 \cdot \sin\widetilde{\alpha_5} & \sin\widetilde{\alpha_5} & -\cos\widetilde{\alpha_5} \end{array}\right] \tag{B.64}$$

Génériquement, les 4 colonnes de la matrice $\mathbf{O'}[:,1:4]$ sont linéairement indépendantes. Néanmoins, on trouve 4 cas singuliers impliquant une chute de rang :

1. Les 5 angles θ_k sont égaux. Dans ce cas, les première et deuxième colonnes de $\mathbf{O'}[:,1:4]$ sont des combinaisons linéaires triviales des troisième et quatrième colonne. La matrice est alors de rang 2. Ainsi, il est indispensable que le robot ait effectué au moins une rotation,

2. Seul un angle θ_k est différent des autres. On montre en effet facilement [11] qu'il existe un noyau non nul lorsque par exemple $\theta_1 = \theta_2 = \theta_3 = \theta_4$. Intuitivement, cela implique que le robot doit avoir effectué un mouvement de rotation suffisament important. En revanche, si l'on a seulement $\theta_1 = \theta_2 = \theta_3$ (les deux autres valeurs étant quelconques) [12], alors les 4 colonnes sont indépendantes,

3. Les 5 angles $\widetilde{\alpha_k}$ sont égaux. Dans ce cas, on peut montrer que les deux premières colonnes de $\mathbf{O'}[:,1:4]$ sont liées et que la matrice est de rang 3. [13] Ceci n'est plus valable dès qu'un des $\widetilde{\alpha_k}$ est différent des autres. On est ici dans un cas particulier : **la sortie est constante**. Les trajectoires permettant d'obtenir une sortie constante tout en respectant la contrainte de non-honolomie sont réduites : il s'agit des cercle centrés sur l'amer (dans ce cas on a nécessairement $\cos\widetilde{\alpha_k} = 90\text{deg}$) ou des trajectoires en ligne droite dans la direction de l'amer (dans ce cas on a nécessairement $\cos\widetilde{\alpha_k} = 0\text{deg}$). Les autres possibilités permettant d'obtenir ces conditions nécessitent des manœuvres pendant lesquelles la sortie va varier (on suppose alors que l'on est capable d'échantilonner pendant ces manœuvres).

4. Le dernier cas correspond au cas où les angles $\theta_k + \widetilde{\alpha_k}$ sont égaux. L'équation B.63 montre alors que les deux premières colonnes sont trivialement liées. L'angle $\theta_k + \widetilde{\alpha_k}$ correspond à l'angle de gisement de l'amer exprimé par rapport à l'axe x du repère global. Lorsque cet angle est

11. A l'aide d'un logiciel de calcul formel comme Maple ou éventuellement " à la main " (mais le calcul n'est pas trivial).

12. Avec éventuellement $\theta_4 = \theta_5$.

13. Ceci quelles que soient les valeurs des θ_k, pourvu que l'on ne soit pas dans l'une des deux conditions précédentes.

constant, cela signifie que le robot se déplace dans l'axe de l'amer : il n'y a aucun parallaxe. Le système est donc naturellement inobservable.

Au final, nous avons montré des conditions nécessaires pour que les 4 premières colonnes soient indépendantes. Il est ainsi nécessaire que le robot ait bénéficié d'un minimum de rotation. On a par ailleurs retrouvé un cas trivial : la nécessité qu'il y ait du parallaxe.

Dernière colonne

D'après l'équation B.62, la dernière colonne est *en général* indépendante des autres lorsque le robot est en déplacement (x_k et y_k non constants). Néanmoins, on retrouve une simplification particulière de cette colonne lorsque le robot se déplace le long d'un cercle en respectant la contrainte de non-honolomie. Dans ce cas, cas, les coordonnées x_k et y_k s'écrivent :

$$\left\{ \begin{array}{rcl} x_k & = & C_x + R\cos\widetilde{\theta_k} \\ y_k & = & C_y + R\sin\widetilde{\theta_k} \end{array} \right. \tag{B.65}$$

où C_x et C_y désignent les coordonnées du centre du cercle, R le rayon du cercle et $\widetilde{\theta_k}$ l'angle permettant de repérer le robot sur le cercle. Lorsque la contrainte de non-honolomie est respectée, les dynamiques de θ_k et $\widetilde{\theta_k}$ sont identiques. On a donc

$$\forall k \in [\![1,5]\!] \quad \widetilde{\theta_k} - \theta_k = \beta \tag{B.66}$$

où β est un angle constant quelconque. Dans ce cas, chaque ligne de la cinquième colonne s'écrit :

$$\begin{array}{rcl} \mathbf{O'}[k,5] & = & \left(C_y + R\sin\widetilde{\theta_k}\right)\sin\left(\theta_k + \widetilde{\alpha_k}\right) + \left(C_x + R\cos\widetilde{\theta_k}\right)\cos\left(\theta_k + \widetilde{\alpha_k}\right) \\ & = & C_x\cos\left(\theta_k + \widetilde{\alpha_k}\right) + C_y\sin\left(\theta_k + \widetilde{\alpha_k}\right) + R\left(\sin\widetilde{\theta_k}\sin\left(\theta_k + \widetilde{\alpha_k}\right) + \cos\widetilde{\theta_k}\cos\left(\theta_k + \widetilde{\alpha_k}\right)\right) \\ & = & C_x\cos\left(\theta_k + \widetilde{\alpha_k}\right) + C_y\sin\left(\theta_k + \widetilde{\alpha_k}\right) + R\cos\left(\theta_k + \widetilde{\alpha_k} - \widetilde{\theta_k}\right) \\ & = & C_x\cos\left(\theta_k + \widetilde{\alpha_k}\right) + C_y\sin\left(\theta_k + \widetilde{\alpha_k}\right) + R\cos\left(\widetilde{\alpha_k} - \beta\right) \\ & = & C_x\cos\left(\theta_k + \widetilde{\alpha_k}\right) + C_y\sin\left(\theta_k + \widetilde{\alpha_k}\right) + R\cos\widetilde{\alpha_k}\cdot\cos\beta + R\sin\widetilde{\alpha_k}\cdot\sin\beta \end{array} \tag{B.67}$$

En notant $\mathbf{C'}[k]$ la $k^{\text{ème}}$ colonne de la matrice $\mathbf{O'}$, l'uilitsation des équations B.62 et B.67 donne :

$$\mathbf{C'}[5] = C_x\cdot\mathbf{C'}[1] + C_y\cdot\mathbf{C'}[2] + R\cos\beta\cdot\mathbf{C'}[3] + R\sin\beta\cdot\mathbf{C'}[4] \tag{B.68}$$

On a donc bien une chute de rang dans le cas d'une trajectoire circulaire. Le système est inobservable.

Remarque B.4 (Ligne droite)

On peut remarquer que le fait d'imposer les (x_k, y_k) à respecter une trajectoire rectiligne ne permet

pas de rendre le système inobservable, ce qui peut paraître surprenant. Néanmoins, dans le cas non-holonome, ce cas est inclu lorsque θ_k est invariant. On remarquera par ailleurs que si le robot évolue en ligne droite à θ_k constant pour les 3 premiers instants, puis s'arrête pour tourner sur place, le système sera observable. En effet, le fait d'avancer en ligne droite induit un rayon de courbure infini, alors que le fait de tourner sur placer induit une trajectoire circulaire pour la caméra (rayon de coubure fini voire nul lorsque la caméra est positionnée sur l'origine du repère robot). En conséquence, on retrouve bien une variation du rayon de courbure de la trajectoire de la caméra (malgré le fait que la trajectoire du robot est incluse dans une droite), ce qui rend logiquement le système observable.

B.4.4 Conclusion quant à l'observabilité du système augmenté

Nous avons présenté dans cette section une étude complète montrant l'observabilité du système discret. Dans un premier temps, l'étude du modèle continu a conduit à montrer que le système n'est pas faiblement localement observable. Néanmoins, un exemple Simulink nous a permis d'illustrer le fait que le système peut être observable après un certain temps : lorsque le rayon de courbure de la trajectoire de la caméra change.

Dans un second temps, nous avons effectué une analyse du système discret. Celle-ci nous a permis d'identifier un certain nombre de trajectoires problématiques. Lorsque le robot se déplace dans la direction d'un amer, lorsqu'il se déplace sans avoir effectué une rotation " suffisament longue " ou lorsqu'il se déplace sur un cercle, alors le système augmenté n'est pas observable. Dans tous les autres cas, le système est observable.

En conséquence, nous avons montré qu'il existe un jeu de commandes permettant de distinguer l'ensemble des états. Le système est donc observable. Nous avons de plus montré que les commandes admissibles incluent un changement du rayon de courbure de la trajectoire de la caméra. Ainsi, tant que le rayon de courbure de la trajectoire de la caméra est constant, l'estimateur ne peut pas améliorer l'estimation de la position du repère de la caméra.

B.5 Conclusion

Nous avons présenté dans cette annexe les principales propriétés d'observabilité du SLAM. Nous avons dans un premier temps rappelé les résultats classiques montrant que ce problème n'est pas observable. La raison de ce fait est liée au déplacement rigide (trajectoire et carte) qu'il est toujours possible de réaliser sans que les mesures en sortie ne changent. Ce problème est commun à toutes les

“ types ” de SLAM (*bearing-only*, *range and bearing*). Néanmoins, le fait de fixer la condition initiale de la position du robot permet de rendre le reste des états observables. Ceci rend donc légitime le fait de créer un estimateur pour résoudre ce problème.

Nous avons ensuite présenté une étude originale concernant l'observabilité du système lorsque la position de la caméra dans le repère du robot est inconnue (cette étude est propre au cas *bearing-only*). Celle-ci montre que le système reste observable, à condition que le système soit suffisament “ excité ” : **le rayon de courbure de la trajectoire doit varier**. En pratique, l'estimation de paramètres sera d'autant meilleure que la trajectoire change souvent de rayon de courbure.

Annexe C

Coefficients associés aux ellipses de confiance

Nous rappelons dans cette annexe comment calculer l'aire (resp. volume) des ellipses (resp. ellipsoïde) à P% de confiances associées aux densités de probabilités gaussienne.

C.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 :

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

D'après l'équation C.1, les zones isoprobables sont des ellipsoïdes centrés sur μ d'équation :

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

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 C.2. Lorsqu'on connaît μ, K et $\mathbf{\Sigma}$, la probabilité que $\mathbf{x}$ soit à l'intérieur de l'ellipsoïde défini en C.2 est :

$$P = \mathrm{P}\left((\mathbf{x}-\mu)^T \mathbf{\Sigma}^{-1}(\mathbf{x}-\mu) \leq K^2\right) \tag{C.3}$$

On a donc :

$$P = \iint_{ellipse} p(\mathbf{x})\,\mathbf{dx} \qquad \text{ou} \qquad P = \iiint_{ellipsoïde} p(\mathbf{x})\,\mathbf{dx} \tag{C.4}$$

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

C.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 :

$$\mathbf{\Sigma} = \mathbf{P}^T\mathbf{D}\mathbf{P} \tag{C.5}$$

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 C.2 est :

$$(\mathbf{Px})^T\,\mathbf{D}^{-1}\,(\mathbf{Px}) = K^2 \tag{C.6}$$

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 C.2) est identique au volume de l'ellipsoïde paramétré par $\mathbf{0}$, $\mathbf{D}$ et K.

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

C.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.

C.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 :

$$P = \iint_{ellipse} \frac{1}{\sqrt{(2\pi)^2 \det \mathbf{\Sigma}}} \exp\left(-\frac{1}{2}\mathbf{x}^T\mathbf{D}^{-1}\mathbf{x}\right)\mathbf{dx} \tag{C.7}$$

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

$$P = \iint_{ellipse} \frac{1}{2\pi\sigma_1\sigma_2} \exp\left(-\frac{x_1^2}{2\sigma_1^2} - \frac{x_2^2}{2\sigma_2^2}\right) dx_1 dx_2 \tag{C.8}$$

soit :

$$P = \frac{1}{2\pi\sigma_1\sigma_2} \iint_{ellipse} \exp\left(-\frac{x_1^2}{2\sigma_1^2} - \frac{x_2^2}{2\sigma_2^2}\right) dx_1 dx_2 \tag{C.9}$$

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

$$\left\{ \begin{array}{rcl} x_1 & = & \rho\sigma_1 \cos\theta \\ x_2 & = & \rho\sigma_1 \sin\theta \end{array} \right. \tag{C.10}$$

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 C.9 devient donc :

$$\begin{array}{rcl} P & = & \dfrac{1}{2\pi\sigma_1\sigma_2} \displaystyle\int_0^{2\pi}\int_0^K \sigma_1\sigma_2\rho \exp\left(-\frac{1}{2}\rho^2\right) d\rho d\theta \\ \Leftrightarrow P & = & \dfrac{2\pi\sigma_1\sigma_2}{2\pi\sigma_1\sigma_2} \displaystyle\int_0^K \exp\left(-\frac{1}{2}\rho^2\right) d\rho \\ \Leftrightarrow P & = & \left[-e^{-\rho^2/2}\right]_0^K \\ \Leftrightarrow P & = & 1 - e^{-K^2/2} \end{array} \tag{C.11}$$

On peut alors déduire de l'équation C.11 :

$$\boxed{K = \sqrt{-2\log(1-P)}} \tag{C.12}$$

C.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 :

$$\mathbf{x}^T\mathbf{D}^{-1}\mathbf{x} = K^2 \tag{C.13}$$

soit :

$$\frac{x_1^2}{(K\sigma_1)^2} + \frac{x_2^2}{(K\sigma_2)^2} = 1 \tag{C.14}$$

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 $\mathbf{\Sigma}$:

$$\boxed{\mathcal{A} = -2\pi \log(1-P) \sqrt{\det \mathbf{\Sigma}}} \tag{C.15}$$

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

$$\boxed{\mathcal{A} = 28.9351 \sqrt{\det \mathbf{\Sigma}}} \tag{C.16}$$

C.4 Cas 3D : ellipsoïdes de confiance

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

C.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 :

$$P = \iiint_{ellipsoïde} \frac{1}{\sqrt{(2\pi)^3 \det \mathbf{\Sigma}}} \exp\left(-\frac{1}{2}\mathbf{x}^T \mathbf{D}^{-1} \mathbf{x}\right) \mathbf{dx} \tag{C.17}$$

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

$$P = \frac{1}{2\pi\sqrt{2\pi}\sigma_1\sigma_2\sigma_3} \iiint_{ellipsoïde} \exp\left(-\frac{x_1^2}{2\sigma_1^2} - \frac{x_2^2}{2\sigma_2^2} - \frac{x_2^2}{2\sigma_3^2}\right) dx_1 dx_2 dx_3 \tag{C.18}$$

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

$$\left\{ \begin{array}{rcl} x_1 & = & \rho\sigma_1 \cos\varphi \cos\theta \\ x_2 & = & \rho\sigma_1 \cos\varphi \sin\theta \\ x_3 & = & \rho\sigma_3 \sin\varphi \end{array} \right. \tag{C.19}$$

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 C.18 devient :

$$P = \frac{1}{2\pi\sqrt{2\pi}} \int_0^{2\pi} \int_{-\pi/2}^{\pi/2} \int_0^K \rho^2 \exp\left(-\frac{1}{2}\rho^2\right) \cos\varphi d\rho d\varphi d\theta \tag{C.20}$$

