POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Localisation et cartographie avec superquadriques comme primitives géométriques

ARNAUD VENET
Département de génie électrique

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie électrique

Décembre 2019

© Arnaud Venet, 2019.

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Localisation et cartographie avec superquadriques comme primitives géométriques

présenté par **Arnaud VENET**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Richard GOURDEAU, président
Jérôme LE NY, membre et directeur de recherche
James FORBES, membre

DÉDICACE

À toutes les personnes que j'ai rencontré à Montréal,
vous me manquerez...

REMERCIEMENTS

Je tiens à remercier d'abord mon directeur de recherche Jérôme Le Ny pour avoir pensé ce projet et m'avoir aiguillé tout au long de ces deux ans.

Je remercie ma famille, ma mère Marie-France, mon père Bruno, ma soeur Amélie, et mes grands parents Marinette et Paul pour leur soutien et leur gentillesse.

Je remercie aussi toutes les personnes rencontrées à Montréal qui ont rendu ces deux ans très sympathiques. Merci à Clémence, Florian, Olivier et Tien mes collègues de laboratoire du MRASL. Merci à Aline, Ibrahim, Magnus, Maxence, Mélanie et Titouan, de grands sportifs qui m'ont aidé à décompresser des rudes journées de travail. Merci à la VLC, à savoir Elliot, Justin et Quentin pour les boomers apaisants, les discussions éclairées et le déneigement optimal.

Enfin, du côté français je suis reconnaissant du service des Relations Internationales de Centrale Marseille, très attentif au bien-être de ses étudiants. Je remercie enfin Serge pour son expertise musicale et Farid pour sa bonhomie.

RÉSUMÉ

Le rôle de la robotique est d'assister l'homme dans des tâches répétitives ou dangereuses. Très présents dans l'industrie, les robots ont commencé à réaliser des missions complexes, notamment en gagnant en autonomie.

Parmi ces tâches, nous retrouvons la cartographie, qui consiste à réaliser une carte représentant un environnement donné. Les robots mobiles sont souvent employés, car ils peuvent évoluer dans ce milieu, et faire des observations de différents points de vue. Dans nos travaux, nos observations sont faites par un capteur laser fixé sur un robot, pouvant effectuer des mesures de distance et de cap. La construction d'une carte passe d'abord par le choix de la représentation de l'environnement. Beaucoup de représentations existent, et doivent répondre aux problèmes de fiabilité et de mémoire. Nous avons choisi d'utiliser des primitives géométriques telles que les superquadriques, encore peu utilisées en cartographie, de par la grande variété de formes qu'elles peuvent prendre, et leur paramétrisation simplifiée.

Un problème inhérent à la robotique mobile est la localisation, et un robot doit pouvoir se positionner dans son environnement pour interagir avec lui. Nous proposons une méthode de localisation qui repose sur la représentation de notre environnement en superquadriques.

Nous traitons séparément dans ce mémoire les deux problèmes de cartographie et de localisation pour un robot mobile se déplaçant dans un plan.

ABSTRACT

Robots are intended to assist people in repetitive or dangerous tasks. With a significant presence in the industry, robots have started to carry out complex missions, especially by becoming more autonomous.

Among these tasks we have mapping, which consists in building a map that represents a given environment. Mobile robots are often used because they can operate in this environment and make observations from different points of view. In our work, our observations are made using a laser sensor fixed on a robot which can take distance and heading measurements. The first step in building a map is to choose the appropriate representation of the environment. Many representations have been developed, and must address reliability and memory issues. We have chosen to use superquadrics as geometric primitives for the wide variety of shapes they can take, and their simplified parametrisation. Those shapes are not yet widely used in cartography.

An inherent problem in mobile robotics is localisation, and a robot must be able to position itself in its environment to be able to interact with it. We provide a method of localisation based on the representation of our environment made of superquadrics.

In this paper, we deal separately with mapping and localisation problems for a mobile robot travelling in a plane.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

ACP	Analyse en Composantes Principales
CCL	Étiquetage des Composantes Connectées
DBSCAN	*Density-Based Spatial Clustering of Applications with Noise*
DOP	Dilution de Précision
EKF	Filtre de Kalman Étendu
GNSS	Géolocalisation et Navigation par un Système de Satellites
HDOP	Dilution de Précision Horizontale
ICP	*Iterative Closest Point*
KFBS	Segmentation Basée sur le Filtrage de Kalman
LRF	*Laser Range Finder*
MC	Moindres Carrés
MCNL	Moindres Carrés Non Linéaires
MCNL-FV	Moindres Carrés Non Linéaires Favorisant la Visibilité
MCNL-PV	Moindres Carrés Non Linéaires avec Pénalité de Volume
SLAM	Localisation et Cartographie Simultanées
SQ	Superquadrique
SVD	Décomposition en Valeurs Singulières
UKF	Filtre de Kalman *Unscented*

LISTE DES ANNEXES

CHAPITRE 1 INTRODUCTION

En robotique mobile, le problème de Localisation et Cartographie Simultanées (SLAM) est défini par la création concurrente d'une carte de l'environnement tout en se positionnant dans celui-ci. La création de la carte a un objectif double puisque celle-ci permet au robot de se positionner relativement à elle, tout en fournissant un produit fini, qui est la carte, pouvant être exploité pour des opérations futures.

Une multitude de méthodes de représentation de cartes existent, et tentent de répondre aux problèmes de mémoire et puissance de calcul requis ou de fiabilité. Notre travail se fait sur la représentation de la carte, où nous utilisons les superquadriques comme primitives géométriques. La figure 1.1 montre un exemple d'une telle carte. Ces volumes paramétrés nous serviront pour décrire les obstacles et éléments de notre environnement.

Le problème de SLAM est complexe, et nous ne l'aborderons pas directement par la suite. Nous le décomposons ici en deux sous problèmes, qui sont la cartographie et la localisation.

1.1 Définitions et concepts de base

1.1.1 Le problème de cartographie

Problème général

Soit un environnement $\mathcal{E}$ borné qui se distingue par les objets qu'il contient. L'objectif est de réaliser une caractérisation de ces objets tout en les localisant dans leur environnement. La réalisation de cet objectif passe par l'utilisation d'un robot mobile dont la position $[x_\mathrm{p}, y_\mathrm{p}, z_\mathrm{p}] \in \mathbb{R}^3$ et l'orientation $\psi_\mathrm{p} \in [-\pi, \pi]$ sont connues en tout temps dans le référentiel de $\mathcal{E}$. Nous appelons pose du robot sa position ainsi que son orientation $\mathbf{x}_\mathrm{p} = [x_\mathrm{p}, y_\mathrm{p}, z_\mathrm{p}, \psi_\mathrm{p}]$. La pose du robot à un instant t est notée $\mathbf{x}_{\mathrm{p},t}$. Ce robot est muni de capteurs pouvant effectuer des mesures externes lui donnant des renseignements sur $\mathcal{E}$. Construire une carte représentant les différents éléments de $\mathcal{E}$ en utilisant ces mesures et les positions du robot revient à résoudre le problème de cartographie.

Dans notre carte, nous avons choisi de représenter un objet de $\mathcal{E}$ avec une superquadrique (SQ), paramétrée par un vecteur de paramètres $\mathbf{P}$. De cette façon, les $M \in \mathbb{N}$ objets de $\mathcal{E}$ sont représentés par M SQs de paramètres $\mathbf{P}_m$, $m \in [\![1, M]\!]$. Chaque vecteur $\mathbf{P}_m$ renferme des informations de position, d'orientation, de taille et de forme pour l'objet m. Notre objectif est de retrouver un ensemble de paramètres $\{\mathbf{P}_m\}$ qui représente le plus fidèlement $\mathcal{E}$.

Système simulé

Nous avons choisi pour $\mathcal{E}$ un espace carré où sont posés M objets tri-dimensionnels en forme de SQ. La figure 1.1 représente un exemple d'environnement avec $M = 4$ obstacles représentés en bleu.

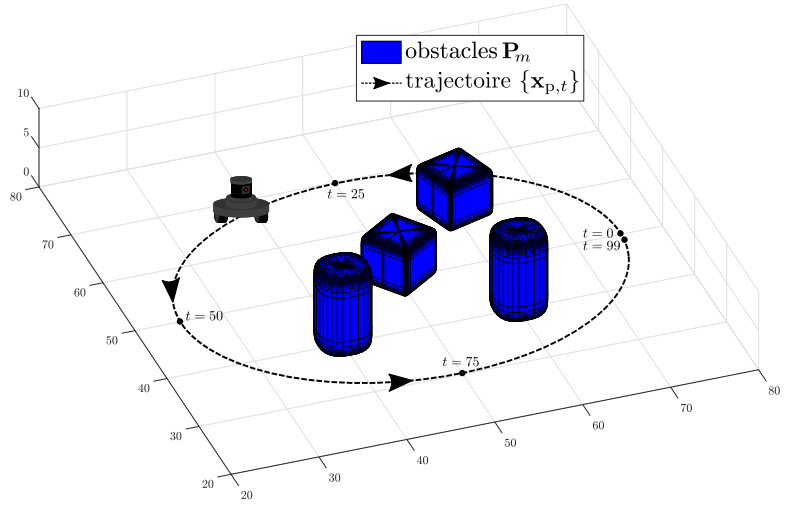


Figure 1.1 Exemple d'environnement $\mathcal{E}$ avec $M = 4$ superquadriques.

Le robot évolue dans cet environnement et effectue une trajectoire elliptique, de sorte à revenir à sa position initiale. La trajectoire du robot s'effectue dans le plan horizontal $z = 0$ sur lequel les objets de $\mathcal{E}$ sont posés. Nous faisons en sorte que le robot n'entre pas en collision avec ces obstacles. La trajectoire du robot est générée avec le fichier `Simulink` donné en annexe A. La trajectoire est ensuite discrétisée en T étapes, et $\mathbf{x}_{\mathrm{p},0} = \mathbf{x}_{\mathrm{p},T}$. La figure 1.1 représente la trajectoire du robot, en prenant $T = 100$ étapes. Le modèle de notre robot est simple, il est inspiré du robot différentiel à deux roues comme Roomba d'iRobot [3], et l'on peut fixer dessus différents capteurs.

Concernant le capteur externe, nous optons pour un *Laser Range Finder* (LRF), ou télémètre laser. Il permet de mesurer des distances et des caps entre sa position et les objets qu'il pointe. Il utilise le temps que mettent les faisceaux pour partir jusqu'à la cible puis lui revenir pour déduire ensuite la distance qui le sépare de cette cible (technique du temps de vol). Nous nous inspirons des caractéristiques du capteur Velodyne LiDAR ULTRA Puck™ [1] en photo figure 1.2 pour dimensionner notre capteur (voir annexe B).

Le champ du vue large nous permet d'avoir une très bonne visibilité des alentours horizontalement, tandis que verticalement, il nous permet de voir des objets posés au sol (au niveau du plan du robot). Concernant les résolutions angulaires et la distance maximum, nous prenons

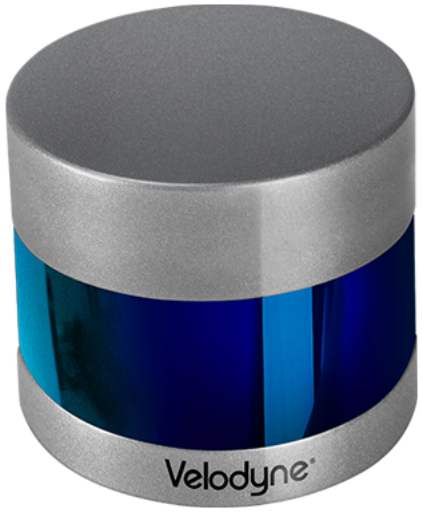


Figure 1.2 Velodyne LiDAR ULTRA Puck$^{\text{TM}}$ [1]

des valeurs plus faibles que pour le capteur Velodyne. À la vue de ces spécificités, le capteur effectue une révolution en 0.2 seconde, et on appelle scan entier les mesures effectuées par une révolution complète du capteur. Pendant une étape t du déplacement du robot, nous ne considérons qu'un scan entier $\tilde{\mathbf{Z}}_t$, effectué à la position $\mathbf{x}_{\text{p},t}$ du robot. $\tilde{\mathbf{Z}}_t$ se décompose en $K_t \in \mathbb{N}$ triplets $\tilde{\mathbf{z}}_k$, où un triplet représente une détection faite par le laser.

$$\tilde{\mathbf{Z}}_t = [\tilde{\mathbf{z}}_1 \ \dots \ \tilde{\mathbf{z}}_k \ \dots \ \tilde{\mathbf{z}}_{K_t}] = \begin{bmatrix} \tilde{d}_1 & \dots & \tilde{d}_k & \dots & \tilde{d}_{K_t} \\ \tilde{\lambda}_1 & \dots & \tilde{\lambda}_k & \dots & \tilde{\lambda}_{K_t} \\ \tilde{\phi}_1 & \dots & \tilde{\phi}_k & \dots & \tilde{\phi}_{K_t} \end{bmatrix}. \tag{1.1}$$

Ces mesures représentent la position des points repérés par le LRF en coordonnées sphériques dans le référentiel du robot, les angles $\lambda \in [-\pi, \pi]$ et $\phi \in [0, \pi/4]$ sont pour le cap et l'élévation, et d la distance, (voir figure 1.3).

Ainsi, connaissant la pose du robot $\mathbf{x}_{\text{p}}$, nous pouvons déterminer la position d'un point mesuré $\tilde{\mathbf{x}}$ dans le référentiel global de $\mathcal{E}$ avec la transformation suivante :

$$\tilde{\mathbf{x}} = \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{bmatrix} = \begin{bmatrix} x_{\text{p}} \\ y_{\text{p}} \\ z_{\text{p}} \end{bmatrix} + \tilde{d} \begin{bmatrix} \cos(\tilde{\lambda} + \psi_{\text{p}}) \cos\tilde{\phi} \\ \sin(\tilde{\lambda} + \psi_{\text{p}}) \cos\tilde{\phi} \\ \sin\tilde{\phi} \end{bmatrix}. \tag{1.2}$$

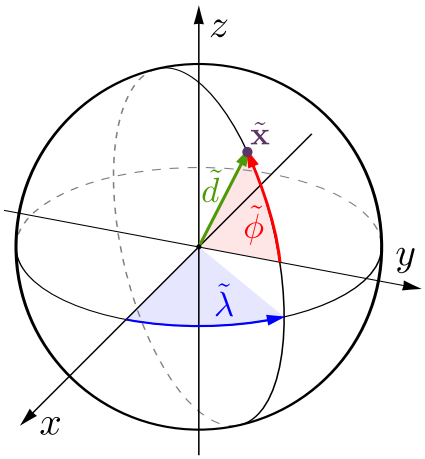


Figure 1.3 Coordonnées sphériques d'un point $\tilde{\mathbf{x}}$ mesuré par le LRF dans le référentiel du robot.

1.1.2 Le problème de localisation

Problème général

Soit un robot mobile évoluant dans l'environnement $\mathcal{E}$. Déterminer la pose $\mathbf{x}_{\mathrm{p},t}$ du robot dans le référentiel de $\mathcal{E}$ pour tout temps t revient à résoudre le problème de localisation. Pour se faire, le robot va s'aider d'une carte de l'environnement $\mathcal{E}$. Cette carte est composée d'une liste de paramètres $\{\mathbf{P}_m\}$ représentant les SQs associées aux objets de $\mathcal{E}$. Par ailleurs, le robot est muni du capteur externe, comme pour la cartographie, lui permettant d'effectuer des observations de $\mathcal{E}$ pour faire des correspondances avec la carte afin de se localiser. De plus, il utilise un capteur interne dont le fonctionnement est détaillé dans le paragraphe suivant. En combinant la connaissance de la carte avec les mesures effectuées par le robot, nous allons pouvoir traiter ce problème de localisation.

Capteur interne

Le second capteur repose sur l'odométrie et est composé de deux encodeurs fixés sur chacune des deux roues. Chaque encodeur calcule le déplacement (la rotation) de sa roue, pouvant alors estimer la position actuelle relativement à une position précédente. Nous considérons les rayons des roues parfaitement déterminés, ainsi que la distance L entre les deux roues. Les mesures d'odométrie pour la roue gauche et droite entre les étapes t et $t+1$ sont enregistrées dans le vecteur $\tilde{\mathbf{U}}_t = [\tilde{u}_{g,t}\ \tilde{u}_{d,t}]^T$. Nos mesures d'odométrie sont simulées à partir de la vitesse

du robot $v = \sqrt{v_x^2 + v_y^2}$ obtenue avec le schémas Simulink représenté figure A.1 :

$$\begin{bmatrix} u_{g,t} \\ u_{d,t} \end{bmatrix} = \begin{bmatrix} \Delta T v_t - \dfrac{L}{2} \left(\psi_{\mathrm{p},t+1} - \psi_{\mathrm{p},t}\right) \\ \Delta T v_t + \dfrac{L}{2} \left(\psi_{\mathrm{p},t+1} - \psi_{\mathrm{p},t}\right) \end{bmatrix}, \tag{1.3}$$

où ΔT représente le temps écoulé entre les étapes de discrétisation t et $t+1$. Ensuite, un bruit gaussien est rajouté artificiellement selon les caractéristiques voulues : $\tilde{\mathbf{U}}_t = \mathbf{U}_t + \mathbf{w}_t$. Tous nos bruits de mesures sont assimilés à des bruits blancs additifs gaussiens.

1.2 Éléments de la problématique

La cartographie pose le problème de représentation de carte. Un environnement peut être décrit par des points, des surfaces, des volumes, des objets, ou toutes ces choses à la fois. Dans la revue de littérature du chapitre 2, nous passons sur les représentations déjà utilisées auparavant. Les méthodes utilisant des nuages de points peuvent dépeindre un environnement précisément, mais demandent énormément de points pour décrire de grands objets. Une bonne représentation doit faire le compromis entre exactitude et compression des données. Ceci a motivé notre travail pour représenter l'environnement avec des superquadriques comme primitives géométriques. L'apport des SQs est une compression de données importante, puisque de gros objets pourraient être modélisés par seulement quelques paramètres. Les SQs peuvent prendre tout un tas de forme, allant du pavé droit aux cylindres. Leur versatilité permet de construire des cartes compressées sans perdre trop d'informations. Des SQs pourraient par exemple servir à modéliser des éoliennes en milieu extérieur, alors qu'en intérieur, elles peuvent décrire bon nombre de mobiliers. L'intégration des SQs apporte aussi son lot de problèmes, et nous tentons dans notre travail d'y apporter des solutions. Nous distinguons ci-dessous trois problèmes.

1.2.1 La création de superquadriques

Le premier problème que l'on rencontre concerne la création de SQs à partir des données des capteurs. Le LRF nous renseigne sur la position de points dans l'espace, et il faut alors, à partir de nuages de points, reconstruire des SQs.

Tout d'abord, nous ne considérons qu'un seul obstacle, et il s'agit de reconstruire une SQ à partir d'un nuage de points. Or, si ce nuage de points est obtenu depuis une position fixe donnée, l'obstacle ne pourra jamais être vu dans sa globalité. Ceci est dû au phénomène d'auto occlusion (voir figure 1.4). La reconstruction ne peut donc pas se faire sans ambiguïté,

et il faut pouvoir recueillir des données depuis différents points de vue. Toutefois, nous ne souhaitons pas garder en mémoire les mesures faites par le LRF d'une étape à l'autre, et d'une position à l'autre. Pour des raisons de mémoire et de capacité de calcul, il n'est pas souhaitable de reconstruire une SQ une fois le nuage entièrement collecté. Nous cherchons une solution itérative, qui nous permet d'estimer un obstacle au fur et à mesure que l'on collecte des points.

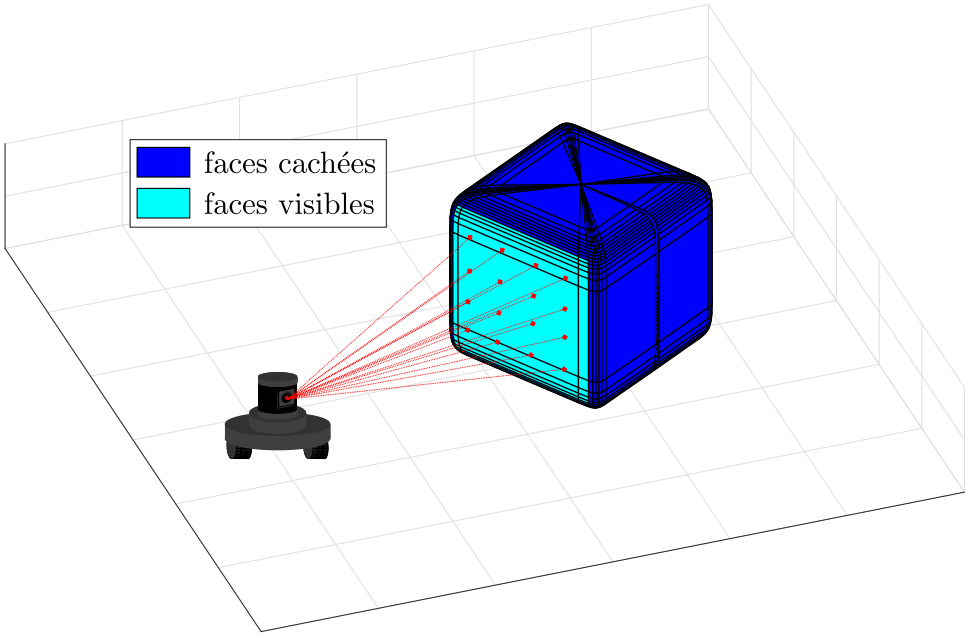


Figure 1.4 Illustration du phénomène d'auto occlusion pour une SQ.

1.2.2 L'association de données

Lorsque plusieurs objets sont observés, il faut pouvoir distinguer à quel objet appartiennent quelles mesures. Il faut donc pouvoir isoler dans le nuage de points les données représentant le même obstacle, et les associer à un obstacle en construction (ou un nouvel obstacle). Cette étape est appelée association de données. Ensuite, il faut pouvoir reconstruire chaque obstacle séparément.

1.2.3 La localisation à partir de superquadriques

Enfin, pour la localisation, notre robot se repère par rapport aux SQs dont il connaît la position et la forme. Toutefois, le robot n'observe pas directement les SQs, mais des points sur leur surface repérés par le LRF. Le problème revient à placer ces points correctement sur

lesdites surfaces des obstacles qui leur correspondent. Là aussi se pose d'abord le problème d'association de données.

1.3 Plan du mémoire

Dans ce mémoire nous commençons au chapitre 2 par une revue de littérature concernant différentes méthodes de positionnement et cartographie. Ensuite, le mémoire est décomposé en quatre thèmes. Le chapitre 3 explique comment reconstruire une SQ à partir d'un nuage de points en utilisant des techniques de minimisation par moindres carrés. Le chapitre 4 explique comment reconstruire une SQ par filtrage de Kalman. Pour ces deux chapitres nous nous attaquons d'abord au problème en 2D plus simple, puis nous généralisons en 3D. Le chapitre 5 concerne le problème de cartographie avec plusieurs obstacles dans l'environnement, en utilisant les techniques vues dans les deux chapitres précédents. Le chapitre 6 aborde le problème de localisation. Enfin nous concluons au chapitre 7 sur le travail qui a été fait, ses limites, et les possibles recherches futures.

CHAPITRE 2 REVUE DE LITTÉRATURE

Cette revue de littérature a vocation à introduire les problématiques de positionnement et cartographie en passant sur des solutions apportées. Nous discutons d'abord des différentes méthodes de localisation à la section 2.1. Nous traitons ensuite la formulation du problème de SLAM à la section 2.2. Il vient ensuite ce qui concerne la cartographie et les méthodes de représentation de l'environnement à la section 2.3. Nous introduirons alors ce que sont les superquadriques dans la dernière section 2.4.

2.1 Méthodes de positionnement

Pour un robot autonome, le positionnement représente l'habilité à se localiser dans son environnement. Plus généralement, on appelle aussi la navigation comme la capacité d'un robot à estimer son état, qui englobe sa position, sa vitesse, (linéaire comme angulaire) son orientation, voire aussi son accélération. La localisation est cruciale puisque sans elle, il est impossible pour un robot mobile de naviguer correctement et ainsi d'effectuer les tâches qui lui sont confiées. Pour se localiser, un robot autonome se base sur deux catégories de systèmes de localisation, et nous distinguons deux types de capteurs [4].

Les capteurs internes au robot auxquels celui-ci a toujours accès lui permettent une navigation à l'estime (*dead reckoning*). Ceux-ci mesurent la vitesse ou l'accélération, et l'intégration de ces mesures permet d'estimer l'orientation et la position. Comme ces mesures sont internes au robot, elles ont l'avantage d'être toujours accessibles et mises à jour à haute fréquence. Par contre, ces mesures ne sont pas faites dans un référentiel de navigation global, et ne renseignent alors que sur la position du robot par rapport à sa dernière position connue. De plus, l'intégration des mesures mène à des erreurs cumulatives responsables d'une erreur d'estimation sur la position qui ne peut que croître.

Il est donc impossible pour un robot autonome de se baser uniquement sur des mesures internes, et l'on comprend naturellement que celui-ci à besoin d'informations provenant de son environnement pour se localiser en accord avec celui-ci. Ces informations proviennent de systèmes externes, et le plus commun est la Géolocalisation et Navigation par un Système de Satellites (GNSS) [5]. Le robot mobile dispose alors d'un capteur pour recevoir ces signaux externes. Ces systèmes permettent alors une localisation absolue dans le référentiel de navigation. En revanche, le taux de rafraîchissement est plus faible pour les systèmes externes (souvent on parle de 1Hz contre 1kHz). Enfin, comme ces systèmes sont externes au robot,

ils ne sont pas toujours disponibles. Pour GNSS, il faut être à l'extérieur à découvert pour que le système fonctionne.

Un système de localisation performant se base souvent sur les mesures internes et externes. La fusion de ces mesures n'est pas triviale et une multitude de techniques ont été développées. L'approche la plus connue repose sur le filtrage de Bayes. Différents filtres ont été développés comme approximation de ce dernier et nous citons le Filtre de Kalman Étendu (EKF) ainsi que certaines variantes. Crassidis [6] utilise un Filtre de Kalman *Unscented* (UKF) pour intégrer des données GNSS à des mesures inertielles, tandis que Kong [7] utilise un EKF pour estimer l'orientation d'un aéronef avec un système inertiel. D'autres approches existent, comme les graphes de facteurs, qui estiment la position en utilisant l'optimisation non linéaire [8], pouvant alors fusionner n'importe quel type de mesure.

Quand les systèmes externes ne sont plus disponibles, il faut s'appuyer directement sur l'environnement dans lequel le robot évolue. Une technique peut être alors de se baser sur une carte de l'environnement. À l'aide de caméras ou de lasers, le robot peut parvenir à détecter à quel endroit il se trouve en se localisant sur la carte.

Corke [9] explique comment localiser un robot mobile à partir de points de l'environnement dont la position est parfaitement connue. Le robot est muni d'un laser pour observer ces points ainsi que d'odomètres. La première technique de localisation repose sur le filtrage de Kalman. La pose est d'abord estimée par intégration de l'odométrie, et elle est corrigée en comparant les observations estimées aux mesures faites par le laser. Les observations estimées sont obtenues à partir des points connus de la carte.

Comparer une mesure estimée à une mesure observée suppose que l'une est associée à l'autre, pourtant, aucun marqueur (en l'absence de systèmes externes) ne permet de distinguer un point d'un autre. Ceci représente le problème bien connu de l'association de données, qui est d'associer une mesure, une observation, à un repère ou un point dans l'environnement. Lors du filtrage, une mauvaise association de données peut mener à une mise à jour incorrecte de la pose, menant elle même à d'autres mauvaises associations. Cette étape est donc cruciale et a motivé le développement d'autres filtres. On cite notamment le filtre à particule [10] basé sur le principe de Monte-Carlo, qui considère plusieurs hypothèses lors de situations ambiguës. Ainsi, lors des observations futures, les hypothèses plus probables et conformes aux observations seront gardées, tandis que les autres hypothèses seront mises de côté.

Enfin, même sans connaissance de quelconque carte, nous pouvons tirer des informations de l'extérieur pour mieux nous localiser. L'odométrie visuelle se sert des observations externes pour ajuster la pose du robot relativement à des positions précédentes connues. Comme son nom l'indique, cette technique ne permet pas de localisation globale. L'odométrie visuelle

consiste à aligner des scans obtenus par deux relevés lasers (ou photos), et de traduire cet alignement comme une correction de la pose du robot. Plusieurs algorithmes d'alignement de points existent, le plus connu étant *Iterative Closest Point* (ICP) introduit par [11]. Par exemple, [12] reçoit des nuages de points d'un laser, et les aligne ensuite sur d'autres scans en utilisant ICP afin de construire une carte. Les techniques d'alignement de scans peuvent aussi être utilisées quand une carte est connue, mais plutôt que d'aligner des scans entre eux, le robot va tenter d'aligner un scan avec la carte elle même, permettant alors ici une localisation globale. Cette technique est notamment employée dans des entrepôts ou des usines où l'environnement est structuré comme [13].

2.2 Localisation et Cartographie Simultanées

Jusqu'ici nous n'avons pas abordé le problème de localisation d'un robot mobile dans son environnement global sans avoir à lui fournir une carte. Pour se faire, le robot peut construire lui-même sa carte. Ce problème est exactement le SLAM, qui représente pour un robot la construction concurrente d'une carte, tout en se localisant dans son environnement. Ainsi, en plus de permettre au robot de se localiser, la carte pourra servir pour un opérateur humain, ou encore faire de la planification de mouvement. Le domaine du SLAM est vaste et encore loin d'être résolu par la communauté scientifique, [14] et [15] présentent les travaux réalisés dans la discipline.

Comme pour le problème de localisation, il s'agit de fusionner les mesures de différents capteurs. Dans les années 1990s, les approches probabilistes sont les plus employées, comme les filtres de Kalman (Bailey [16] présente une analyse des différents EKF-SLAM). Des filtres à particules (FastSLAM [17] et FastSLAM2 [18]), et des algorithmes de maximisation d'espérance (maximum de vraisemblance) [19] sont aussi employés.

D'autres approches existent, et nous retrouvons les graphes de poses, où chaque noeud du graphe représente une position du robot à un instant donné, tandis que les arêtes reliant les noeuds représentent les contraintes. Lu et Milios [20] calculent les arêtes avec les contraintes odométriques (liens faibles entre deux noeuds successifs) et les contraintes liées au scan laser (liens forts entre deux noeuds où les scans se recoupent suffisamment). Ces derniers liens sont calculés avec un algorithme de superposition de scans [21]. Ensuite les positions sont estimées de façon à maximiser la vraisemblance des mesures (des contraintes du graphe) et [20] utilise simplement une estimation linéaire. Plus tard ont été introduits les graphes de facteurs, qui rajoutent des noeuds dans les graphes de poses pour représenter des marqueurs de l'environnement. Plusieurs solveurs non linéaires ont à leur tour été développés comme iSAM2 [22] et g2o [23].

Un problème phare de SLAM est l'association de données. Celle-ci permet au robot d'observer mieux comment il évolue vis à vis de son environnement. L'association de données est la capacité à reconnaitre dans l'environnement des marqueurs, et de pouvoir associer plusieurs mesures externes à ces mêmes marqueurs. Ces marqueurs sont diversifiés et peuvent être un angle [24], un descripteur SIFT [25] ou ORB [26], ou plus élaboré comme une ligne, un point de fuite ou un plan, tous très utilisés dans le SLAM visuel (vSLAM [27], MonoSLAM [28], ORB-SLAM [29]). L'utilisation de marqueurs plus élaborés améliore la fiabilité de l'association de données, et facilite les techniques d'ajustement de faisceaux et d'odométrie visuelle [30]. À plus long terme dans le trajet du robot, l'association de données mène à ce qu'on appelle la fermeture de boucle. Il s'agit de reconnaître un lieu déjà visité auparavant par le robot, permettant alors d'ajouter des contraintes dans la carte et raffiner l'estimation de la position du robot.

2.3 Carte métrique de l'environnement

Lors de la cartographie vient la question de la représentation de la géométrie de l'environnement. Sous quelle forme stocker toutes les informations reçues par les capteurs externes ? En SLAM, on parle de représentation métrique, ou de carte métrique pour désigner l'encodage de la carte.

La première représentation concerne les cartes quadrillées d'occupation. Souvent utilisée en 2D, cette représentation discrétise la carte en cellules, et assigne une probabilité d'occupation à chacune d'entre elle. Cette représentation très simple est inefficace en terme de mémoire, et [31] propose une représentation compressée OctoMap basée sur l'Octree.