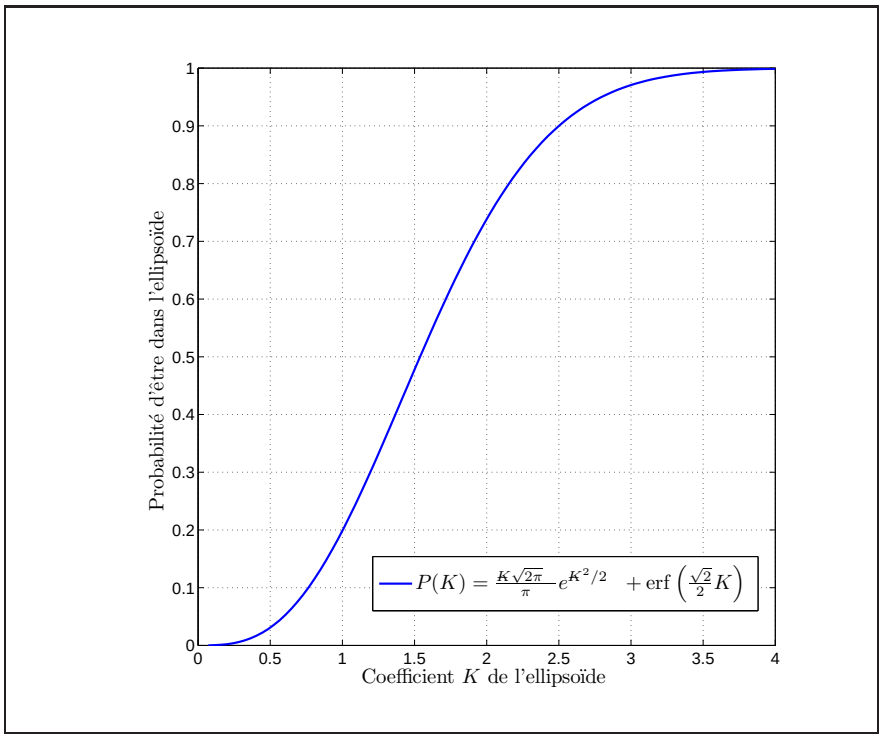


FIGURE C.1 – Probabilité d'être à l'intérieur de l'ellipsoïde en fonction de K

soit :

$$P = \frac{1}{2\pi\sqrt{2\pi}} \underbrace{\int_0^{2\pi} d\theta}_{2\pi} \underbrace{\int_{-\pi/2}^{\pi/2} \cos\varphi d\varphi}_{2} \int_0^K \rho^2 \exp\left(-\frac{1}{2}\rho^2\right) d\rho \tag{C.21}$$

on a donc :

$$P = \frac{\sqrt{2\pi}}{\pi} \int_0^K \rho^2 \exp\left(-\frac{1}{2}\rho^2\right) d\rho \tag{C.22}$$

L'intégrale à calculer dans l'équation C.22 n'est pas triviale. On choisit d'effectuer une intégration par parties :

$$\begin{aligned} \int_0^K \rho^2 \exp\left(-\frac{1}{2}\rho^2\right) d\rho &= \int_0^K \rho \cdot \rho e^{-\rho^2/2} d\rho \\ &= \left[-\rho e^{-\rho^2/2}\right]_0^K - \int_0^K -e^{-\rho^2/2} d\rho \\ &= -Ke^{-K^2/2} + \int_0^K e^{-\rho^2/2} d\rho \end{aligned} \tag{C.23}$$

L'intégrale restante dans l'équation C.23 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 :

$$\begin{array}{rccc} \text{erf}: & \mathbb{R} & \longrightarrow & [-1\ \ 1] \\ & x & \longmapsto & \dfrac{2}{\sqrt{2\pi}}\displaystyle\int_0^x e^{-t^2}dt \end{array} \tag{C.24}$$

Un changement de variable approprié permet d'obtenir :

$$\int_0^K -e^{-\rho^2/2}d\rho = \frac{\sqrt{2\pi}}{2}\text{erf}\left(\frac{\sqrt{2}}{2}K\right) \tag{C.25}$$

En utilisant les équations C.22, C.23 et C.25, on obtient :

$$\boxed{P = \frac{-K\sqrt{2\pi}}{\pi}e^{-K^2/2} + \text{erf}\left(\frac{\sqrt{2}}{2}K\right)} \tag{C.26}$$

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 graphe de P est donné sur la figure C.1.

Mais l'expression de P donnée dans l'équation C.26 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 :

$$\boxed{K = 3.368214175}$$

C.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 $\mathbf{\Sigma}$:

$$\boxed{\mathcal{V} = \tfrac{4}{3}\pi K^3\sqrt{\det\mathbf{\Sigma}}} \tag{C.27}$$

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

$$\boxed{\mathcal{V} = 160.0618\sqrt{\det\mathbf{\Sigma}}} \tag{C.28}$$

Annexe D

Changement du point de conditionnement

Nous avons présenté dans le chapitre 8 un théorème permettant de construire des cartes locales prenant en compte l'intégralité des mesures passées. Celui-ci est basé sur l'utilisation de l'équation suivante : [1]

$$p\left(\mathbf{x}_{t_1+1:t_2}, \mathbf{m}^2, \mathbf{m}^3 | \mathbf{z}_{0:t_1}, \mathbf{u}_{0:t_1}, \mathbf{z}_{t_1+1:t_2}, \mathbf{u}_{t_1+1:t_2}, \mathbf{x}_{t_1}\right) \propto p\left(\mathbf{x}_{t_1+1:t_2}, \mathbf{m}^2, \mathbf{m}^3 | \mathbf{z}_{t_1+1:t_2} \mathbf{u}_{t_1+1:t_2}, \mathbf{x}_{t_1}\right) \times p\left(\mathbf{m}^2 | \mathbf{z}_{0:t_1} \mathbf{u}_{0:t_1}, \mathbf{x}_{t_1}\right) \tag{D.1}$$

Lors de la description du théorème, nous avions laissé en suspens le calcul de la densité de probabilité reconditionnée $p\left(\mathbf{m}^2 | \mathbf{z}_{0:t_1} \mathbf{u}_{0:t_1}, \mathbf{x}_{t_1}\right)$. Nous proposons dans cette annexe de voir en détail ce calcul.

Cette annexe s'articule en 4 parties. Nous montrons dans la section D.1 que pour le cas de la première carte, cette densité de probabilité peut être déduite facilement à l'aide du théorème classique de reconditionement des variables aléatoires gaussiennes. Nous verrons alors que cette méthode ne peut être utilisée que sur la première carte. La section D.2, quant à elle, est consacrée à la définition du nouveau problème. Nous introduirons l'ensemble des hypothèses ainsi que les nouvelles notations. La méthode détaillée pour aboutir au résultat est présentée dans la section D.3. Nous concluons enfin par une comparaison de notre méthode avec l'algorithme introduit dans [Piniés et Tardós, 2008].

D.1 Méthode de calul sur la première carte

D.1.1 Utilisation du théorème de conditionnement

La toute première carte est calculée sans l'utilisation d'aucune information *a priori* (à l'exception de la pose initiale du robot fixée à zéro). Dans ce cas, il s'agit de l'algorithme de SAM classique qui

1. Voir chapitre 8 pour la description des notations.

est utilisé pour le calcul de la solution. D'après l'équation 1.89 page 57, il est possible de calculer directement les paramètres d'information associés à $p\left(\mathbf{x}_{0:t_1}, \mathbf{m}^1, \mathbf{m}^2 | \mathbf{z}_{0:t_1} \mathbf{u}_{1:t_1}\right)$.[2] **On dispose ainsi des paramètres de la densité jointe, incluant à la fois $\mathbf{x}_0$ et le reste de la trajectoire**. Disposant de la densité jointe, on peut utiliser le théorème de conditionnement des densités de probabilité gaussiennes pour conditionner par rapport à $\mathbf{x}_{t_1} = \mu_{t_1}$ (théorème 1.4 page 23).

Remarque D.1 (Choix de la valeur pour conditionner)

Il est a priori *possible d'utiliser n'importe quelle valeur pour appliquer le théorème de conditionnement des densités de probabilité gaussiennes. Néanmoins, il convient de remarquer que les paramètres d'information obtenus ne sont valables qu'autour des points de fonctionnement utilisés. Admettons par exemple que l'estimation de la pose du robot à t_1 soit μ_{t_1} (supposée très loin de zero). Pour reconditionner la densité de probabilité, il est préférable d'utiliser la valeur μ_{t_1}. Si l'on veut conditionner par rapport à $\mathbf{x}_{t_1} = \mathbf{0}$, on pourra dans un premier temps conditionner par rapport à $\mathbf{x}_{t_1} = \mu_{t_1}$ puis transporter la solution avec une transformation déterministe pour amener $\mathbf{x}_{t_1}$ à zéro (cette transformation est exacte : elle implique une translation/rotation du vecteur d'information et une rotation de la matrice d'information, voir proposition D.1). Utiliser le théorème 1.4 pour conditionner directement par rapport à $\mathbf{x}_{t_1} = \mathbf{0}$ donnerait des résultats désastreux (cf. figure D.1).*

Proposition D.1 (Transport des paramètres d'information)

Soit $\mathbf{x}$ un vecteur aléatoire dont les paramètres d'information sont ξ et $\mathbf{\Omega}$ et dont les paramètres classiques sont μ et $\mathbf{\Sigma}$. Soient $\mathbf{R}$ une matrice de rotation et $\mathbf{t}$ un vecteur, tous deux déterministes et donnés. Soit $\mathbf{x}'$ la variable aléatoire définie par $\mathbf{x}' = \mathbf{R}\mathbf{x} + \mathbf{t}$. Les paramètres associés à $\mathbf{x}'$ sont donnés par :

$$\left.\begin{aligned} \mu' &= \mathbf{R}\mu + \mathbf{t} \\ \mathbf{\Sigma}' &= \mathbf{R}\mathbf{\Sigma}\mathbf{R}^T \end{aligned}\right\} \text{ pour les paramètres classiques}$$
$$\left.\begin{aligned} \xi' &= \mathbf{R}\left(\xi + \mathbf{\Omega}\mathbf{R}^T\mathbf{t}\right) \\ \mathbf{\Omega}' &= \mathbf{R}\mathbf{\Omega}\mathbf{R}^T \end{aligned}\right\} \text{ pour les paramètres d'information} \tag{D.2}$$

La preuve concernant les paramètres classiques est immédiate et fait intervenir les règles de propagations classiques des matrices de variances-covariances. La partie concernant les paramètres d'information utilisent la relation $\mathbf{R}\mathbf{R}^T = \mathbf{I}$ ainsi que les relations de passage entre les paramètres classiques et les paramètres d'information.

2. On avait alors montré qu'il était nécessaire de conditionner cette densité de probabilité par rapport à $\mathbf{x}_0$, sans quoi la matrice d'information n'était pas inversible.

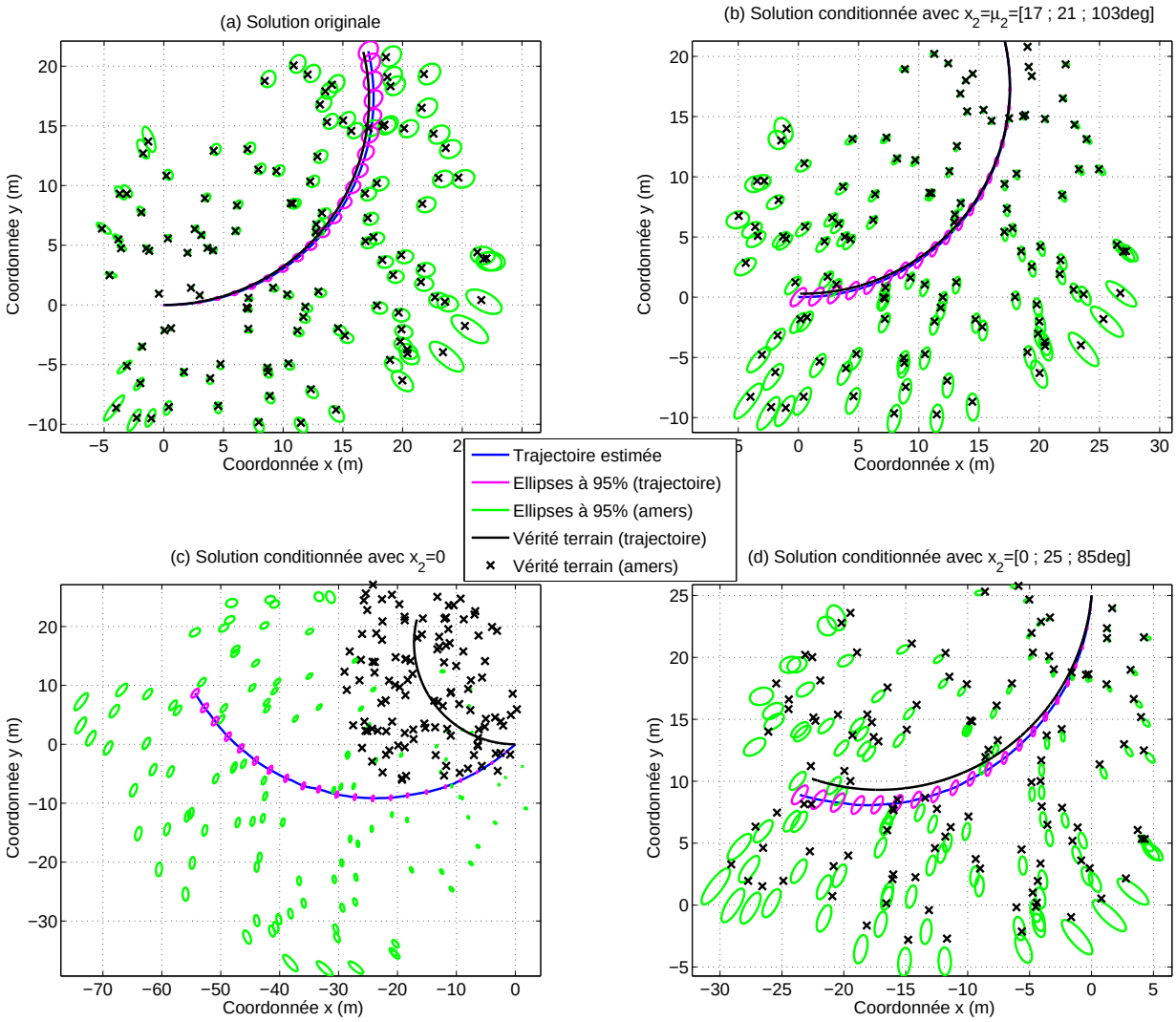


FIGURE D.1 – Reconditionnement par rapport à diverses valeurs de $\mathbf{x}_2$ dans le cas d'une trajectoire simulée circulaire. Dans les cas (b–d), la vérité terrain correspond au transport de la vérité terrain d'origine, de sorte à faire coïncider la valeur finale de la trajectoire — (a) Solution originale, conditionnée par rapport à $\mathbf{x}_1 = \mathbf{0}$. On peut voir que le résultat initial est consistant. Le reconditionnement doit donc l'être également – (b) Reconditionnement par rapport à $\mathbf{x}_2 = \mu_2$. Dans ce cas, la solution est parfaite. L'ensemble des ellipses de confiance incluent bien la vérité terrain, et on voit bien que les incertitudes sont plus grandes autour de la position initiale. On pourrait alors très bien transporter la solution de sorte à changer la valeur de $\mathbf{x}_2$ graĉe à l'équation D.2 – (c) Cas où on applique le théorème de conditionnement avec $\mathbf{x}_2 = \mathbf{0}$. Ici, le résultat est très inconsistant. La trajectoire obtenue n'est même plus tangente à la vérité terrain en $\mathbf{x}_2$ – (d) Cas où on applique le théorème avec une valeur de $\mathbf{x}_2$ loin de μ_2 pour la partie translation, mais " assez proche " pour la partie rotation. Le résultat est presque consistant et est de meilleure qualité que celui de (c). On constate finalement que la partie qui pose problème dans le conditionnement est la troisième composante (associée à l'angle de rotation). La solution n'est alors valable que dans un bassin de convergence assez serré. En ce qui concerne le positionnement, il n'y a finalement pas de limitation. Ceci est logique et provient du fait que dans le cas d'une translation pure, la transformation de l'espérance et du vecteur d'information est une application linéaire (les matrices d'information et de variances-covariances restent quant à elles constantes).

D.1.2 Autres cartes : indisponibilité de la densité jointe

Considérons désormais le cas de la seconde carte. A la fin de celle-ci, la densité de probabilité finale s'obtient par application de l'équation D.1. Pour commencer une nouvelle carte afin d'appliquer à nouveau le théorème, il faut que l'on soit capable de reconditionner la densité de probabilité obtenue avec l'équation D.1. L'équation D.1 **ne nous donne que la densité de probabilité déjà conditionnée par rapport à $\mathbf{x}_{t_1}$**. Contrairement au cas précédent, nous ne disposons plus de la densité jointe, si bien qu'une application du théorème de conditionnement sur le résultat de l'équation D.1 conduirait à conditionner la solution à la fois par rapport à $\mathbf{x}_{t_1}$ et à la fois par rapport à $\mathbf{x}_{t_2}$; ceci n'est pas le but cherché et n'a pas " d'interprétation physique ".

Il est donc nécessaire de développer une méthode alternative à la simple application du théorème de conditionnement afin de résoudre ce problème de déplacement du point connu.

D.2 Nouvelle formulation du problème et hypothèses

D.2.1 Formulation du problème

Nous cherchons dans cette section à résoudre le problème du reconditionnement de la solution par rapport à la dernière position de la trajectoire.

Dans la suite de cette annexe, nous nous intéressons à trouver la densité de probabilité $p\left(\mathbf{m}^2, \mathbf{m}^3 | \mathbf{z}_{0:t_2}, \mathbf{u}_{0:t_2}, \mathbf{x}_{t_2} = \mu_2\right)$ à partir de la densité de probabilité connue $p\left(\mathbf{x}_{t_1+1:t_2}, \mathbf{m}^2, \mathbf{m}^3 | \mathbf{z}_{0:t_2}, \mathbf{u}_{0:t_2}, \mathbf{x}_{t_1} = \mathbf{0}\right)$. Dans la suite, nous allons résoudre ce problème en utilsant uniquement la connaissance de $p\left(\mathbf{x}_{t_2}, \mathbf{m}^2, \mathbf{m}^3 | \mathbf{z}_{0:t_2}, \mathbf{u}_{0:t_2}, \mathbf{x}_{t_1} = \mathbf{0}\right)$ (qui peut être déduite par marginalisation de la précédente densité).

Par ailleurs, les méthodes que nous employons utilisent une approximation au premier ordre de certaines variables. Nous prenons en compte la remarque D.1 pour le choix de la valeur de conditionnement. μ_2 est donc choisi compatible avec les contraintes de linéarisation. Pour cela, nous choisissons de fixer μ_2 à la valeur du point de fonctionnement pour les linéarisations. [3]

Sans perte de généralité, nous simplifions les notations du problème en utilisant des notations plus génériques :

Position initiale : $\mathbf{x}_{t_1} \longrightarrow \mathbf{x}_1$

3. Pour rappel, ce choix s'effectue sans aucune perte de généralité dans le raisonnement. En effet, une fois le reconditionnement effectué avec $\mathbf{x}_2 = \mu_2$, il sera toujours possible d'introduire une transformation rigide de la solution de sorte à utiliser n'importe quelle autre valeur souhaitée pour conditionner $\mathbf{x}_2$ (équation D.2).

Position finale : $\mathbf{x}_{t_2} \longrightarrow \mathbf{x}_2$

Ensemble des amers : $\{\mathbf{m}^2, \mathbf{m}^3\} \longrightarrow \mathbf{m}$

Mesures : $\{\mathbf{z}_{0:t_2}, \mathbf{u}_{0:t_2}\} \longrightarrow \mathbf{z}$

Le problème peut être reformulé ainsi :

Comment calculer la densité $p(\mathbf{m}|\mathbf{z}, \mathbf{x_2} = \mu_2)$ *à partir des paramètres de* $p(\mathbf{x_2}, \mathbf{m}|\mathbf{z}, \mathbf{x_1} = \mathbf{0})$ *?*

D.2.2 Hypothèses

Dans les développements qui suivent, nous supposons que $p(\mathbf{x_2}, \mathbf{m}|\mathbf{z}, \mathbf{x_1} = \mathbf{0})$ est connu. Il s'agit d'une denstité de probabilité supposée gaussienne dont les paramètres d'information ξ^0 et $\mathbf{\Omega}^0$ sont donnés par : [4]

$$\xi^0 = \begin{bmatrix} \xi^0_2 \\ \xi^0_{\mathbf{m}} \end{bmatrix} \qquad \text{et} \qquad \mathbf{\Omega}^0 = \begin{bmatrix} \mathbf{\Omega}^0_{22} & \mathbf{\Omega}^0_{2\mathbf{m}} \\ \mathbf{\Omega}^0_{\mathbf{m}2} & \mathbf{\Omega}^0_{\mathbf{m}2} \end{bmatrix} \tag{D.3}$$

L'estimation de l'espérance de $\mathbf{m}$ et de $\mathbf{x}_2$ est donnée par $\left(\mathbf{\Omega}^0\right)^{-1}\xi^0$. La valeur obtenue constitue la meilleure hypothèse possible quant à la linéarisation des équations. Soit $\mu_{\mathbf{m}}$ la valeur du point de fonctionnement associé à $\mathbf{m}$ (pour rappel, le point de fonctionnement associé à $\mathbf{x}_1$ est $\mathbf{0}$, et celui associé à $\mathbf{x}_2$ est μ_2). On suppose donc que μ_2 et $\mu_{\mathbf{m}}$ vérifient l'équation suivante :

$$\begin{bmatrix} \xi^0_2 \\ \xi^0_{\mathbf{m}} \end{bmatrix} = \begin{bmatrix} \mathbf{\Omega}^0_{22} & \mathbf{\Omega}^0_{2\mathbf{m}} \\ \mathbf{\Omega}^0_{\mathbf{m}2} & \mathbf{\Omega}^0_{\mathbf{m}2} \end{bmatrix} \cdot \begin{bmatrix} \mu_2 \\ \mu_{\mathbf{m}} \end{bmatrix} \Longleftrightarrow \begin{cases} \xi^0_2 &= \mathbf{\Omega}^0_{22}\mu_2 + \mathbf{\Omega}^0_{2\mathbf{m}}\mu_{\mathbf{m}} \\ \xi^0_{\mathbf{m}} &= \mathbf{\Omega}^0_{\mathbf{m}2}\mu_2 + \mathbf{\Omega}^0_{\mathbf{mm}}\mu_{\mathbf{m}} \end{cases} \tag{D.4}$$

D.3 Résolution

Nous décrivons dans cette section tous les développements permettant d'obtenir le résultat souhaité. Les calculs effectués sont appliqués au cas du SLAM à 3 degrés de liberté avec des amers 2D (la généralisation avec des amers 3D est immédiate, et celle avec un modèle à 6 degrés de liberté complexifie juste certains calculs).

Nous n'allons pas calculer la densité voulue directement. Dans un premier temps, nous cherchons les paramètres de la densité jointe $p(\mathbf{x}_1, \mathbf{m}|\mathbf{z}, \mathbf{x_2} = \mu_2)$. La solution finale sera finalement obtenue en utilisant le théorème classique de marginalisation.

4. L'exposant 0 renvoie à $\mathbf{x_1} = \mathbf{0}$.

D.3.1 Equation principale

En utilisant la règle de Bayes, la densité de probabilité $p(\mathbf{x_1}, \mathbf{m}|\mathbf{z}, \mathbf{x_2} = \mu_2)$ peut être factorisée de la façon suivante :

$$p(\mathbf{x_1}, \mathbf{m}|\mathbf{z}, \mathbf{x_2} = \mu_2) = p(\mathbf{m}|\mathbf{z}, \mathbf{x_1}, \mathbf{x_2} = \mu_2) .p(\mathbf{x_1}|\mathbf{z}, \mathbf{x_2} = \mu_2) \tag{D.5}$$

Dans l'équation D.5, $\mathbf{x_1}$ et $\mathbf{m}$ sont des *variables*. Les termes constants sont uniquement μ_2 et $\mathbf{z}$. En conséquence, le premier facteur de l'équation D.5 ($p(\mathbf{m}|\mathbf{z}, \mathbf{x_1}, \mathbf{x_2} = \mu_2)$) est une fonction à la fois de $\mathbf{m}$ et de $\mathbf{x_1}$ (bien que $\mathbf{x}_1$ apparaisse dans le conditionnement).

Le lecteur pourra remarquer que le premier facteur de cette factorisation n'a pas de sens physique, ce qui est dû au fait que l'on conditionne à la fois par rapport à la pose initiale et la pose finale. Néanmoins, le calcul de ce facteur n'est qu'un artifice de calcul intermédiaire.

Dans la suite, nous allons nous intéressons à retrouver $p(\mathbf{m}|\mathbf{z}, \mathbf{x_1}, \mathbf{x_2} = \mu_2)$ et $p(\mathbf{x_1}|\mathbf{z}, \mathbf{x_2} = \mu_2)$.

D.3.2 Calcul de $p(\mathbf{m}|\mathbf{z}, \mathbf{x_1}, \mathbf{x_2} = \mu_2)$

Le calcul de $p(\mathbf{m}|\mathbf{z}, \mathbf{x_1}, \mathbf{x_2} = \mu_2)$ implique de conditionner la probabilité de $\mathbf{m}$ par rapport aux poses $\mathbf{x_1}$ et $\mathbf{x_2} = \mu_2$. Ceci peut être réalisé en utilisant le théorème de conditionnement sur $p(\mathbf{x_2}, \mathbf{m}|\mathbf{z}, \mathbf{x_1})$. Néanmoins, nous ne disposons pas de cette dernière densité de probabilité dans le cas général ; nous ne possédons que son instanciation avec $\mathbf{x_1} = \mathbf{0}$. En conséquence, nous allons exprimer $p(\mathbf{x_2}, \mathbf{m}|\mathbf{z}, \mathbf{x_1})$ pour n'importe quelle valeur de $\mathbf{x_1}$.

D.3.2.1 Représentation de $p(\mathbf{x_2}, \mathbf{m}|\mathbf{z}, \mathbf{x_1})$ comme une fonction de $\mathbf{x_1}$

Calculer $p(\mathbf{x_2}, \mathbf{m}|\mathbf{z}, \mathbf{x_1} = \mathbf{0})$ signifie que l'on cherche la distribution jointe des variables aléatoires $\mathbf{m}$ et $\mathbf{x_2}$ lorsque la position initiale est connue et fixée à zéro. Une formulation équivalente consiste à dire que l'on cherche la distribution jointe de $\mathbf{m}$ et $\mathbf{x_2}$ **exprimée dans le repère associé à la position initiale du robot**. Supposons désormais que la position initiale du robot *soit connue* et vaille $\mathbf{x_1} = [x_1, y_1, \theta_1]^T$ (plus nécessairement zéro). Définissons par ailleurs les variables suivantes :

- $\mathbf{m}^1$ et $\mathbf{x_2}^1$ les coordonnées des amers et de la position finale du robot exprimées dans **le repère associé à la position initiale du robot**,
- $\mathbf{m}^g$ et $\mathbf{x_2}^g$ les coordonnées des amers et de la position finale du robot exprimées dans **le repère global**.

On a donc :

$$\mathbf{x_2}^g = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{x_2}^1 + \begin{bmatrix} x_1 \\ y_1 \\ \theta_1 \end{bmatrix} \tag{D.6}$$

et

$$\mathbf{m}^g = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & & & & & & \\ \sin\theta_1 & \cos\theta_1 & 0 & & & & & & \\ 0 & 0 & 1 & & & & & & \\ & & & \ddots & \ddots & \ddots & & & \\ & & & \ddots & \ddots & \ddots & & & \\ & & & \ddots & \ddots & \ddots & & & \\ & & & & & & \cos\theta_1 & -\sin\theta_1 & 0 \\ & & & & & & \sin\theta_1 & \cos\theta_1 & 0 \\ & & & & & & 0 & 0 & 1 \end{bmatrix} \mathbf{m}^1 + \begin{bmatrix} x_1 \\ y_1 \\ 0 \\ \vdots \\ x_1 \\ y_1 \\ 0 \end{bmatrix} \tag{D.7}$$

Dans le but d'alléger l'écriture de l'équation D.7, nous définissons les matrices :

$$\mathbf{R}(\mathbf{x_1}) = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \overline{\mathbf{R}}(\mathbf{x}1) = \mathrm{diag}(\mathbf{R}(\mathbf{x_1}), \ldots, \mathbf{R}(\mathbf{x_1})) \qquad \mathbf{H} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{D.8}$$

On obtient alors :

$$\begin{bmatrix} \mathbf{x_2}^g \\ \mathbf{m}^g \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{R}(\mathbf{x_1}) & \mathbf{0} \\ \mathbf{0} & \overline{\mathbf{R}}(\mathbf{x_1}) \end{bmatrix}}_{\mathfrak{R}(\mathbf{x_1})} \begin{bmatrix} \mathbf{x_2}^1 \\ \mathbf{m}^1 \end{bmatrix} + \underbrace{\begin{bmatrix} \mathbf{x_1} \\ \mathbf{H}\mathbf{x_1} \end{bmatrix}}_{\mathbf{T}(\mathbf{x_1})} \tag{D.9}$$

En conséquence, lorsque $\mathbf{x_1}$ est différent de zéro, $[\mathbf{x_2}^{gT}, \mathbf{m}^{gT}]^T$ et $[\mathbf{x_2}^{1T}\mathbf{m}^{1T}]^T$ diffèrent uniquement par une matrice de rotation et un vecteur de translation. Il s'agit d'une transformation affine paramétrée par $\mathbf{x_1}$. La densité $p(\mathbf{x_2}, \mathbf{m}|\mathbf{z}, \mathbf{x_1})$ est donc gaussienne et ses paramètres d'information ($\tilde{\xi}(\mathbf{x_1})$ et $\tilde{\mathbf{\Omega}}(\mathbf{x_1})$) sont donnés par le transport de ξ^0 et de $\mathbf{\Omega}^0$:

$$\tilde{\xi}(\mathbf{x_1}) = \mathfrak{R}(\mathbf{x_1})\left(\xi^0 + \mathbf{\Omega}^0\left(\mathfrak{R}(\mathbf{x_1})\right)^{-1}\mathbf{T}(\mathbf{x_1})\right) \qquad \text{et} \qquad \tilde{\mathbf{\Omega}}(\mathbf{x_1}) = \mathfrak{R}(\mathbf{x_1})\mathbf{\Omega}^0\left(\mathfrak{R}(\mathbf{x_1})\right)^{-1} \tag{D.10}$$

La figure D.2 illustre le changement des paramètres de la densité cherchée en fonction de $\mathbf{x_1}$.