Une autre méthode est de représenter la scène comme un ensemble de points repérés dans l'environnement (*landmarks*), comme le fait ORB-SLAM [29]. La caractéristique de ces points c'est qu'ils peuvent être distingués parmi les autres dans leur environnement, grâce à certains descripteurs. L'avantage de cette représentation facilite l'association de données et les fermetures de boucle grâce aux descripteurs.

Les méthodes directes reconstruisent les cartes avec des données brutes, en utilisant par exemple directement les pixels d'images ou les nuages de points d'un laser. Par exemple DTAM [32] et REMODE [33] reconstruisent une carte dense en utilisant un flux monoculaire. Ces méthodes fonctionnent mieux dans des environnements plus pauvres, ou quand les données sont plus bruitées (caméra mal calibrée) et où il est difficile de reconnaître de manière fiable des formes géométriques. Le désavantage majeur réside dans la puissance de calcul nécessaire pour traiter et garder en mémoire tout ce flux d'information.

Il existe des représentations plus complexes où la carte est représentée par des surfaces (des plans) ou des volumes. Ceci à l'avantage de rendre mieux compte de la physique de l'environnent, et par exemple, un robot pourra plus facilement gérer l'évitement et la collision d'obstacles, ou encore la manipulation d'objets. Castle [34] utilise une caméra pour localiser des plans dans son environnement et [35] utilise un capteur RGB-D afin de construire des plans infinis. Ce type de primitive géométrique plus élaboré que des points ou des lignes permet une meilleure précision quant à l'estimation des cartes et la robustesse lors du suivi. D'autres objets géométriques peuvent encore être utilisés comme des surfaces courbées (courbes de Bézier, B-splines, NURBS [36]).

Enfin, les dernières représentations sont encore plus riches au niveau sémantique, et certains algorithmes utilisent des objets 3D de la vie courante. Ces représentations plus récentes tirent profit des capteurs de profondeur et de leur association avec une caméra (images RGB-D type Kinect). Modéliser des objets permet de leur associer un volume, une masse, et donc de créer une carte facilitant encore plus l'interaction robot-environnement. SLAM++ [37] utilise une base de données préalable d'objets 3D et essaie de les associer dans l'environnement, tandis que [38] distingue des objets dans une carte dense en les segmentant pour les réaligner ensuite sur le plan qui les supporte. Ces représentations tirent profit des améliorations de performance récentes en détection d'objets et en méthodes d'apprentissage automatique profond.

2.4 Les superquadriques comme primitives géométriques

Nous nous proposons d'utiliser les superquadriques (SQs) pour représenter notre environnement. Ces objets ont pour avantages d'être paramétriques et compactes en terme de mémoire, car seuls cinq paramètres suffisent à décrire leur forme, plus six autres pour leur pose dans l'espace 3D. Les SQs découlent des superellipses (courbes bi-dimensionnelles) décrites pour la première fois par Gabriel Lamé en 1818 [39]. Les SQs ont été introduites en vision par ordinateur par [40] au début des années 1980s, et représentent dès lors un outil de modélisation géométrique, notamment en infographie et en robotique. Jaklic [41] raconte l'évolution de la place des SQs en vison par ordinateur et utilise celles-ci afin de segmenter une image 3D (*range image)* en différentes SQs. Il explique entre autre comment nous pouvons décrire une image avec des SQs.

CHAPITRE 3 RECONSTRUCTION D'UNE SUPERQUADRIQUE PAR LES MOINDRES CARRES

Nous nous intéressons dans ce chapitre à la construction d'une SQ à partir d'un nuage de points. Ce problème nous intéresse ici car lors de la cartographie et de la localisation, le LRF va renvoyer des nuages de points qu'il va falloir transformer en SQs. Nous commençons avec le problème plus simple des ellipses en deux dimensions, pour finir avec le modèle généralisé des SQs.

3.1 Reconstruction d'une ellipse

Dans cette section nous expliquons comment construire une ellipse à partir d'un nuage de points bi-dimensionnel. Ainsi nous supposons que notre nuage représente une unique ellipse affectée d'un bruit gaussien non biaisé. Le modèle à retrouver est composé de cinq paramètres $\mathbf{P} = [x_c, y_c, a, b, \psi]$. Le couple (x_c, y_c) représente le centre de l'ellipse, (a, b) ses demis grands axes, et ψ son orientation par rapport au référentiel global. Pour lever les ambiguïtés, tous nos angles sont pris dans l'intervalle $[-\pi, \pi]$. Nous faisons donc face à un problème d'ajustement (*fitting*) de données.

3.1.1 Méthode des moindres carrés linéaires

La méthode des moindres carrés permet de comparer nos données expérimentales entachées d'erreurs de mesure à un modèle mathématique. En l'occurrence, notre modèle est celui de l'ellipse et comporte cinq paramètres :

$$F(\mathbf{x}; \mathbf{P}) = \mathbf{x}^T \mathbf{A} \mathbf{x} + \mathbf{b}^T \mathbf{x} = 1. \tag{3.1}$$

Le vecteur $\mathbf{x} = (x, y)^T$ représente un point sur le contour de l'ellipse. La matrice $\mathbf{A}$ est symétrique définie positive et $\mathbf{b}$ est un vecteur. Nous pouvons réécrire l'équation de notre ellipse sous la forme :

$$F(\mathbf{x}; \mathbf{P}) = Ax^2 + By^2 + Cxy + Dx + Ey = 1. \tag{3.2}$$

Trouver notre modèle revient à trouver les cinq coefficients $\boldsymbol{\beta} = [A, B, C, D, E]^T$. En consi-

dérant notre ensemble de points $\{\mathbf{x}_k\}_{k\in[\![1,K]\!]}$, nous construisons la matrice :

$$\mathbf{X} = \begin{bmatrix} x_1^2 & y_1^2 & x_1y_1 & x_1 & y_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_K^2 & y_K^2 & x_Ky_K & x_K & y_K \end{bmatrix}. \tag{3.3}$$

Il s'agit de résoudre pour $\boldsymbol{\beta}$:

$$\mathbf{X}\boldsymbol{\beta} = \mathbf{1}, \tag{3.4}$$

avec $\mathbf{1}$ le vecteur de taille K rempli de 1. Ensuite, nous devons nous assurer que la matrice $\mathbf{X}$ et de plein rang (c'est à dire ici $\text{rank}(\mathbf{X}) = 5$). Cette condition est vérifiée si nous avons au moins 5 points $\mathbf{x}$ différents et non nuls. En effet, avec moins de 5 points, plusieurs ellipses peuvent êtres candidates et la solution n'est pas unique, tandis que le point $\mathbf{x} = (0, 0)$ ne peut être solution de $F(\mathbf{x}, \mathbf{P}) = 1$. Une fois s'être assuré de ces conditions, les coefficients de $\boldsymbol{\beta}$ minimisant l'erreur quadratique vérifient (estimateur des Moindres Carrés (MC)) :

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{1}. \tag{3.5}$$

Enfin, il faut identifier dans $\hat{\boldsymbol{\beta}}$ les paramètres estimés $\hat{\mathbf{P}}$. Les correspondances sont détaillées dans [42]. Nous représentons figure 3.1 la reconstruction d'une ellipse à partir de points bruités. La reconstruction se passe très bien dans le cas où nous avons des points bien répartis autour de l'ellipse, puis se dégrade si l'on ne considère que les points d'un côté de l'ellipse.

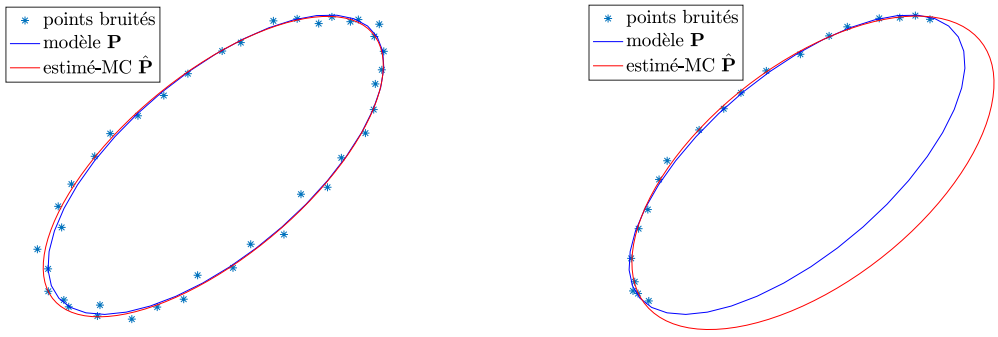


Figure 3.1 Reconstruction d'une ellipse par la méthode des MC linéaires.

3.1.2 Méthode des moindres carrés non linéaires

Nous utilisons maintenant l'autre forme de l'équation de l'ellipse :

$$F(x,y;\mathbf{P}) = \left(\frac{(x-x_\mathrm{c})\cos\psi + (y-y_\mathrm{c})\sin\psi}{a}\right)^2 + \left(\frac{(x-x_\mathrm{c})\sin\psi - (y-y_\mathrm{c})\cos\psi}{b}\right)^2 = 1. \tag{3.6}$$

Nous souhaitons trouver directement les paramètres de $\mathbf{P}$ sans passer par des coefficients. La formulation du problème des MC devient :

$$\hat{\mathbf{P}} = \underset{\mathbf{P}}{\operatorname{argmin}} \sum_{i=k}^{K} (F(x_k, y_k; \mathbf{P}) - 1)^2. \tag{3.7}$$

Cette formulation n'est ni linéaire, ni polynomiale, et nous devons alors utiliser une méthode non linéaire de résolution. Ces méthodes fonctionnent itérativement, et calculent un estimé $\hat{\mathbf{P}}_i$ à chaque étape, jusqu'à ce qu'un critère de convergence soit vérifié. Nous choisissons un algorithme à région de confiance, décrit dans [43]. Celui-ci à l'avantage de supporter les contraintes de frontière, et peut effectuer de grands pas lorsque la solution initiale est éloignée de l'optimale. Un estimé initial $\hat{\mathbf{P}}_0$ doit être fourni à l'algorithme. Nous remarquons dans le cas des ellipses que l'initialisation n'est pas cruciale, mais nous prenons $(\hat{x}_{\mathrm{c},0}, \hat{y}_{\mathrm{c},0})$ égal au barycentre de notre nuage de points $\{\mathbf{x}_k\}$. Concernant les paramètres $\hat{a}_0$, $\hat{b}_0$ et $\hat{\psi}_0$, nous pouvons effectuer une Analyse en Composantes Principales (ACP) [44] sur le nuage de point. Cette analyse est faite en considérant la matrice suivante :

$$\mathbf{M} = \begin{bmatrix} \widetilde{\mathbb{V}}(\{x_k\}) & \widetilde{\mathbb{V}}(\{x_k\},\{y_k\}) \\ \widetilde{\mathbb{V}}(\{y_k\},\{x_k\}) & \widetilde{\mathbb{V}}(\{y_k\}) \end{bmatrix}. \tag{3.8}$$

Nous définissons les notations suivantes.

- $\widetilde{\mathbb{V}}(\{x_k\}) = \widetilde{\mathbb{E}}((x_k - \widetilde{\mathbb{E}}(x_k))^2)$ dénote la variance empirique des composantes x des points.
- $\widetilde{\mathbb{V}}(\{x_k\},\{y_k\}) = \widetilde{\mathbb{E}}((x_k - \widetilde{\mathbb{E}}(x_k))(y_k - \widetilde{\mathbb{E}}(y_k)))$ dénote la covariance empirique entre les composantes x et y des points.

Ensuite, nous effectuons la décomposition de cette matrice en éléments propres [45]. Cette décomposition est toujours possible car $\mathbf{M}$ est symétrique et réelle [45]. Nous obtenons :

$$\mathbf{M} = \mathbf{V}\mathbf{D}\mathbf{V}^T, \tag{3.9}$$

avec $\mathbf{V}$ une matrice orthogonale et $\mathbf{D}$ une matrice diagonale. Enfin nous choisissons :

$$\begin{cases} \psi_0 = \text{atan2}\left(\mathbf{V}_{2,1}, \mathbf{V}_{1,1}\right), \\ a_0 = \sqrt{2\mathbf{D}_{2,2}}, \\ b_0 = \sqrt{2\mathbf{D}_{1,1}}. \end{cases} \tag{3.10}$$

Notons que a_0 et b_0 sont toujours définis car la diagonale de la matrice $\mathbf{D}$ ne comporte que des coefficients positifs. Ceci s'explique en démontrant que $\mathbf{M}$ est une matrice définie positive :

. $\mathbf{M}$ est symétrique car $\widetilde{\mathbb{V}}(\{x_k\},\{y_k\}) = \widetilde{\mathbb{V}}(\{y_k\},\{x_k\})$,

. $\mathbf{M}_{1,1} > 0$ car $\widetilde{\mathbb{V}}(\{x_k\}) > 0$ (une variance est toujours positive dans notre situation),

. $\mathbf{M}_{1,1}\mathbf{M}_{2,2} - (\mathbf{M}_{2,1})^2 > 0$ car par Cauchy-Schwartz on a $\widetilde{\mathbb{V}}(\{x_k\})\widetilde{\mathbb{V}}(\{y_k\}) > \widetilde{\mathbb{V}}(\{x_k\},\{y_k\})^2$.

Ainsi, d'après [46], $\mathbf{M}$ est définie positive, et ses valeurs propres sont positives.

Cette méthode d'initialisation est particulièrement efficace pour les nuages de points homogènement distribués autour de l'ellipse, comme montré figure 3.2 à gauche. Pour un nuage de points concentré que sur la moitié de l'ellipse, nous voyons figure 3.2 à droite que l'ACP nous donne un résultat moins satisfaisant.

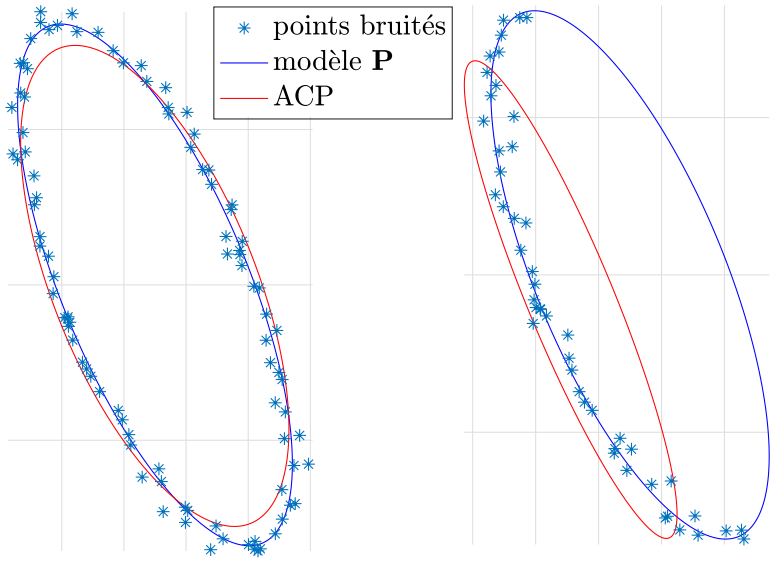


Figure 3.2 Résultat de l'ACP à partir de différents nuages de points.

Revenons un instant à notre environnement global. Nous avons un LRF fixé sur un robot qui scanne son environnement. Du point de vue du robot, il sera impossible d'observer un obstacle en entier. C'est le problème d'auto occlusion. Il faut garder ceci à l'esprit lorsque nous voulons faire correspondre un modèle à des données partielles. La figure 3.3 à gauche montre la reconstruction d'une ellipse par la méthode des Moindres Carrés Non Linéaire (MCNL)

lors d'un phénomène d'auto occlusion. L'estimé $\hat{\mathbf{P}}$ se superpose bien aux données, mais est trop grand par rapport au modèle. Il est dans notre intérêt de favoriser des estimés petits, c'est à dire avec des paramètres estimés $(\hat{a}, \hat{b})$ plus petits. En effet, nous voulons rendre compte que si le LRF n'a encore rien observé dans une zone, c'est qu'il n'y a rien à détecter pour le moment. Par ce fait, nous voulons trouver le plus petit modèle pouvant correspondre aux points détectés. Pour ce faire [47] introduit une pénalité sur le volume en changeant la fonction à minimiser :

$$\hat{\mathbf{P}} = \underset{\mathbf{P}}{\operatorname{argmin}} \sum_{k=1}^{K} (\sqrt{ab}(F(x_k, y_k; \mathbf{P}) - 1))^2. \tag{3.11}$$

La pénalité se traduit dans le facteur ab qui est proportionnel à l'aire de l'ellipse $A_e = \pi ab$. Cette modification avantage le modèle de plus petite taille quand plusieurs modèles recouvrent les points de manière équivalente. La figure 3.3 montre la reconstruction de la même ellipse, mais avec cette méthode de Moindres Carrés Non Linéaires avec Pénalité de Volume (MCNL-PV). Nous observons que la reconstruction se passe mieux. L'estimé est notablement plus petit et mieux aligné avec le modèle. Une autre technique pour limiter les trop grandes reconstructions est aussi de rajouter lors de la minimisation des contraintes (bornes supérieurs) pour les paramètres a et b.

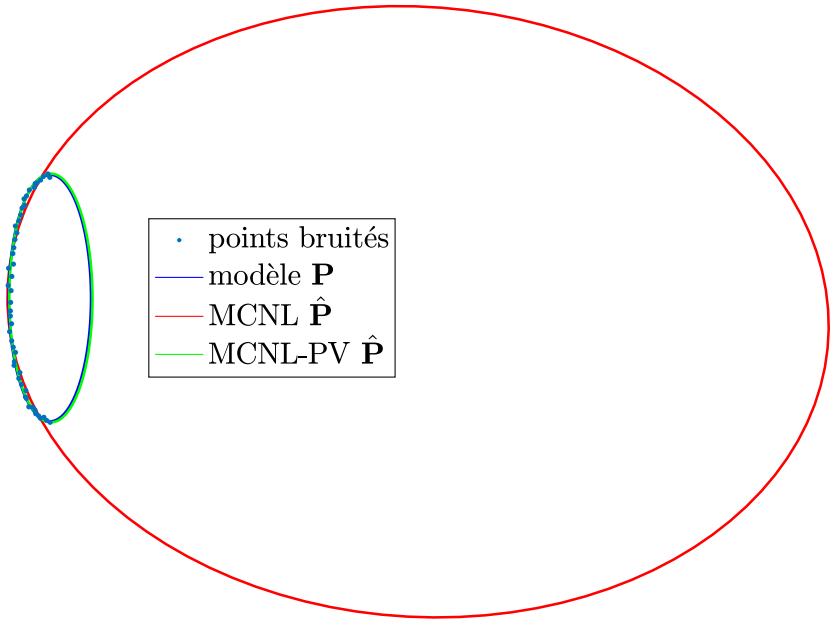


Figure 3.3 Reconstruction d'une ellipse par la méthode des MCNL en rouge, et MCNL-PV en vert, lors d'un phénomène d'auto occlusion.

3.2 Reconstruction d'une superellipse

3.2.1 Les superellipses

Les superellipses sont l'extension des ellipses et ont été définies pour la première fois par Gabriel Lamé [39]. Elles sont définies dans le plan par les points $\{(x, y)\}$ vérifiant :

$$\left|\frac{x}{a}\right|^n + \left|\frac{y}{b}\right|^n = 1. \tag{3.12}$$

Pour $n = 2$ nous retrouvons l'équation de l'ellipse. Un paramètre vient donc s'ajouter pour décrire la forme de la superellipse. Nous réécrivons l'équation de la superellipse comme telle :

$$F(\mathbf{x}; \mathbf{P}) = \left(\frac{x^{\mathsf{s}}}{a}\right)^{\frac{2}{\epsilon}} + \left(\frac{y^{\mathsf{s}}}{b}\right)^{\frac{2}{\epsilon}} = 1. \tag{3.13}$$

Nous remplaçons n par $2/\epsilon$. Aussi, et ce jusqu'à la fin, nous utilisons $x^{\frac{2}{\epsilon}}$ pour désigner $(x^2)^{\frac{1}{\epsilon}} = |x|^{\frac{2}{\epsilon}}$, notation souvent utilisée dans la littérature. Nous rajoutons le paramètre ϵ à $\mathbf{P} = [x_{\mathrm{c}}, y_{\mathrm{c}}, a, b, \psi, \epsilon]$. De plus, nous notons $\mathbf{x}^{\mathsf{s}} = [x^{\mathsf{s}}\ y^{\mathsf{s}}]^T$ les coordonnées d'un point dans le référentiel de la quadrique :

$$\begin{bmatrix} x^{\mathsf{s}} \\ y^{\mathsf{s}} \end{bmatrix} = \mathbf{R}^T \begin{bmatrix} x - x_{\mathrm{c}} \\ y - y_{\mathrm{c}} \end{bmatrix}, \quad \mathbf{R} = \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix}. \tag{3.14}$$

La figure 3.4 illustre l'allure de la superellipse selon la valeur que prend $\epsilon \in [0, 2]$. Pour $\epsilon > 2$ les figures obtenues ne sont plus convexes, et dans notre travail nous nous restreignons aux figures convexes.

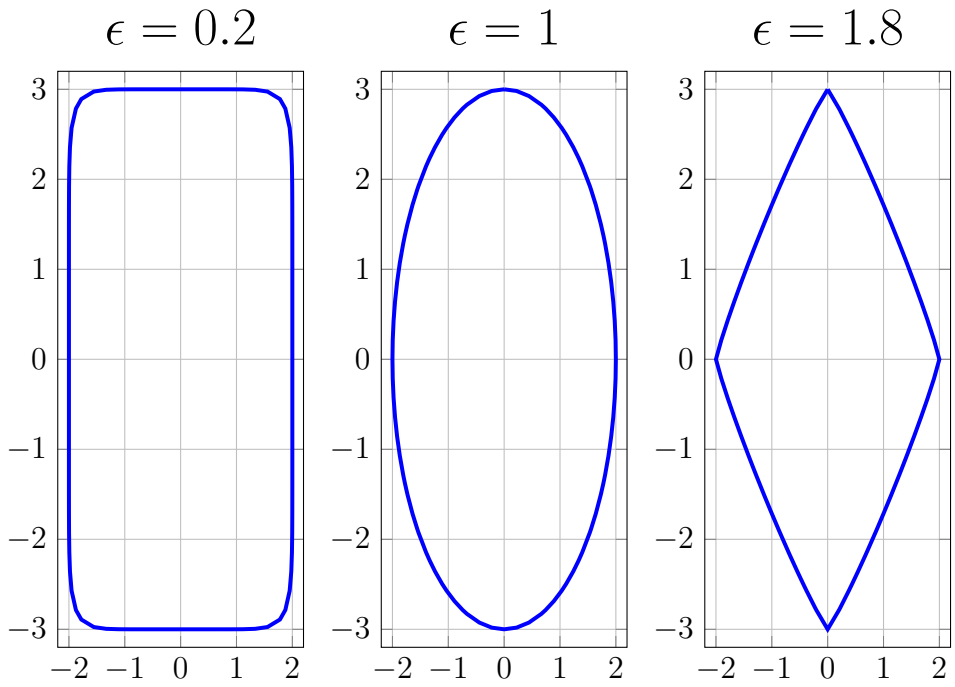


Figure 3.4 Superellipses d'allure dépendante de ϵ, $x_c = y_c = 0,\ a = 3,\ b = 2,\ \psi = \pi/2$.

3.2.2 Moindres Carrés linéaires pour les superellipses

De la même façon que pour les ellipses, nous pouvons essayer de retrouver par moindres carrés linéaires nos superellipses. Pour ceci, il faut pouvoir exprimer l'équation 3.13 de $F(\mathbf{x}; \mathbf{P})$ sous forme polynomiale en terme de $\mathbf{x}$. C'est ce que nous avons fait précédemment quand $\epsilon = 1$, dans le cas de l'ellipse simple. L'exercice ici est moins évident, et en fait il n'existe pas de forme polynomiale quand $2/\epsilon$ n'est pas entier. Nous pouvons nous intéresser aux courbes de Lamé, prenant $2/\epsilon = n \in \mathbb{N}$, définies ci dessous :

$$F(x, y; \mathbf{P}) = \left(\frac{(x - x_c)\cos\psi + (y - y_c)\sin\psi}{a} \right)^n + \left(\frac{(x - x_c)\sin\psi - (y - y_c)\cos\psi}{b} \right)^n = 1. \tag{3.15}$$

Nous réécrivons la fonction comme suit :

$$\begin{aligned} F(x, y; \mathbf{P}) = &\left(\frac{x\cos\psi + y\sin\psi - (x_c\cos\psi + y_c\sin\psi)}{a} \right)^n \\ &+ \left(\frac{x\sin\psi + y\cos\psi - (x_c\sin\psi - y_c\cos\psi)}{b} \right)^n = 1. \end{aligned} \tag{3.16}$$

En utilisant la formule du multinôme de Newton, on a par exemple :

$$\begin{aligned}&\left(x\cos\psi + y\sin\psi - (x_c\cos\psi + y_c\sin\psi)\right)^n \\ &= \sum_{k_1+k_2+k_3=n} \begin{bmatrix} n \\ k_1, k_2, k_3 \end{bmatrix} [x^{k_1}\ y^{k_2}\ 1] \begin{bmatrix} \cos^{k_1}\psi \\ \sin^{k_2}\psi \\ (-x_c\cos\psi - y_c\sin\psi)^{k_3} \end{bmatrix}\end{aligned} \tag{3.17}$$

La forme polynomiale développée compte alors $\frac{(n+1)(n+2)}{2}$ coefficients et n'est donc absolument pas indépendante de l'ordre de la courbe de Lamé. Ceci signifie que le problème des moindres carrés linéaires devrait être posé différemment selon la forme de la superellipse. Aussi, le nombre de coefficients à calculer est équivalent à $(n^2/2)$. Ainsi, pour une superellipse plutôt rectangulaire, avec le paramètre $\epsilon = 0.1$, soit $n = 20$, nous aurions besoin au moins de 200 points pour calculer autant de coefficients. Une solution serait de trouver des relations algébriques entre ces coefficients afin d'en réduire le nombre, mais ceci reste extrêmement fastidieux. Il faut aussi pouvoir finalement transformer tout ces coefficients en cinq paramètres contenus dans $\mathbf{P}$. Pour toutes ces raisons, nous favorisons par la suite l'approche non linéaire.

3.2.3 Moindres Carrés Non Linéaires pour les superellipses

De même que pour les ellipses, nous souhaitons à partir d'un nuage de points, retrouver le modèle $\hat{\mathbf{P}}$ qui correspond le mieux, et [41] utilise la fonction de regression suivante :

$$F_S(\mathbf{x};\mathbf{P}) = \sqrt{ab}\,(F(\mathbf{x};\mathbf{P})^\epsilon - 1) = \sqrt{ab}\left(\left(\left(\frac{x^{\mathsf{s}}}{a}\right)^{\frac{2}{\epsilon}} + \left(\frac{y^{\mathsf{s}}}{b}\right)^{\frac{2}{\epsilon}}\right)^{\epsilon} - 1\right). \tag{3.18}$$

Ceci rend la métrique d'erreur indépendante de la forme de la superellise, sans affecter la forme de la superellipse quand $F = 1$. Sur la figure 3.6 à gauche nous observons le résultat de la minimisation de cette fonction par notre solveur. Le modèle estimé est bien aligné sur les points, mais dans le mauvais sens. Nous aimerions trouver plutôt quelque chose comme sur la figure 3.6 à droite. Il faut alors modifier la fonction objective à minimiser. Nous introduisons la méthode de Moindres Carrés Non Linéaires Favorisant la Visibilité (MCNL-FV), pensée par [2]. Cette méthode prend en compte le volume de la quadrique, la distance des points à sa surface, mais aussi sa visibilité. La fonction à minimiser pensée par [2] comporte trois termes :

$$G = \alpha_1 G_V + \alpha_2 G_F + \alpha_3 G_\nabla, \tag{3.19}$$

avec $G_V = ab$ qui correspond à la pénalité sur le volume et $G_F = (F^\epsilon - 1)^2$ qui correspond

à la distance radiale d'un point à la surface. La nouveauté se cache dans le terme G_∇, qui rend compte que de la position $\mathbf{x}_\mathrm{p}$ du robot, la partie cachée de la quadrique doit se trouver derrière sa partie observée. Ceci rendrait la reconstruction figure 3.6 à droite impossible, et à gauche préférable. G_∇ est calculé à l'aide d'un produit scalaire utilisant la position du robot dans le référentiel global $[x_\mathrm{p}\ y_\mathrm{p}]^T$. On appelle $\nabla_r F(\mathbf{x};\mathbf{x}_\mathrm{p},\mathbf{P})$ la dérivée directionnelle du point $\mathbf{x}$ associée à la superellipse $\mathbf{P}$ relativement à la position d'observation $\mathbf{x}_\mathrm{p}$:

$$\nabla_r F(\mathbf{x};\mathbf{x}_\mathrm{p},\mathbf{P}) = \mathbf{u}(\mathbf{x};\mathbf{x}_\mathrm{p}) \cdot \nabla F(\mathbf{x};\mathbf{P}), \tag{3.20}$$

avec :

$$\mathbf{u}(\mathbf{x};\mathbf{x}_\mathrm{p}) = \frac{\begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x_\mathrm{p} \\ y_\mathrm{p} \end{bmatrix}}{\left\| \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x_\mathrm{p} \\ y_\mathrm{p} \end{bmatrix} \right\|_2}\ , \text{ et } \nabla F(\mathbf{x};\mathbf{P}) \text{ le gradient de } F \text{ au point } \mathbf{x} \text{ (la normale à la surface).} \tag{3.21}$$

Une illustration du produit scalaire est montrée figure 3.5. Avoir un produit scalaire négatif pour chaque point $\mathbf{x}_k$ signifie que nous sommes dans le cas désiré, c'est à dire que l'obstacle est observé depuis son extérieur.

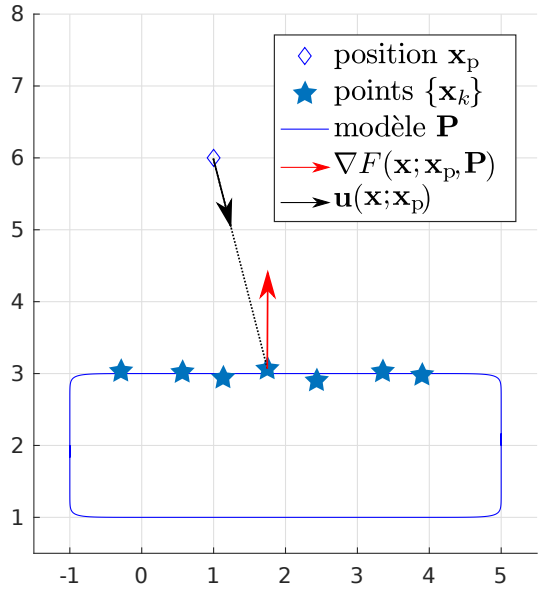


Figure 3.5 Illustration du calcul de la dérivée directionnelle de [2].

Finalement, nous avons le terme G_∇ :

$$G_\nabla = \frac{1}{2}(1 + \tanh(\alpha_0 \nabla_r F)). \tag{3.22}$$

Ainsi, lorsque le produit scalaire est très négatif, G_∇ devient proche de 0, tandis que lorsqu'il devient positif, alors G_∇ s'approche de 1, ce qui pénalise la fonction de coût totale G. Les paramètres $[\alpha_0, \alpha_1, \alpha_2, \alpha_3]$ sont fixés par [2] à $[2, 1, 80, 30]$. Nous avons finalement :

$$G(\mathbf{x}; \mathbf{x}_\mathrm{p}, \mathbf{P}) = \alpha_1 ab + \alpha_2 \left(F(\mathbf{x}; \mathbf{P})^\epsilon - 1\right)^2 + \alpha_3 \frac{1}{2}(1 + \tanh(\alpha_0 \nabla_r F(\mathbf{x}; \mathbf{x}_\mathrm{p}, \mathbf{P}))). \tag{3.23}$$

Figure 3.6 montre donc l'apport de la méthode MCNL-FV devant la méthode MCNL-PV.