Tous calculs faits, la substitution des équations D.3 et D.9 dans l'équation D.10 donne :

$$\tilde{\xi}(\mathbf{x_1}) = \begin{bmatrix} \tilde{\xi}_2(\mathbf{x_1}) \\ \tilde{\xi}_\mathbf{m}(\mathbf{x_1}) \end{bmatrix} = \begin{bmatrix} \mathbf{R}(\mathbf{x_1})\Big(\xi_2^0 + \left(\Omega_{2\mathbf{m}}^0\overline{\mathbf{R}}(\mathbf{x_1})^T\mathbf{H} + \Omega_{22}^0\mathbf{R}(\mathbf{x_1})^T\right)\mathbf{x_1}\Big) \\ \overline{\mathbf{R}}(\mathbf{x_1})\Big(\xi_\mathbf{m}^0 + \left(\Omega_{\mathbf{m}2}^0\mathbf{R}(\mathbf{x_1})^T + \Omega_{\mathbf{mm}}^0\overline{\mathbf{R}}(\mathbf{x_1})^T\mathbf{H}\right)\mathbf{x_1}\Big) \end{bmatrix} \tag{D.11}$$

et

$$\tilde{\mathbf{\Omega}}(\mathbf{x_1}) = \begin{bmatrix} \tilde{\mathbf{\Omega}}_{22}(\mathbf{x_1}) & \tilde{\mathbf{\Omega}}_{2\mathbf{m}}(\mathbf{x_1}) \\ \tilde{\mathbf{\Omega}}_{\mathbf{m}2}(\mathbf{x_1}) & \tilde{\mathbf{\Omega}}_{\mathbf{mm}}(\mathbf{x_1}) \end{bmatrix} = \begin{bmatrix} \mathbf{R}(\mathbf{x_1})\mathbf{\Omega}_{22}^0\mathbf{R}(\mathbf{x_1})^T & \mathbf{R}(\mathbf{x_1})\mathbf{\Omega}_{2\mathbf{m}}^0\overline{\mathbf{R}}(\mathbf{x_1})^T \\ \overline{\mathbf{R}}(\mathbf{x_1})\mathbf{\Omega}_{\mathbf{m}2}^0\mathbf{R}(\mathbf{x_1})^T & \overline{\mathbf{R}}(\mathbf{x_1})\mathbf{\Omega}_{\mathbf{mm}}^0\overline{\mathbf{R}}(\mathbf{x_1})^T \end{bmatrix} \tag{D.12}$$

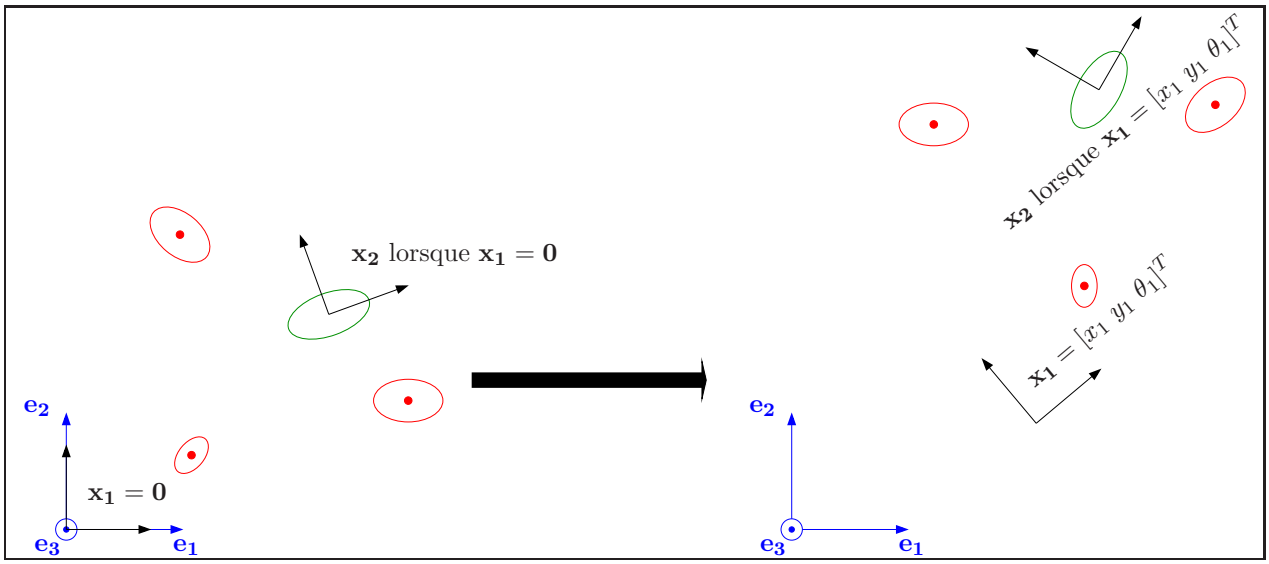


FIGURE D.2 – Illustration des conséquence du changement $\mathbf{x_1}$ sur les probabilités — Le repère de référence est en bleu. Les repère initial et le repère final du robot sont en noir. L'ellipse verte est associée à la position finale du robot, alors que l'ellipse rouge est associée aux positions des amers. La partie gauche de la figure représente le cas où le repère associé à la première pose du robot coïncide avec le repère global. La partie droite, quant à elle, représente le cas général (ie. $\mathbf{x}_1 \neq \mathbf{0}$). On pourra noter que les ellipses de confiance ont subi une rotation dont l'angle est égal à l'orientation intiale du robot.

D.3.2.2 Représentation de $p(\mathbf{m}|\mathbf{z}, \mathbf{x_1}, \mathbf{x_2} = \mu_2)$ comme une fonction de $\mathbf{x_1}$

Nous allons désormais déduire $p(\mathbf{m}|\mathbf{z}, \mathbf{x_1}, \mathbf{x_2} = \mu_2)$ de $p(\mathbf{m}, \mathbf{x_2}|\mathbf{z}, \mathbf{x_1})$. Pour cela, nous allons dans un premier temps appliquer le théorème de conditionnement des variables aléatoires gaussiennes. On aura alors les paramètres associés à la densité $p(\mathbf{m}|\mathbf{z}, \mathbf{x_1}, \mathbf{x_2} = \mu_2)$. Néanmoins, ceci ne sera pas suffisant pour exprimer complètement $p(\mathbf{m}|\mathbf{z}, \mathbf{x_1}, \mathbf{x_2} = \mu_2)$. En effet, le facteur devant l'exponentielle dépend de la matrice et du vecteur d'informations (qui dépendent eux-même de $\mathbf{x}_1$). Dans un second temps, nous donnons l'expression complète de $\log p(\mathbf{m}|\mathbf{z}, \mathbf{x_1}, \mathbf{x_2} = \mu_2)$ en fonction de la valeur de $\mathbf{x}_1$.

D.3.2.2.1 Paramètres de $p(\mathbf{m}|\mathbf{z}, \mathbf{x_1}, \mathbf{x_2} = \mu_2)$

On obtient les paramètres d'information de $p(\mathbf{m}|\mathbf{z}, \mathbf{x_1}, \mathbf{x_2} = \mu_2)$ en appliquant le théorème de conditionnement sur $p(\mathbf{x_2}, \mathbf{m}|\mathbf{z}, \mathbf{x_1})$. Les paramètres obtenus, notés $\xi^{|\mathbf{x_2}=\mu_2}(\mathbf{x_1})$ et $\mathbf{\Omega}^{|\mathbf{x_2}=\mu_2}(\mathbf{x_1})$, sont

donnés par :

$$\begin{aligned}\xi^{|\mathbf{x_2}=\mu_2}(\mathbf{x_1}) &= \tilde{\xi}_{\mathbf{m}}(\mathbf{x_1}) - \tilde{\mathbf{\Omega}}_{\mathbf{m}2}(\mathbf{x_1})\mu_2 \\ &= \overline{\mathbf{R}}(\mathbf{x_1})\Big(\xi^0_{\mathbf{m}} + \left(\Omega^0_{\mathbf{m}2}\mathbf{R}(\mathbf{x_1})^T + \Omega^0_{\mathbf{mm}}\overline{\mathbf{R}}(\mathbf{x_1})^T\mathbf{H}\right)\mathbf{x_1} - \mathbf{\Omega}^0_{\mathbf{m}2}\mathbf{R}(\mathbf{x_1})^T\mu_2\Big) \quad \text{(D.13)}\end{aligned}$$

et

$$\begin{aligned}\mathbf{\Omega}^{|\mathbf{x_2}=\mu_2}(\mathbf{x_1}) &= \tilde{\mathbf{\Omega}}_{\mathbf{mm}}(\mathbf{x_1}) \\ &= \overline{\mathbf{R}}(\mathbf{x_1})\mathbf{\Omega}^0_{\mathbf{mm}}\overline{\mathbf{R}}(\mathbf{x_1})^T \quad \text{(D.14)}\end{aligned}$$

Nous utiliserons par la suite les paramètres classiques de $p\left(\mathbf{m}|\mathbf{z}, \mathbf{x_1}, \mathbf{x_2} = \mu_2\right)$, ie. son espérance $\mu^{|\mathbf{x_2}=\mu_2}(\mathbf{x_1})$ et sa matrice de variances-covariances $\mathbf{\Sigma}^{|\mathbf{x_2}=\mu_2}(\mathbf{x_1})$. Ces paramètres sont donnés par :

$$\left\{\begin{array}{rcl}\mu^{|\mathbf{x_2}=\mu_2}(\mathbf{x_1}) &=& \overline{\mathbf{R}}(\mathbf{x_1})\left(\mathbf{\Omega}^0_{\mathbf{mm}}\right)^{-1}\Big(\xi^0_{\mathbf{m}} + \left(\Omega^0_{\mathbf{m}2}\mathbf{R}(\mathbf{x_1})^T + \Omega^0_{\mathbf{mm}}\overline{\mathbf{R}}(\mathbf{x_1})^T\mathbf{H}\right)\mathbf{x_1} - \mathbf{\Omega}^0_{\mathbf{m}2}\mathbf{R}(\mathbf{x_1})^T\mu_2\Big) \\ \mathbf{\Sigma}^{|\mathbf{x_2}=\mu_2}(\mathbf{x_1}) &=& \overline{\mathbf{R}}(\mathbf{x_1})\left(\mathbf{\Omega}^0_{\mathbf{mm}}\right)^{-1}\overline{\mathbf{R}}(\mathbf{x_1})^T\end{array}\right. \quad \text{(D.15)}$$

D.3.2.2.2 Calcul de $\log\left(p\left(\mathbf{m}|\mathbf{z}, \mathbf{x_1}, \mathbf{x_2} = \mu_2\right)\right)$ en fonction de $\mathbf{x_1}$

En prenant le logarithme $p\left(\mathbf{m}|\mathbf{z}, \mathbf{x_1}, \mathbf{x_2} = \mu_2\right)$, on obtient :

$$\begin{aligned}&\log\left(p\left(\mathbf{m}|\mathbf{z}, \mathbf{x_1}, \mathbf{x_2} = \mu_2\right)\right) = \\ &\quad= \log\left(\frac{\exp\left(\mathbf{m} - \mu^{|\mathbf{x_2}=\mu_2}(\mathbf{x_1})\right)^T\overline{\mathbf{R}}(\mathbf{x_1})\mathbf{\Omega}^0_{\mathbf{mm}}\overline{\mathbf{R}}(\mathbf{x_1})^T\left(\mathbf{m} - \mu^{|\mathbf{x_2}=\mu_2}(\mathbf{x_1})\right)}{\sqrt{(2\pi)^{\dim(\mathbf{m})}\det\left(\mathbf{\Sigma}^{|\mathbf{x_2}=\mu_2}(\mathbf{x_1})\right)}}\right) \\ &\quad= \underbrace{-\left(\frac{\dim(\mathbf{m})}{2}\right)\log(2\pi)}_{\text{constant}} - \frac{1}{2}\log\left(\det\left(\mathbf{\Sigma}^{|\mathbf{x_2}=\mu_2}(\mathbf{x_1})\right)\right) \\ &\qquad - \frac{1}{2}\left(\mathbf{m} - \mu^{|\mathbf{x_2}=\mu_2}(\mathbf{x_1})\right)^T\left(\mathbf{\Sigma}^{|\mathbf{x_2}=\mu_2}(\mathbf{x_1})\right)^{-1}\left(\mathbf{m} - \mu^{|\mathbf{x_2}=\mu_2}(\mathbf{x_1})\right)\end{aligned} \quad \text{(D.16)}$$

On peut remarquer que le second terme de l'équation D.17 (issu de la normalisation de l'exponentielle) semble dépendre de $\mathbf{x}_1$. Nous proposons d'étudier plus en détail le terme $\log\left(\det\left(\mathbf{\Sigma}^{|\mathbf{x_2}=\mu_2}(\mathbf{x_1})\right)\right)$. On a :

$$\begin{aligned}\det\left(\mathbf{\Sigma}^{|\mathbf{x_2}=\mu_2}(\mathbf{x_1})\right) &= \det\left(\overline{\mathbf{R}}(\mathbf{x_1})\left(\mathbf{\Omega}^0_{\mathbf{mm}}\right)^{-1}\overline{\mathbf{R}}(\mathbf{x_1})^T(\mathbf{x_1})\right) \\ &= \det\left(\overline{\mathbf{R}}(\mathbf{x_1})\right)\det\left(\left(\mathbf{\Omega}^0_{\mathbf{mm}}\right)^{-1}\right)\det\left(\overline{\mathbf{R}}(\mathbf{x_1})^T(\mathbf{x_1})\right)\end{aligned} \quad \text{(D.17)}$$

$\overline{\mathbf{R}}(\mathbf{x_1})$ est une matrice de rotation ; son déterminant est donc indépendant de $\mathbf{x}_1$ et vaut 1. D'après l'équation D.17, on a $\det\left(\mathbf{\Sigma}^{|\mathbf{x_2}=\mu_2}(\mathbf{x_1})\right) = \det\left(\left(\mathbf{\Omega}^0_{\mathbf{mm}}\right)^{-1}\right)$, qui est **indépendant de $\mathbf{x}_1$**. On a finale-

ment :

$$\log\left(p\left(\mathbf{m}|\mathbf{z},\mathbf{x_1},\mathbf{x_2}=\mu_2\right)\right) = cst - \frac{1}{2}\left(\mathbf{m}-\mu^{|\mathbf{x_2}=\mu_2}(\mathbf{x_1})\right)^T \overline{\mathbf{R}}(\mathbf{x_1})\mathbf{\Omega}^0_{\mathbf{mm}}\overline{\mathbf{R}}(\mathbf{x_1})^T\left(\mathbf{m}-\mu^{|\mathbf{x_2}=\mu_2}(\mathbf{x_1})\right) \tag{D.18}$$

L'équation D.18 n'est ni linéaire, ni quadratique en $\mathbf{x}_1$. Néanmoins, l'ensemble des raisonnements effectués jusque ici ont été réalisé à l'aide de linéarisations en supposant que le résultat est valide autour d'un point de fonctionnement (ce qui permet de supposer valide la propagation des densités gaussiennes). Ainsi, les valeurs de $\mathbf{x}_1$ et de $\mathbf{m}$ doivent nécessairement être comprises autour de ces points de fonctionnement. Ceci nous autorise donc à effectuer un développement limité de l'équation D.18.