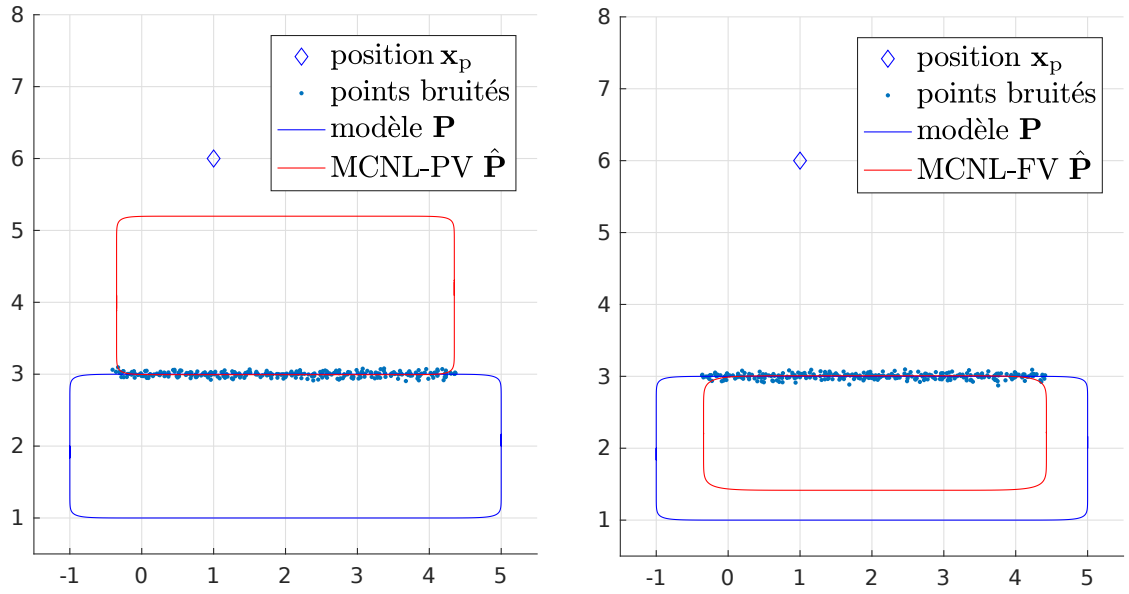


Figure 3.6 Reconstruction d'une superellipse par la méthode des MCNL-PV à gauche, et MCNL-FV à droite, lors d'un phénomène marqué d'auto occlusion.

3.3 Reconstruction d'une superquadrique

3.3.1 Les superquadriques

Les SQs sont l'extension des ellipsoïdes (ellipses tri-dimensionnelles). En quelques sorte, les SQs sont aux ellipsoïdes ce que sont les superellipses aux ellipses. Une SQ est définie par onze

paramètres, qu'on ajoute au vecteur $\mathbf{P}$:

$$\mathbf{P} = [x_c, y_c, z_c, a, b, c, \phi, \theta, \psi, \epsilon_1, \epsilon_2], \tag{3.24}$$

on compte trois paramètres pour son centre, trois pour sa taille, trois pour son orientation, et deux pour sa forme. L'équation de la SQ est :

$$F(\mathbf{x}; \mathbf{P}) = \left(\left(\frac{x^{\mathsf{s}}}{a} \right)^{\frac{2}{\epsilon_2}} + \left(\frac{y^{\mathsf{s}}}{b} \right)^{\frac{2}{\epsilon_2}} \right)^{\frac{\epsilon_2}{\epsilon_1}} + \left(\frac{z^{\mathsf{s}}}{c} \right)^{\frac{2}{\epsilon_1}} = 1. \tag{3.25}$$

Comme nous sommes désormais en trois dimensions, nous avons ajouté à $\mathbf{x}$ sa troisième composante. Les figures 3.7 et 3.8 montrent plusieurs exemples de superquadriques. De même que pour les superellipses, $\mathbf{x}^{\mathsf{s}}$ désigne les coordonnées d'un point dans le référentiel de la superquadrique. On définit le changement de base dans ce référentiel :

$$\mathbf{R}_\phi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}, \ \mathbf{R}_\theta = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}, \ \mathbf{R}_\psi = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$\mathbf{R} = \mathbf{R}_\psi \mathbf{R}_\theta \mathbf{R}_\phi.$$

$$\begin{bmatrix} x^{\mathsf{s}} \\ y^{\mathsf{s}} \\ z^{\mathsf{s}} \end{bmatrix} = \mathbf{R}^T \begin{bmatrix} x - x_c \\ y - y_c \\ z - z_c \end{bmatrix}. \tag{3.26}$$

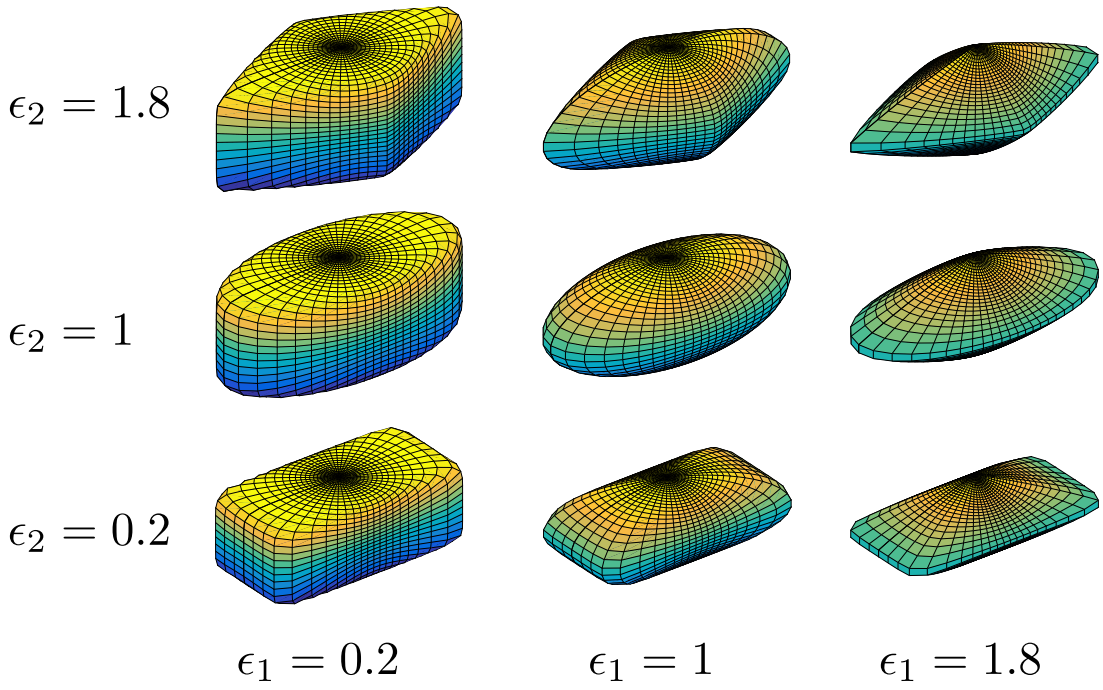


Figure 3.7 Plusieurs superquadriques de même taille $a = 8,\ b = 4,\ c = 3$.

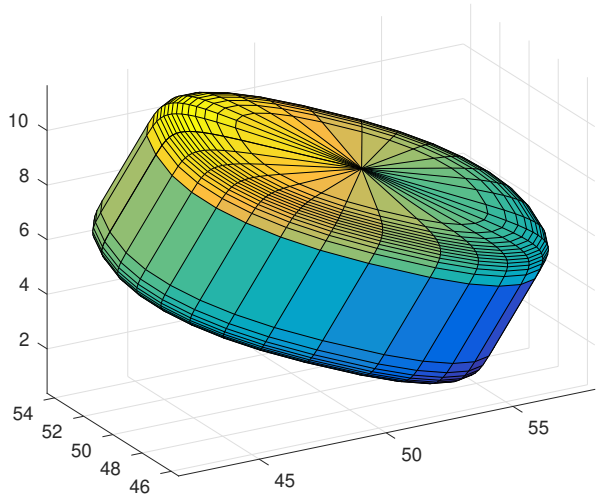


Figure 3.8 Superquadrique de paramètres $\mathbf{P} = [50, 50, 6, 8, 4, 3, \pi/10, \pi/8, 0, 0.2, 0.8]$.

3.3.2 Moindres Carrés pour les superquadriques

Il s'agit maintenant de retrouver une SQ à partir d'un nuage de points en trois dimensions. Les fonctions de coût vues précédemment pour les superellipses s'étendent facilement au modèle 3D, et [41] utilise :

$$F_S(\mathbf{x};\mathbf{P}) = \sqrt{abc}\,(F(\mathbf{x};\mathbf{P})^{\epsilon_1} - 1) = \sqrt{abc}\left(\left(\left(\left(\frac{x^s}{a}\right)^{\frac{2}{\epsilon_2}} + \left(\frac{y^s}{b}\right)^{\frac{2}{\epsilon_2}}\right)^{\frac{\epsilon_2}{\epsilon_1}} + \left(\frac{z^s}{c}\right)^{\frac{2}{\epsilon_1}}\right)^{\epsilon_1} - 1\right). \tag{3.27}$$

La figure 3.9 montre le résultat de la minimisation de F_S^2 appliquée aux points en vert, avec la méthode de région de confiance.

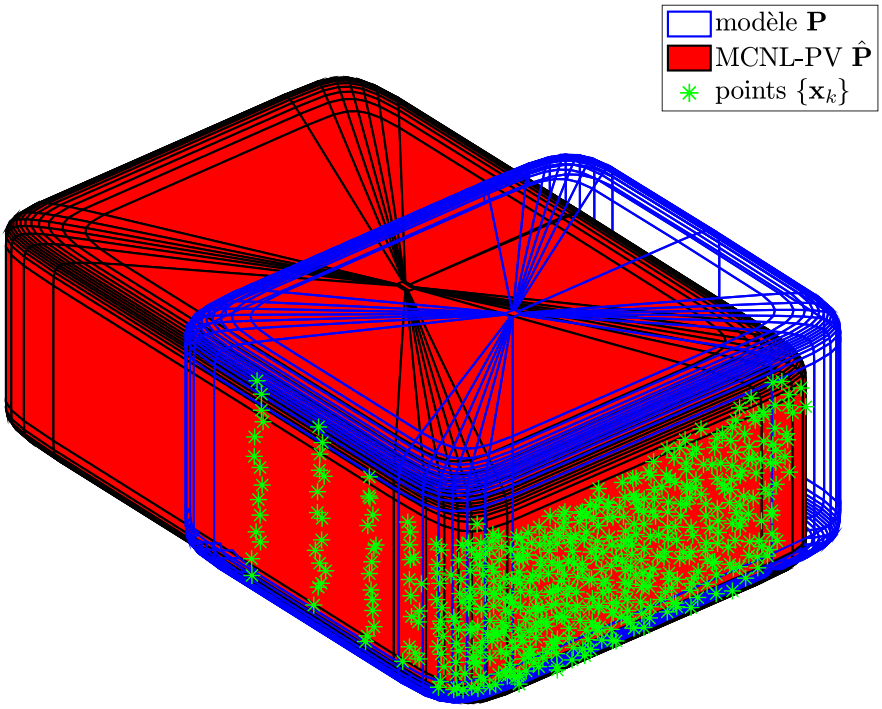


Figure 3.9 Reconstruction en rouge de la superquadrique bleue à partir des points verts.

Pour la suite nous gardons cette fonction objective plutôt que celle de [2]. Nous faisons ce choix car, en trois dimensions, le phénomène d'auto occlusion peut être encore plus marqué, et dégrade alors les reconstructions faites par cette fonction objective (voir section 3.3.3 figure 3.11 à gauche). De plus, la présence des paramètres α rend cette méthode fastidieuse à régler. Nous nous retrouvons alors de nouveau avec des reconstructions faites dans le mauvais sens (comme figure 3.6), et pour contrer ce phénomène, nous introduisons des contraintes sur les paramètres $\hat{\mathbf{P}}$ à retrouver.

3.3.3 Contraintes sur le modèle à retrouver

Les reconstructions faites dans le mauvais sens, comme figure 3.6 à gauche, concerne les figures dont un seul côté a été observé. Dans le cas 2D seule une ligne a été vue, et dans le cas 3D seul un plan. Notre robot n'a alors pas assez d'informations pour reconstruire une SQ. Ainsi, nous voulons pouvoir détecter ces cas ci pour ne pas avoir à reconstruire directement par moindres carrés une SQ sur ces points. La détection se fait par ACP sur la matrice de covariance de nos points (matrice de l'équation 3.8). En 2D comme en 3D, nous pouvons vérifier le conditionnement de cette matrice, et si celui-ci est trop grand, alors nous considérons que nos points sont distribués selon une droite en 2D ou un plan en 3D.

Cas 2D

Sur la figure 3.10 à droite, nous voyons de nouveau que la reconstruction par les moindres carrés ne se passe pas comme souhaitée. Nous procédons alors autrement. Grâce à la matrice de covariance des points $\mathbf{M}$, nous détectons que nos points sont alignés selon une droite. La décomposition de la matrice $\mathbf{M}$ en éléments propres nous donne $\mathbf{M} = \mathbf{V}\mathbf{D}\mathbf{V}^T$.

En s'inspirant de la dérivée directionnelle de [2], nous calculons le vecteur unitaire $\mathbf{u}(\{\mathbf{x}_k\}, \mathbf{x}_\mathrm{p})$ de point d'application $\mathbf{x}_\mathrm{p}$ dirigé vers le milieu $\mathbf{C}$ du segment formé par les points $\{\mathbf{x}_k\}$ (voir figure 3.10 à droite) :

$$\mathbf{u}(\{\mathbf{x}_k\}, \mathbf{x}_\mathrm{p}) = \frac{\mathbf{x}_\mathrm{p} - \mathbf{C}}{\|\mathbf{x}_\mathrm{p} - \mathbf{C}\|_2}, \mathbf{C} = \frac{1}{2}\begin{bmatrix} \min\limits_k x_k + \max\limits_k x_k \\ \min\limits_k y_k + \max\limits_k y_k \end{bmatrix}. \tag{3.28}$$

Nous choisissons alors nos bornes $(\mathbf{b}_{x_\mathrm{c}}, \mathbf{b}_{y_\mathrm{c}})$ pour les paramètres $(x_\mathrm{c}, y_\mathrm{c})$ de telle manière :

$$\begin{cases} \mathbf{v} = \mathbf{C} - t\mathbf{u}, \\ \hat{x}_\mathrm{c} \in \mathbf{b}_{x_\mathrm{c}} = [\min(\mathbf{v}_1,\ \mathbf{C}_1), \max(\mathbf{v}_1,\ \mathbf{C}_1)], \\ \hat{y}_\mathrm{c} \in \mathbf{b}_{y_\mathrm{c}} = [\min(\mathbf{v}_2,\ \mathbf{C}_2), \max(\mathbf{v}_2,\ \mathbf{C}_2)]. \end{cases} \tag{3.29}$$

Le paramètre t permet d'ajuster la taille de l'intervalle, il faut faire attention à ne pas le prendre trop petit (supérieur à $2\hat{b}$). Concernant les paramètres $(\hat{a}, \hat{b}, \hat{\psi}, \hat{\epsilon})$ nous les fixons directement et n'auront pas à être calculés dans la minimisation. Nous fixons $\hat{\epsilon} = 0.1$, car en effet si une droite a été détectée c'est que notre superellipse s'approche plus du rectangle (voir figure 3.4). Concernant le paramètre b, il est impossible de l'estimer car nous avons accès qu'à une unique dimension de notre quadrique (qui sera contenue dans a), on l'estime

alors arbitrairement, (par exemple 3 fois plus petit que $\hat{a}$). Il reste alors les deux paramètres (a, ψ). Nous utilisons maintenant la matrice de passage $\mathbf{V}$. Cette matrice représente en fait une matrice de rotation (d'après le théorème spectrale [45], $\mathbf{M}$ étant symétrique, $\mathbf{V}$ est orthogonale). Cette matrice va nous servir pour se placer dans le repère de la droite, et $\hat{\psi}$ est calculé de la sorte :

$$\hat{\psi} = \text{atan2}\left(\mathbf{V}_{2,1}, \mathbf{V}_{1,1}\right). \tag{3.30}$$

Ensuite, nous nous plaçons dans le référentiel de la superellipse :

$$\mathbf{x}^{\mathsf{s}} = \mathbf{V}^T \mathbf{x}. \tag{3.31}$$

Enfin nous prenons :

$$\hat{a} = \frac{\max\left\{x_k^{\mathsf{s}}\right\} - \min\left\{x_k^{\mathsf{s}}\right\}}{2}. \tag{3.32}$$

Nous pouvons à présent déterminer les paramètres $(x_{\mathrm{c}}, y_{\mathrm{c}})$ par la minimisation suivante :

$$(\hat{x}_{\mathrm{c}}, \hat{y}_{\mathrm{c}}) = \underset{(x_{\mathrm{c}}, y_{\mathrm{c}})}{\text{argmin}} \sum_{k=1}^{K} (\sqrt{ab}(F(x_k, y_k; \check{\mathbf{P}})^{\epsilon} - 1))^2, \;\; \check{\mathbf{P}} = \left[x_{\mathrm{c}}, y_{\mathrm{c}}, \hat{a}, \hat{b}, \hat{\psi}, \hat{\epsilon}\right]. \tag{3.33}$$

Sur la figure 3.10 à droite nous observons le résultat de cette minimisation. Grâce aux contraintes ($(x_{\mathrm{c}}, y_{\mathrm{c}})$ doit être contenu dans le rectangle vert), nous forçons la reconstruction à être orientée correctement.

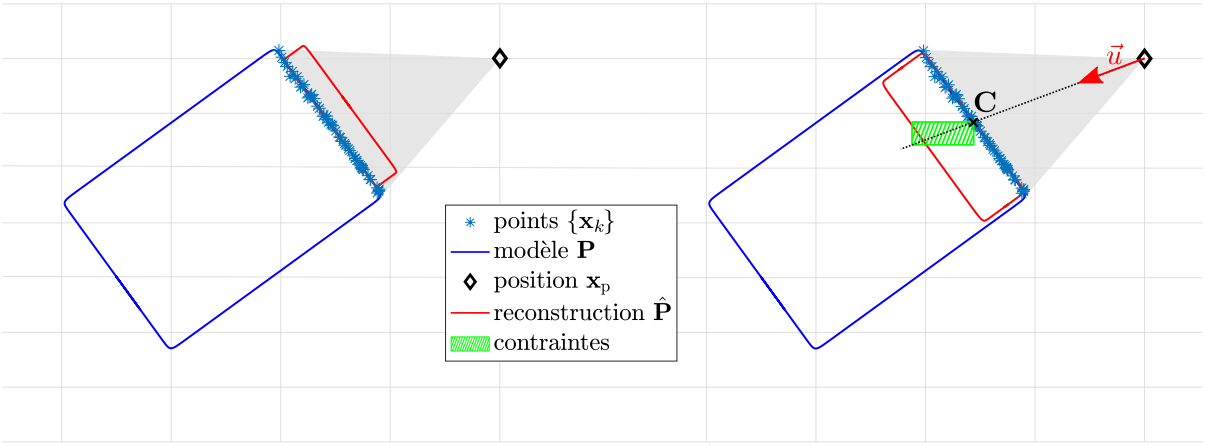


Figure 3.10 Reconstruction d'une superellipse à partir de points situés le long d'une droite. À gauche sans contrainte sur $(x_{\mathrm{c}}, y_{\mathrm{c}})$, à droite avec.

Cas 3D

Nous testons figure 3.11 à gauche la reconstruction d'une quadrique par MCNL-FV. Nous observons que l'estimé rouge n'est pas fidèle au modèle initial bleu, et ne passe pas bien par les points en vert. Nous étendons alors la méthode en deux dimensions vue au paragraphe précédent. La détection du plan se fait encore une fois avec la matrice de covariance $\mathbf{M}$:

$$\mathbf{M} = \begin{bmatrix} \widetilde{\mathbb{V}}(\{x_k\}) & \widetilde{\mathbb{V}}(\{x_k\},\{y_k\}) & \widetilde{\mathbb{V}}(\{x_k\},\{z_k\}) \\ \widetilde{\mathbb{V}}(\{y_k\},\{x_k\}) & \widetilde{\mathbb{V}}(\{y_k\}) & \widetilde{\mathbb{V}}(\{y_k\},\{z_k\}) \\ \widetilde{\mathbb{V}}(\{z_k\},\{x_k\}) & \widetilde{\mathbb{V}}(\{z_k\},\{y_k\}) & \widetilde{\mathbb{V}}(\{z_k\}) \end{bmatrix}. \tag{3.34}$$

Un conditionnement trop grand de cette matrice indique que les points de coordonnées $\{[x_k \;\; y_k \;\; z_k]\}$ sont distribués sur un même plan. Nous procédons alors aux étapes de reconstruction suivantes. D'abord, il faut décomposer $\mathbf{M}$ en éléments propres, $\mathbf{M} = \mathbf{VDV}^T$. Nous construisons ensuite le vecteur $\mathbf{u}$ comme équation (3.28), et définissons les bornes pour $(\hat{x}_c, \hat{y}_c, \hat{z}_c)$ comme équation (3.29), en rajoutant simplement la dimension en z. Il reste maintenant à fixer tous les autres paramètres. Premièrement, nous posons $\hat{\epsilon}_1 = \hat{\epsilon}_2 = 0.1$, en supposant être face à un pavé droit (voir figure 3.7). Ensuite, nous trouvons l'orientation du plan (ou de la SQ) grâce à la matrice orthogonale $\mathbf{V}$:

$$\begin{cases} \hat{\phi} = \text{atan2}(\mathbf{V}_{3,2}, \mathbf{V}_{3,3}), \\ \hat{\theta} = -\text{asin}(\mathbf{V}_{3,1}), \\ \hat{\psi} = \text{atan2}(\mathbf{V}_{2,1}, \mathbf{V}_{1,1}). \end{cases} \tag{3.35}$$

Pour la taille (a, b, c), nous nous plaçons d'abord dans le référentiel de la superquadrique, $\mathbf{x}^{\mathsf{s}} = \mathbf{V}^T\mathbf{x}$, et nous prenons :

$$\begin{aligned} \hat{a} &= \frac{\max\{x_k^{\mathsf{s}}\} - \min\{x_k^{\mathsf{s}}\}}{2}, \\ \hat{b} &= \frac{\max\{y_k^{\mathsf{s}}\} - \min\{y_k^{\mathsf{s}}\}}{2}. \end{aligned} \tag{3.36}$$

Concernant la profondeur de la SQ représentée par c, nous n'avons pas les informations nécessaires pour la déterminer. En effet, nos points sont distribués ici selon un plan, et nous fixons arbitrairement la profondeur $\hat{c} = \min(\hat{a}, \hat{b})/3$ par exemple. Enfin, nous déterminons les

paramètres (x_c, y_c, z_c) par la minimisation suivante :

$$(\hat{x}_c, \hat{y}_c, \hat{z}_c) = \underset{(x_c,y_c,z_c)}{\operatorname{argmin}} \sum_{k=1}^{K} (\sqrt{abc}(F(x_k, y_k, z_k; \check{\mathbf{P}})^{\epsilon_1} - 1))^2, \ \check{\mathbf{P}} = \left[x_c, y_c, z_c, \hat{a}, \hat{b}, \hat{c}, \hat{\phi}, \hat{\theta}, \hat{\psi}, \hat{\epsilon}_1, \hat{\epsilon}_2\right]. \tag{3.37}$$

Sur la figure 3.11 à droite nous observons le résultat de cette minimisation. $\hat{\mathbf{P}}$ est orientée correctement, et s'aligne bien aux points verts sans être trop grand.

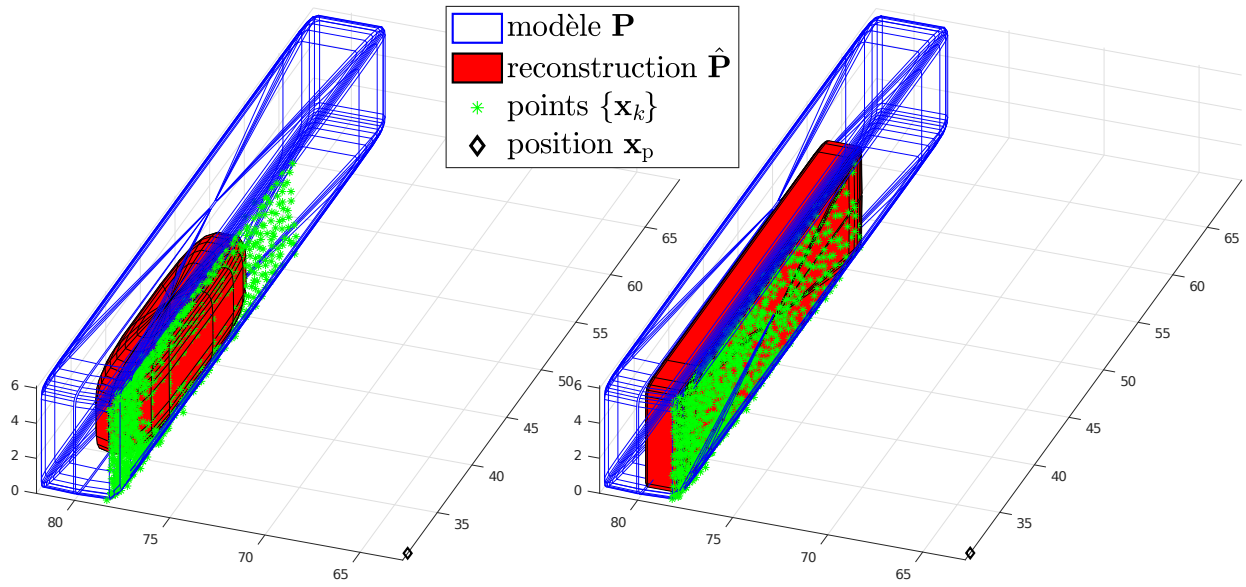


Figure 3.11 Reconstruction d'une superquadrique à partir de points situés le long d'un plan. À gauche avec la méthode MCNL-FV, et à droite avec la méthode MCNL-PV avec contraintes sur (x_c, y_c, z_c).

3.4 Conclusion

Plusieurs méthodes ont été abordées dans ce chapitre pour reconstruire une superquadrique à partir d'un nuage de points. Nous retenons deux méthodes. Pour savoir laquelle utiliser, nous utilisons un critère sur le conditionnement de la matrice équation (3.34). Si le conditionnement n'est pas trop grand (inférieur à un seuil pris ici à 100, à régler dépendant du bruit des points), alors nous entamons le processus de reconstruction sur les points en utilisant la fonction de minimisation F_S vue équation (3.27). Au contraire, si le conditionnement est grand, nous considérons que nos points sont distribués selon un plan (cas de forte auto occlusion). La reconstruction de la SQ se fait donc plus artificiellement, en construisant d'abord un plan, puis en bornant son centre, comme décrit section 3.3.3. Ceci nous évite de nous retrouver avec des reconstructions dans le mauvais sens.

CHAPITRE 4 RECONSTRUCTION D'UNE SUPERQUADRIQUE PAR FILTRAGE DE KALMAN

Dans ce chapitre nous traitons d'une technique de filtrage de superquadriques. Comme précédemment, nous commençons avec les cas 2D plus simples en traitant d'abord le cas elliptique, puis le cas des superellipses. Nous finissons avec le filtrage de SQs.

4.1 Présentation du problème

Nous avons un robot muni d'un LRF se déplaçant autour d'un obstacle. Il s'agit de reconstruire une quadrique ressemblant au mieux à l'obstacle, à l'aide de nuages de points obtenus par le laser. Au début, seule une partie de la quadrique est visible, puis au fur et à mesure que le robot se déplace autour d'elle, nous pouvons voir l'obstacle sous tous ses angles. Le but est de retrouver les paramètres de cet obstacle, sans devoir utiliser la méthode des MC à chaque nouvelle acquisition de points. En effet, cette méthode est coûteuse en temps de calcul. Par ailleurs, le but de notre algorithme et de notre représentation de carte est d'enregistrer seulement des paramètres de SQs, et ainsi d'une étape à l'autre nous ne mémorisons pas les mesures faites par le LRF. Notre algorithme doit pouvoir mettre à jour son estimé avec les mesures courantes et l'estimé précédent, sans utiliser les mesures faites au préalable. Pour cette raison, nous avons choisi d'utiliser le filtrage de Kalman pour mettre à jour les paramètres de nos quadriques en intégrant les mesures courantes du LRF.

Nous discrétisons notre espace temporel. Soit T le nombre d'étapes que met notre robot à parcourir son trajet, et soit $t \in [\![0, T]\!]$ une étape quelconque. À l'étape t, notre robot se trouve à la pose $\mathbf{x}_{\mathrm{p},t} = [x_{\mathrm{p},t}, y_{\mathrm{p},t}, z_{\mathrm{p},t}, \psi_{\mathrm{p},t}]$, avec $\psi_{\mathrm{p},t}$ son orientation selon l'axe vertical z. Au cour de cette étape, le laser effectue des mesures $\tilde{\mathbf{Z}}_t$. $\tilde{\mathbf{Z}}_t$ contient K_t mesures $\{\tilde{\mathbf{z}}_k\}_{k\in[\![1,K_t]\!]}$ correspondant aux K_t détections du laser. Les détections sont dues uniquement à l'obstacle $\mathbf{P}$ présent dans l'environnement. Nous considérons que toutes les mesures faites durant la même étape sont effectuées depuis la même position (la vitesse du robot est négligeable devant celle du capteur). La figure 4.1 illustre l'acquisition de mesures par un LRF monté sur un robot au cour de trois étapes. Nous devons retrouver les paramètres $\mathbf{P}$ de l'obstacle en bleu.

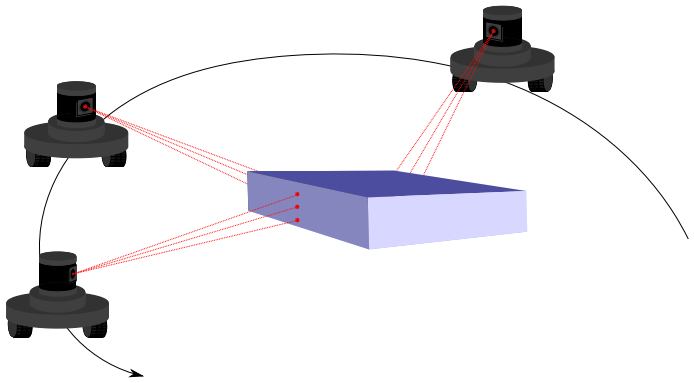


Figure 4.1 Acquisition de mesures par un laser monté sur un robot au cours de trois étapes.

4.2 Filtrage d'une ellipse avec l'EKF

Nous commençons avec le cas 2D de l'ellipse. Nous rappelons l'équation de l'ellipse, de paramètres $\mathbf{P} = [x_c, y_c, a, b, \psi]$, où (x_c, y_c) représente son centre, (a, b) ses demis grands axes et ψ son orientation dans le référentiel de navigation. Si (x, y) est un point compris sur l'ellipse, alors :

$$F(x, y; \mathbf{P}) = \left(\frac{(x - x_c)\cos\psi + (y - y_c)\sin\psi}{a} \right)^2 + \left(\frac{(x - x_c)\sin\psi - (y - y_c)\cos\psi}{b} \right)^2 = 1. \tag{4.1}$$

Pour le filtrage nous avons besoin des coordonnées des points détectés par le LRF dans le référentiel de navigation. Ces coordonnées se calculent connaissant la position du capteur $\mathbf{x}_p$ (renvoyée par la simulation) ainsi que les mesures d'angle et de distance $\{\tilde{\mathbf{z}}_k\}$ que celui-ci a effectué. En deux dimensions, le laser mesure des couples distance angle : $\tilde{\mathbf{z}}_k = (\tilde{d}_k, \tilde{\lambda}_k)$. On donne la position des points détectés par le LRF $\{\tilde{\mathbf{x}}_k\}$:

$$\tilde{\mathbf{x}}_k = \begin{bmatrix} x_p \\ y_p \end{bmatrix} + \tilde{d}_k \begin{bmatrix} \cos(\tilde{\lambda}_k + \psi_p) \\ \sin(\tilde{\lambda}_k + \psi_p) \end{bmatrix}, \tag{4.2}$$

et la figure 4.2 illustre le calcul des coordonnées des points $\tilde{\mathbf{x}}_k$ dans le référentiel global.

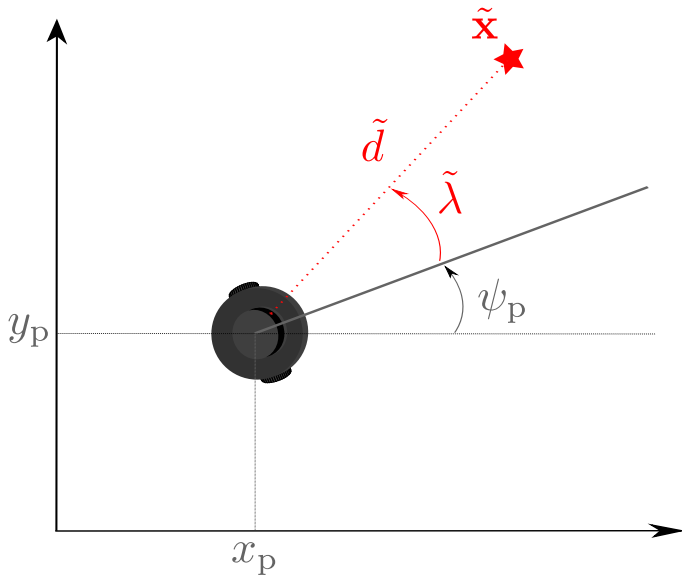


Figure 4.2 Transformation des mesures du capteur en coordonnées dans le référentiel de navigation.

4.2.1 Équations de Kalman

L'idée est d'utiliser un Filtre de Kalman Étendu (EKF) [48], où nous avons l'état à estimer $\mathbf{P} = [x_{\mathrm{c}}\ y_{\mathrm{c}}\ a\ b\ \psi]^T$:

$$\begin{cases} \mathbf{P}_{t+1} = \mathbf{P}_t + \mathbf{W}_t, \\ \mathbf{Y}_t = \mathbf{g}_t(\mathbf{P}_t) + \mathbf{V}_t. \end{cases} \tag{4.3}$$

$\mathbf{W}_t$ et $\mathbf{V}_t$ sont des bruits blancs gaussiens non corrélés, de matrices de covariance respectives $\mathbf{Q}$ et $\mathbf{R}$. L'estimation des paramètres de l'ellipse $\mathbf{P}$ suit une dynamique constante. C'est l'étape de mise à jour qui est importante. Celle-ci utilise les mesures externes du LRF. $\mathbf{Y}_t$ concatène les positions des points détectés par le capteur à l'étape t dans le référentiel de navigation :

$$\mathbf{Y}_t = [\tilde{x}_{t,1}\ \tilde{x}_{t,2}\ \ldots\ \tilde{x}_{t,K_t}\ \tilde{y}_{t,1}\ \tilde{y}_{t,2}\ \ldots\ \tilde{y}_{t,K_t}]^T. \tag{4.4}$$

Il faut maintenant trouver la fonction $\mathbf{g}_t(\hat{\mathbf{P}}_t)$, qui à partir des paramètres courants estimés $\hat{\mathbf{P}}_t$ de l'ellipse renvoie les positions estimées $\hat{\mathbf{Y}}_t^-$ des points détectés par le laser.

4.2.2 Calcul de la fonction g

Trouver $\mathbf{g}_t(\hat{\mathbf{P}}_t^-)$ revient à calculer $\hat{\mathbf{Y}}_t^-$ qui est donc la position estimée des points avant d'avoir pris en compte les mesures du LRF. Pour la mesure k du laser nous avons une mesure de

distance $\tilde{d}_k$ et un cap $\tilde{\lambda}_k$, nous pouvons ainsi remonter au point mesuré $\tilde{\mathbf{x}}_k$. Pour avoir $\hat{\mathbf{Y}}_t^-$, nous avons besoin de calculer une estimation de ces points $\{\hat{\mathbf{x}}_k^-\}_k$, en accord avec notre état estimé $\hat{\mathbf{P}}_t^-$. Puis, pour calculer $\hat{\mathbf{x}}_k^-$, nous prenons le point d'intersection entre la droite partant du robot de cap $(\tilde{\lambda}_k + \psi_\mathrm{p})$ et l'ellipse formée par les paramètres courants $\hat{\mathbf{P}}_t^-$. La figure 4.3 illustre comment nous voulons obtenir $\hat{\mathbf{x}}_k^-$.

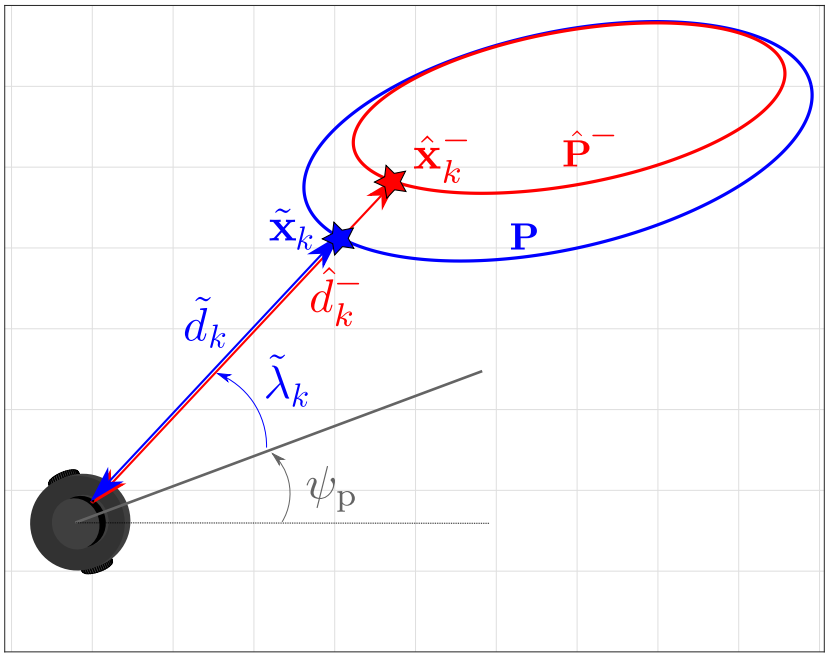


Figure 4.3 Illustration du calcul de $\hat{\mathbf{x}}_k^-$ en fonction de la pose robot $\mathbf{x}_\mathrm{p}$, de la mesure $\tilde{\lambda}_k$, et de l'estimation courante de l'ellipse $\hat{\mathbf{P}}^-$.

Mathématiquement, il s'agit de trouver le couple $(\hat{x}_k^-,\ \hat{y}_k^-)$ tel que :

$$\begin{cases} \hat{y}_k^- = m_k \hat{x}_k^- + p_k, \\ F(\hat{x}_k^-, \hat{y}_k^-; \hat{\mathbf{P}}_t^-) = 1. \end{cases} \tag{4.5}$$

Les coefficients m_k et p_k caractérisent la droite partant du robot et de cap $(\tilde{\lambda}_k + \psi_\mathrm{p})$ dans le référentiel de navigation.

$$\begin{cases} m_k = \dfrac{\sin(\tilde{\lambda}_k + \psi_\mathrm{p})}{\cos(\tilde{\lambda}_k + \psi_\mathrm{p})}, \\ p_k = \hat{y}_k^- - m_k \hat{x}_k^-. \end{cases} \tag{4.6}$$

En se plaçant dans le référentiel de l'ellipse, nous posons :

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos\hat{\psi}_t^- & \sin\hat{\psi}_t^- \\ -\sin\hat{\psi}_t^- & \cos\hat{\psi}_t^- \end{bmatrix} \begin{bmatrix} \hat{x}_k^- - \hat{x}_{\mathrm{c},t}^- \\ \hat{y}_k^- - \hat{y}_{\mathrm{c},t}^- \end{bmatrix}. \tag{4.7}$$

Et il faut résoudre pour (x, y) :

$$\begin{cases} x\sin\hat{\psi}_t^- + y\cos\hat{\psi}_t^- + \hat{y}_{\mathrm{c},t}^- = m_k(x\cos\hat{\psi}_t^- - y\sin\hat{\psi}_t^- + \hat{x}_{\mathrm{c},t}^-) + p_k, \\ \left(\dfrac{x}{\hat{a}_t^-}\right)^2 + \left(\dfrac{y}{\hat{b}_t^-}\right)^2 = 1, \end{cases} \tag{4.8}$$

$$\begin{cases} y = M_k x + P_k, \\ \left(\dfrac{x}{\hat{a}_t^-}\right)^2 + \left(\dfrac{y}{\hat{b}_t^-}\right)^2 = 1, \end{cases} \tag{4.9}$$

avec

$$\begin{cases} M_k = \dfrac{m_k\cos\hat{\psi}_t^- - \sin\hat{\psi}_t^-}{m_k\sin\hat{\psi}_t^- + \cos\hat{\psi}_t^-}, \\ P_k = \dfrac{m_k\hat{x}_{\mathrm{c},t}^- - \hat{y}_{\mathrm{c},t}^- + p_k}{m_k\sin\hat{\psi}_t^- + \cos\hat{\psi}_t^-}. \end{cases} \tag{4.10}$$

En substituant y on obtient une équation d'ordre deux en x, de discriminant

$$\Delta_k = 4\left(\frac{1}{\hat{a}_t^{-2}} + \frac{M_k^2}{\hat{b}_t^{-2}} - \frac{P_k^2}{\hat{a}_t^{-2}\hat{b}_t^{-2}}\right). \tag{4.11}$$

Si le discriminant est positif nous avons deux solutions pour x.

$$x = \frac{-\dfrac{M_k P_k}{\hat{b}_t^{-2}} \pm \sqrt{\Delta_k}}{\dfrac{1}{\hat{a}_t^{-2}} + \dfrac{M_k^2}{\hat{b}_t^{-2}}}. \tag{4.12}$$

On retrouve y avec $y = M_k x + P_k$, puis on repasse aux coordonnées dans le référentiel de navigation :

$$\begin{bmatrix} \hat{x}_k^- \\ \hat{y}_k^- \end{bmatrix} = \begin{bmatrix} x\cos\hat{\psi}_t^- - y\sin\hat{\psi}_t^- + \hat{x}_{\mathrm{c},t}^- \\ x\sin\hat{\psi}_t^- + y\cos\hat{\psi}_t^- + \hat{y}_{\mathrm{c},t}^- \end{bmatrix}. \tag{4.13}$$

Finalement on retrouve $\mathbf{g}_t(\hat{\mathbf{P}}_t^-) = \hat{\mathbf{Y}}_t^-$. Quand il n'y a pas de solution (discriminant négatif), alors on ne considère tout simplement pas les mesures dans la suite, et s'il y a deux solutions, on considère celle qui est la plus proche du robot (en norme euclidienne).

4.2.3 Filtrage de Kalman Étendu

Afin de réaliser le filtrage, il faut linéariser la fonction $\mathbf{g}_t$. Ceci est fastidieux, mais l'on peut trouver $\dfrac{\partial \mathbf{g}_t}{\partial \hat{\mathbf{P}}_t^-} = \mathbf{C}_t$ en dérivant les expressions des $\hat{x}_k^-$ et $\hat{y}_k^-$ selon chaque paramètre de l'ellipse $\hat{\mathbf{P}}_t^-$:

$$\frac{\partial \mathbf{g}_t}{\partial \hat{\mathbf{P}}_t^-} = \mathbf{C}_t = \begin{bmatrix} \dfrac{\partial \hat{x}_{t,1}^-}{\partial \hat{x}_{\mathrm{c},t}^-} & \dfrac{\partial \hat{x}_{t,1}^-}{\partial \hat{y}_{\mathrm{c},t}^-} & \dfrac{\partial \hat{x}_{t,1}^-}{\partial \hat{a}_t^-} & \dfrac{\partial \hat{x}_{t,1}^-}{\partial \hat{b}_t^-} & \dfrac{\partial \hat{x}_{t,1}^-}{\partial \hat{\psi}_t^-} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \dfrac{\partial \hat{x}_k^-}{\partial \hat{x}_{\mathrm{c},t}^-} & \dfrac{\partial \hat{x}_k^-}{\partial \hat{y}_{\mathrm{c},t}^-} & \dfrac{\partial \hat{x}_k^-}{\partial \hat{a}_t^-} & \dfrac{\partial \hat{x}_k^-}{\partial \hat{b}_t^-} & \dfrac{\partial \hat{x}_k^-}{\partial \hat{\psi}_t^-} \\ \dfrac{\partial \hat{y}_{t,1}^-}{\partial \hat{x}_{\mathrm{c},t}^-} & \dfrac{\partial \hat{y}_{t,1}^-}{\partial \hat{y}_{\mathrm{c},t}^-} & \dfrac{\partial \hat{y}_{t,1}^-}{\partial \hat{a}_t^-} & \dfrac{\partial \hat{y}_{t,1}^-}{\partial \hat{b}_t^-} & \dfrac{\partial \hat{y}_{t,1}^-}{\partial \hat{\psi}_t^-} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \dfrac{\partial \hat{y}_k^-}{\partial \hat{x}_{\mathrm{c},t}^-} & \dfrac{\partial \hat{y}_k^-}{\partial \hat{y}_{\mathrm{c},t}^-} & \dfrac{\partial \hat{y}_k^-}{\partial \hat{a}_t^-} & \dfrac{\partial \hat{y}_k^-}{\partial \hat{b}_t^-} & \dfrac{\partial \hat{y}_k^-}{\partial \hat{\psi}_t^-} \end{bmatrix}. \tag{4.14}$$

Quant à la propagation, $\mathbf{P}_{t+1} = \mathbf{f}_t(\mathbf{P}_t) + \mathbf{W}_t$, et $\mathbf{f}_t(\mathbf{P}_t) = \mathbf{P}_t$, donc $\dfrac{\partial \mathbf{f}_t}{\partial \mathbf{P}_t} = \mathbf{I}_5$. Les équations du filtrage sont données ci-dessous.

Étape de propagation :

- $\hat{\mathbf{P}}_t^- = \hat{\mathbf{P}}_{t-1}^+$,
- $\mathbf{\Sigma}_t^- = \mathbf{\Sigma}_{t-1}^+ + \mathbf{Q}$.

Incorporation des mesures du laser :

- $\hat{\mathbf{Y}}_t^- = \mathbf{g}_t(\hat{\mathbf{P}}_t^-)$
- $\mathbf{K}_t = \mathbf{\Sigma}_t^- \mathbf{C}_t^T (\mathbf{C}_t \mathbf{\Sigma}_t^- \mathbf{C}_t^T + \mathbf{R}_t)^{-1}$
- $\hat{\mathbf{P}}_t^+ = \hat{\mathbf{P}}_t^- + \mathbf{K}_t(\mathbf{Y}_t - \hat{\mathbf{Y}}_t^-)$
- $\mathbf{\Sigma}_t^+ = (\mathbf{I}_5 - \mathbf{K}_t \mathbf{C}_t)\mathbf{\Sigma}_t^-$

Il est difficile de caractériser $\mathbf{Q}$, il faut le régler de sorte à obtenir le meilleur filtrage possible. Celui-ci est choisi petit puisque l'ellipse à filtrer est la même à chaque étape, et on peut donc bien faire confiance au modèle. Quant à $\mathbf{R}_t$, il reflète la précision du capteur, qui est de l'ordre de $\sigma = 0.1$ mètre. On considère les mesures entre deux détections indépendantes entres elles, et par simplicité les composantes x et y indépendantes aussi pour un même point, on a donc choisi $\mathbf{R}_t = 10^{-2}\mathbf{I}_{2K_t}$.

4.2.4 Résultats

Nous testons notre filtrage en faisant tourner le robot avec le laser autour d'un obstacle en forme d'ellipse. À la position initiale du robot, nous effectuons un premier scan. À l'étape d'initialisation pour l'estimé $\mathbf{P}_0^+$ nous utilisons les techniques vues au chapitre 3.

La figure 4.4 montre l'état du filtrage pour quatre étapes. Nous avons volontairement pris un cas des plus défavorables, avec une ellipse fortement allongée. Initialement (image en haut à gauche), notre robot est placé en bas à gauche par rapport à l'obstacle, et le laser détecte mal l'ellipse étant données la position du robot et la forte excentricité de l'ellipse. Ainsi, l'estimé initial $\hat{\mathbf{P}}_0^+$ obtenu par les moindres carrés n'est pas très bon. Le robot se déplace (image en haut à droite) jusqu'à aller en haut à droite de l'obstacle. L'estimé obtenu après cinquante étapes de filtrage apparaît en rouge. Même chose en bas à gauche après un tour complet du robot (étape $t = 100$), puis en bas à droite après deux tours complets (étape $t = 200$). Les estimés en rouge deviennent très proches de l'obstacle en bleu. Dans cet exemple nous avons pris $\mathbf{Q} = \mathbf{\Sigma}_0 = 10^{-3}\mathbf{I}_5$. Il semble préférable de prendre un $\mathbf{Q}$ plus faible pour des ellipses moins excentrées.

4.2.5 Conclusion

L'algorithme fonctionne relativement bien, mais reste sensible aux réglages des matrices $\mathbf{Q}$ et $\mathbf{R}$. Aussi cet algorithme parait compliqué à appliquer sur les superquadriques, (fonction $\mathbf{g}$ impossible à trouver analytiquement, et à dériver).

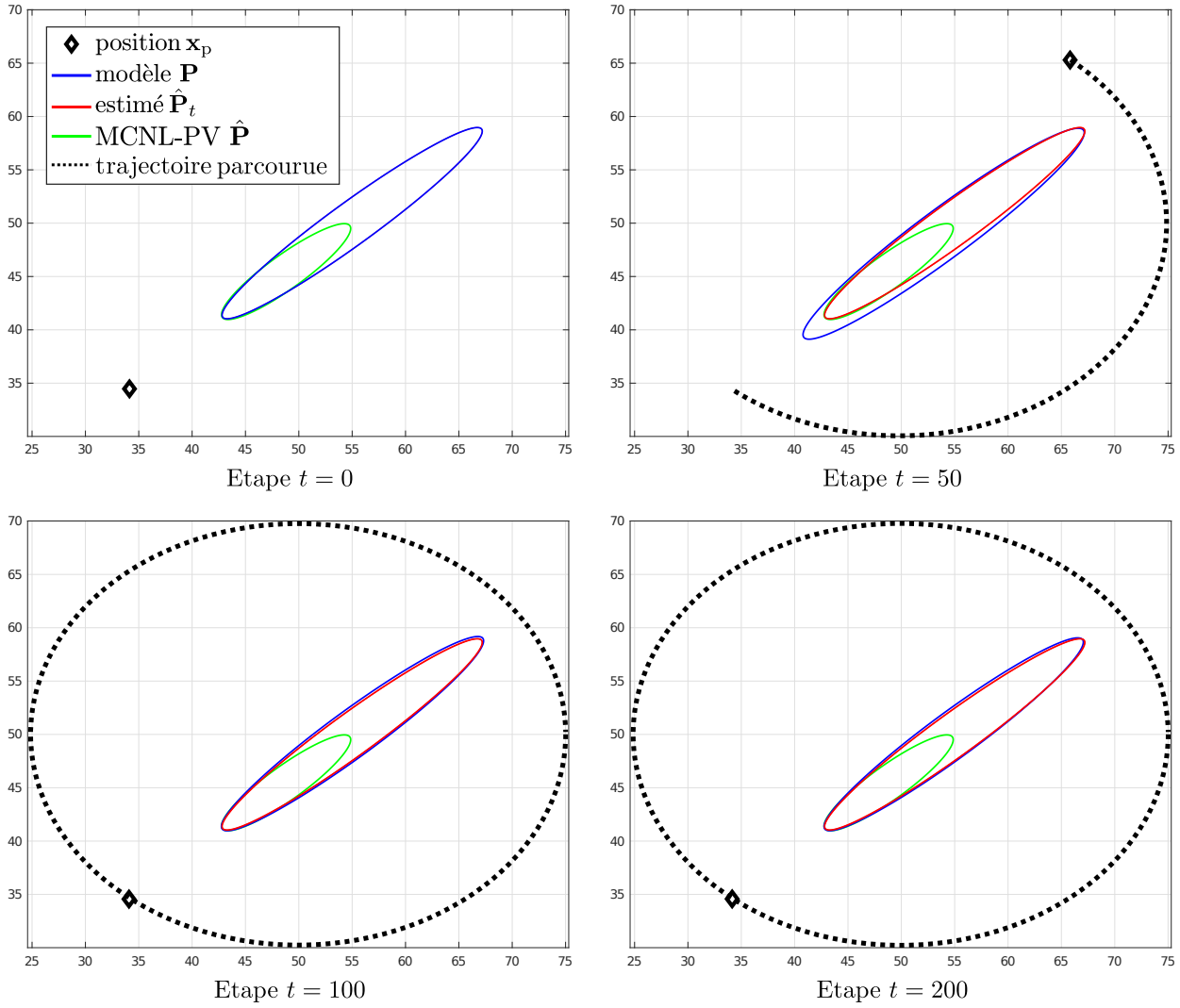


Figure 4.4 Filtrage d'une ellipse avec l'EKF.

4.3 Filtrage d'une superellipse avec l'UKF

Le filtre de Kalman Unscented (UKF) propose une alternative à l'EKF qui permet d'éviter le calcul de la fonction $\mathbf{g}$ analytiquement, et donc d'avoir à la dériver. Nous allons pouvoir utiliser par la suite ce type de filtrage pour les superquadriques. L'UKF repose sur la transformée *Unscented* [49]. Celle-ci s'appuie sur le principe qu'une distribution probabiliste est plus facile à approximer qu'une fonction non linéaire. Il s'agit ici de choisir un échantillon $\mathcal{P}^{(i)}$ de paramètres $\hat{\mathbf{P}}_t$ précisément défini, dépendant notamment de $\mathbf{\Sigma}_t$. L'échantillon est donc choisi de manière déterministe. À partir de cet échantillon, nous voulons extrapoler la distribution de $\hat{\mathbf{Y}}_t$ connaissant celle de $\hat{\mathbf{P}}_t$ (supposée gaussienne) et la relation $\hat{\mathbf{Y}}_t = \mathbf{g}_t(\hat{\mathbf{P}}_t)$.

4.3.1 Application de l'UKF sur les ellipses

Nous avons les mêmes équations que précédemment.

$$\begin{cases} \mathbf{P}_{t+1} = \mathbf{P}_t + \mathbf{W}_t, \\ \mathbf{Y}_t = \mathbf{g}_t(\mathbf{P}_t) + \mathbf{V}_t, \end{cases} \tag{4.15}$$

avec $\mathbf{P}$ les $N = 5$ paramètres de l'ellipse. Pour les ellipses, la fonction $\mathbf{g}$ est connue, mais nous ne voulons plus utiliser sa jacobienne. Nous détaillons les étapes de l'UKF [50] :

— Propager l'état à travers le modèle constant :

$$\begin{aligned} \hat{\mathbf{P}}_t^- &= \hat{\mathbf{P}}_{t-1}^+, \\ \mathbf{\Sigma}_t^- &= \mathbf{\Sigma}_{t-1}^+ + \mathbf{Q}. \end{aligned} \tag{4.16}$$

— Choisir $2N + 1$ *sigma points* $\mathcal{P}$ représentant un échantillon d'états possibles :

$$\begin{aligned} \mathcal{P}^{(0)} &= \hat{\mathbf{P}}_t^-, \\ \mathcal{P}^{(i)} &= \hat{\mathbf{P}}_t^- + \sqrt{N+\lambda}\left[\sqrt{\mathbf{\Sigma}_t^-}\right]_i, \\ \mathcal{P}^{(i+N)} &= \hat{\mathbf{P}}_t^- - \sqrt{N+\lambda}\left[\sqrt{\mathbf{\Sigma}_t^-}\right]_i, i \in [\![1, N]\!]. \end{aligned} \tag{4.17}$$

Le coefficient λ représente à quelle distance de la moyenne nous choisissons nos *sigma points*. Jusqu'à la fin nous prenons $\lambda = 1$. La notation $\left[\sqrt{\mathbf{M}}\right]_i$ désigne la colonne i de la racine d'une matrice $\mathbf{M}$, vérifiant $\sqrt{\mathbf{M}}\sqrt{\mathbf{M}}^T = \mathbf{M}$.

— Choisir les poids W correspondant à ces *sigma points* :

$$\begin{aligned} W^{(0)} &= \frac{\lambda}{N+\lambda}, \\ W^{(i)} &= \frac{1}{2(N+\lambda)}, i \in [\![1, 2N]\!]. \end{aligned} \tag{4.18}$$

— Propager les *sigma points* à travers la fonction $\mathbf{g}_t$:

$$\mathcal{Y}^{(i)} = \mathbf{g}_t\left(\mathcal{P}^{(i)}\right), i \in [\![0, 2N]\!]. \tag{4.19}$$

— Calculer $\hat{\mathbf{Y}}_t^-$:

$$\hat{\mathbf{Y}}_t^- = \sum_{i=0}^{2N} W^{(i)} \mathcal{Y}^{(i)}. \tag{4.20}$$

— Calculer les matrices de covariances $\mathbf{S}_t$ et $\mathbf{D}_t$:

$$\begin{aligned}\mathbf{S}_t &= \mathbf{R}_t + \sum_{i=0}^{2N} W^{(i)}(\mathcal{Y}^{(i)} - \hat{\mathbf{Y}}_t^-)(\mathcal{Y}^{(i)} - \hat{\mathbf{Y}}_t^-)^T, \\ \mathbf{D}_t &= \sum_{i=0}^{2N} W^{(i)}(\mathcal{P}^{(i)} - \hat{\mathbf{P}}_t^-)(\mathcal{Y}^{(i)} - \hat{\mathbf{Y}}_t^-)^T.\end{aligned} \tag{4.21}$$

— Mettre à jour l'état estimé $\hat{\mathbf{P}}_t^+$:

$$\begin{aligned}\mathbf{K}_t &= \mathbf{D}_t\mathbf{S}_t^{-1}, \\ \hat{\mathbf{P}}_t^+ &= \hat{\mathbf{P}}_t^- + \mathbf{K}_t(\mathbf{Y}_t - \hat{\mathbf{Y}}_t^-), \\ \mathbf{\Sigma}_t^+ &= \mathbf{\Sigma}_t^- - \mathbf{K}_t\mathbf{S}_t\mathbf{K}_t^T.\end{aligned} \tag{4.22}$$

4.3.2 Initialisation de l'UKF et dilution de précision

Lors des différents tests, on remarque que la convergence du filtre est très sensible à l'initialisation. Nous conservons pour $\hat{\mathbf{P}}_0^+$ ce qui est retourné par notre solveur des moindres carrés. Quant à $\mathbf{\Sigma}_0^+$, nous utilisons la Dilution de Précision (DOP). La DOP [51] est un facteur qui amplifie les erreurs de mesures dues à la géométrie de l'environnement. Elle dépend de $\hat{\mathbf{P}}_0^+$, des premières mesures du laser $\{\tilde{\mathbf{x}}_{t=0,k}\}_k$, et du bruit σ (écart-type) sur ces mesures. En considérant la fonction F de l'ellipse, (équation 4.1), nous calculons la jacobienne suivante :

$$J_F = \left[\frac{\partial F}{\partial \hat{x}_{\mathrm{c},0}^+} \frac{\partial F}{\partial \hat{y}_{\mathrm{c},0}^+} \frac{\partial F}{\partial \hat{a}_0^+} \frac{\partial F}{\partial \hat{b}_0^+} \frac{\partial F}{\partial \hat{\theta}_0^+}\right]. \tag{4.23}$$

Nous définissons la matrice $\mathbf{M}$ à partir de cette Jacobienne :

$$\mathbf{M}_{t=0} = \begin{bmatrix} J_F(x_{t=0,k=1}, y_{t=0,k=1}; \hat{\mathbf{P}}_0^+) \\ J_F(x_{t=0,k=2}, y_{t=0,k=2}; \hat{\mathbf{P}}_0^+) \\ \vdots \\ J_F(x_{0,K}, y_{0,K}; \hat{\mathbf{P}}_0^+) \end{bmatrix}, \tag{4.24}$$

ainsi que l'opérateur $\mathfrak{A}$:

$$\mathfrak{A}(\mathbf{M}) = \left(\mathbf{M}\mathbf{M}^T\right)^{-1}. \tag{4.25}$$

Finalement, nous utilisons cette formule pour initialiser l'incertitude de notre estimation $\mathbf{\Sigma}_\mathbf{0}^+$:

$$\mathbf{\Sigma}_0^+ = \sigma^2\mathfrak{A}(\mathbf{M}_0). \tag{4.26}$$

La figure 4.5 illustre le phénomène de DOP. Du point de vue du robot rouge, la grande

longueur de l'ellipse est bien observée, on aura donc une faible incertitude (DOP) sur le paramètre a. En opposition, à cause de l'auto occlusion, la petite longueur de l'ellipse n'est pas observée, ainsi nous auront une grande incertitude sur le paramètre b. Pour le robot bleu, c'est l'inverse, la petite longueur est bien observée contrairement à la grande longueur. Au final, c'est le robot vert qui observe le mieux l'ellipse et qui aura alors globalement (pour tous les paramètres) une plus faible DOP de son point de vue. Il est d'ailleurs le seul des trois robots à pouvoir estimer précisément où est le centre (x_c, y_c) de l'ellipse.

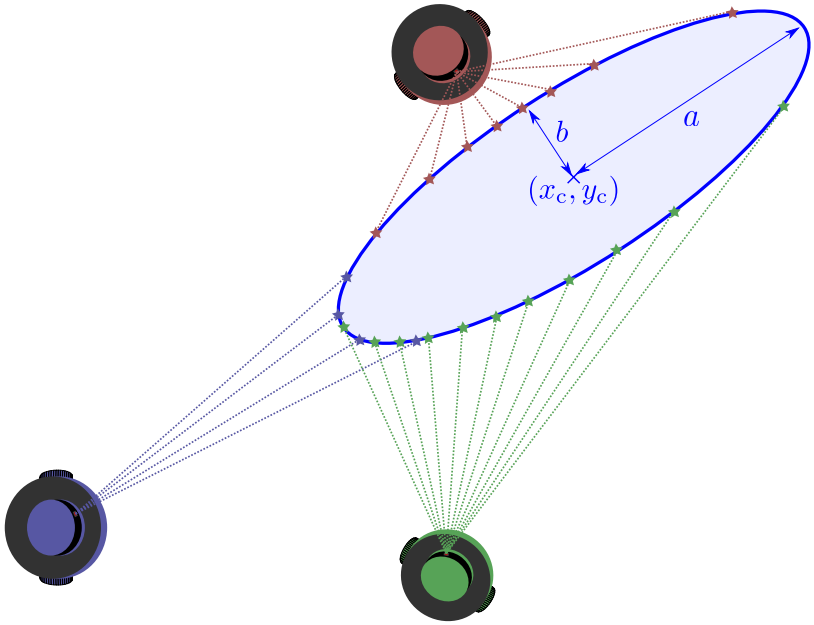


Figure 4.5 Illustration des différentes dilutions de précision selon la position du robot.

Exemple de DOP

Les figures 4.6 et 4.7 montrent chacune une superellipse observée par un LRF depuis un point fixe. Nous donnons les dilutions de précision correspondantes, fonctions des points observés $\{\mathbf{x}_k\}$, de la reconstruction faite $\hat{\mathbf{P}}$, et prenant ici pour l'exemple $\sigma = 1$. Nous comparons aussi ces valeurs avec celles calculées en prenant le vrai modèle $\mathbf{P}$. Nous souhaitons savoir ici quels paramètres de $\mathbf{P}$ ont été estimés plus ou moins facilement, et nous regardons alors les valeurs sur la diagonale.

Sur la figure 4.6, nous voyons qu'avec les points $\{\mathbf{x}_k\}$ observés, l'estimation de a est bien plus fiable que celle de b (DOP estimée 30 fois plus faible). Effectivement, les points $\{\mathbf{x}_k\}$ nous donnent aucune information quant à la profondeur de la superellipse. Aussi, nous voyons que l'estimation de x_c est plus fiable devant celle de y_c. Ceci se comprend facilement en

observant la figure, sachant aussi que l'inclinaison ψ de l'ellipse est nulle dans le référentiel de navigation. Enfin, concernant ψ, ce paramètre a très bien été estimé (DOP très faible). Notre robot ayant vu des points agencés sur une ligne, l'orientation de la superellipse est estimée précisément.