D.3.2.2.3 Développement limité de l'équation D.18

Nous présentons dans ce paragraphe le développement limité de l'équation D.18. Les hypothèses effectuées quant à la linéarisation sont :

- $\mathbf{x_1}$ est proche de son point de fonctionnement, fixé à zéro,
- $\mathbf{m}$ est également proche de son point de fonctionnement, fixé à $\mu_{\mathbf{m}}$.

On peut donc écrire :

$$\begin{array}{rcl} \mu^{|\mathbf{x_2}=\mu_2}(\mathbf{x_1}) & \approx & \mu^{|\mathbf{x_2}=\mu_2}(\mathbf{0}) + \mathbf{J_1}\mathbf{x_1} = \overbrace{\left(\mathbf{\Omega}^0_{\mathbf{mm}}\right)^{-1}\left(\xi^0_{\mathbf{m}} - \mathbf{\Omega}^0_{\mathbf{m}2}\mu_2\right)}^{\mu_{\mathbf{m}} \text{ d'après (D.4)}} + \mathbf{J_1}\mathbf{x_1} = \mu_{\mathbf{m}} + \mathbf{J_1}\mathbf{x_1} \\ \mathbf{R}(\mathbf{x_1}) & \approx & \mathbf{I} + \mathbf{A}\theta_1 \\ \overline{\mathbf{R}}(\mathbf{x_1}) & \approx & \mathbf{I} + \overline{\mathbf{A}}\theta_1 \end{array} \tag{D.19}$$

où :

- $\mathbf{J_1}$ est la matrice jacobienne de $\mu^{|\mathbf{x_2}=\mu_2}(\mathbf{x_1})$ par rapport à $\mathbf{x}_1$ évaluée en $\mathbf{0}$. Son calcul est décrit au paragraphe suivant,
- $\mathbf{A}$ est la dérivée de $\mathbf{R}(\mathbf{x}_1)$ par rapport à θ_1 en 0.

$$\mathbf{A} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{D.20}$$

- $\overline{\mathbf{A}}$ est la dérivée de $\overline{\mathbf{R}}(\mathbf{x_1})$ par rapport à θ_1 en 0. On a donc :

$$\overline{\mathbf{A}} = \text{diag}\left(\mathbf{A},\mathbf{A},\ldots,\mathbf{A}\right) \tag{D.21}$$

On peut donc réécrire l'équation D.18) :

$$\log\left(p\left(\mathbf{m}|\mathbf{z},\mathbf{x_1},\mathbf{x_2}=\mu_2\right)\right) \approx cst - \frac{1}{2}\left(\mathbf{m}-\mu_{\mathbf{m}}-\mathbf{J_1}\mathbf{x_1}\right)^T\left(\mathbf{I}+\overline{\mathbf{A}}\theta_1\right)\mathbf{\Omega}^0_{\mathbf{mm}}\left(\mathbf{I}-\overline{\mathbf{A}}\theta_1\right)\left(\mathbf{m}-\mu_{\mathbf{m}}-\mathbf{J_1}\mathbf{x_1}\right) \tag{D.22}$$

En ne gardant que les ordres 0,1 et 2 du développement limité de l'équation D.22, on obtient :

$$\begin{aligned}\log\left(p\left(\mathbf{m}|\mathbf{z},\mathbf{x_1},\mathbf{x_2}=\mu_2\right)\right) \approx & cst - \frac{1}{2}\left(\mathbf{m}-\mu_\mathbf{m}\right)^T \mathbf{\Omega}^0_{\mathbf{mm}}\left(\mathbf{m}-\mu_\mathbf{m}\right) - \frac{1}{2}\mathbf{x_1}^T\mathbf{J_1}^T\mathbf{\Omega}^0_{\mathbf{mm}}\mathbf{J_1}\mathbf{x_1} \\ & + \left(\mathbf{m}-\mu_\mathbf{m}\right)^T\mathbf{\Omega}^0_{\mathbf{mm}}\mathbf{J_1}\mathbf{x_1} \\ = & cst - \frac{1}{2}\mathbf{m}^T\mathbf{\Omega}^0_{\mathbf{mm}}\mathbf{m} - \frac{1}{2}\mathbf{x_1}^T\mathbf{J_1}^T\mathbf{\Omega}^0_{\mathbf{mm}}\mathbf{J_1}\mathbf{x_1} + \mathbf{m}^T\mathbf{\Omega}^0_{\mathbf{mm}}\mathbf{J_1}\mathbf{x_1} + \mathbf{m}^T\mathbf{\Omega}^0_{\mathbf{mm}}\mu_\mathbf{m} \\ & - \mathbf{x_1}^T\mathbf{J_1}^T\mathbf{\Omega}^0_{\mathbf{mm}}\mu_\mathbf{m} - \frac{1}{2}\underbrace{\mu_\mathbf{m}{}^T\mathbf{\Omega}^0_{\mathbf{mm}}\mu_\mathbf{m}}_{\text{constant}} \\ = & cst' - \frac{1}{2}\mathbf{m}^T\mathbf{\Omega}^0_{\mathbf{mm}}\mathbf{m} - \frac{1}{2}\mathbf{x_1}^T\mathbf{J_1}^T\mathbf{\Omega}^0_{\mathbf{mm}}\mathbf{J_1}\mathbf{x_1} + \mathbf{m}^T\mathbf{\Omega}^0_{\mathbf{mm}}\mathbf{J_1}\mathbf{x_1} + \mathbf{m}^T\mathbf{\Omega}^0_{\mathbf{mm}}\mu_\mathbf{m} \\ & - \mathbf{x_1}^T\mathbf{J_1}^T\mathbf{\Omega}^0_{\mathbf{mm}}\mu_\mathbf{m}\end{aligned} \tag{D.23}$$

En utilisant le fait que $\mathbf{\Omega}^0_{\mathbf{mm}}\mu_\mathbf{m} = \xi^0_\mathbf{m} - \mathbf{\Omega}^0_{\mathbf{m}2}\mu_2$ (équation D.4), on a :

$$\begin{aligned}\log\left(p\left(\mathbf{m}|\mathbf{z},\mathbf{x_1},\mathbf{x_2}=\mu_2\right)\right) \approx & cst' - \frac{1}{2}\mathbf{m}^T\mathbf{\Omega}^0_{\mathbf{mm}}\mathbf{m} - \frac{1}{2}\mathbf{x_1}^T\mathbf{J_1}^T\mathbf{\Omega}^0_{\mathbf{mm}}\mathbf{J_1}\mathbf{x_1} + \mathbf{m}^T\mathbf{\Omega}^0_{\mathbf{mm}}\mathbf{J_1}\mathbf{x_1} \\ & + \mathbf{m}^T\left(\xi^0_\mathbf{m} - \mathbf{\Omega}^0_{\mathbf{m}2}\mu_2\right) - \mathbf{x_1}^T\mathbf{J_1}^T\left(\xi^0_\mathbf{m} - \mathbf{\Omega}^0_{\mathbf{m}2}\mu_2\right)\end{aligned} \tag{D.24}$$

On a donc au final :

$$\boxed{\begin{aligned}\log\left(p\left(\mathbf{m}|\mathbf{z},\mathbf{x_1},\mathbf{x_2}=\mu_2\right)\right) \approx cst' - \frac{1}{2}\begin{bmatrix}\mathbf{x_1}^T & \mathbf{m}^T\end{bmatrix}\begin{bmatrix}\mathbf{J_1}^T\mathbf{\Omega}^0_{\mathbf{mm}}\mathbf{J_1} & -\mathbf{J_1}^T\mathbf{\Omega}^0_{\mathbf{mm}} \\ -\mathbf{\Omega}^0_{\mathbf{mm}}\mathbf{J_1} & \mathbf{\Omega}^0_{\mathbf{mm}}\end{bmatrix}\begin{bmatrix}\mathbf{x_1} \\ \mathbf{m}\end{bmatrix} \\ + \begin{bmatrix}\mathbf{x_1}^T & \mathbf{m}^T\end{bmatrix}\begin{bmatrix}-\mathbf{J_1}^T\left(\xi^0_\mathbf{m} - \mathbf{\Omega}^0_{\mathbf{m}2}\mu_2\right) \\ \left(\xi^0_\mathbf{m} - \mathbf{\Omega}^0_{\mathbf{m}2}\mu_2\right)\end{bmatrix}\end{aligned}} \tag{D.25}$$

D.3.2.2.4 Calcul de $\mathbf{J_1}$

$\mathbf{J_1}$ est la matrice jacobienne de $\mu^{|\mathbf{x_2}=\mu_2}(\mathbf{x_1})$ par rapport à $\mathbf{x_1}$ et évaluée en $\mathbf{x}_1 = \mathbf{0}$:

$$\mathbf{J_1} = \frac{\partial}{\partial \mathbf{x_1}}\left(\overline{\mathbf{R}}(\mathbf{x_1})\left(\mathbf{\Omega}^0_{\mathbf{mm}}\right)^{-1}\left(\xi^0_\mathbf{m} + \left(\Omega^0_{\mathbf{m}2}\mathbf{R}(\mathbf{x_1})^T + \Omega^0_{\mathbf{mm}}\overline{\mathbf{R}}(\mathbf{x_1})^T\mathbf{H}\right)\mathbf{x_1} - \mathbf{\Omega}^0_{\mathbf{m}2}\mathbf{R}(\mathbf{x_1})^T\mu_2\right)\right)\Big|_{\mathbf{x_1}=\mathbf{0}} \tag{D.26}$$

Nous calculons la matrice $\mathbf{J_1}$ en deux étapes :

1. dans un premier temps, on ne dérive que par rapport à x_1 et y_1. L'opération est relativement facile car $\overline{\mathbf{R}}(\mathbf{x_1})$ et $\mathbf{R}(\mathbf{x_1})$ dépendent uniquement du paramètre θ_1,
2. ensuite, la dérivée par rapport à θ_1 est calculée séparément.

Le calcul de la matrice jacobienne par rapport à x_1, y_1 est donné par :

$$\begin{aligned}
\mathbf{J_1}[:,1:2] &= \left(\overline{\mathbf{R}}(\mathbf{x_1}) \left(\mathbf{\Omega^0_{mm}}\right)^{-1} \left(\Omega^0_{\mathbf{m}2} \mathbf{R}(\mathbf{x_1})^T + \Omega^0_{\mathbf{mm}} \overline{\mathbf{R}}(\mathbf{x_1})^T \mathbf{H} \right) \frac{\partial}{\partial x_1 y_1} (\mathbf{x_1}) \right) \Big|_{\mathbf{x_1}=\mathbf{0}} \\
&= \overline{\mathbf{R}}(\mathbf{0}) \left(\mathbf{\Omega^0_{mm}}\right)^{-1} \left(\Omega^0_{\mathbf{m}2} \mathbf{R}(\mathbf{0})^T + \Omega^0_{\mathbf{mm}} \overline{\mathbf{R}}(\mathbf{0})^T \mathbf{H} \right) \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \\
&= \left(\left(\mathbf{\Omega^0_{mm}}\right)^{-1} \Omega^0_{\mathbf{m}2} + \mathbf{H} \right) \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}
\end{aligned} \tag{D.27}$$

La dernière colonne de $\mathbf{J_1}$ est quant à elle donnée par :

$$\begin{aligned}
\mathbf{J_1}[:,3] = &\left(\frac{\partial}{\partial \theta_1} \left(\overline{\mathbf{R}}(\mathbf{x_1}) \left(\mathbf{\Omega^0_{mm}}\right)^{-1} \right) \left(\xi^0_{\mathbf{m}} + \left(\Omega^0_{\mathbf{m}2} \mathbf{R}(\mathbf{x_1})^T + \Omega^0_{\mathbf{mm}} \overline{\mathbf{R}}(\mathbf{x_1})^T \mathbf{H} \right) \mathbf{x_1} - \mathbf{\Omega}^0_{\mathbf{m}2} \mathbf{R}(\mathbf{x_1})^T \mu_2 \right) \right. \\
&\left. + \left(\overline{\mathbf{R}}(\mathbf{x_1}) \left(\mathbf{\Omega^0_{mm}}\right)^{-1} \right) \frac{\partial}{\partial \theta_1} \left(\xi^0_{\mathbf{m}} + \left(\Omega^0_{\mathbf{m}2} \mathbf{R}(\mathbf{x_1})^T + \Omega^0_{\mathbf{mm}} \overline{\mathbf{R}}(\mathbf{x_1})^T \mathbf{H} \right) \mathbf{x_1} - \mathbf{\Omega}^0_{\mathbf{m}2} \mathbf{R}(\mathbf{x_1})^T \mu_2 \right) \right) \Big|_{\mathbf{x_1}=\mathbf{0}} \\
&= \overline{\mathbf{A}} \left(\mathbf{\Omega^0_{mm}}\right)^{-1} \left(\xi^0_{\mathbf{m}} + \left(\Omega^0_{\mathbf{m}2} \mathbf{R}(\mathbf{0})^T + \Omega^0_{\mathbf{mm}} \overline{\mathbf{R}}(\mathbf{0})^T \mathbf{H} \right) \times \mathbf{0} - \mathbf{\Omega}^0_{\mathbf{m}2} \mathbf{R}(\mathbf{0})^T \mu_2 \right) \\
&+ \overline{\mathbf{R}}(\mathbf{0}) \left(\mathbf{\Omega^0_{mm}}\right)^{-1} \left(\left(\Omega^0_{\mathbf{m}2} \mathbf{A}^T + \Omega^0_{\mathbf{mm}} \overline{\mathbf{A}}^T \mathbf{H} \right) \times \mathbf{0} + \left(\Omega^0_{\mathbf{m}2} \mathbf{R}(\mathbf{0})^T + \Omega^0_{\mathbf{mm}} \overline{\mathbf{R}}(\mathbf{0})^T \mathbf{H} \right) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right. \\
&\qquad \left. - \mathbf{\Omega}^0_{\mathbf{m}2} \mathbf{R}(\mathbf{0})^T \mu_2 \right) \\
&= \overline{\mathbf{A}} \left(\mathbf{\Omega^0_{mm}}\right)^{-1} \left(\xi^0_{\mathbf{m}} - \mathbf{A}^T \mu_2 \right) + \left(\mathbf{\Omega^0_{mm}}\right)^{-1} \left(\left(\Omega^0_{\mathbf{m}2} + \Omega^0_{\mathbf{mm}} \mathbf{H} \right) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} - \mathbf{\Omega}^0_{\mathbf{m}2} \mathbf{A}^T \mu_2 \right) \\
&= \overline{\mathbf{A}} \left(\mathbf{\Omega^0_{mm}}\right)^{-1} \left(\xi^0_{\mathbf{m}} - \mathbf{\Omega}^0_{\mathbf{m}2} \mu_2 \right) + \left(\left(\mathbf{\Omega^0_{mm}}\right)^{-1} \Omega^0_{\mathbf{m}2} + \mathbf{H} \right) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + \left(\mathbf{\Omega^0_{mm}}\right)^{-1} \mathbf{\Omega}^0_{\mathbf{m}2} \mathbf{A} \mu_2
\end{aligned} \tag{D.28}$$

On peut finalement fusionner les équations D.27 et D.28 pour avoir une expression compacte de $\mathbf{J_1}$:

$$\mathbf{J_1} = \mathbf{J_1}[:,1:2] \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} + \mathbf{J_1}[:,3] \times \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \tag{D.29}$$

Le résultat final est donc donné par :

$$\boxed{\mathbf{J_1} = \left(\mathbf{\Omega^0_{mm}}\right)^{-1} \Omega^0_{\mathbf{m}2} + \mathbf{H} + \left(\overline{\mathbf{A}} \left(\mathbf{\Omega^0_{mm}}\right)^{-1} \left(\xi^0_{\mathbf{m}} - \mathbf{\Omega}^0_{\mathbf{m}2} \mu_2 \right) + \left(\mathbf{\Omega^0_{mm}}\right)^{-1} \mathbf{\Omega}^0_{\mathbf{m}2} \mathbf{A} \mu_2 \right) \times \mathbf{e}_3^T} \tag{D.30}$$

où $\mathbf{e}_3 = [0\ 0\ 1]^T$ désigne le troisième vecteur de la base canonique associée à $\mathbb{R}^3$.