Sur la figure 4.7 nous prenons un autre exemple plus elliptique, avec une inclinaison non nulle. Globalement, nous voyons que les DOPs sont plus élevées que pour le modèle précédent, et l'inclinaison ψ ainsi que sa forme ϵ sont déterminées moins précisément. L'estimation de a est plus fiable devant celle de b. Aussi, nous voyons que les corrélations entre les paramètres x_{c} et y_{c} sont plus élevées que pour le modèle précédent, ce qui est dû à l'inclinaison ψ non nulle de l'ellipse. Nous remarquons dans les deux cas présentés ici que les DOPs sont sous estimées, les coefficients diagonaux étant plus faibles quand le calcul est réalisé avec $\hat{\mathbf{P}}$ plutôt que $\mathbf{P}$.

x_c	y_c	a	b	ψ	ϵ
0.2587	-0.0789	-0.0002	0.0781	-0.0001	-0.0033
-0.0789	8.0323	0.3231	-8.0187	-0.0002	0.3730
-0.0002	0.3231	0.2674	-0.3239	0.0000	0.0155
0.0781	-8.0187	-0.3239	8.0053	0.0002	-0.3724
-0.0001	-0.0002	0.0000	0.0002	0.0000	0.0000
-0.0033	0.3730	0.0155	-0.3724	0.0000	0.0174

x_c	y_c	a	b	ψ	ϵ
1.0665	-8.3379	-0.8769	8.3331	-0.0022	-0.1127
-8.3379	319.1700	21.0710	-318.9700	0.0101	4.3871
-0.8769	21.0710	2.2050	-21.0670	0.0013	0.2932
8.3331	-318.9700	-21.0670	318.7800	-0.0101	-4.3849
-0.0022	0.0101	0.0013	-0.0101	0.0000	0.0001
-0.1127	4.3871	0.2932	-4.3849	0.0001	0.0612

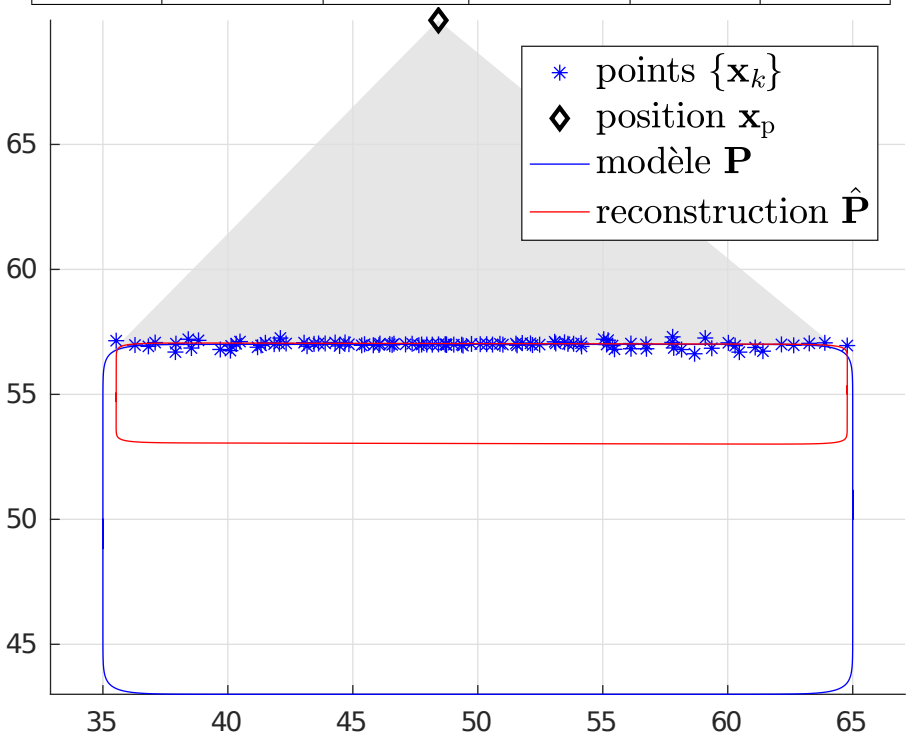


Figure 4.6 Illustration du phénomène de DOP. Premier tableau calculé avec $\hat{\mathbf{P}}$ (en rouge sur la figure), le second avec $\mathbf{P}$ (en bleu). $\sigma = 1$.

x_c	y_c	a	b	ψ	ϵ
6.4229	-4.4428	-0.0313	6.6078	-0.1240	-1.5002
-4.4428	17.4450	-4.7742	-16.8540	-0.1240	3.4874
-0.0313	-4.7742	5.0822	4.1484	0.0675	-0.3223
6.6078	-16.8540	4.1484	17.3800	0.0512	-3.5617
-0.1240	-0.1240	0.0675	0.0512	0.0081	-0.0017
-1.5002	3.4874	-0.3223	-3.5617	-0.0017	0.9713

x_c	y_c	a	b	ψ	ϵ
112.2200	-21.4870	-29.0380	71.7480	-3.6590	-11.0920
-21.4870	214.9300	-159.9400	-193.6400	-4.4140	5.8892
-29.0380	-159.9400	157.6200	123.6400	4.9666	2.5669
71.7480	-193.6400	123.6400	199.7300	2.0326	-10.0240
-3.6590	-4.4140	4.9666	2.0326	0.2523	0.2928
-11.0920	5.8892	2.5669	-10.0240	0.2928	1.7052

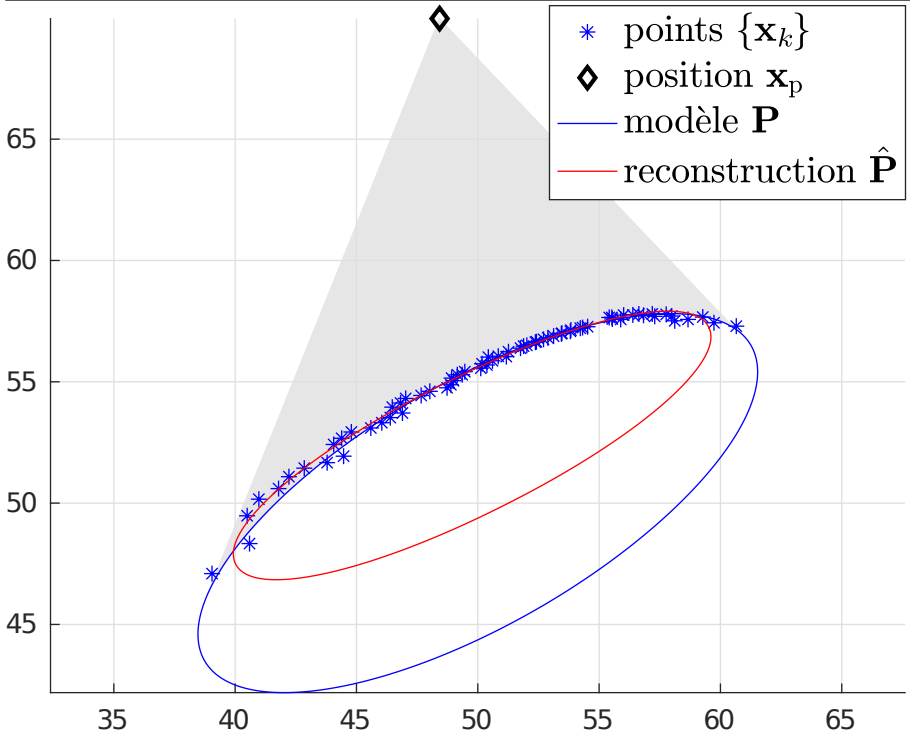


Figure 4.7 Illustration du phénomène de DOP. Premier tableau calculé avec $\hat{\mathbf{P}}$ (en rouge sur la figure), le second avec $\mathbf{P}$ (en bleu). $\sigma = 1$.

4.3.3 Résultats

Dans nos simulations, nous prenons $\mathbf{R}_t = 10^{-2}\mathbf{I}_{2K_t}$, et choisissons $\mathbf{Q} = 10^{-6}\mathbf{I}_5$.

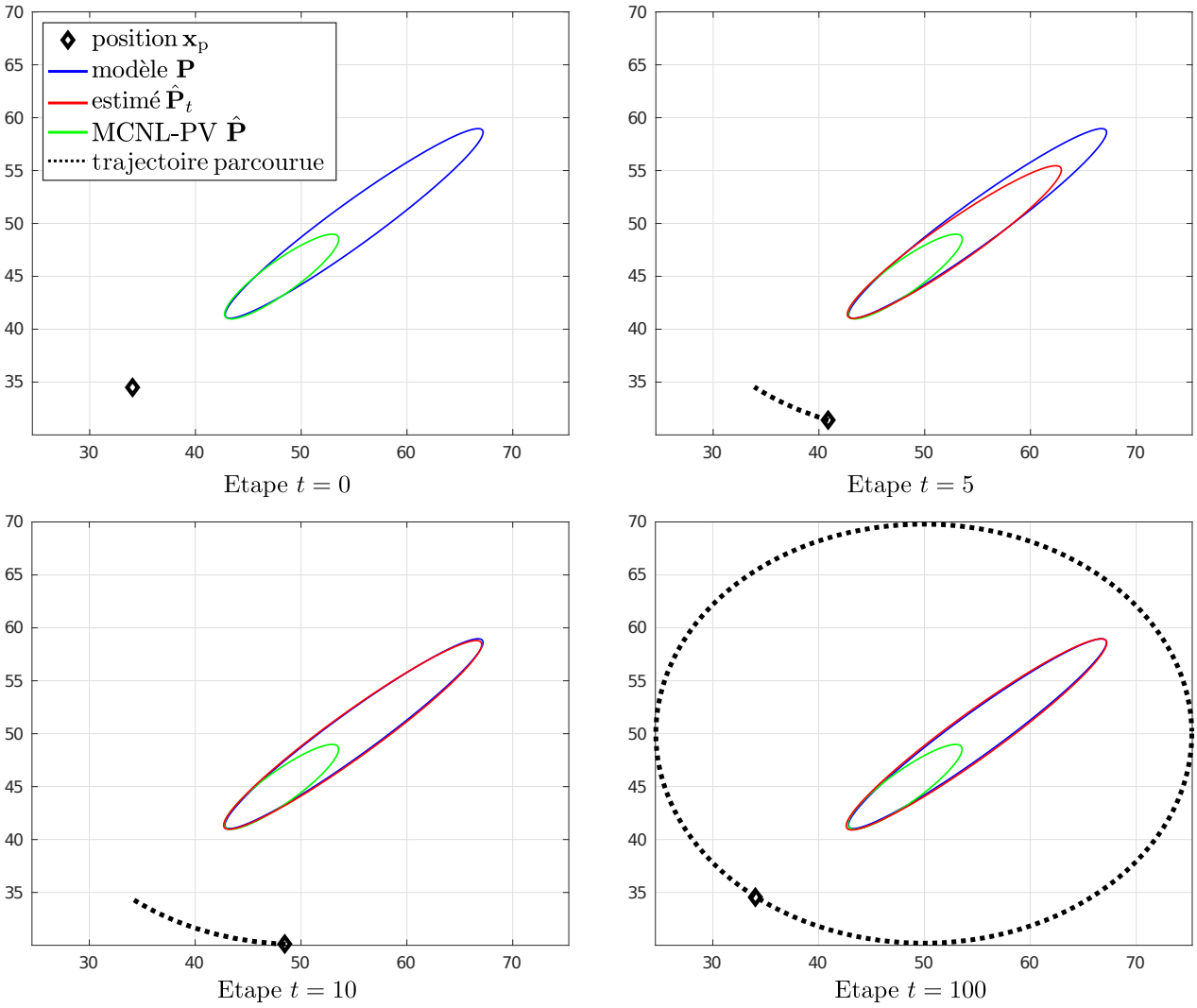


Figure 4.8 Filtrage d'une ellipse avec l'UKF.

La figure 4.8 montre l'état du filtrage pour quatre étapes. Initialement (image en haut à gauche), notre robot est placé en bas à gauche par rapport à l'obstacle, et le laser détecte mal l'ellipse étant donnée la position du robot et de la forte excentricité de l'ellipse. Ainsi, l'estimé initial (en vert) obtenu par les moindres carrés n'est donc pas très bon. Le robot se déplace un peu (image en haut à droite) et découvre un peu plus l'obstacle (étape $t = 5$). L'estimé obtenu par filtrage apparaît en rouge. En bas à gauche même chose après un déplacement un peu plus grand (étape $t = 10$), puis en bas à droite après un tour complet autour de

l'obstacle (étape $t = 100$). Les estimés en rouge deviennent très proches de l'obstacle en bleu même lors d'un faible déplacement du robot. Les résultats sont meilleurs que pour l'EKF, mais un peu plus coûteux en temps de calcul. L'EKF et l'UKF sont tous les deux des approximations plus au moins fidèles du filtre de Bayes [50]. L'EKF tente d'approximer en linéarisant les intégrales du filtre Bayésien tandis que l'UKF utilise des *sigma-points* pour faire une quadrature Gaussienne. Dans notre cas où nous avons de fortes non linéarités pour l'intégration des mesures externes (fonction $\mathbf{g}$), nous pouvions nous attendre à ce que les quadratures (UKF) fonctionnent mieux que les linéarisations (EKF).

4.3.4 Cas des superellipses

Nous allons utiliser le même algorithme pour filtrer des superellipses. L'état à estimer $\mathbf{P}$ devient de taille $N = 6$ puisque l'on rajoute le paramètre ϵ. Par contre, nous devons calculer de nouveau la fonction $\mathbf{g}_t$ nous permettant d'obtenir $\hat{\mathbf{Y}}_t^-$. Cette fois-ci, plutôt que d'effectuer l'innovation dans le filtre avec les positions des points $\{\mathbf{x}_k\}$, nous utilisons directement les mesures $\{\tilde{\mathbf{z}}_k\} = \{(\tilde{d}_k, \tilde{\lambda}_k)\}$ du LRF. Ainsi, nous avons :

$$\mathbf{Y}_t = \begin{bmatrix} \tilde{d}_1 & \tilde{\lambda}_1 \\ \vdots & \vdots \\ \tilde{d}_{\tilde{K}_t} & \tilde{\lambda}_{\tilde{K}_t} \end{bmatrix} \text{ et } \hat{\mathbf{Y}}_t^- = \begin{bmatrix} \hat{d}_1^- & \hat{\lambda}_1^- \\ \vdots & \vdots \\ \hat{d}_{\hat{K}_t}^- & \hat{\lambda}_{\hat{K}_t}^- \end{bmatrix}. \tag{4.27}$$

La comparaison de ces matrices n'est pas triviale, car elles n'ont déjà à priori pas le même nombre de lignes. Cependant, nous savons que les mesures de cap se recoupent, car le LRF prend toujours ses mesures avec les mêmes positions angulaires mesurées $\tilde{\lambda}$, mais pour certains λ, il se peut que $\mathbf{Y}_t$ aura un écho $\tilde{d}_k$ tandis que $\hat{\mathbf{Y}}_t^-$ ne présentera pas d'écho. L'inverse peut aussi se produire. Nous décidons de comparer en fait les échos d, c'est à dire les distances mesurées et estimées obtenues pour la même mesure angulaire λ. En réordonnant les distances, nous prenons alors finalement :

$$\mathbf{Y}_t = \begin{bmatrix} \tilde{d}_1 \\ \vdots \\ \tilde{d}_{K_t} \end{bmatrix} \text{ et } \hat{\mathbf{Y}}_t^- = \begin{bmatrix} \hat{d}_1^- \\ \vdots \\ \hat{d}_{K_t}^- \end{bmatrix}, \tag{4.28}$$

où cette fois-ci $\tilde{d}_k$ et $\hat{d}_k$ correspondent à la même mesure de cap $\tilde{\lambda}_k$. Étant donné que les mesures de cap $\tilde{\lambda}$ correspondent pour $\mathbf{Y}_t$ et $\hat{\mathbf{Y}}_t^-$, il est inutile de les inclure dans ces vecteurs, puisque lors de l'innovation, $\mathbf{Y}_t - \hat{\mathbf{Y}}_t^-$ fera apparaître une colonne de zéro. Cette manipulation nous permet de diviser par deux la taille du vecteur $\mathbf{Y}$, en ne conservant que des données

pouvant être comparées entre elles.

En s'appuyant sur les calculs fait pour les ellipses, il s'agit cette fois de résoudre :

$$\begin{cases} y = M_k x + P_k, \\ \left(\dfrac{x}{\hat{a}_t^-}\right)^{2/\hat{\epsilon}_t^-} + \left(\dfrac{y}{\hat{b}_t^-}\right)^{2/\hat{\epsilon}_t^-} = 1. \end{cases} \tag{4.29}$$

On utilise la fonction :

$$h(x) = \left(\frac{x}{\hat{a}_t^-}\right)^{2/\hat{\epsilon}_t^-} + \left(\frac{M_k x + P_k}{\hat{b}_t^-}\right)^{2/\hat{\epsilon}_t^-} - 1. \tag{4.30}$$

Nous cherchons les solutions satisfaisant $h(x) = 0$, ce qui ne se résout pas analytiquement. Les variations de $h(x)$ sont semblables à celle d'une parabole, (cas où $\hat{\epsilon}_t^- = 1$), qui est convexe et diverge en l'infini en plus et moins l'infini. Afin d'optimiser les temps de calcul, nous pouvons nous assurer qu'il y ait une solution avant de résoudre l'équation $h(x) = 0$. Ceci peut se faire efficacement en vérifiant qu'au point x^* tel que $\dfrac{\partial h(x^*)}{\partial x} = 0$ nous ayons $h(x^*) \leq 0$. Comme $h(x)$ est convexe, la fonction atteint son minimum en ce point. En vérifiant qu'au minimum de la fonction, celle-ci est négative ou nulle, nous nous assurons d'après ses variations que $h(x) = 0$ présente au moins une solution. Ce point d'inflexion peut être calculé et est donné par :

$$x^* = \frac{\gamma_k P_k}{1 - \gamma_k M_k}, \text{ avec } \gamma_k = -\frac{\hat{a}_t^-}{\hat{b}_t^-}\left(\frac{\hat{a}_t^- M_k}{\hat{b}_t^-}\right)^{\frac{\hat{\epsilon}_t^-}{2-\hat{\epsilon}_t^-}}. \tag{4.31}$$

Une fois que nous nous sommes assuré de l'existence de solutions, nous utilisons une méthode de Newton. Celle-ci est initialisée avec la coordonnée $x_{\mathrm{p}}^{\mathbf{s}}$ du robot placé dans le référentiel de la superellipse, de cette façon, nous nous assurons de converger vers la bonne solution. Une fois le couple solution $(\hat{x}_k^-, \hat{y}_k^-)$ obtenu, nous pouvons remonter à la distance $\hat{d}_k^- = \left\| \begin{bmatrix} \hat{x}_k^- - x_{\mathrm{p}} \\ \hat{y}_k^- - y_{\mathrm{p}} \end{bmatrix} \right\|_2$.

4.3.5 Résultats

Nous prenons $\mathbf{Q} = 10^{-6}\mathbf{I}_6$ et $\mathbf{R} = 10^{-2}\mathbf{I}_{K_t}$. La figure 4.9 montre différentes étapes de la reconstruction d'une ellipse avec l'UKF pour superellipses. La figure 4.10 montre différentes étapes de la reconstruction d'une superellipse. Dans les deux cas, nous observons que les reconstructions se passent bien.

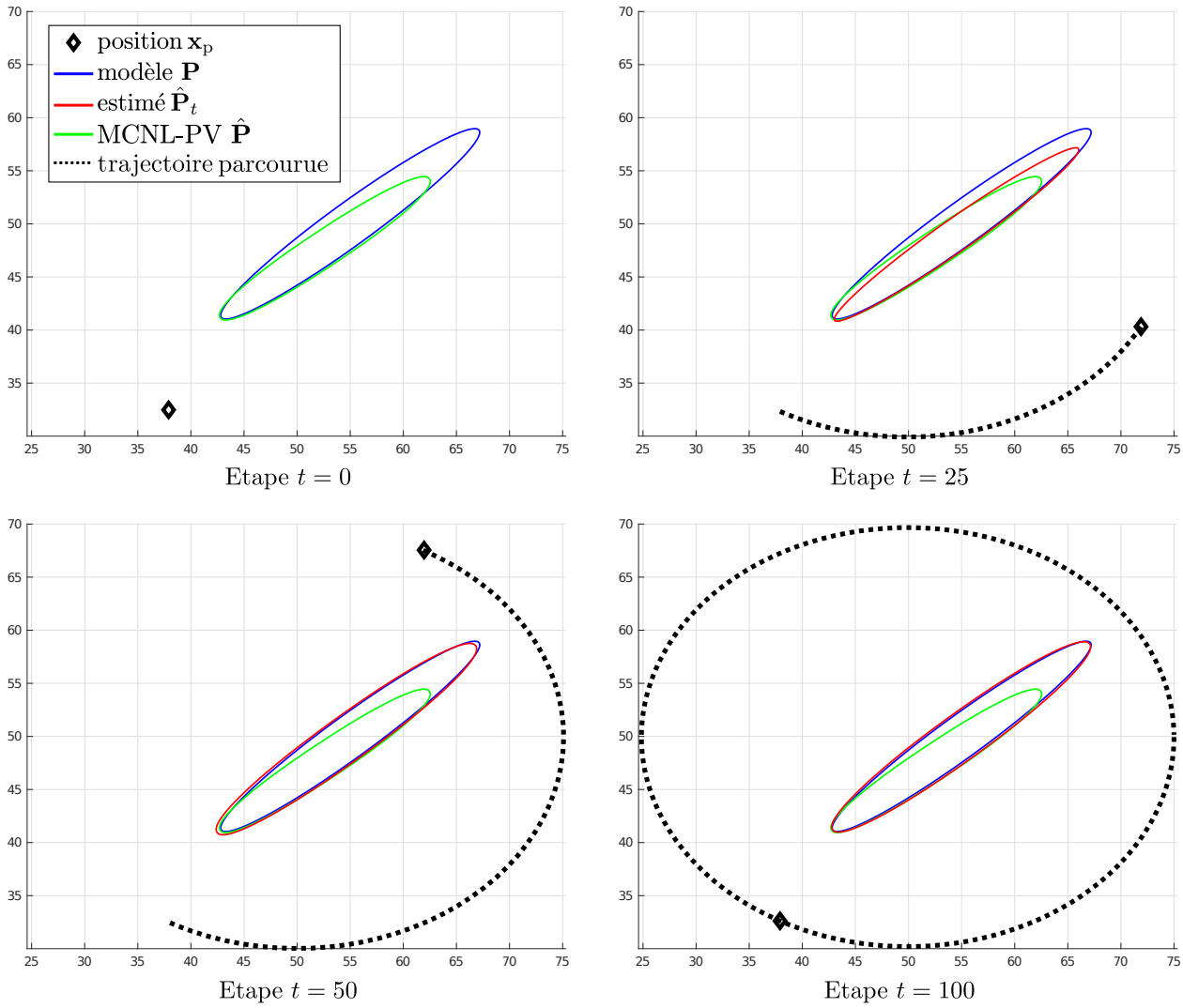


Figure 4.9 Filtrage d'une ellipse avec l'UKF pour superellipse.

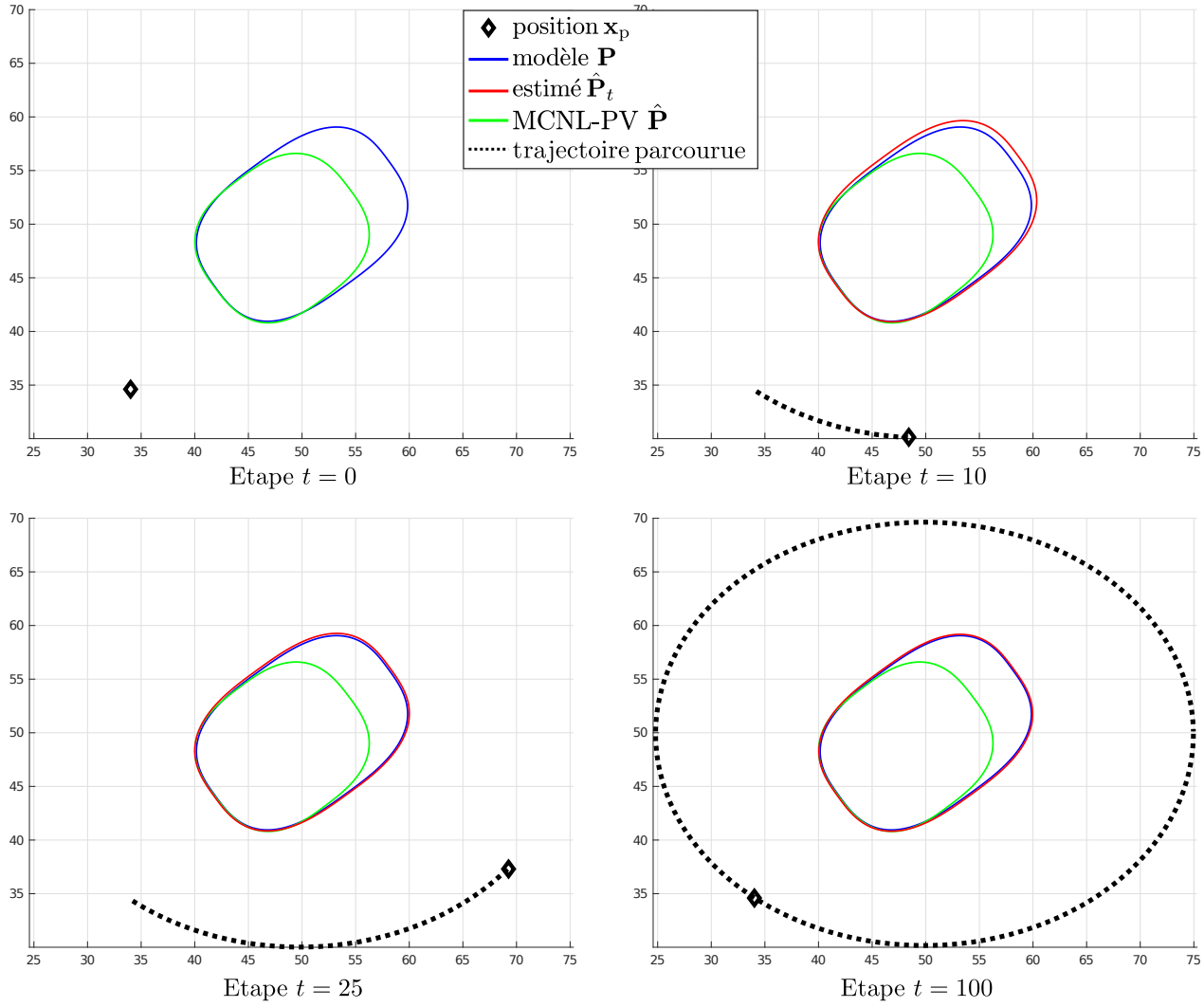


Figure 4.10 Filtrage d'une superellipse avec l'UKF pour superellipse.

4.4 Filtrage d'une superquadrique

Maintenant que tous les outils ont été mis en place, nous passons au cas tri-dimensionnel avec le filtrage de SQs.

4.4.1 Acquisition des données

Un robot tourne autour de la superquadrique et effectue des mesures avec un LRF qui scanne en trois dimensions. Le capteur choisi scanne verticalement selon un angle $\phi_l \in [0, \pi/4]$ et effectue une mesure par degré (soit 45 mesures). Puis ce laser tourne autour de l'axe vertical selon l'angle $\lambda_l \in [-\pi, \pi]$ avec une résolution de un degré aussi. Par ailleurs, le laser ne peut pas voir ce qui se situe trop loin de lui (30 mètres ici). La figure 4.11 illustre un scan du capteur pour un angle λ_l arbitraire.

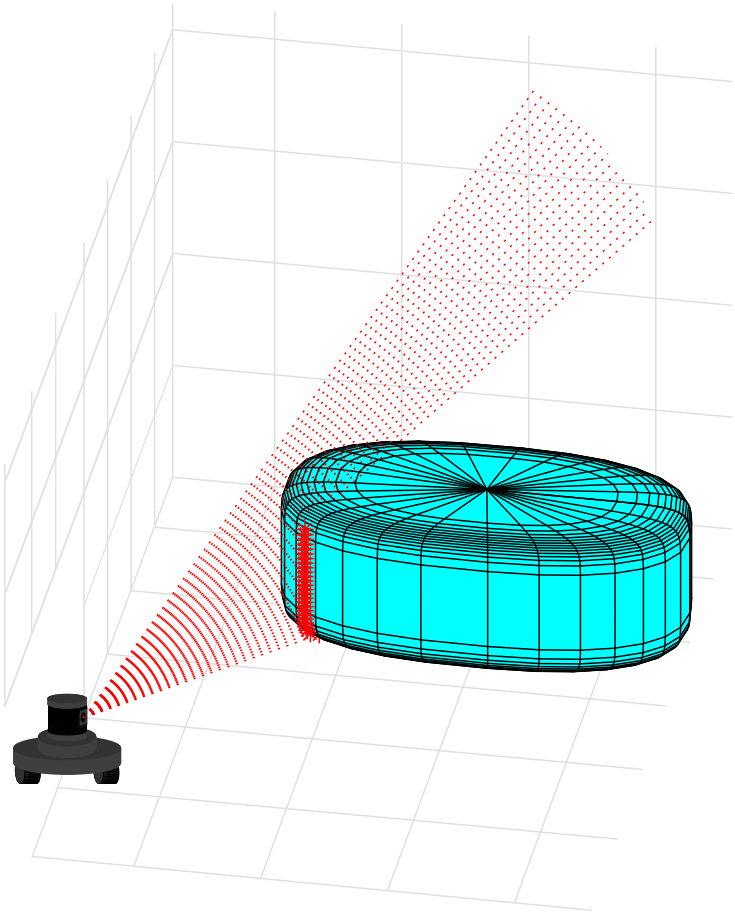


Figure 4.11 Scan d'une superquadrique effectué par le LRF.

4.4.2 Points d'intersection

Dans cette partie nous voulons déterminer quand est-ce que le faisceau laser intersecte la SQ. Ceci servira pour générer les données du capteur et aussi dans l'UKF lors du filtrage. Soit $[x_\mathrm{p}, y_\mathrm{p}, z_\mathrm{p}, \psi_\mathrm{p}]$ la position et orientation (pose) du robot (dans nos simulations z_p est toujours nul). Les points $\{(x, y, z)\}$ vérifiant les équations paramétriques suivantes sont sur le faisceau du laser de direction angulaire (λ_l, ϕ_l) :

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x_\mathrm{p} \\ y_\mathrm{p} \\ z_\mathrm{p} \end{bmatrix} + s \begin{bmatrix} \cos(\lambda_\mathrm{l} + \psi_\mathrm{p}) \cos\phi_\mathrm{l} \\ \sin(\lambda_\mathrm{l} + \psi_\mathrm{p}) \cos\phi_\mathrm{l} \\ \sin\phi_\mathrm{l} \end{bmatrix}, s \in \mathbb{R}^+. \tag{4.32}$$

En passant dans le référentiel de la SQ avec la matrice de rotation $\mathbf{R}$ et l'origine $(x_\mathrm{c}, y_\mathrm{c}, z_\mathrm{c})$:

$$\mathbf{R}^T \begin{bmatrix} x - x_\mathrm{c} \\ y - y_\mathrm{c} \\ z - z_\mathrm{c} \end{bmatrix} = \mathbf{R}^T \begin{bmatrix} x_\mathrm{p} - x_\mathrm{c} \\ y_\mathrm{p} - y_\mathrm{c} \\ z_\mathrm{p} - z_\mathrm{c} \end{bmatrix} + s\mathbf{R}^T \begin{bmatrix} \cos(\lambda_\mathrm{l} + \psi_\mathrm{p}) \cos\phi_\mathrm{l} \\ \sin(\lambda_\mathrm{l} + \psi_\mathrm{p}) \cos\phi_\mathrm{l} \\ \sin\phi_\mathrm{l} \end{bmatrix}, s \in \mathbb{R}. \tag{4.33}$$

En posant :

$$\mathbf{L} = \mathbf{R}^T \begin{bmatrix} \cos(\lambda_\mathrm{l} + \psi_\mathrm{p}) \cos\phi_\mathrm{l} \\ \sin(\lambda_\mathrm{l} + \psi_\mathrm{p}) \cos\phi_\mathrm{l} \\ \sin\phi_\mathrm{l} \end{bmatrix}, \quad \begin{bmatrix} x_\mathrm{p}^\mathsf{s} \\ y_\mathrm{p}^\mathsf{s} \\ z_\mathrm{p}^\mathsf{s} \end{bmatrix} = \mathbf{R}^T \begin{bmatrix} x_p - x_c \\ y_p - y_c \\ z_p - z_c \end{bmatrix}, \tag{4.34}$$

nous avons

$$\begin{bmatrix} x^\mathsf{s} \\ y^\mathsf{s} \\ z^\mathsf{s} \end{bmatrix} = \begin{bmatrix} x_\mathrm{p}^\mathsf{s} \\ y_\mathrm{p}^\mathsf{s} \\ z_\mathrm{p}^\mathsf{s} \end{bmatrix} + s\mathbf{L}, \ s \in \mathbb{R}. \tag{4.35}$$

Le vecteur $\begin{bmatrix} x_\mathrm{p}^\mathsf{s} \\ y_\mathrm{p}^\mathsf{s} \\ z_\mathrm{p}^\mathsf{s} \end{bmatrix}$ représente la position du robot dans le référentiel de la SQ, et $\mathbf{L}$ est un vecteur constant dépendant de l'orientation du robot et de la SQ, et de la direction du faisceau laser. En isolant s dans l'équation de z^s, on a $s = \dfrac{z^\mathsf{s} - z_\mathrm{p}^\mathsf{s}}{\mathbf{L}_3}$, et nous obtenons l'équation de droite suivante :

$$\begin{cases} x^\mathsf{s} = f_x(z^\mathsf{s}) = x_\mathrm{p}^\mathsf{s} + \dfrac{z^\mathsf{s} - z_\mathrm{p}^\mathsf{s}}{\mathbf{L}_3}\mathbf{L}_1, \\ y^\mathsf{s} = f_y(z^\mathsf{s}) = y_\mathrm{p}^\mathsf{s} + \dfrac{z^\mathsf{s} - z_\mathrm{p}^\mathsf{s}}{\mathbf{L}_3}\mathbf{L}_2. \end{cases} \tag{4.36}$$

À l'intersection entre la SQ et le faisceau laser, le point (x, y, z) vérifie aussi $F(x, y, z; \mathbf{P}) = 1$.

D'où l'équation en z^{s} à résoudre :

$$F(z^{\mathsf{s}};\mathbf{P}) = \left(\left(\frac{f_x(z^{\mathsf{s}})}{a}\right)^{\frac{2}{\epsilon_2}} + \left(\frac{f_y(z^{\mathsf{s}})}{b}\right)^{\frac{2}{\epsilon_2}}\right)^{\frac{\epsilon_2}{\epsilon_1}} + \left(\frac{z^{\mathsf{s}}}{c}\right)^{\frac{2}{\epsilon_1}} = 1. \tag{4.37}$$

Cette équation ne présente aucune solution si le faisceau n'intersecte pas la SQ, une si le faisceau lui est tangent, et deux sinon. Il n'existe pas dans le cas général de solution analytique à cette équation. Nous utilisons une méthode de Newton simple visant à résoudre :

$$F(z^{\mathsf{s}};\mathbf{P}) - 1 = 0. \tag{4.38}$$

La méthode de Newton est initialisée au point $z_{\mathrm{p}}^{\mathsf{s}}$, ceci nous assure de trouver la bonne solution (celle étant la plus proche du capteur). Nous voyons sur la figure 4.12 à gauche la difficulté avec laquelle notre algorithme trouve la solution dans ce cas de figure. Dans cet exemple, ϵ_1 est pris égal à 0.2, et pour des valeurs ϵ_1 plus faibles, il arrive que l'algorithme n'arrive pas à converger du tout. Ainsi dans nos travaux nous résolvons plutôt :

$$F(z^{\mathsf{s}};\mathbf{P})^{\frac{\epsilon_1}{2}} - 1 = 0. \tag{4.39}$$

Ceci ne change en rien la solution. Sur la figure 4.12 à droite nous constatons que la méthode de Newton converge en une ou deux itérations seulement (pour une précision désirée d'un centimètre).

Une fois que l'algorithme a convergé vers une solution z_*^{s}, nous retrouvons notre point d'intersection dans le référentiel de navigation :

$$\begin{bmatrix} x_* \\ y_* \\ z_* \end{bmatrix} = \mathbf{R} \begin{bmatrix} f_x(z_*^{\mathsf{s}}) \\ f_x(z_*^{\mathsf{s}}) \\ z_*^{\mathsf{s}} \end{bmatrix} + \begin{bmatrix} x_{\mathrm{c}} \\ y_{\mathrm{c}} \\ z_{\mathrm{c}} \end{bmatrix}. \tag{4.40}$$

Nous pouvons finalement remonter à la mesure de distance :

$$\hat{d} = \left\| \begin{bmatrix} x_* - x_{\mathrm{p}} \\ y_* - y_{\mathrm{p}} \\ z_* - z_{\mathrm{p}} \end{bmatrix} \right\|_2. \tag{4.41}$$

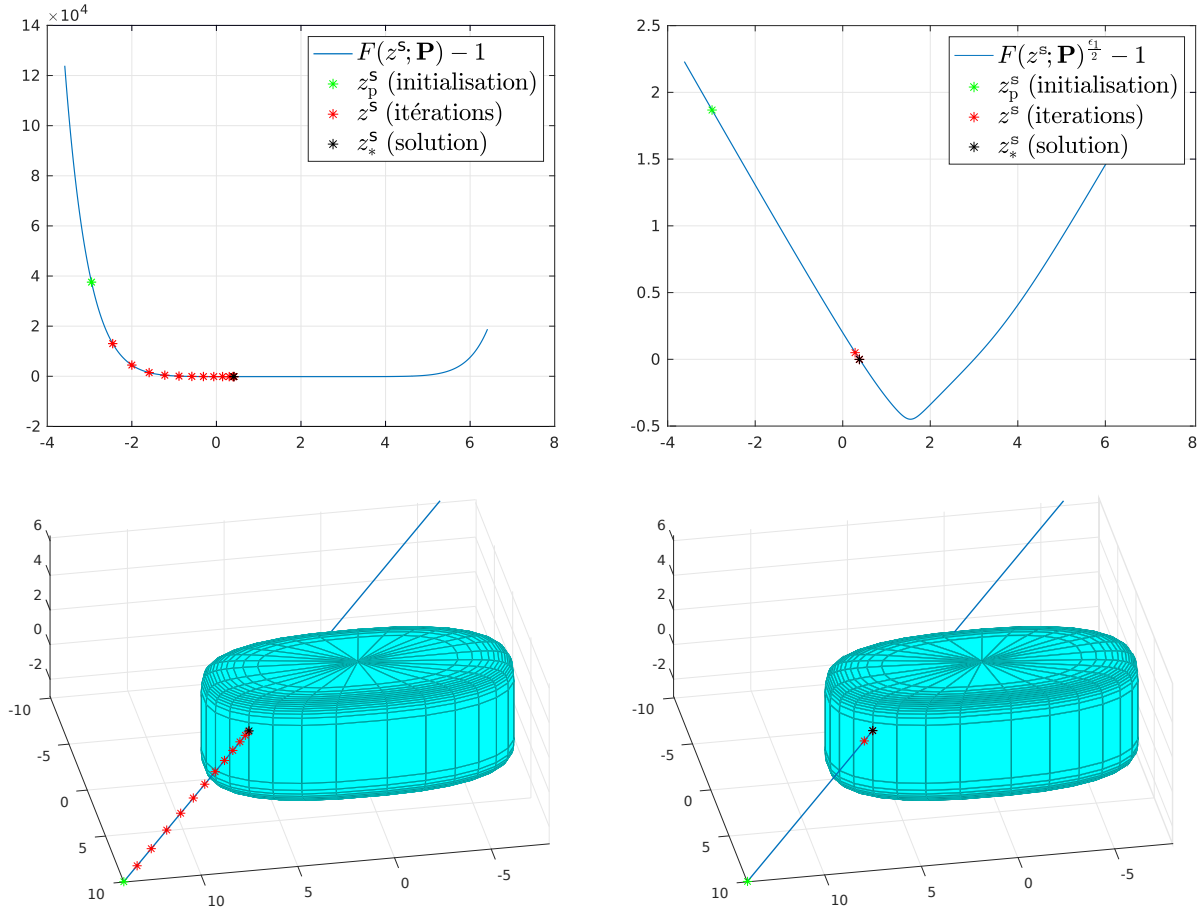


Figure 4.12 Comparaison des vitesses de convergence des deux méthodes de Newton.

4.4.3 Simulations

Nous pouvons faire tourner l'UKF comme vu précédemment, l'état $\mathbf{P}$ à estimer est simplement rallongé à $N = 11$ paramètres, et les gains sont changés. Il a été fixé pour les matrices de covariance de l'UKF :

$$\mathbf{Q} = 10^{-6}\mathrm{diag}(1,1,1,1,1,1,10^{-3},10^{-3},10^{-3},0.3,0.3),\ \mathbf{R} = 10^{-2}\mathbf{I}_{K_t}. \tag{4.42}$$

Les estimés sont initialisés par moindres carrés comme vu dans le chapitre 3, et $\mathbf{\Sigma}_0$ est basée sur la dilution de précision. Pour l'initialisation, nous nous plaçons dans un cas défavorable (où la DOP est grande, c'est à dire où l'objet est peu descriptible depuis la position du robot). Ainsi, les estimés initiaux obtenus par moindres carrés sont mauvais. L'UKF est testé sur deux SQs, de paramètres :

$$\mathbf{P}^{(1)} = [50,50,3,6,5,3,0,0,0,0.2,0.8],\ \mathbf{P}^{(2)} = [50,50,3,8,4,3,0,0,0,0.1,0.2], \tag{4.43}$$

et les figures 4.13 et 4.14 présentent différentes étapes lors de la reconstruction de ces deux SQs. Les figures 4.15 et 4.16 présentent les différences entre les paramètres $\hat{\mathbf{P}}_t$ estimés par l'UKF et les paramètres réels $\mathbf{P}$ en fonction de l'étape $t \in [\![0, 100]\!]$. On observe globalement que les paramètres convergent correctement, exceptions faites pour z_c et c qui sont sous estimés car le robot, d'où il est, ne perçoit ni au dessus ni en dessous des volumes.

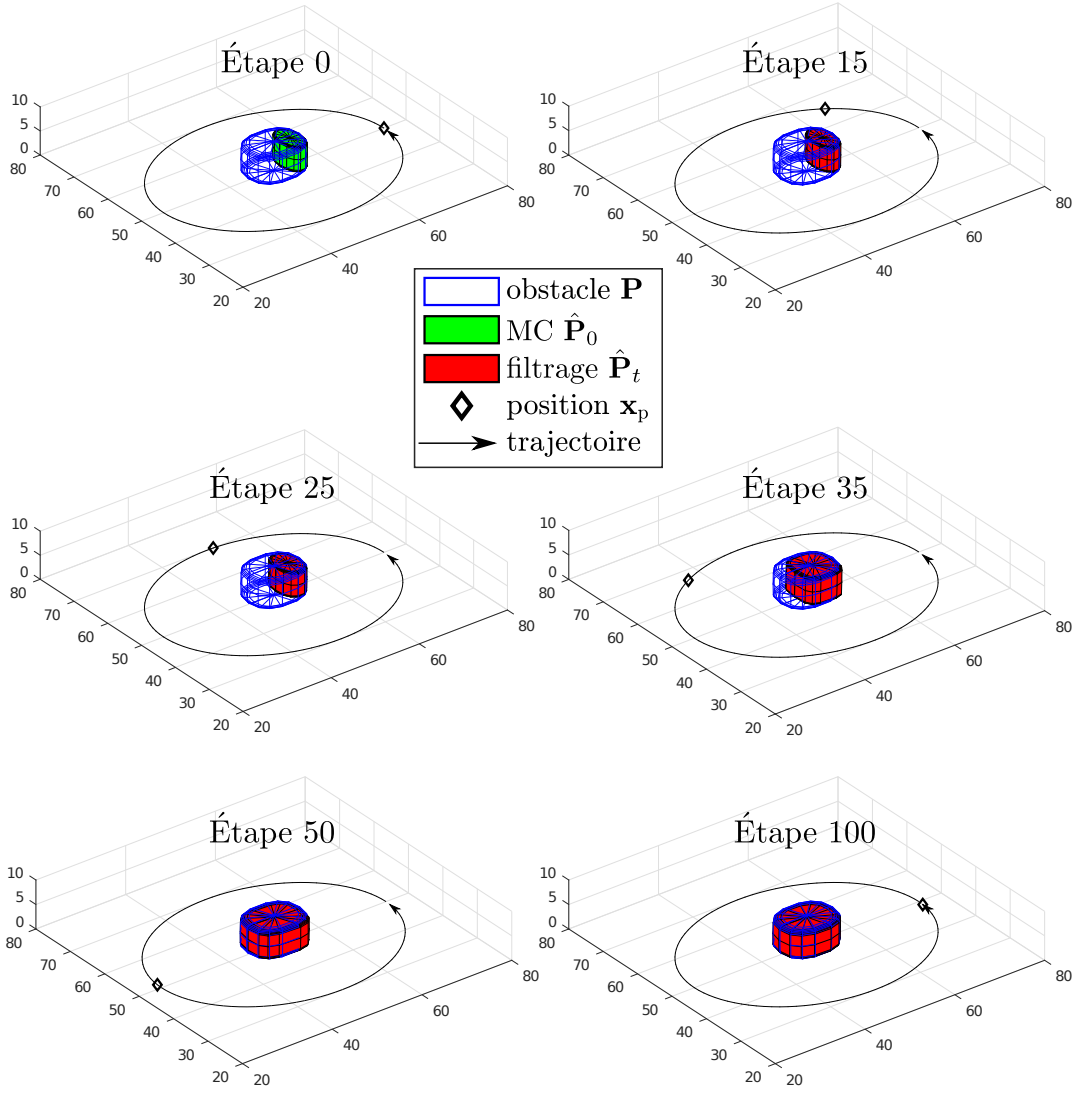


Figure 4.13 Représentation du filtrage de $\mathbf{P}^{(1)}$ lors de plusieurs étapes.

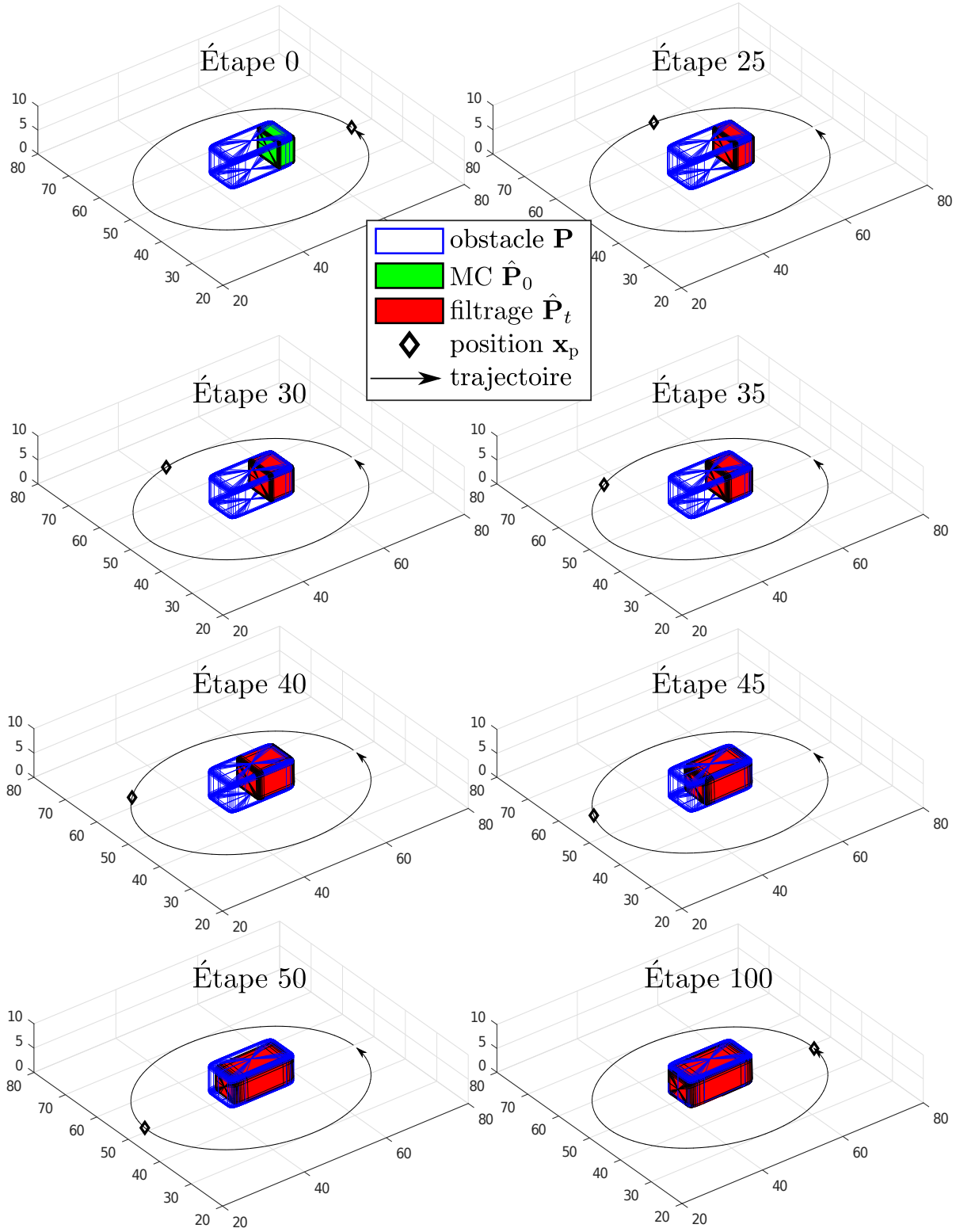


Figure 4.14 Représentation du filtrage de $\mathbf{P}^{(2)}$ lors de plusieurs étapes.

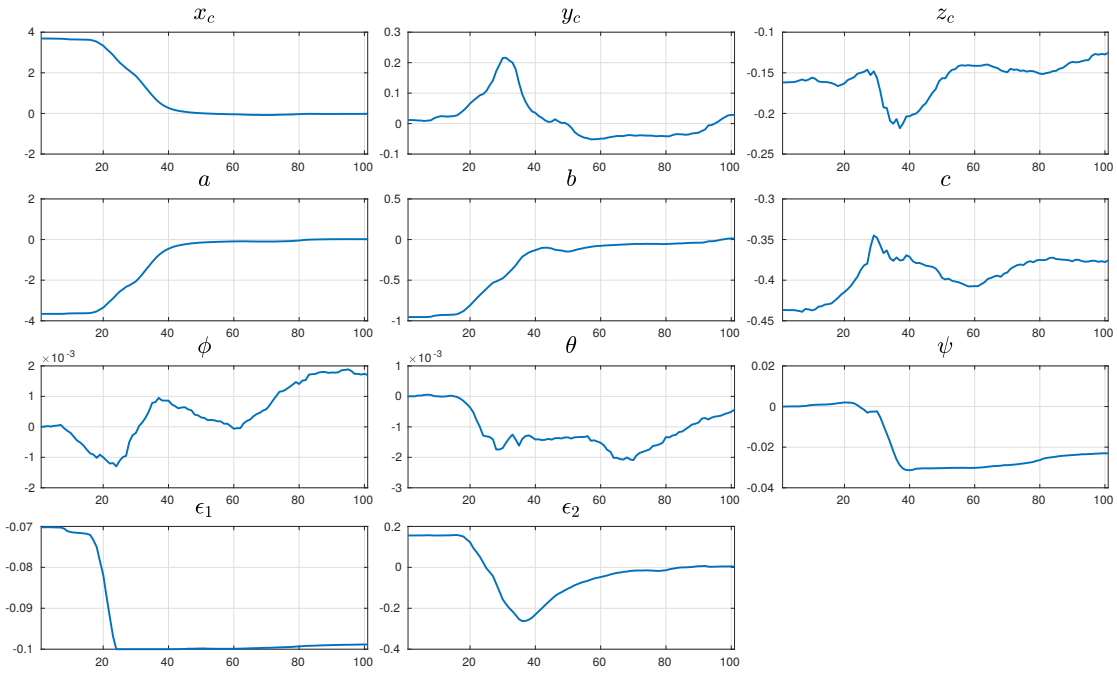


Figure 4.15 Différence entre les paramètres estimés et réels pour $\mathbf{P}^{(1)}$.

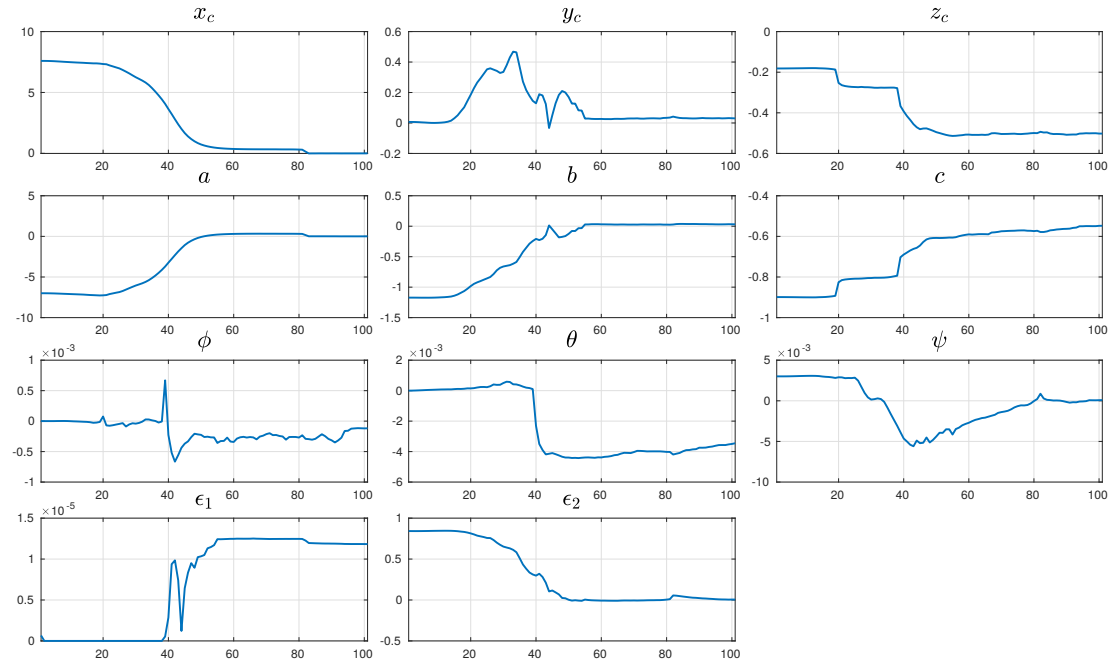
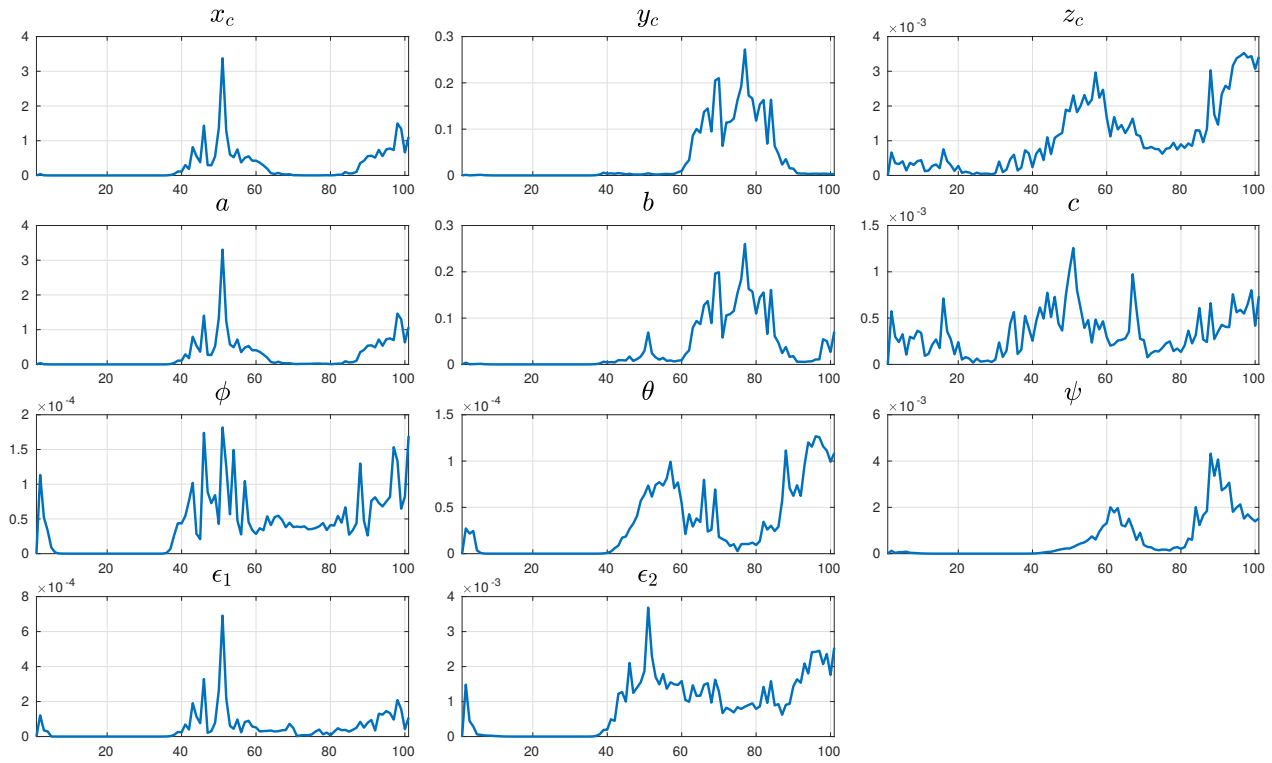


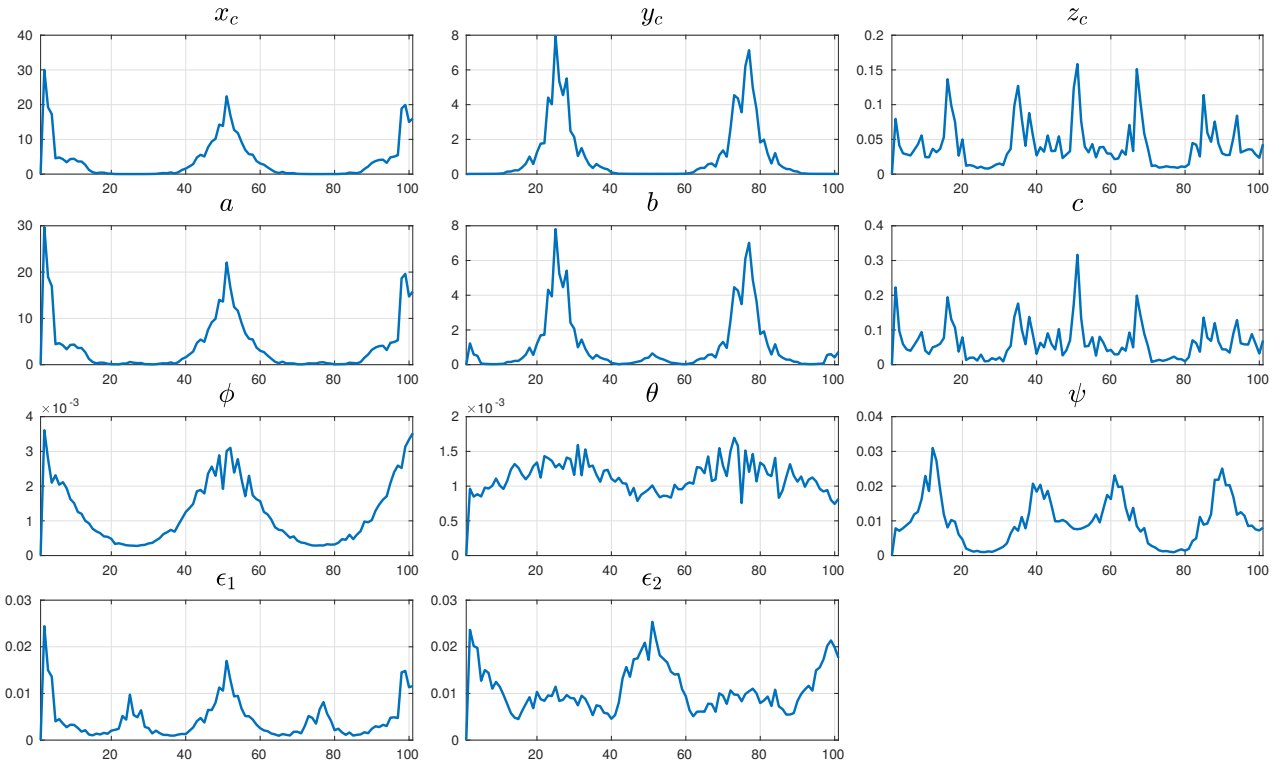
Figure 4.16 Différence entre les paramètres estimés et réels pour $\mathbf{P}^{(2)}$.

4.4.4 Considérations sur la dilution de précision

Les figures 4.17 et 4.18 montrent les DOPs. Nous distinguons les DOPs estimées calculées avec les paramètres estimés $\hat{\mathbf{P}}_t$ par le filtre et les DOPs vraies calculées avec les paramètres réels $\mathbf{P}$. Ces graphiques rendent compte de la difficulté à estimer certains paramètres selon la position du robot. Par exemple, autour de l'étape $t = 50$, les paramètres x_c et a sont plus difficilement estimables. Même constat pour les paramètres y_c et b aux étapes $t = 25$ et $t = 75$. Par ailleurs on remarque ici aussi que les DOPs estimées sont largement sous-estimées par rapports aux DOPs réelles, d'autant plus lors des premières étapes d'estimations. On remarque aussi les variations très importantes de DOPs pour le deuxième exemple pour les paramètres x_c, y_c, a et b, dues à la forme plus rectangulaire (ϵ_2 pris plus petit) que pour le premier exemple. En conclusion nous n'utilisons les mesures de DOP que pour initialiser la matrice $\mathbf{\Sigma}$ du filtre de Kalman.



DOPs estimées



DOPs vraies

Figure 4.17 DOPs calculées avec les paramètres estimés (haut) et les paramètres réels (bas) pour $\mathbf{P}^{(1)}$.

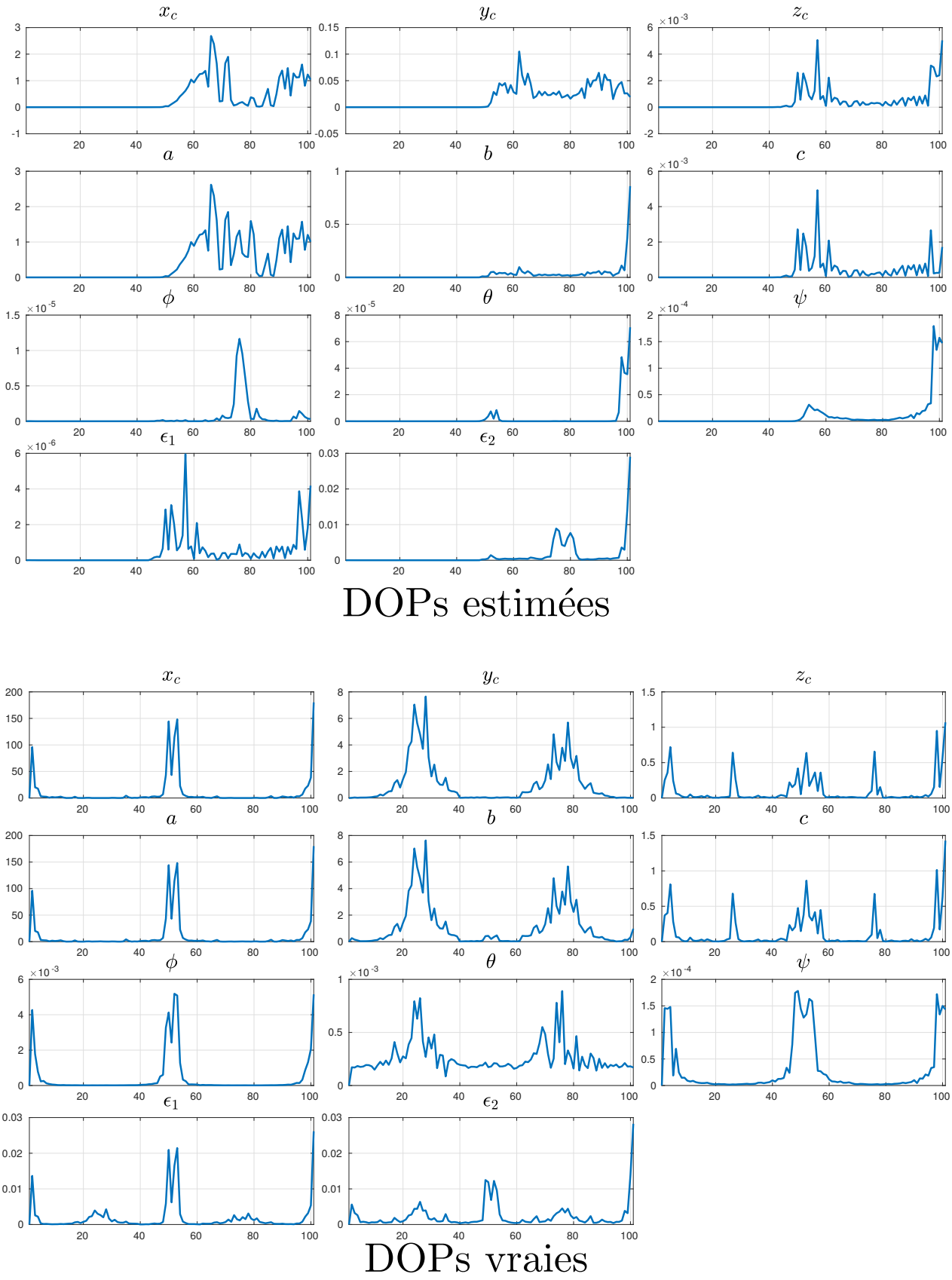


Figure 4.18 DOPs calculées avec les paramètres estimés (haut) et les paramètres réels (bas) pour $\mathbf{P}^{(2)}$.