D.3.3 Calcul de $p(\mathbf{x_1}|\mathbf{z}, \mathbf{x_2} = \mu_2)$

Le calcul de $p(\mathbf{x_1}|\mathbf{z}, \mathbf{x_2} = \mu_2)$ implique une transformation préalable de l'expression. En effet, nous ne disposons que de $p(\mathbf{x_2}|\mathbf{z}, \mathbf{x_1} = \mathbf{0})$: il s'agit d'une densité de probabilité conditionnée par rapport à $\mathbf{x_1} = \mathbf{0}$. Nous n'avons pas de densité concernant directement $\mathbf{x_1}$. En conséquence, nous allons dans un premier trouver une formulation équivalente du problème.

D.3.3.1 Transformation de $p(\mathbf{x_1}|\mathbf{z}, \mathbf{x_2} = \mu_2)$

En utilisant la règle de Bayes sur $p(\mathbf{x_1}|\mathbf{z}, \mathbf{x_2} = \mu_2)$, on obtient :

$$p(\mathbf{x_1}|\mathbf{z}, \mathbf{x_2}) = p(\mathbf{x_2}|\mathbf{z}, \mathbf{x_1}) \times \frac{p(\mathbf{x_2}|\mathbf{z})}{p(\mathbf{x_1}|\mathbf{z})} \tag{D.31}$$

On peut noter que l'ensemble des mesures (vecteur $\mathbf{z}$) n'est pas suffisant pour estimer la position du robot, que ce soit $\mathbf{x_2}$ ou $\mathbf{x_1}$. En effet, le vecteur $\mathbf{z}$ nous permet d'estimer la position du robot et de tous les amers par rapport à une position connue. Il s'agit du problème classique d'observabilité associé au SLAM : si aucune des poses de la trajectoire n'est connue, $\mathbf{x_1}$ et $\mathbf{x_2}$ sont *inobservables*. Ainsi, $\mathbf{x_1}$ peut valoir n'importe quelle valeur lorsque seul $\mathbf{z}$ est connu ; il en est de même pour $\mathbf{x_2}$. En conséquence, les densités $p(\mathbf{x_2}|\mathbf{z})$ et $p(\mathbf{x_1}|\mathbf{z})$ peuvent être vue comme la limite d'une densité uniforme étendue sur l'intégralité de l'espace : aucune position ne peut être privilégiée. Le rapport $p(\mathbf{x_2}|\mathbf{z}) / p(\mathbf{x_1}|\mathbf{z})$ peut donc être fixé à 1, et ce quels que soient $\mathbf{x}_1$ et $\mathbf{x}_2$.[5] L'équation D.31 peut donc être réécrite :

$$p(\mathbf{x_1}|\mathbf{z}, \mathbf{x_2}) = p(\mathbf{x_2}|\mathbf{z}, \mathbf{x_1}) \tag{D.32}$$

L'équation D.32 est valide quelles que soient les valeurs de $\mathbf{x_1}$ et $\mathbf{x_2}$. Dans la suite, nous allons dans un premier temps calculer $p(\mathbf{x_2}|\mathbf{z}, \mathbf{x_1})$ comme une fonction de $\mathbf{x_1}$ et de $\mathbf{x_2}$. Nous évaluerons ensuite cette fonction en $\mathbf{x_2} = \mu_2$. D'après l'équation D.32, on aura alors finalement une fonction de $\mathbf{x}_1$ directement égale à $p(\mathbf{x_1}|\mathbf{z}, \mathbf{x_2})$.

D.3.3.2 Calcul de $p(\mathbf{x_2}|\mathbf{z}, \mathbf{x_1})$ en fonction de $\mathbf{x_1}$ et $\mathbf{x_2}$

On utilise dans ce paragraphe une méthode très proche de celle proposée au paragraphe D.3.2.1 ; on va déduire $p(\mathbf{x_2}|\mathbf{z}, \mathbf{x_1})$ de la densité connue $p(\mathbf{x_2}|\mathbf{z}, \mathbf{x_1} = \mathbf{0})$.

On peut d'abord remarquer que $p(\mathbf{x_2}|\mathbf{z}, \mathbf{x_1} = \mathbf{0})$ est une densité gaussienne dont les paramètres sont donnés par marginalisation de $p(\mathbf{x_2}, \mathbf{m}|\mathbf{z}, \mathbf{x_1} = \mathbf{0})$:

5. Plus particulièrement, cette propriété est vraie pour $\mathbf{x_2} = \mu_2$.

- $\mathbf{\Omega}^0_{2|1} = \mathbf{\Omega}^0_{22} - \mathbf{\Omega}^0_{2\mathbf{m}} \left(\mathbf{\Omega}^0_{\mathbf{mm}}\right)^{-1} \mathbf{\Omega}^0_{\mathbf{m}2}$ est la matrice d'information de $p\left(\mathbf{x_2}|\mathbf{z}, \mathbf{x_1} = \mathbf{0}\right)$,
- $\xi^0_{2|1} = \xi^0_2 - \mathbf{\Omega}^0_{2\mathbf{m}} \left(\mathbf{\Omega}^0_{\mathbf{mm}}\right)^{-1} \xi^0_{\mathbf{m}}$ est le vecteur d'information de $p\left(\mathbf{x_2}|\mathbf{z}, \mathbf{x_1} = \mathbf{0}\right)$.

L'espérance mathématique et la matrice de variances-covariances de $p\left(\mathbf{x_2}|\mathbf{z}, \mathbf{x_1} = \mathbf{0}\right)$ sont quant à elles données par :

- $\mathbf{\Sigma}^0_{2|1} = \left(\mathbf{\Omega}^0_{2|1}\right)^{-1}$, pour la matrice de variances-covariance,
- $\mu^0_{2|1} = \left(\mathbf{\Omega}_{2|1}\right)^{-1} \xi^0_{2|1} = \mu_2$, pour l'espérance mathématique. L'égalité avec μ_2 provient de l'hypothèse effectuée quant au choix de μ_2 (égal à l'espérance associée à $\mathbf{x}_2$ dans $p\left(\mathbf{x}_2, \mathbf{m}|\mathbf{z}, \mathbf{x}_1 = 0\right)$) ; voir page 372.

Supposons désormais que $\mathbf{x_1}$ est connu et vaut $[x_1\ y_1\ \theta_1]^T$ (et non plus zéro). On peut facilement montrer (par un raisonnement identique à celui effectué au paragraphe D.3.2.1 page 374) que $p\left(\mathbf{x_2}|\mathbf{z}, \mathbf{x_1}\right)$ désigne une densité de probabilité gaussienne dont la matrice de variances-covariances et l'espérance mathématique sont données par : [6]

- $\mathbf{\Sigma}_{2|1}(\mathbf{x_1}) = \mathbf{R}(\mathbf{x_1}) \mathbf{\Sigma}^0_{2|1} \mathbf{R}(\mathbf{x_1})^T$,
- $\mu_{2|1}(\mathbf{x_1}) = \mathbf{R}(\mathbf{x_1}) \left(\mu^0_{2|1} + \mathbf{R}(\mathbf{x_1})^T \mathbf{x_1}\right) = \mathbf{R}(\mathbf{x_1}) \left(\mu_2 + \mathbf{R}(\mathbf{x_1})^T \mathbf{x_1}\right)$.

où $\mathbf{R}(\mathbf{x_1})$ a été défini dans l'équation D.8 page 375.

On peut alors déduire de ces paramètres le logarithme de $p\left(\mathbf{x_2}|\mathbf{z}, \mathbf{x_1}\right)$:

$$\begin{aligned} \log\left(p\left(\mathbf{x_2}|\mathbf{z}, \mathbf{x_1}\right)\right) = & - \underbrace{\left(\frac{\dim(\mathbf{x_2})}{2}\right)}_{=3/2} \log(2\pi) - \log \underbrace{\left(\det\left(\mathbf{\Sigma}_{2|1}(\mathbf{x_1})\right)\right)}_{=\det\left(\mathbf{\Sigma}^0_{2|1}\right)} \\ & - \frac{1}{2} \left(\mathbf{x_2} - \mu_{2|1}(\mathbf{x_1})\right)^T \underbrace{\left(\mathbf{\Sigma}_{2|1}(\mathbf{x_1})\right)^{-1}}_{=\mathbf{R}(\mathbf{x_1}) \mathbf{\Omega}^0_{2|1} \mathbf{R}(\mathbf{x_1})^T} \left(\mathbf{x_2} - \mu_{2|1}(\mathbf{x_1})\right) \\ = & \ cst'' - \frac{1}{2} \left(\mathbf{x_2} - \mu_{2|1}(\mathbf{x_1})\right)^T \mathbf{R}(\mathbf{x_1}) \mathbf{\Omega}^0_{2|1} \mathbf{R}(\mathbf{x_1})^T \left(\mathbf{x_2} - \mu_{2|1}(\mathbf{x_1})\right) \end{aligned} \tag{D.33}$$

D.3.3.3 Développement limité de $\log\left(p\left(\mathbf{x_2}|\mathbf{z}, \mathbf{x_1}\right)\right)$

En raison des hypothèses de linéarisations effectuées, l'équation D.33 n'est valide que lorsque $\mathbf{x_1}$ est proche de zero et $\mathbf{x_2}$ est proche de μ_2. Nous pouvons approcher $\log\left(p\left(\mathbf{x_2}|\mathbf{z}, \mathbf{x_1}\right)\right)$ par son développement limité à l'ordre 2 (de sorte à ne garder que des termes représentatifs d'une densité de probabilité gaussienne en vue de l'écriture du résultat final). Pour cela, un développement au premier

6. Ce résultat peut également être montré en utilisant les équations (D.11-D.12) page 375 puis en appliquant le théorème de marginalisation. On obtient le même résultat après simplification.

ordre de $\mu_{2|1}(\mathbf{x_1})$ et de $\mathbf{R}(\mathbf{x_1})$ est suffisant :

$$\begin{array}{rcl} \mu_{2|1}(\mathbf{x_1}) & \approx & \mu_{2|1}(\mathbf{0}) + \mathbf{J_2 x_1} = \mu_2 + \mathbf{J_2 x_1} \\ \mathbf{R}(\mathbf{x_1}) & \approx & \mathbf{I} + \mathbf{A}\theta_1 \end{array} \tag{D.34}$$

où :

- $\mathbf{J_2}$ est la matrice jacobienne de $\mu_{2|1}(\mathbf{x_1})$ par rapport à $\mathbf{x}_1$ évaluée en $\mathbf{0}$. Son calcul est donné au paragraphe D.3.3.5,
- $\mathbf{A}$ a été défini dans l'équation D.20 page 378.

On peut donc réécrire l'équation D.33 :

$$\log\left(p\left(\mathbf{x_2}|\mathbf{z},\mathbf{x_1}\right)\right) \approx cst'' - \frac{1}{2}\left(\mathbf{x_2} - \mu_2 - \mathbf{J_2 x_1}\right)^T \left(\mathbf{I} + \mathbf{A}\theta_1\right) \mathbf{\Omega}^0_{2|1} \left(\mathbf{I} - \mathbf{A}\theta_1\right)\left(\mathbf{x_2} - \mu_2 - \mathbf{J_2 x_1}\right) \tag{D.35}$$

D.3.3.4 Calcul de $\log\left(p\left(\mathbf{x_1}|\mathbf{z},\mathbf{x_2}=\mu_2\right)\right)$

En utilisant le résultat de l'équation D.35 et l'équation D.32 (page 381) avec $\mathbf{x_2} = \mu_2$, on obtient directement la valeur de $\log\left(p\left(\mathbf{x_1}|\mathbf{z},\mathbf{x_2}=\mu_2\right)\right)$:

$$\log\left(p\left(\mathbf{x_1}|\mathbf{z},\mathbf{x_2}=\mu_2\right)\right) \approx cst'' - \frac{1}{2}\left(-\mathbf{J_2 x_1}\right)^T \left(\mathbf{I} + \mathbf{A}\theta_1\right) \mathbf{\Omega}^0_{2|1} \left(\mathbf{I} - \mathbf{A}\theta_1\right)\left(-\mathbf{J_2 x_1}\right) \tag{D.36}$$

En ne gardant que les termes jusqu'au second ordre de l'équation D.36, on obtient :

$$\log\left(p\left(\mathbf{x_1}|\mathbf{z},\mathbf{x_2}=\mu_2\right)\right) \approx cst'' - \frac{1}{2}\mathbf{x_1}^T \mathbf{J_2}^T \mathbf{\Omega}^0_{2|1} \mathbf{J_2 x_1} \tag{D.37}$$

On obtient le résultat final en substituant la valeur de $\mathbf{\Omega}^0_{2|1}$ dans l'équation D.37 :

$$\boxed{\log\left(p\left(\mathbf{x_1}|\mathbf{z},\mathbf{x_2}=\mu_2\right)\right) \approx cst'' - \tfrac{1}{2}\mathbf{x_1}^T \mathbf{J_2}^T \left(\mathbf{\Omega}^0_{22} - \mathbf{\Omega}^0_{2\mathbf{m}} \left(\mathbf{\Omega}^0_{\mathbf{mm}}\right)^{-1} \mathbf{\Omega}^0_{\mathbf{m}2}\right) \mathbf{J_2 x_1}} \tag{D.38}$$

D.3.3.5 Calcul de $\mathbf{J_2}$

$\mathbf{J_2}$ est la matrice jacobienne de $\mu^{2|1}(\mathbf{x_1})$ par rapport à $\mathbf{x_1}$ évaluée en zéro :

$$\mathbf{J_2} = \frac{\partial}{\partial \mathbf{x_1}} \left(\mathbf{R}(\mathbf{x_1}) \left(\mu_2 + \mathbf{R}(\mathbf{x_1})^T \mathbf{x_1}\right) \right) \Bigg|_{\mathbf{x_1}=\mathbf{0}} \tag{D.39}$$

Tout comme pour le cas de $\mathbf{J_1}$, $\mathbf{J_2}$ est calculée en deux étapes :

1. dans un premier temps, on ne dérive que par rapport à x_1 et y_1. L'opération est relativement facile car $\overline{\mathbf{R}}(\mathbf{x_1})$ dépend uniquement du paramètre θ_1,
2. ensuite, la dérivée par rapport à θ_1 est calculée séparément.