4.5 Cas de mauvaise/lente convergence

Nous constatons à la section précédente, figure 4.14, que la convergence est lente. En effet des étapes 0 à 30, l'estimé évolue peu alors qu'il dispose d'informations suffisantes pour se mettre à jour. Nous proposons dans cette section une méthode pour pallier ce défaut.

4.5.1 Explication du phénomène

L'UKF a des difficultés à mettre l'estimé à jour quand celui-ci est trop éloigné du modèle réel, car il n'arrive pas à prendre en compte les mesures. La figure 4.19 montre un cas où la mise à jour est difficile. Les détections entourées en vert coïncident très bien avec les prédictions calculées par le filtre, et ces mesures ne contribueront alors à aucune mise à jour. Les détections entourées en orange n'ont pas pu être appareillées avec des prédictions du filtre, car le modèle estimé est trop petit, et n'intersecte pas les rayons du lidar. Seul le couple de points entouré en cyan peut contribuer à la mise à jour du modèle, ce qui fait au final très peu, d'où une difficulté d'évolution de l'estimé.

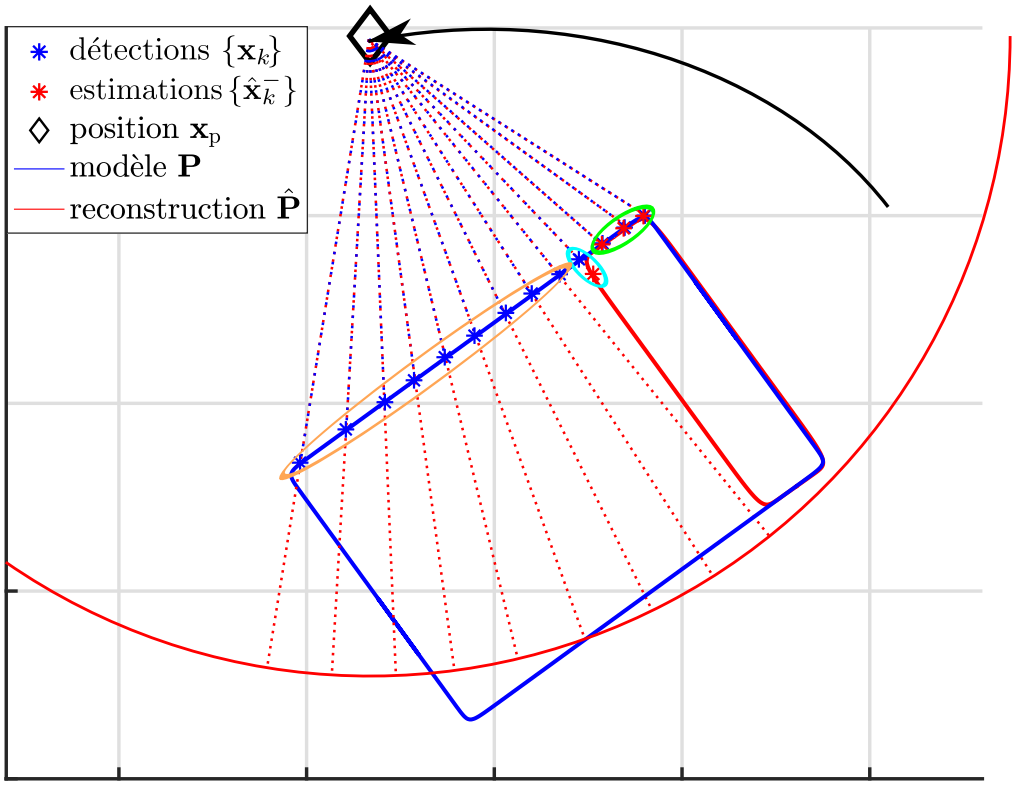


Figure 4.19 Illustration d'un cas où l'estimé est difficilement mis à jour.

4.5.2 Détection

Ce phénomène peut être détecté en analysant les mesures reçues par le capteur. Nous regardons la part des mesures ne pouvant pas être comparées à des estimations lors du calcul de l'innovation (cas des points en orange sur la figure 4.19). Si cette part est trop grande, alors nous détectons que le modèle $\hat{\mathbf{P}}$ n'est pas bon.

4.5.3 Rectification

Nous choisissons dans ce cas de détruire l'estimé et de le reconstruire par moindres carrés. Ce moindre carré peut être refait à partir des points courants seulement, ou en ajoutant aussi des points post-générés avec la SQ précédente. Cet ajout peut-être intéressant si l'estimé précédent est jugé pertinent (en regardant la part des mesures qui correspond bien aux estimations). Il faut bien penser aussi à mettre à jour la variance de l'estimé $\mathbf{\Sigma}$, sachant que lors de la destruction puis reconstruction du modèle, les paramètres des axes peuvent êtres permutés (a peut devenir b et vice-versa). Cette nouvelle variance est calculée avec la DOP comme pour l'initialisation du filtre.

4.5.4 Conclusion

Nous pouvons comparer les résultats obtenus en regardant les figures 4.14 et 4.20. Avec la modification de l'UKF, nous voyons qu'à l'étape $t = 35$, l'estimé est bien plus proche de l'obstacle réel.

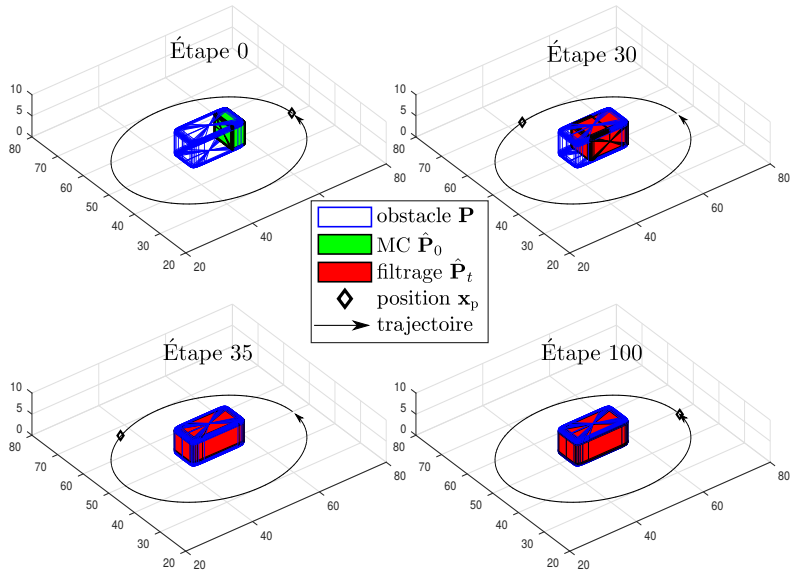


Figure 4.20 Illustration de l'efficacité de l'UKF modifié.

CHAPITRE 5 CARTOGRAPHIE

Dans ce chapitre nous traitons du problème de cartographie avec un robot mobile. Nous assumons connaître en tout temps la pose du robot $\mathbf{x}_\mathrm{p}$ dans le référentiel global de navigation $\mathcal{E}$. L'objectif est de retrouver les paramètres des différentes superquadriques disposées dans $\mathcal{E}$. Le LRF monté sur le robot nous permet d'observer ces SQs à travers des nuages de points. Nous traitons d'abord à la section 5.1 de la segmentation de ces nuages de points afin d'y séparer et distinguer les différentes SQs. Le problème d'association de données est présent dans ce chapitre, car il faut associer les mesures obtenues par le LRF aux différentes SQs en cours de reconstruction. La section 5.2.1 présente une méthode de cartographie naïve qui associe les données sans utiliser le modèle des SQs, tandis que section 5.3 nous utilisons directement les SQs en reconstruction pour associer nos points. Il arrive souvent que plusieurs SQs soient construites pour le même obstacle réel, et nous proposons à la section 5.4 un algorithme permettant de fusionner des SQs. Enfin nous présentons une simulation de notre algorithme de cartographie à la section 5.5.

5.1 Segmentation

5.1.1 Présentation du problème

Soit T le nombre d'étapes que met notre robot à parcourir son trajet, et soit $t \in [\![0, T]\!]$ une étape quelconque. À l'étape t, notre robot se trouve à la pose $\mathbf{x}_{\mathrm{p},t}$, et au cour d'une étape t, le laser effectue des mesures $\tilde{\mathbf{Z}}_t$. $\tilde{\mathbf{Z}}_t$ contient K_t mesures $\tilde{\mathbf{z}}_k$ correspondant aux détections du laser lors de l'étape t. Une mesure $\tilde{\mathbf{z}}_k$ comporte trois composantes, $\tilde{\mathbf{z}}_k = [\tilde{d}_k, \tilde{\lambda}_k, \tilde{\phi}_k]$, correspondant au cap, à l'élévation et à la distance de la détection par rapport au référentiel du robot. Les détections sont dues aux M obstacles (superquadriques) $\mathbf{P}_m, m \in [\![1, M]\!]$ de l'environnement $\mathcal{E}$. Le but de la segmentation, lors d'une étape t, est de classer ensemble les mesures de $\tilde{\mathbf{Z}}_t$ associées au même obstacle.

5.1.2 Segmentation basée sur le filtrage de Kalman en 2D

Nous avons d'abord utilisé un filtre de Kalman pour segmenter les données en deux dimensions. En deux dimensions les données provenant du capteur lors d'une étape apparaissent comme ceci : $\tilde{\mathbf{z}}_k = [\tilde{d}_k, \tilde{\lambda}_k],\ k \in [\![1, K]\!]$ où $K \leq 360$ étant donné que notre LRF est dimensionné pour faire une mesure par degré. De plus, ces mesures sont ordonnées selon $\lambda \in [-\pi, \pi]$, et $\lambda_{k+1} - \lambda_k > 0$.

La Segmentation Basée sur le Filtrage de Kalman (KFBS) [52] regarde successivement chaque mesure, et prédit si la mesure en cours d'étude correspond au modèle (ou état) estimé avec les mesures précédentes. Pour réaliser ceci, un filtre de Kalman linéaire est utilisé. L'état $\mathbf{X}_k = [\tilde{d}_k \ \frac{\partial \tilde{d}_k}{\partial \lambda}]^T$ représente la distance d'un obstacle pour la mesure k donnée par le LRF, ainsi que la variation de distance selon le cap de mesure à cette étape. Voici les équations de Kalman :

$$\begin{cases} \mathbf{X}_{k+1} = \mathbf{A}\mathbf{X}_k + \mathbf{w}_k, \\ d_k = \mathbf{C}\mathbf{X}_k + v_k, \end{cases} \tag{5.1}$$

où $\mathbf{A} = \begin{bmatrix} 1 & \Delta\lambda \\ 0 & 1 \end{bmatrix}$ et $\mathbf{C} = [1 \ 0]$. Dans notre cas $\Delta\lambda = 2\pi/360$. L'algorithme 1 présente le fonctionnement de KFBS.

Initialisation du filtre $\mathbf{\Sigma}_0^+, \hat{\mathbf{X}}_0^+$
for $k = 1 : K$ **do**
  Propagation de l'état
  $\hat{\mathbf{X}}_k^- \leftarrow \hat{\mathbf{X}}_{k-1}^+$
  $\mathbf{\Sigma}_k^- \leftarrow \mathbf{A}\mathbf{\Sigma}_{k-1}^+\mathbf{A}^T + \mathbf{Q}$
  $\mathbf{I}_k \leftarrow \tilde{d}_k - \mathbf{C}\hat{\mathbf{X}}_k^-$
  $\mathbf{S}_k \leftarrow \mathbf{C}\mathbf{\Sigma}_k^-\mathbf{C}^T + \mathbf{R}$
  if $\mathbf{I}_k^T\mathbf{S}_k^{-1}\mathbf{I}_k > D_{th}$ **then**
    Extraire les extrémités et identifier les segments
    Réinitialisation du filtre $\mathbf{\Sigma}_0^+, \hat{\mathbf{X}}_0^+$
  else
    Mise à jour de l'état
    $\mathbf{K}_k \leftarrow \mathbf{\Sigma}_k^-\mathbf{C}^T\mathbf{S}^{-1}$
    $\hat{\mathbf{X}}_k^+ \leftarrow \hat{\mathbf{X}}_k^- + \mathbf{K}_k\mathbf{I}_k$
    $\mathbf{\Sigma}_k^+ \leftarrow \mathbf{\Sigma}_k^- - \mathbf{K}_k\mathbf{C}\mathbf{\Sigma}_k^-$
  end if
end for

Algorithm 1 Algorithme KFBS

La décision de l'algorithme de savoir si le point courant k appartient au même groupement que celui du modèle se fait à partir de l'innovation. En effet, si l'innovation est trop grande, c'est que la prédiction du filtre a été trop faible. Or, cette prédiction est basée sur l'état précédent. Ainsi, si cette prédiction est trop faible, c'est que l'état précédent est trop éloigné de l'état que représente le point k. Une innovation grande traduira ainsi l'apparition d'un nouveau groupement, (ou d'un outlier au groupement présent) : la segmentation est donc faite. Pour plus de robustesse, nous décidons d'effectuer KFBS avec la liste des mesures

ordonnées dans les deux sens, (la liste originale et la liste renversée), et de vérifier que pour un même groupement, deux mesures successives ont un cap de mesure proche. Enfin, on rejette aussi les groupements contenant trop peu de mesures. La figure 5.1 illustre le résultat de KFBS. Nous voyons que les points extrémaux ont tendance à être rejetés par le filtre, de même que les points trop écartés sur l'obstacle rectangulaire en haut. Ces points sont effectivement caractérisés par une innovation plus importante car ils sont à la limite entre deux modèles, et ne correspondent donc pas à un modèle précis. Par la suite, nous n'allons pas garder cette technique de segmentation car elle ne convient pas pour les mesures dans l'espace 3D. En effet, l'ordre des mesures importe beaucoup dans cet algorithme, et il est difficile d'ordonner celles-ci en 3D.

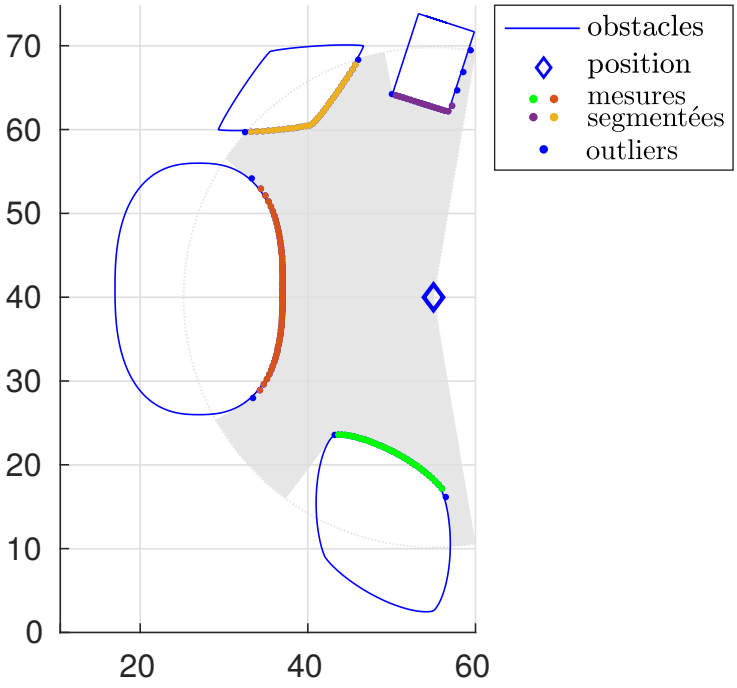


Figure 5.1 Illustration de l'algorithme KFBS.

5.1.3 Étiquetage des composantes connectées

Nous utilisons la technique d'Étiquetage des Composantes Connectées (CCL), définie par [53]. Il s'agit de créer une image labellisée qui associe pour une image donnée chaque composante connectée à un label (une étiquette) unique.

Nous appliquons cette technique à nos mesures. Notre espace de mesures $\tilde{\mathbf{Z}}$ est tri-dimensionnel, et nous le discrétisons en voxels. Afin de rester fidèle à la vision du LRF, nous choisissons les coordonnées sphériques pour créer la grille de l'espace visible par le laser. Une fois la grille

créée, nous associons à chaque cellule le nombre de détections faites par le laser. Une cellule est dite occupée si elle contient au moins une mesure. La grille ainsi remplie est assez sparse (beaucoup de cases sont inoccupées). La figure 5.2 montre comment cette grille est construite en deux dimensions. Notre capteur est dimensionné pour prendre une mesure chaque degré dans le plan horizontal, et nous avons discrétisé en conséquence pour avoir environ deux mesures par cellule, quelque soit sa distance par rapport au robot. Les angles sont ainsi plus finement discrétisés que la distance.

Une fois la grille obtenue, nous commençons la segmentation en repérant chaque ensemble de cellules occupées connecté (connexe) dans l'espace 3D. Nous définissons le voisinage d'une cellule comme les 8 cellules qui lui sont les plus proches en 2D, et 26 en 3D. Nous utilisons l'algorithme décrit par [54], qui binarise d'abord la grille précédente (une case vaut 1 si au moins une détection est faite, et 0 sinon). Cet algorithme est implémenté dans `Matlab` avec `bwlabeln`. Un exemple figure 5.2 révèle par exemple trois composantes connexes.

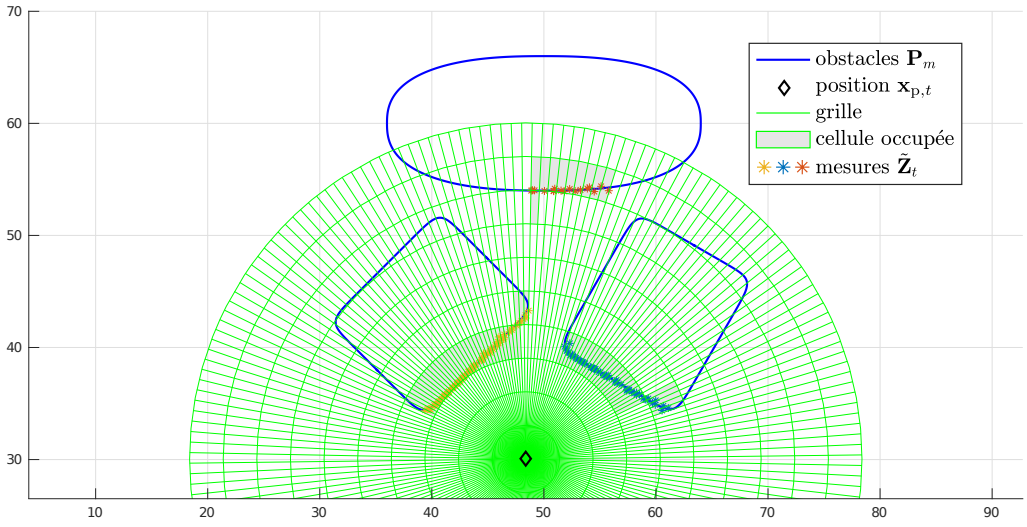


Figure 5.2 Illustration de la grille de segmentation.

5.1.4 Groupage spatial basé sur la densité

Nous proposons un autre algorithme décrit par [55], nommé *Density-Based Spatial Clustering of Applications with Noise* (DBSCAN), qui partitionne des données et qui représente une approche efficace pour la segmentation. Il a d'abord été utilisé pour identifier des classes d'étoiles dans des bases de données spatiales. C'est un algorithme basé sur la densité dans le sens où les zones denses appartiendront à différentes classes, tandis que les zones moins denses sont considérées comme du bruit. DBSCAN a besoin de deux paramètres pour fonctionner,

une distance η et un nombre minimum de point p devant se trouver dans un rayon η pour que lesdits points soient considérés comme un même cluster. Cet algorithme présente l'avantage d'être simple à implémenter, et ne nécessite pas de savoir au préalable le nombre de clusters à identifier. Il peut aussi gérer les données aberrantes. Son principal défaut réside dans la difficulté à traiter des clusters de densités trop différentes. Or, ceci est souvent le cas pour nous. En effet, pour un obstacle proche, nous auront plus de données rapprochées que pour un obstacle éloigné.

5.2 Cartographie par segmentation puis association de données

L'association de données à l'étape t revient à associer des groupes de mesures de $\tilde{\mathbf{Z}}_t$ à des obstacles $\mathbf{P}_m$ de $\mathcal{E}$. Dans cette section, nous détaillons une méthode d'association de données en utilisant au préalable un algorithme de segmentation (voir section 5.1). Dans la section suivante 5.3, nous expliquons comment faire l'association de données et la segmentation en même temps.

5.2.1 Méthode des barycentres

Initialisation. À l'étape $t = 0$, nous recevons les mesures $\tilde{\mathbf{Z}}_0$. Celles-ci sont d'abord segmentées (voir section 5.1) en $N_0 \in \mathbb{N}$ classes $\mathbf{C}_n, n \in [\![1, N_0]\!]$. Pour chaque classe, nous ajoutons un obstacle (superquadrique) $\hat{\mathbf{P}}_m$ à l'environnement (voir section 3). Par soucis de mémoire, nous ne souhaitons pas conserver $\tilde{\mathbf{Z}}_0$, mais nous devons tout de même garder une trace de ce qui a été observé. Nous décidons pour chaque classe de garder uniquement le barycentre des mesures effectuées. Pour ceci, il faut pouvoir associer une mesure du laser à un point dans l'espace, ce qui est possible puisque l'on connaît la pose $\mathbf{x}_\mathrm{p}$ du robot. Connaissant la transformation qui lie les mesures $\tilde{\mathbf{Z}}$ et la position du robot $\mathbf{x}_\mathrm{p}$ aux positions des détections $\{\tilde{\mathbf{x}}\}$ dans le référentiel global $\mathcal{E}$ (voir l'équation 1.2), nous pouvons alors calculer un barycentre $\mathbf{b}_n$ pour chaque classe :

$$\mathbf{b}_n = \frac{1}{|\mathbf{C}_n|} \sum_{\tilde{\mathbf{x}} \in \mathbf{C}_n} \tilde{\mathbf{x}}. \tag{5.2}$$

Ce sont ces barycentres, en plus des SQs, qui seront mémorisés par l'algorithme. Ces barycentres seront mis à jour lors des étapes suivantes à chaque fois que de nouvelles mesures leurs seront associées. Nous conservons aussi à chaque fois les barycentres de l'étape précédente. Pour récapituler, nous aurons une variable globale $\mathbf{B}_\mathrm{g}$ qui met à jour les barycentres pondérés de chaque obstacle depuis l'étape $t = 0$. Puis nous aurons une variable $\mathbf{B}_t$ qui change complè-

tement d'une étape à l'autre mémorisant uniquement les barycentres des classes identifiées à l'étape t courante.

Hérédité. Lors d'une étape $t > 0$, nous procédons de même pour la segmentation et la création de classes. Il s'agit maintenant d'associer ces classes aux obstacles précédemment identifiés, en utilisant un critère sur les barycentres. La première chose à faire est alors de calculer $\mathbf{B}_t$. Ensuite, ce processus d'identification passe par deux étapes :

Nous voulons d'abord voir si les classes de l'étape t correspondent à des objets identifiés à l'étape précédente $t-1$. En effet il est fort probable d'observer les mêmes objets entre deux étapes successives. Nous comparons alors $\mathbf{B}_t$ à $\mathbf{B}_{t-1}$. Si certains barycentres sont proches (distance euclidienne inférieure à un certain seuil τ_1), alors nous pouvons associer certaines classes de t à celles de $t-1$. Or, le problème d'association de données étant résolu à l'étape $t-1$, il le devient pour ces classes de l'étape t.

Ensuite, si nous ne pouvons pas associer au moins une classe à des objets de l'étape précédente, alors deux cas de figure se présentent :

- L'objet correspondant à cette classe a déjà été observé mais plus loin dans le passé, (pas à l'étape précédente mais à des étapes antérieures). Nous cherchons ici une fermeture de boucle. Ainsi, nous comparons maintenant $\mathbf{B}_t$ à $\mathbf{B}_{\mathrm{g}}$. Comme précédemment, si certains barycentres sont proches (distance inférieure à un autre seuil τ_2, plus grand à priori), alors nous pouvons associer certaines classes aux obstacles de la carte, et le problème d'association de données est résolu.
- L'objet correspondant à cette classe n'a encore jamais été observé, et il faut en créer un nouveau pour l'ajouter à la carte.

Finalement, il ne faut pas oublier de mettre à jour $\mathbf{B}_{\mathrm{g}}$, ainsi que les estimés $\hat{\mathbf{P}}_m$ (voir la section 4).

Cette technique repose sur le calcul des barycentres et n'utilise pas les informations $\hat{\mathbf{P}}_m$ que nous avons sur les SQs. De plus les seuils τ_1 et τ_2 sont déterminant car décident de la création d'un nouvel objet, et leur réglage reste délicat. Souvent, il arrive que de nouvelles SQs soient créées pour un obstacle déjà identifié, et nous nous retrouvons avec plusieurs estimés $\hat{\mathbf{P}}_m$ pour le même obstacle réel $\mathbf{P}$. Pour résoudre ce problème, nous proposons de fusionner ces estimés pour n'en garder qu'un. La technique de fusion est expliquée par la suite à la section 5.4.

5.3 Cartographie par association directe d'un point à une superquadrique

Nous proposons ici une autre technique s'appuyant directement sur les estimés en cours, associant directement chaque point à une SQ. Soit un point $\tilde{\mathbf{x}}$ mesuré dans notre environnement, nous souhaitons savoir à quelle SQ estimée il appartient. Nous souhaitons utiliser les incertitudes $\mathbf{\Sigma}$ sur ces quadriques renvoyées par l'UKF (voir section 4). Nous utilisons la notion de distance d'un point $\tilde{\mathbf{x}}$ à une SQ pour effectuer notre association de données.

5.3.1 Distance d'un point à une superquadrique

La formule de la distance d'un point à une superquadrique (ou superellipse) peut être calculée par minimisation numérique, mais il n'y a pas de forme analytique proche connue. Nous utilisons la distance euclidienne radiale définie dans [41] et illustrée figure 5.3. Celle-ci représente la distance entre un point $\mathbf{x}$ et une superquadrique $\mathbf{P}$ le long d'une ligne passant par ce point et le centre $\mathbf{x}_c$ de la superquadrique.

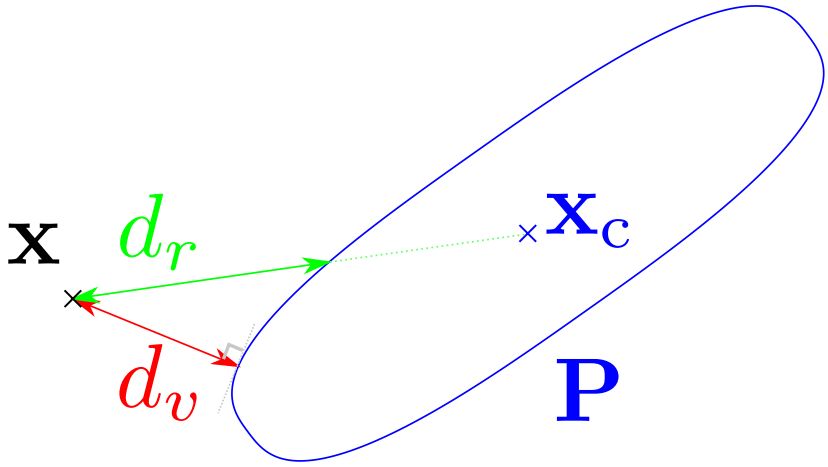


Figure 5.3 Distance d'un point $\mathbf{x}$ à une superellipse $\mathbf{P}$. En vert, d_r représente la distance euclidienne radiale, et en rouge, d_v la distance euclidienne vraie.

La formule de la distance radiale d_r [41] entre un point $\mathbf{x}$ et une superquadrique $\mathbf{P}$ est donnée par :

$$d_r(\mathbf{x};\mathbf{P}) = \|\mathbf{x}-\mathbf{x}_c\|_2 \left(1 - F(\mathbf{x};\mathbf{P})^{-\frac{\epsilon_1}{2}}\right). \tag{5.3}$$

5.3.2 Distance estimée

Soit une superellipse de paramètres $\mathbf{P}$ dont nous avons une estimation $\hat{\mathbf{P}}$ avec une erreur $\mathbf{\Sigma}$. Nous aimerions utiliser $\mathbf{\Sigma}$ afin de savoir si notre mesure de distance est fiable ou non.

Approche linéaire. En prenant pour hypothèse que l'erreur est suffisamment petite, nous pouvons tenter de linéariser l'expression de d_r autour de $\hat{\mathbf{P}} = \mathbf{P} + \mathrm{d}\mathbf{P}$:

$$d_r(\mathbf{x};\mathbf{P}) \sim d_r(\mathbf{x};\hat{\mathbf{P}}) + \mathbf{J_P}\mathrm{d}\mathbf{P}, \tag{5.4}$$

avec la jacobienne $\mathbf{J_P} = \left.\dfrac{\partial d_r(\mathbf{x};\mathbf{P})}{\partial \mathbf{P}}\right|_{\mathbf{P}=\hat{\mathbf{P}}}$. Nous donnons alors l'erreur sur l'estimation de la distance provenant de l'erreur d'estimation de $\hat{\mathbf{P}}$:

$$\begin{aligned}
\sigma^2_{\mathbf{P}} &= \mathbb{E}\left[\left(d_r(\mathbf{x};\mathbf{P}) - d_r(\mathbf{x};\hat{\mathbf{P}})\right)\left(d_r(\mathbf{x};\mathbf{P}) - d_r(\mathbf{x};\hat{\mathbf{P}})\right)^T\right], \\
&= \mathbb{E}\left[\mathbf{J_P}\mathrm{d}\mathbf{P}\mathrm{d}\mathbf{P}^T\mathbf{J}_\mathbf{P}^T\right], \\
&= \mathbf{J_P}\mathbb{E}\left[\mathrm{d}\mathbf{P}\mathrm{d}\mathbf{P}^T\right]\mathbf{J}_\mathbf{P}^T, \\
\sigma^2_{\mathbf{P}} &= \mathbf{J_P}\mathbf{\Sigma}\mathbf{J}_\mathbf{P}^T.
\end{aligned} \tag{5.5}$$

Approche de Monte Carlo. La distance d_r à une dépendance fortement non linéaire en $\mathbf{P}$. L'approche linéarisante peut donner des résultats non pertinents dans le cas où l'estimé $\hat{\mathbf{P}}$ est éloigné du vrai modèle $\mathbf{P}$. On peut tenter une seconde approche plus empirique, basée sur la simulation de Monte Carlo [56]. Nous choisissons un échantillon de N valeurs $\mathcal{P}^{(i)}$ distribuées selon une loi normale $\mathcal{N}(\hat{\mathbf{P}}, \mathbf{\Sigma})$. Nous calculons ensuite les distances $d_r^{(i)}$ du point $\mathbf{x}$ à chaque $\mathcal{P}^{(i)}$. Nous calculons finalement :

$$\begin{aligned}
\hat{d}_r &= \tilde{\mathbb{E}}_i[d_r^{(i)}] = \frac{1}{N}\sum_i d_r^{(i)}, \\
\sigma^2_{\mathbf{P}} &= \tilde{\mathbb{E}}_i[(d_r^{(i)} - \hat{d}_r)^2] = \frac{1}{N}\sum_i (d_r^{(i)} - \hat{d}_r)^2.
\end{aligned} \tag{5.6}$$

Approche de la transformée *Unscented*. Une troisième approche peut se faire via la transformée *Unscented* [49], déjà utilisée à la section 4.3. Contrairement aux méthodes de Monte-Carlo, l'échantillon des $\mathcal{P}^{(i)}$ est précisément défini et l'estimation est donc déterministe. Nous voulons extrapoler la distribution de d_r connaissant celle de $\mathbf{P}$ et la relation $d_r = f(\mathbf{P})$. Nous procédons comme suit [49] :

— Choisir l'échantillon $\{\mathcal{P}^{(i)}\}$ de départ (*sigma points*) :

$$\begin{aligned}
\mathcal{P}^{(0)} &= \hat{\mathbf{P}}, \\
\mathcal{P}^{(i)} &= \hat{\mathbf{P}} + \sqrt{N+\lambda}\left[\sqrt{\mathbf{\Sigma}}\right]_i, \\
\mathcal{P}^{(i+N)} &= \hat{\mathbf{P}} - \sqrt{N+\lambda}\left[\sqrt{\mathbf{\Sigma}}\right]_i, i \in [\![1, N]\!].
\end{aligned} \tag{5.7}$$

Le coefficient λ représente à quelle distance de la moyenne nous choisissons notre échantillon, nous prenons $\lambda = 1$. N représente la dimension du vecteur $\mathbf{P}$.