La dérivation par rapport à x_1, y_1 donne :

$$\mathbf{J_2}[:, 1:2] = \mathbf{R}(\mathbf{0}) \times \mathbf{R}(\mathbf{0})^T \times \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \tag{D.40}$$

La dernière colonne de $\mathbf{J_2}$, quant à elle, est donnée par :

$$\begin{aligned} \mathbf{J_2}[:, 3] &= \left(\left(\frac{\partial}{\partial \theta_1} \mathbf{R}(\mathbf{x_1}) \right) \left(\mu_2 + \mathbf{R}(\mathbf{x_1})^T \mathbf{x_1} \right) + \mathbf{R}(\mathbf{x_1}) \left(\left(\frac{\partial}{\partial \theta_1} \mathbf{R}(\mathbf{x_1})^T \right) \mathbf{x_1} + \mathbf{R}(\mathbf{x_1})^T \left(\frac{\partial}{\partial \theta_1} \mathbf{x_1} \right) \right) \right) \Bigg|_{\mathbf{x_1}=\mathbf{0}} \\ &= \mathbf{A} \left(\mu_2 + \mathbf{R}(\mathbf{0})^T \times \mathbf{0} \right) + \mathbf{R}(\mathbf{0}) \left(\mathbf{A}^T \times \mathbf{0} + \mathbf{R}(\mathbf{0})^T \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right) \\ &= \mathbf{A}\mu_2 + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \end{aligned} \tag{D.41}$$

on peut finalement fusionner les équations D.40 et D.41) pour obtenir $\mathbf{J_2}$:

$$\boxed{\mathbf{J_2} = \mathbf{I} + \mathbf{A}\mu_2 \times \mathbf{e}_3^T} \tag{D.42}$$

D.3.4 Résultat final

On obtient le résultat final grâce à l'équation D.5 que l'on rappelle ici :

$$p\left(\mathbf{x_1}, \mathbf{m} | \mathbf{z}, \mathbf{x_2} = \mu_2\right) = p\left(\mathbf{m} | \mathbf{z}, \mathbf{x_1}, \mathbf{x_2} = \mu_2\right) . p\left(\mathbf{x_1} | \mathbf{z}, \mathbf{x_2} = \mu_2\right) \tag{D.5}$$

En utilisant les résultats des équations D.25 et D.37 dans l'équation D.5 permet d'obtenir :

$$\begin{aligned} \log\left(p\left(\mathbf{x_1}, \mathbf{m} | \mathbf{z}, \mathbf{x_2} = \mu_2\right)\right) \approx & cst' + cst'' - \frac{1}{2} \begin{bmatrix} \mathbf{x_1}^T & \mathbf{m}^T \end{bmatrix} \begin{bmatrix} \mathbf{J_1}^T \mathbf{\Omega}^0_{\mathbf{mm}} \mathbf{J_1} & -\mathbf{J_1}^T \mathbf{\Omega}^0_{\mathbf{mm}} \\ -\mathbf{\Omega}^0_{\mathbf{mm}} \mathbf{J_1} & \mathbf{\Omega}^0_{\mathbf{mm}} \end{bmatrix} \begin{bmatrix} \mathbf{x_1} \\ \mathbf{m} \end{bmatrix} \\ & - \frac{1}{2} \mathbf{x_1}^T \mathbf{J_2}^T \mathbf{\Omega}^0_{2|1} \mathbf{J_2} \mathbf{x_1} + \begin{bmatrix} \mathbf{x_1}^T & \mathbf{m}^T \end{bmatrix} \begin{bmatrix} -\mathbf{J_1}^T \left(\xi^0_{\mathbf{m}} - \mathbf{\Omega}^0_{\mathbf{m2}} \mu_2 \right) \\ \xi^0_{\mathbf{m}} - \mathbf{\Omega}^0_{\mathbf{m2}} \mu_2 \end{bmatrix} \end{aligned} \tag{D.43}$$

On a donc finalement :

$$\log\left(p\left(\mathbf{x_1},\mathbf{m}|\mathbf{z},\mathbf{x_2}=\mu_2\right)\right) \approx \alpha - \frac{1}{2}\begin{bmatrix}\mathbf{x_1}^T & \mathbf{m}^T\end{bmatrix}\begin{bmatrix}\mathbf{\Omega}^{end}_{11} & \mathbf{\Omega}^{end}_{1\mathbf{m}} \\ \mathbf{\Omega}^{end}_{\mathbf{m}1} & \mathbf{\Omega}^{end}_{\mathbf{mm}}\end{bmatrix}\begin{bmatrix}\mathbf{x_1} \\ \mathbf{m}\end{bmatrix} + \begin{bmatrix}\mathbf{x_1}^T & \mathbf{m}^T\end{bmatrix}\begin{bmatrix}\xi^{end}_1 \\ \xi^{end}_{\mathbf{m}}\end{bmatrix} \tag{D.44}$$

où α est une valeur constante, et où on a posé :

$$\begin{aligned}
\mathbf{\Omega}^{end}_{11} &= \mathbf{J_1}^T\mathbf{\Omega}^0_{\mathbf{mm}}\mathbf{J_1} + \mathbf{J_2}^T\left(\mathbf{\Omega}^0_{22} - \mathbf{\Omega}^0_{2\mathbf{m}}\left(\mathbf{\Omega}^0_{\mathbf{mm}}\right)^{-1}\mathbf{\Omega}^0_{\mathbf{m}2}\right)\mathbf{J_2} \\
\mathbf{\Omega}^{end}_{1\mathbf{m}} &= -\mathbf{J_1}^T\mathbf{\Omega}^0_{\mathbf{mm}} \\
\mathbf{\Omega}^{end}_{\mathbf{m}1} &= \mathbf{\Omega}^{end^T}_{1\mathbf{m}} \\
\mathbf{\Omega}^{end}_{\mathbf{mm}} &= \mathbf{\Omega}^0_{\mathbf{mm}} \\
\xi^{end}_1 &= -\mathbf{J_1}^T\left(\xi^0_{\mathbf{m}} - \mathbf{\Omega}^0_{\mathbf{m}2}\mu_2\right) \\
\xi^{end}_{\mathbf{m}} &= \xi^0_{\mathbf{m}} - \mathbf{\Omega}^0_{\mathbf{m}2}\mu_2
\end{aligned} \tag{D.45}$$

La densité de probabilité $p\left(\mathbf{x_1},\mathbf{m}|\mathbf{z},\mathbf{x_2}=\mu_2\right)$ est donc bien une densité gaussienne (aux approximations du développement limité près). Ses paramètres d'information sont donnés dans les équations D.44 et D.45. Finalement, les paramètres d'informations de $p\left(\mathbf{m}|\mathbf{z},\mathbf{x_2}=\mu_2\right)$ sont donnés par le théorème de marginalisation :

- $\mathbf{\Omega}^{end}_{\mathbf{mm}} - \mathbf{\Omega}^{end}_{\mathbf{m}1}\left(\mathbf{\Omega}^{end}_{11}\right)^{-1}\mathbf{\Omega}^{end}_{1\mathbf{m}}$ est la matrice d'information de $p\left(\mathbf{m}|\mathbf{z},\mathbf{x_2}=\mu_2\right)$
- $\xi^{end}_{\mathbf{m}} - \mathbf{\Omega}^{end}_{\mathbf{m}1}\left(\mathbf{\Omega}^{end}_{11}\right)^{-1}\xi^{end}_{\mathbf{m}}$ est le vecteur d'information de $p\left(\mathbf{m}|\mathbf{z},\mathbf{x_2}=\mu_2\right)$

D.4 Conclusion

Au final, nous avons donné dans cette annexe une méthode rigoureuse et consistante permettant de déplacer le point de conditionnement de la carte. Nous avons vu que si l'on veut conditionner par rapport à la dernière pose, il faut que la valeur associée au conditionement soit très proche de la valeur courante du point de fonctionnement (au moins pour la partie de l'état associée à la rotation). Dans le cas contraire, on sortirait très rapidement du domaine de validité des linéarisations. Ensuite, il est toujours possible de transporter les paramètres obtenus de sorte à obtenir la valeur par rapport à laquelle on souhaite réellement conditionner $\mathbf{x}_2$ (pour réinitialiser à l'origine par exemple).

Enfin, les développements effectués montrent que le fait de reconditionner la carte n'est pas trivial. Ce sujet n'est pratiquement pas abordé dans les publications traitant de cartes locales. Par ailleurs, les auteurs de [Piniés et Tardós, 2008] soulèvent indirectement ce problème avec la version de l'algorithme qui réinitialise les cartes locales à zéro. Les auteurs suggèrent alors d'utiliser les matrices jacobiennes associées au changement de repère entre $\mathbf{x}_1$ et $\mathbf{x}_2$ sans réelle justification mathématique. Nous avons

finalement prouvé ce résultat par les calculs effectués dans cette annexe.

Bibliographie

A. Angeli, D. Filliat, S. Doncieux et J.-A. Meyer : A fast and incremental method for loop-closure detection using bags of visual words. *IEEE Transactions On Robotics, Special Issue on Visual SLAM*, 2008.

A. Angeli, D. Filliat, S. Doncieux et J.-A. Meyer : Visual topological slam and global localization. *In Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2009.

Tim Bailey, Juan Nieto, Jose Guivant, Michael Stevens et Eduardo Nebot : Consistency of the ekf-slam algorithm. *In International Conference on Intelligent Robots and Systems*, 2006a.

Tim Bailey, Juan Nieto et Eduardo Nebot : Consistency of the fastslam algorithm. *In IEEE International Conference on Robotics and Automation*, 2006b.

João P. Barreto : *General Central Projection Systems, modeling, calibration and visual servoing.* Thèse de doctorat, Department of electrical and computer engineering, 2003.

H Bay, T Tuytelaars et L Van Gool : Surf : Speeded up robust features. *In Computer Vision–ECCV 2006 - Springer*, 2006.

P. Beaudet : Rotationally invariant image operators. *In Proc. 4th Int. Joint Conf. Patt. Recog., Kyoto, Japan*, pages 579–583, 1978.

S. Benhimane et E. Malis : Real-time image-based tracking of planes using efficient second-order minimization. *In IEEE International Conference on Intelligent Robots and Systems*, 2004.

S. Benhimane et E. Malis : Homography-based 2d visual tracking and servoing. *Joint Issue of the International Journal of Computer Vision and the International Journal of Robotic Research, 2006.*, 2006.

Selim Benhimane : *Vers une Approche Unifiée Pour la Suivi Temps-Réel et l'Asservissement Vusuel.* Thèse de doctorat, Ecole des Mines de Paris, INRIA Sophia Antipolis, 2007.

Philippe Bonnifait : *Localisation précise en position et attitude des robots mobiles d'extérieur à évolutions lentes.* Thèse de doctorat, Ecole Centrale de Nantes, 1997.

M. Bosse, P. Newman, J. Leonard, M. Soika, W. Feiten et S. Teller : An atlas framework for scalable mapping. *In IEEE International Conference on Robotics and Automation*, 2003.

Michael Bosse, Paul Newman, John Leonard et Seth Teller : Simultaneous Localization and Map Building in Large-Scale Cyclic Environments Using the Atlas Framework. *The International Journal of Robotics Research*, 23(12):1113–1139, 2004.

Vincent Brandou : *Stéréovision Locale et Reconstruction 3D/4D*. Thèse de doctorat, Université de Nice-Sophia Antipolis, 2008.

GEORGE CASELLA et CHRISTIAN P. ROBERT : Rao-Blackwellisation of sampling schemes. *Biometrika*, 83(1):81–94, 1996. URL `http://biomet.oxfordjournals.org/cgi/content/abstract/83/1/81`.

J.A. Castellanos, R. Martinez-Cantin, J.D. Tardós et J. Neira : Robocentric map joining : Improving the consistency of ekf-slam. *Robotics and Autonomous Systems*, 55(1):21 – 29, 2007. ISSN 0921-8890. Simultaneous Localisation and Map Building.

Jose A. Castellanos, J. M. M. Montiel, J. Neira et J. D. Tardós : The spmap : A probabilistic framework for simultaneous localization and map building. *IEEE Transactions on Robotics and Automation*, 15(5), 1999.

Javier Civera, Andrew J. Davison et J. M. Martínez Montiel : Inverse Depth Parametrization for Monocular SLAM. *IEEE Transactions on Robotics*, 24(5):932–945, 2008.

Javier Civera, Andrew J. Davison et J.M.M. Montiel : Inverse depth to depth conversion for monocular slam. *In International Conference on Robotics and Automation (ICRA)*, 2007.

Frank Dellaert et Michael Kaess : Square Root SAM : Simultaneous Location and Mapping via Square Root Information Smoothing. *Journal of Robotics Research*, 2006.

G. Dissanayake, P. Newman, H. F. Durrant-Whyte, S. Clark, et M. Csorba : A solution to the simultaneous localization and map building (slam) problem. *IEEE Transactions on Robotics and Automation*, 17(3):229–241, 2001.

Arnaud Doucet, Nando de Freitas, Kevin Murphy et Stuart Russell : Rao-blackwellised particle filtering for dynamic bayesian networks. *In Uncertainty in Artificial Intelligence*, 2000.

Arnaud Doucet, Simon Godsill et Christophe Andrieu : On sequential monte carlo sampling methods for bayesian filtering. *Statistics and Computing*, 10(3):197–208, 2004.

H. Durrant-Whyte, S. Majumder, S. Thrun, M. de Battista, et S. Scheding : A Bayesian algorithm for simultaneous localization and map building. *In Proceedings of the 10th International Symposium of Robotics Research (ISRR)*, 2001.

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

C. Estrada, J. Neira et J.D. Tardos : Hierarchical SLAM : Real-Time Accurate Mapping of Large Environments. *IEEE Transactions on Robotics*, 21(4):588–596, August 2005.

Olivier D. Faugeras, Quang-Tuan Luong et Stephen J. Maybank : Camera self-calibration : Theory and experiments. *In ECCV '92 : Proceedings of the Second European Conference on Computer Vision*, pages 321–334, London, UK, 1992. Springer-Verlag. ISBN 3-540-55426-2.

Martin A. Fischler et Robert C. Bolles : Random sample consensus : a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981.

J. Folkesson, P. Jensfelt et H.I. Christensen : Graphical SLAM using vision and the measurement subspace. *In IEE/RSJ Inernational Conference on Intelligent Robots and Systems (IROS), Edmonton, Canada*, 2005.

Christopher Geyer : *Catadioptric Projective Geometry : theory and applications*. Thèse de doctorat, University of Pennsylvania, 2003.

H. Hadj-Abdelkader, E. Malis et P. Rives : Spherical Image Processing for Accurate Odometry with Omnidirectional Cameras. *In The Eighth Workshop on Omnidirectional Vision*, October .

C. Harris et M. Stephens : A Combined Corner and Edge Detection. *In Proceedings of The Fourth Alvey Vision Conference*, pages 147–151, 1977.

Richard Hartley et Andrew Zisserman : *Multiple View geometry in Computer vision*. Cambridge university press, 2000.

Wei He, Takayoshi Yamashita, Hongtao Lu et Shihong Lao : Surf tracking. *In Computer Vision, 2009 IEEE 12th International Conference on*, pages 1586 –1592, sept. 2009.

R. Hermann et A. J. Krener : Nonlinear controllability and observability. *IEEE Transactions On Automatic Control*, 22(5):728–740, October 1977.

Kin Ho et Paul Newman : Loop closure detection in slam by combining visual and spatial appearance. *Robotics and Autonomous Systems*, 54(9):740–749, 2006.