— Choisir les poids $\{\mathcal{W}^{(i)}\}$ correspondants :

$$\begin{aligned} \mathcal{W}^{(0)} &= \frac{\lambda}{N+\lambda}, \\ \mathcal{W}^{(i)} &= \frac{1}{2(N+\lambda)}, i \in [\![1, 2N]\!]. \end{aligned} \tag{5.8}$$

— Propager notre échantillon pour calculer les distances correspondantes :

$$d_r^{(i)} = d_r(\mathbf{x}; \mathcal{P}^{(i)}), i \in [\![0, 2N]\!]. \tag{5.9}$$

— Calculer les moyenne et variance pondérées de l'échantillon :

$$\begin{aligned} \hat{d}_r &= \frac{1}{N} \sum_i \mathcal{W}^{(i)} d_r^{(i)}, \\ \sigma_{\mathbf{P}}^2 &= \frac{1}{N} \sum_i \mathcal{W}^{(i)} (d_r^{(i)} - \hat{d}_r)^2. \end{aligned} \tag{5.10}$$

Comparaison des différentes méthodes La figure 5.4 montre l'allure des échantillons $\{\mathcal{P}^{(i)}\}$ pour la méthode de Monte-Carlo et la méthode *Unscented*. Nous observons que du côté où le robot effectue son observation, les $\mathcal{P}^{(i)}$ sont semblables et se recoupent. Par contre, du côté non observé de la quadrique, les courbes des $\mathcal{P}^{(i)}$ ne se superposent plus que ce soit pour la méthode de Monte-Carlo ou *Unscented*, et nous constatons une distribution de plus grande variance autour de l'estimé $\hat{\mathbf{P}}$.

Nous plaçons maintenant un point $\mathbf{x}$ sur la quadrique (voir figure 5.5). Ainsi, $d_r = 0$, et nous estimons $\hat{d}_r$ ainsi que $\sigma_{\mathbf{P}}$ avec nos trois méthodes. Nous commençons à l'étape $t = 0$ pour finir à l'étape $t = 99$. La figure 5.6 montre la valeur de nos estimateurs. En haut figure 5.6, $\hat{\mathbf{P}}$ est obtenu par filtrage comme décrit section 4. Ainsi, à chaque étape nous rassemblons de plus en plus d'informations et de mesures, ce qui explique la décroissance de $\sigma_{\mathbf{P}}$. En bas figure 5.6, nous reconstruisons l'estimé par moindre carré (section 3) à chaque étape. L'estimation est donc difficile (voir figure 5.5) aux étapes $t \in [\![0, 20]\!]$. Aux étapes $t \in [\![20, 25]\!]$ nous avons une faible dilution de précision, et nous pouvons obtenir beaucoup d'informations au vue de notre position par rapport à la quadrique, d'où une baisse de $\sigma_{\mathbf{P}}$. Aux étapes $t \in [\![25, 50]\!]$, nous observons qu'un côté de la quadrique, ce qui explique une nouvelle augmentation de $\sigma_{\mathbf{P}}$. Enfin, à partir de l'étape $t = 50$, nous passons du côté du point $\mathbf{p}$, et l'estimation de la distance atteint sa précision maximum.

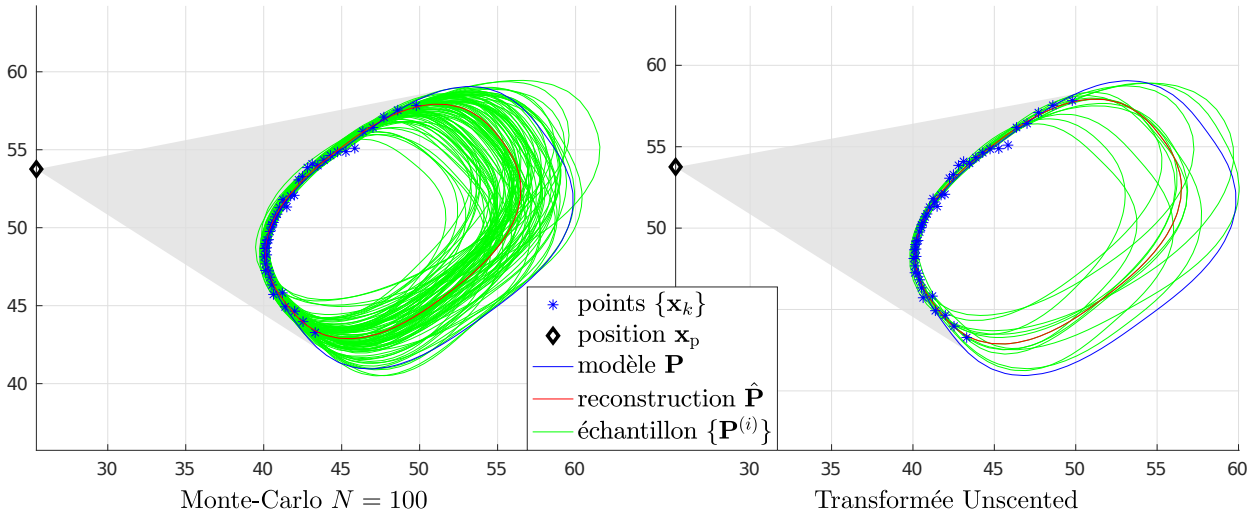


Figure 5.4 Illustration des échantillons pour la méthode de Monte-Carlo à gauche, et la transformée *Unscented* à droite.

Nous observons figure 5.6 que nos estimés renvoient des résultats équivalents pour les trois méthodes sur cet exemple. Pour la suite, nous choisirons l'estimé calculé avec la méthode *Unscented*.

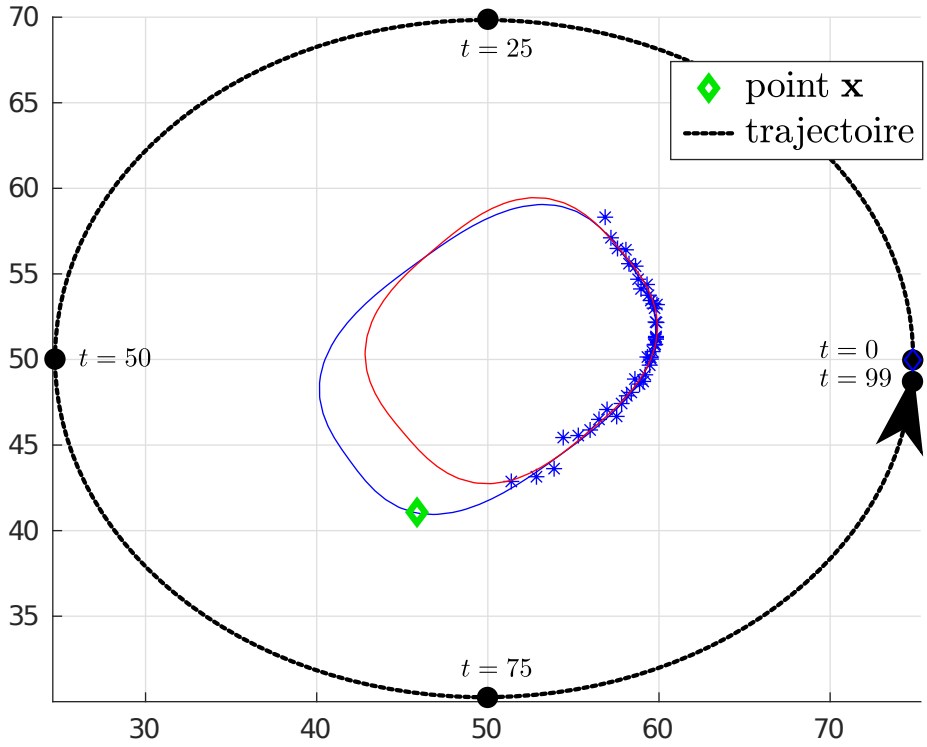


Figure 5.5 Illustration de l'estimation de la distance $\hat{d}_r$ du point $\mathbf{x}$ au modèle $\mathbf{P}$.

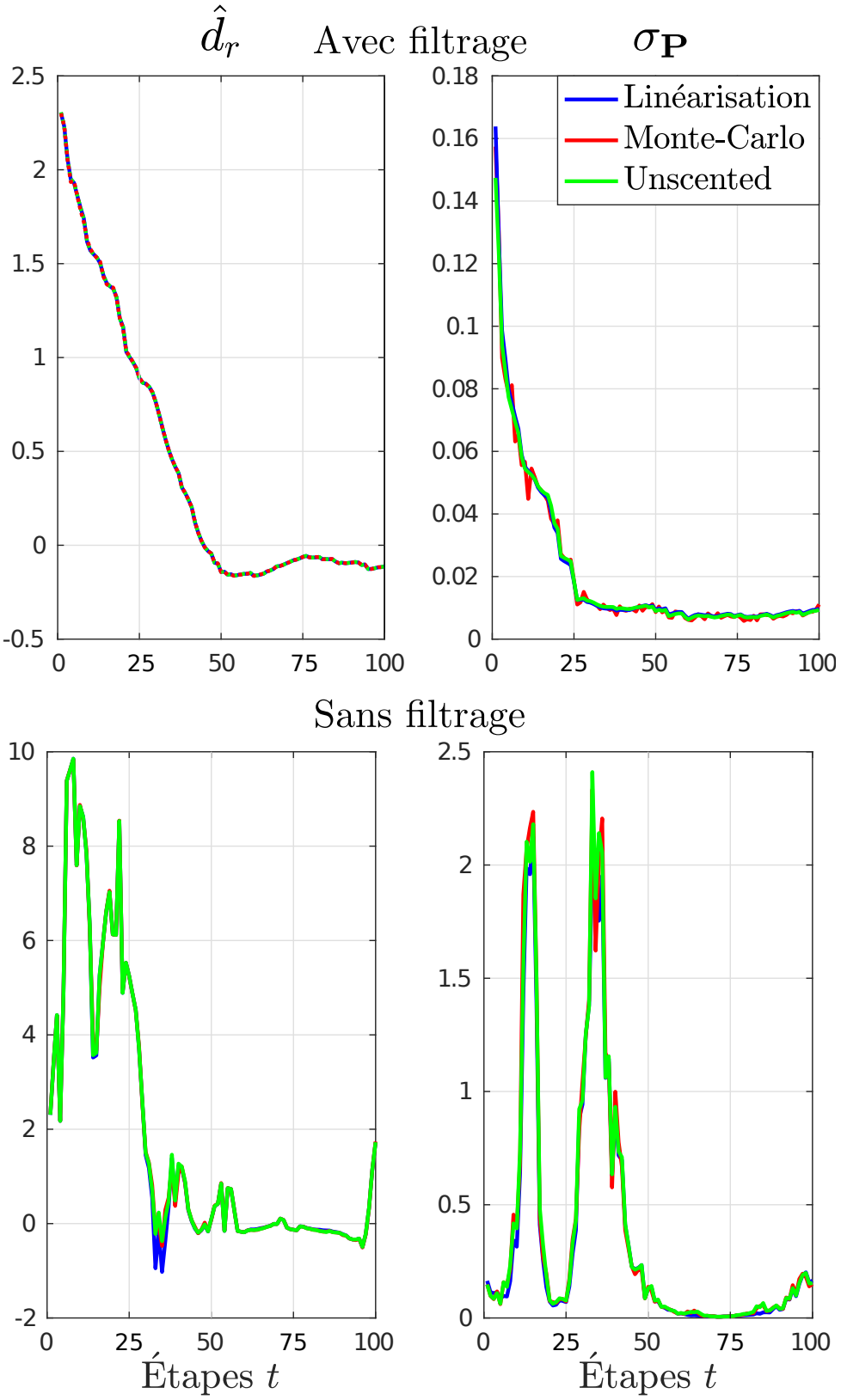


Figure 5.6 Distance estimée $\hat{d}_r$ du point $\mathbf{x}$ à $\mathbf{P}$ et erreur estimée sur cette distance σ_{d_r} pour chaque étape, avec chacune des trois méthodes.

5.3.3 Considération du bruit de mesure

Il ne faut pas oublier que le point $\mathbf{x}$ provient lui aussi d'une mesure bruitée, et nous pouvons linéariser l'expression de d_r autour de $\tilde{\mathbf{x}} = \mathbf{x} + \mathrm{d}\mathbf{x}$:

$$d_r(\mathbf{x}; \hat{\mathbf{P}}) \sim d_r(\tilde{\mathbf{x}}; \hat{\mathbf{P}}) + \mathbf{J}_\mathbf{x}\mathrm{d}\mathbf{x}, \tag{5.11}$$

avec $\mathbf{J}_\mathbf{x} = \left.\dfrac{\partial d_r(\mathbf{x}; \hat{\mathbf{P}})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}}$. De la même façon qu'à l'équation (5.5), nous avons :

$$\sigma^2_\mathbf{x} = \mathbf{J}_\mathbf{x}\mathbf{\Sigma}_\mathbf{x}\mathbf{J}^T_\mathbf{x}. \tag{5.12}$$

La matrice $\mathbf{\Sigma}_\mathbf{x}$ représente l'erreur sur l'estimation de $\mathbf{x}$ dans le référentiel de navigation. Nous avons, dans le cas bi-dimensionel :

$$\tilde{\mathbf{x}} = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_\mathrm{p} \\ y_\mathrm{p} \end{bmatrix} + \tilde{d} \begin{bmatrix} \cos(\psi_\mathrm{p} + \tilde{\lambda}) \\ \sin(\psi_\mathrm{p} + \tilde{\lambda}) \end{bmatrix}. \tag{5.13}$$

Nous rappelons que dans le problème de cartographie, la pose $\mathbf{x}_\mathrm{p}$ du robot est parfaitement connue. L'erreur sur l'estimation de $\mathbf{x}$ provient uniquement des bruits du capteur. Ces bruits peuvent être décrits grâce aux caractéristiques du capteur :

$$\tilde{\mathbf{z}} = \begin{bmatrix} \tilde{d} \\ \tilde{\lambda} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} d \\ \lambda \end{bmatrix}, \mathbf{R} \right), \tag{5.14}$$

et nous utilisons ceci pour calculer l'incertitude $\mathbf{\Sigma}_\mathbf{x}$ sur $\tilde{\mathbf{x}}$:

$$\mathbf{\Sigma}_\mathbf{x} = \mathbf{J}_\mathbf{z}\mathbf{R}\mathbf{J}^T_\mathbf{z}, \tag{5.15}$$

avec $\mathbf{J}_\mathbf{z} = \left.\dfrac{\partial \mathbf{x}(\mathbf{z}; \mathbf{x}_\mathrm{p})}{\partial \mathbf{z}}\right|_{\mathbf{z}=\tilde{\mathbf{z}}}$. Nous avons alors :

$$\sigma^2_\mathbf{x} = \mathbf{J}_\mathbf{x}\mathbf{J}_\mathbf{z}\mathbf{R}\mathbf{J}^T_\mathbf{z}\mathbf{J}^T_\mathbf{x}. \tag{5.16}$$

Finalement, en considérant les erreurs d'estimation sur $\mathbf{P}$ et sur $\mathbf{x}$ indépendantes, nous avons l'erreur sur l'estimation $\hat{d}_r$:

$$\sigma^2_{d_r} = \sigma^2_\mathbf{P} + \sigma^2_\mathbf{x}. \tag{5.17}$$

5.3.4 Association d'un point à une superquadrique

À présent, nous voulons associer des nouveaux points à nos estimations $\hat{\mathbf{P}}_m$ afin de pouvoir les mettre à jour. La figure 5.7 illustre notre situation, nous avons découvert un nouveau point $\tilde{\mathbf{x}}$ avec le LRF, et nous souhaitons savoir à quel modèle il appartient. Nous nous basons sur le travail fait précédemment avec la distance radiale. Pour chaque estimé $\hat{\mathbf{P}}_m$ nous allons calculer $\hat{d}_r^{(m)} = d_r(\tilde{\mathbf{x}}, \hat{\mathbf{P}}_m)$ et les erreurs $\sigma_{d_r}^{(m)}$ associées. Ensuite, l'association de données se fait en considérant les quantités suivantes :

$$\frac{1}{\sigma_{d_r}^{(m)}\sqrt{2\pi}} \exp^{-\frac{1}{2}\left(\frac{\hat{d}_r^{(m)}}{\sigma_{d_r}^{(m)}}\right)^2}. \tag{5.18}$$

Il s'agit de trouver le couple $(\hat{d}_r^{(m)}, \sigma_{d_r}^{(m)})$ correspondant à l'estimé m qui minimise cette quantité. En utilisant l'approche par maximum de vraisemblance, il s'agit de trouver l'estimé m qui maximise la quantité :

$$\left(\frac{\sigma_{d_r}^{(m)}}{\hat{d}_r^{(m)}}\right)^2, \tag{5.19}$$

et c'est ainsi que l'on associe finalement une mesure $\tilde{\mathbf{x}}$ à un estimé $\hat{\mathbf{P}}_m$.

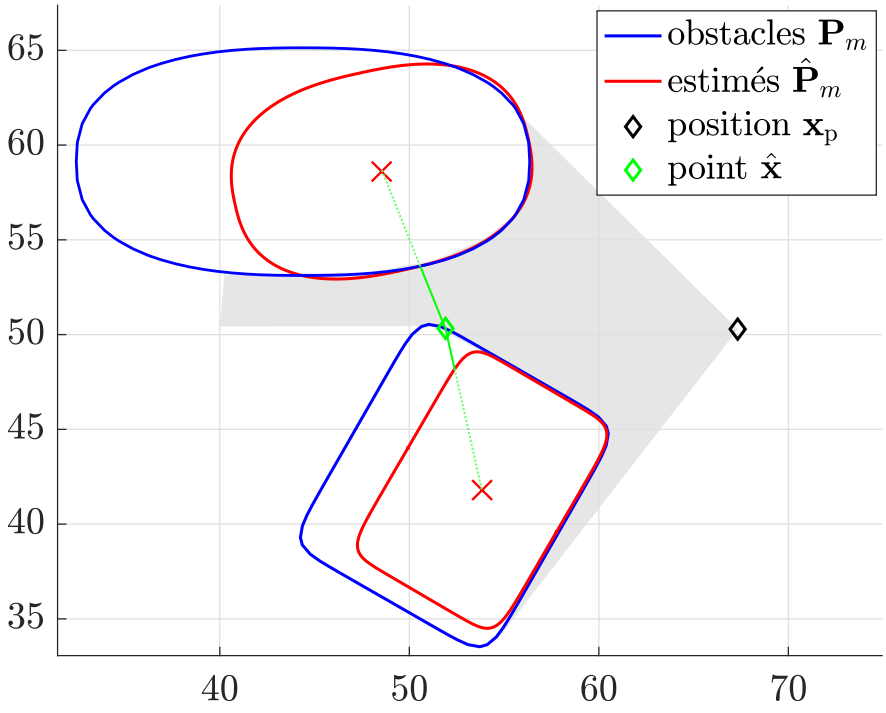


Figure 5.7 Illustration de l'association du point $\hat{\mathbf{x}}$ en vert à l'un des modèles estimés $\hat{\mathbf{P}}_m$.

5.3.5 Détection et création d'un nouvel obstacle

Certains points ne peuvent être associés à aucun modèle $\hat{\mathbf{P}}_m$ car ils sont trop loin (tous les $\hat{d}_r^{(m)}$ sont trop grands), ou nous considérons que la quantité de l'équation (5.19) est trop faible pour tous les estimés m. Dans ce cas, il faut envisager la création d'un nouvel obstacle. Nous procédons comme suit :

- Recueil des points $\hat{\mathbf{x}}_k$ et des mesures $\tilde{\mathbf{z}}_k$ du LRF qui n'ont été associés à aucun obstacle existant.
- Segmentation de ces points à partir des mesures $\tilde{\mathbf{z}}_k$ avec la technique de CCL (voir section 5.1.3).
- Création de nouveaux modèles $\hat{\mathbf{P}}_m$ par les moindres carrés pour chaque classe regroupant un nombre de mesures suffisant.

Ainsi nous créons et associons en même temps de nouveaux objets aux nouvelles mesures.

5.4 Fusion de superquadriques

Lors de l'association de données, il est fréquent de considérer comme nouveau un objet déjà observé précédemment. Ceci arrive avant la fermeture de boucle notamment, quand le robot n'a pas encore compris qu'il revenait dans une zone déjà cartographiée (voir étape 2, figure 5.8). Ainsi, pour le même obstacle $\mathbf{P}$, nous nous retrouvons avec deux estimés $\hat{\mathbf{P}}_1$ et $\hat{\mathbf{P}}_2$. Le but est alors de garder qu'un seul estimé $\hat{\mathbf{P}}$. La figure 5.8 illustre la démarche que nous voudrions implémenter.

5.4.1 Critère de fusion

Il faut d'abord pouvoir détecter que les deux estimés appartiennent bien au même obstacle. Pour ceci nous nous basons sur un critère d'intersection. Si deux SQs $\mathbf{P}_1$ et $\mathbf{P}_2$ s'intersectent, alors celles-ci appartiennent au même obstacle. Le problème revient à savoir s'il existe un point $\mathbf{x}^* \in \mathbb{R}^3$ à l'intérieur (ou sur la surface) de chacune des SQs :

$$F(\mathbf{x}^*; \mathbf{P}_1) \leq 1, \ F(\mathbf{x}^*; \mathbf{P}_2) \leq 1. \tag{5.20}$$

La recherche de l'existence d'une telle solution est équivalente à trouver $\mathbf{x}^* \in \mathbb{R}^3$ telle que :

$$F(\mathbf{x}^*; \mathbf{P}_1) = F(\mathbf{x}^*; \mathbf{P}_2) = 1. \tag{5.21}$$

Si un tel point s'avérait exister, alors nous pourrions engager le processus de fusion. Mais, il

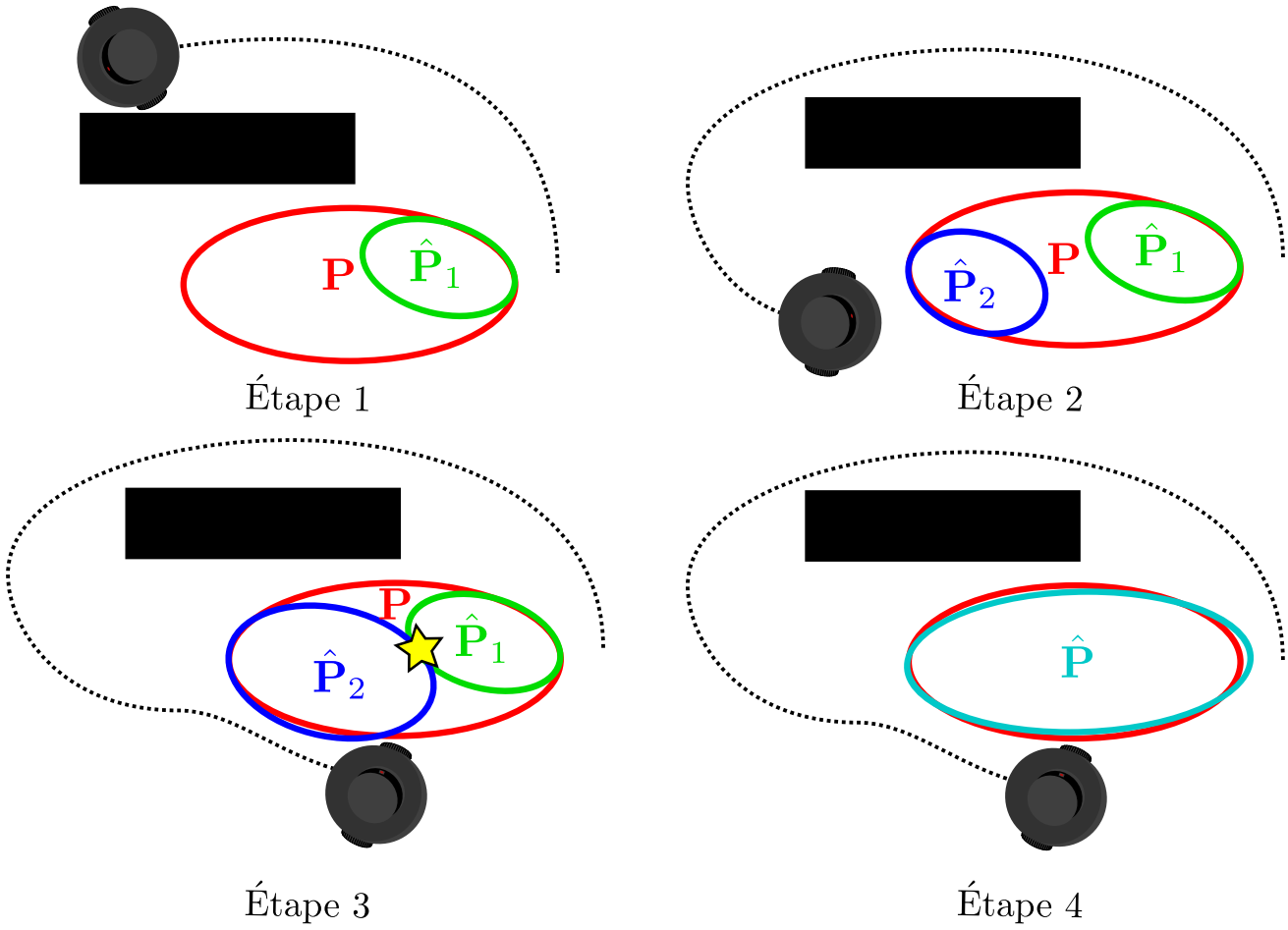


Figure 5.8 Illustration du processus de fusion de deux SQs estimés.

n'existe pas analytiquement de solution à ces équations. Plutôt que d'appeler systématiquement un solveur, nous avons développé toute une démarche à suivre permettant de savoir le plus rapidement possible si deux SQs s'intersectent. Cette démarche est schématisée figure 5.10. Elle comprend des tests rapidement vérifiables détaillés dans les paragraphes suivants.

Distances entre SQs. Ce premier test est le plus simple à vérifier. Nous traçons deux sphères autour de chacune des SQs et testons leur intersection. Si nous avons

$$||\mathbf{x}_{c,1} - \mathbf{x}_{c,2}||_2 > \sqrt{a_1^2 + b_1^2 + c_1^2} + \sqrt{a_2^2 + b_2^2 + c_2^2}, \tag{5.22}$$

alors nos sphères ne s'intersectent pas, tout comme nos SQs car celles-ci sont inclues dans leur sphere respective.

Intersection d'ellipsoïdes. Les SQs vérifiant $0 \leq \epsilon_1, \epsilon_2 \leq 1$ contiennent leur ellipsoïde tandis que celles vérifiant $1 \leq \epsilon_1, \epsilon_2 \leq 2$ sont inclues dans leur ellipsoïde (voir [57] pour la démonstration). Ainsi, l'étude de l'intersection d'ellipsoïdes peut fournir une condition nécessaire et/ou suffisante quant à l'intersection ou non de deux SQs. Une ellipsoïde peut être représentée dans l'espace par 9 paramètres (deux de moins que pour les SQs, et nous faisons l'analogie avec le vecteur $\mathbf{P}$). On en compte trois pour sa taille, trois pour sa position et trois pour son orientation. Voici l'équation d'une ellipsoïde, caractérisée par la fonction Q à l'instar de la fonction F pour les SQs :

$$Q(\mathbf{x}; \mathbf{P}) = \mathbf{x}^T \mathbf{A} \mathbf{x} + \mathbf{b}^T \mathbf{x} + c = 1, \tag{5.23}$$

avec $\mathbf{A}$ symétrique positive semi-définie. Afin de valider si deux ellipsoïdes s'intersectent, nous nous ramenons au problème d'optimisation suivant, décrit par [58] :

$$\begin{cases} \min\limits_{\mathbf{x}} & Q(\mathbf{x}; \mathbf{P}_1), \\ \text{s.l.c.} & Q(\mathbf{x}; \mathbf{P}_2) - 1 = 0. \end{cases} \tag{5.24}$$

Les courbes de différents niveaux $Q(\mathbf{x}; \mathbf{P}) = \lambda$ représentent toutes des ellipsoïdes. $\lambda = 1$ signifie que $\mathbf{x}$ est sur la surface de l'ellipsoïde, $\lambda > 1$ et $\mathbf{x}$ est en dehors de l'ellipsoïde, $\lambda \in [0, 1]$ et $\mathbf{x}$ est dedans. Si $\lambda = 0$ alors $\mathbf{x} = \mathbf{x}_c$ ($\mathbf{x}_c$ est le centre de $\mathbf{P}$). La résolution du problème d'optimisation (5.24) va mener à trouver les points $\mathbf{x}^*$ de $\mathbf{P}_2$ pour lesquels nous avons des λ critiques vérifiant $Q(\mathbf{x}^*; \mathbf{P}_1) = \lambda$. Il s'agira ensuite de choisir parmi ces λ le plus grand, λ_g, et le plus petit, λ_p. Trois cas de figure se présentent :

- $0 \leq \lambda_p \leq \lambda_g$, alors nos ellipsoïdes sont distinctes,
- $\lambda_p \leq 0 \leq \lambda_g$, alors nos ellipsoïdes s'intersectent,
- $\lambda_p \leq \lambda_g \leq 0$, alors une ellipsoïde est inclue dans l'autre.

Nous donnons un exemple en 2D figure 5.9, où nous avons trois points critiques. Le point $\mathbf{x}_1^*$ correspond à un $\lambda_1 < 0$, tandis que les points $\mathbf{x}_2^*$ et $\mathbf{x}_3^*$ vérifient $0 < \lambda_2 < \lambda_3$. On a dans ce cas $\lambda_p = \lambda_1$ et $\lambda_g = \lambda_3$ et nous nous retrouvons dans la configuration $\lambda_p \leq 0 \leq \lambda_g$: nos ellipses s'intersectent.

Le problème (5.24) est résolu par la méthode des multiplicateurs de Lagrange, en posant :

$$\mathcal{L}(\mathbf{x}, t; \mathbf{P}_1, \mathbf{P}_2) = Q(\mathbf{x}; \mathbf{P}_1) + t(Q(\mathbf{x}; \mathbf{P}_2) - 1), \tag{5.25}$$

nous devons résoudre :

$$\begin{cases} \nabla\mathcal{L}(\mathbf{x}, t; \mathbf{P}_1, \mathbf{P}_2) & = 0, \\ Q(\mathbf{x}; \mathbf{P}_2) - 1 & = 0. \end{cases} \tag{5.26}$$

La résolution se fait comme suit :

$$\nabla\mathcal{L}(\mathbf{x}, t; \mathbf{P}_1, \mathbf{P}_2) = \nabla Q(\mathbf{x}; \mathbf{P}_1) + t\nabla Q(\mathbf{x}; \mathbf{P}_2) = 2(\mathbf{A}_1 + t\mathbf{A}_2)\mathbf{x} + \mathbf{b}_1 + t\mathbf{b}_2 = 0. \tag{5.27}$$

Nous isolons $\mathbf{x}$:

$$\mathbf{x} = -(\mathbf{A}_1 + t\mathbf{A}_2)^{-1}(\mathbf{b}_1 + t\mathbf{b}_2)/2 = \frac{1}{\delta(t)}\mathbf{Y}(t). \tag{5.28}$$

On identifie $\delta(t)$ le déterminant de $(\mathbf{A}_1 + t\mathbf{A}_2)$, et $\mathbf{Y}(t)$ une matrice dépendante de t. Puis nous injectons dans la contrainte d'égalité de telle sorte :

$$\mathbf{Y}(t)^T\mathbf{A}_2\mathbf{Y}(t) + \delta(t)\mathbf{b}_2^T\mathbf{Y}(t) + \delta(t)^2 c_2 - 1 = 0. \tag{5.29}$$

Ceci est en fait un polynôme de degré 6, il suffit de trouver ses racines t^*, d'où découlent les $\mathbf{x}^*$, puis nous calculons les $\lambda = Q(\mathbf{x}^*; \mathbf{P}_1)$, pour ne garder que les λ extrémaux et ainsi vérifier dans quel cas de figure nous sommes.