G.P. Huang, A.I. Mourikis et S.I. Roumeliotis : Analysis and improvement of the consistency of extended kalman filter based slam. *In Proceedings of the 2008 IEEE International Conference on Robotics and Automation (ICRA)*, 2008.

Carine Hue, Jean-Pierre Le Cadre et Patrick Pérez : Étude comparative des techniques de filtrage non-linéaire appliquées à la localisation 2D d'un véhicule en temps réel. *IEEE Transactions on Aerospace and Electronic Systems*, 28(3):791–812, 2002.

Luc Jaulin : A nonlinear set-membership approach for the localization and map building of underwater robots. *IEEE Transactions On Robotics*, 2009a.

Luc Jaulin : Robust set membership state estimation. *Automatica*, 45(1):202,206, 2009b.

Cyril Joly : Caractérisation métrologique de capteurs intertiels et fusion avec un récepteur GPS EGNOS en navigation automobile avancée. Mémoire de D.E.A., Laboratoire Central des Ponts et Chaussées / Ecole Centrale de Nantes, September 2006.

Cyril Joly, David Bétaille et François Peyret : Étude comparative des techniques de filtrage non-linéaire appliquées à la localisation 2D d'un véhicule en temps réel. *Traitement du Signal*, 5 (3):201–220, 2009.

Cyril Joly et Patrick Rives : Bearing-only SLAM : comparison between probabilistic and deterministic methods. Research Report RR-6602, INRIA, 2008. URL `http://hal.inria.fr/inria-00308722/en/`.

Cyril Joly et Patrick Rives : Building Consistent Local Submaps with Omnidirectional SLAM. *In The Ninth Workshop on Omnidirectional Vision, Camera Networks and Non-classical Cameras, Kyoto, Japon*, October 2009.

Cyril Joly et Patrick Rives : Bearing-Only SAM Using a Minimal Inverse Depth Parametrization – Application do Omnidirectional SLAM. *In 7^{th} International Conference on Informatics in Control, Automation and Robotics (ICINCO 2010), Funchal, Madeira - Portugal*, June 2010.

S. Julier : The stability of covariance inflation methods for SLAM. *In Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 3, pages 2749 – 2754 vol.3, 27-31 2003.

Simon Julier et Jeffrey K. Uhlmann : A counter example to the theory of simultaneous localization and map building. *In International Conference on Robotics and Automation*, 2001.

Simon J. Julier et Jeffery K. Uhlmann : A new extension of the kalman filter to nonlinear systems. *In Proceedings of AeroSense : The 11th International Symposium on Aerospace/Defense Sensing, Simulation and Controls, Multi Sensor Fusion, Tracking and Resource Management*, 1997.

Michael Kaess, Ananth Ranganathan et Frank Dellaert : iSAM : Fast Incremental Smoothing and Mapping with Efficient Data Association. *In International Conference on Robotics and Automation (ICRA)*, 2007.

L. Kitchen et A. Rosenfeld : Gray level corner detection. *Pattern Recognition Letters*, 1(2):95–102, december 1982.

N. M. Kwok et G. Dissanayake : An efficient multiple hypothesis filter for bearing-only slam. *In IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004.

Kwang Wee Lee, W. Sardha Wijesoma et Javier Ibanez Guzman : On the observability and observability analysis of slam. *In IROS*, pages 3569–3574, 2006.

Thomas Lemaire, Simon Lacroix et Joan Solá : A practical 3D bearing only SLAM algorithm. *In IEE/RSJ Inernational Conference on Intelligent Robots and Systems (IROS), Edmonton, Canada*, 2005.

J.J. Leonard et H.J.S Feder : Decoupled stochastic mapping. *IEEE Journal of Oceanic Engineering*, 26(5-6):561–571, 2001.

Shigang Li et Yuta Shimomura : Lane Marking Detection by Side Fisheye Camera. *In IEE/RSJ Int.Conf.on Intelligent Robots and Systems (IROS), Acropolis Convention Center, Nice, France*, 2008.

David G. Lowe : Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 2(60):91–110, 2004.

Yi Ma, Stefano Soatto, Jana Kosecka et S. Shankar Sastry : *An Invitation to 3-D Vision : From Images to Geometric Models*. Springer-Verlag, 2003. ISBN 0387008934.

E. Malis : Improving vision-based control using efficient second-order minimization techniques. *In IEEE International Conference on Robotics and Automation*, 2004.

Clide F. Martin, XiaoChang Wang et Mark Stamp : Discrete Observability and Numerical Quadrature. *IEEE Transactions On Automatic Control*, 36(11):1337–1340, November 1991.

Ruben Martinez-Cantin et José A. Castellanos : Bouding Uncertainty in EKF-SLAM : The Robocentric Local Approach. *In IEEE International Conference on Robotics and Automation (ICRA), Orlando, Florida*, May 2006.

C. Mei et P. Rives : Calibrage non biaise d'un capteur central catadioptrique. *In RFIA*, January 2006a.

C. Mei et P. Rives : Calibration between a central catadioptric camera and a laser range finder for robotic applications. *In IEEE International Conference on Robotics and Automation*, May 2006b.

C. Mei, G. Sibley, M. Cummins, P. Newman et I. Reid : A constant time efficient stereo slam system. *In BMVC*, 2009.

C. Mei, G. Sibley, M. Cummins, P. Newman et I. Reid : Real : A system for large-scale mapping in constant-time using stereo. *International Journal of Computer Vision*, 2010. Special issue of BMVC, accepted for publication.

Christopher Mei : *Laser-Augmented Omnidirectional Vision for 3D Localisation and Mapping*. Thèse de doctorat, Ecole des Mines de Paris, INRIA Sophia Antipolis, 2007.

K. Mikolajczyk et C. Schmid : A Performance Evaluation of Local Descriptors. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 27(10):1615–1630, 2005.

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

M Montemerlo, S Thrun, D Koller et 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.

Michael Montemerlo : *FastSLAM : A Factored Solution to the Simultaneous Localization and Mapping Problem with Unknown Data Association*. Thèse de doctorat, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, July 2003.

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

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

P. E. Moraal et J. W. Grizzle : Observer design for nonlinear systems with discrete-time measurements. *IEEE Transactions On Automatic Control*, 40(3):395–404, March 1995.

Hans Moravec : Towards automatic visual obstacle avoidance. *In International Joint Conferences on Artificial Intelligence, Cambridge, MA*, page 584, 1977.

P. Newman : *On the Structure and Solution of the Simultaneous Localisation and Map Building Problem*. Thèse de doctorat, Australian Centre for Field Robotics, University of Sydney, Sydney, Australia, 2000.

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

N Paparoditis : STEREOPOLIS : une plateforme mobile multi-capteurs pour le suivi des politiques — Présentation au format pdf disponible à l'adresse `http://recherche.ign.fr/doc/Stereopolis_Advancity_080320.pdf`, 2000.

Athanansios Papoulis : *Methods and Applications of Interval Analysis, second edition.* McGraw Hill Higher Education, 1984.

Martin P. Parsley et Simon J. Julier : Avoiding Negative Depth in Inverse Depth Bearing-Only SLAM. *In IEE/RSJ Int.Conf.on Intelligent Robots and Systems (IROS), Acropolis Convention Center, Nice, France*, 2008.

Lina M. Paz et José Neira : Optimal local map size for EKF-based SLAM. *In IEE/RSJ Int.Conf.on Intelligent Robots and Systems (IROS), Beijing, China*, 2006.

P. Piniés et J.D. Tardós : Scalable slam building conditionally independent local maps. *In IEE/RSJ Int.Conf.on Intelligent Robots and Systems (IROS)*, 2007.

P. Piniés et J.D. Tardós : Large scale slam building conditionally independent local maps : Application to monocular vision. *IEEE Transactions On Robotics*, 24(5), 2008.

Pedro Piniés, Juan D. Tardós et José Neira : Localization of avalanche victims using robocentric SLAM. *In IEE/RSJ Int.Conf.on Intelligent Robots and Systems (IROS), Beijing, China*, 2006.

Roger Pissard-Gibollet : *Conception et Commande par Asservissement Visuel d'un Robot Mobile.* Thèse de doctorat, Ecole des Mines de Paris, 1993.

Eric Royer, Maxime Lhuillier, Michel Dhome et Jean-Marc Lavest : A Monocular Vision for Mobile Robot Localization and Autonomous Navigation. *International Journal of Computer Vision*, 74(3):237–260, 2007.

A. Salazar-Garibay et E. Malis : Direct approach to the self-calibration of omnidirectional cameras. *In Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 2149 –2155, sept. 2009.

A. Salazar-Garibay, E. Malis et C. Mei : Visual tracking of planes with an uncalibrated central catadioptric camera. *In Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 2999 –3004, 10-15 2009.

Torsten Sattler, Bastian Leibe et Leif Kobbelt : Scramsac : Improving ransac's efficiency with a spatial consistency filter. *In Computer Vision, 2009 IEEE 12th International Conference on*, pages 2090 –2097, sept. 2009.

Linda Smail : *Algorithmique pour les Réseaux Bayésiens et leurs Extensions.* Thèse de doctorat, Université de Marne-la-Vallée, 2004.

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

Padhraic Smyth, David Heckerman et Michael Jordan : Probabilistic independence networks for hidden markov probability models. Rapport technique, 1996.

J. Solà, M. Devy, A. Monin et T. Lemaire : Undelayed initialization in bearing only slam. *In IEEE International Conference on Intelligent Robots and Systems*, 2005.

Joan Solà : *Towards Visual Localization Mapping and Moving Objects Tracking by a Mobile Robot : a Geometric and Probabilistic Approach.* Thèse de doctorat, Université de Toulouse, LAAS, 2007.

Peter Sturm : *Vision 3D non calibrée : contributions à la reconstruction projective et étude des mouvements critiques pour l'auto-calibrage.* Thèse de doctorat, Institut National Polytechnique de Grenoble - INPG, 12 1997. URL `http://tel.archives-ouvertes.fr/tel-00004964/en/`. theses/1997/Sturm.Peter.

R. Swaminathan, M. D. Grossberg et S. K. Nayar : Non-Single Viewpoint Catadioptric Cameras : Geometry and Analysis. *International Journal of Computer Vision*, 66(3):211–229, Mar 2006.

J. Tardos, J. Neira, P. Newman et J. Leonard : Robust mapping and localization in indoor environments using sonar data. *International Journal of Robotics Research*, 21(4):311–330, 2002.

S. Thrun, Y. Liu, D. Koller, A.Y. Ng, Z. Ghahramani et H. Durrant-Whyte : Simultaneous localization and mapping with sparse extended information filters. *International Journal of Robotics Research*, 23(7/8), 2004.

Sebastian Thrun : Robotic mapping : A survey, 2002.

Sebastian Thrun et 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.

Nicolas Trawny et Stergios I. Roumeliotis : A unified framework for nearby and distant landmarks in bearing-only slam. *In International Conference on Robotics and Automation (ICRA)*, 2006.

Tinne Tuytelaars et Krystian Mikolajczyk : *Local Invariant Feature Detectors : A Survey.* Now Publishers Inc., Hanover, MA, USA, 2008. ISBN 1601981384, 9781601981387. URL `http://portal.acm.org/citation.cfm?id=1481563`.

Wikipedia : Bayesian network — Article Wikipedia disponible à l'adresse `http://en.wikipedia.org/wiki/Bayesian_network`.

Yasushi Yagi : Omnidirectional sensing and its applications. *IEICE Trans, on Information and Systems*, E82-D(3):568–579, 1999.

Jonathan S. Yedidia, William T. Freeman et Yair Weiss : Understanding Belief Propagation and its Generalizations. Technical Report TR-2001-22, MITSUBISHI ELECTRIC RESEARCH LABORATORIES, January 2002. URL `http://www.merl.com/publications/TR2001-022/`.

Contributions aux méthodes de localisation et cartographie simultanées par vision omnidirectionnelle

Résumé :

Estimer le mouvement d'un robot mobile tout en construisant une représentation de son environnement est un problème essentiel pour le développement des robots autonomes : il est connu sous l'acronyme de SLAM (Simultaneous Localization and Mapping). Nous nous intéressons dans cette thèse au SLAM visuel avec caméra omnidirectionnelle. L'environnement est représenté par des amers ponctuels dont la profondeur n'est pas mesurée directement.

Nous nous sommes d'abord intéressés aux méthodes de résolution du SLAM. Deux approches ont retenu notre attention de part leur consistance : le SAM (Smoothing and Mapping : une méthode de lissage probabiliste prenant en compte toute la trajectoire) et l'analyse par intervalles. Une étude en simulation a été menée et conclut que la méthode d'analyse par intervalles est inadaptée au SLAM visuel en raison de la faible redondance d'informations. Le SAM a ensuite été validé sur des données réelles, dans les cas de deux trajectoires planes et d'une à 6 degrés de liberté.

Un problème du SLAM est sa complexité calculatoire quadratique avec le nombre d'amers. Il est souvent résolu en segmentant l'environnement en cartes locales. Nous avons repris ce concept en l'améliorant sur deux points. Premièrement, chaque nouvelle carte intègre les informations de la précédente de façon consistante. Ensuite, le critère déterminant quand changer de carte impose désormais que chaque carte locale soit fortement corrélée, ce qui permet de limiter l'augmentation des incertitudes. Nous avons finalement utilisé cet aspect pour appliquer des algorithmes déterministes pour augmenter la représentation ponctuelle par des plans texturés et une visualisation pertinente de l'espace libre.

Mots clés : SLAM, localisation, cartographie, Vision omnidirectionnelle, lissage probabiliste, cartes locales, analyse par intervalles

Contributions to Omnidirectional Visual Simultaneous Localization and Mapping

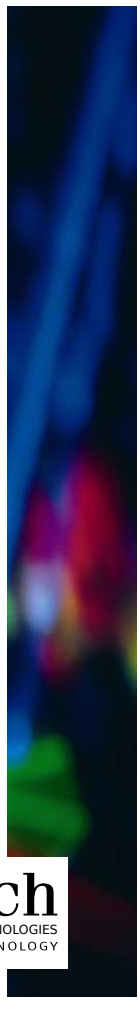
Abstract:

Estimating the motion of a mobile robot while simultaneously building a representation of its environment is a key problem for autonomous robotics. This problem is known as SLAM (Simultaneous Localization and Mapping). In this thesis, we address the problem of Bearing-only Visual SLAM based on an omnidirectional camera. The environment is made of point-based landmarks whose depth is never measured directly.

We first made a state of the art of SLAM solving methods. We finally kept two algorithms because of their good consistency: the SAM (Smoothing and Mapping) which is a probabilistic method based on the smoothing of the whole robot trajectory, and the interval analysis. Simulation results shown that the interval method is not well adapted to the bearing-only SLAM: the system is not redundant enough. The SAM algorithm was then validated on real data, with two planar trajectories and a full six degrees of freedom trajectory.

Complexity is a classical issue of SLAM methods: computational cost quadratically rises with the number of landmarks. This issue is often solved by splitting the whole environment in local submaps. We used this concept and improved it on two points. First, each new local map consistently takes advantages of all the measurements made in the previous one. Then, we developed a new criterion to decide when starting a new map which ensures that the current map remains strongly correlated. This leads to limit the growth of uncertainties in local maps. Finally, we used the latter property to apply deterministic algorithms to enhance the point-based representation by textured planes and a pertinent representation of the free space.

Keywords: SLAM, localization, mapping, omnidirectional vision, probabilistic smoothing, local maps, interval analysis



MINES ParisTech

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY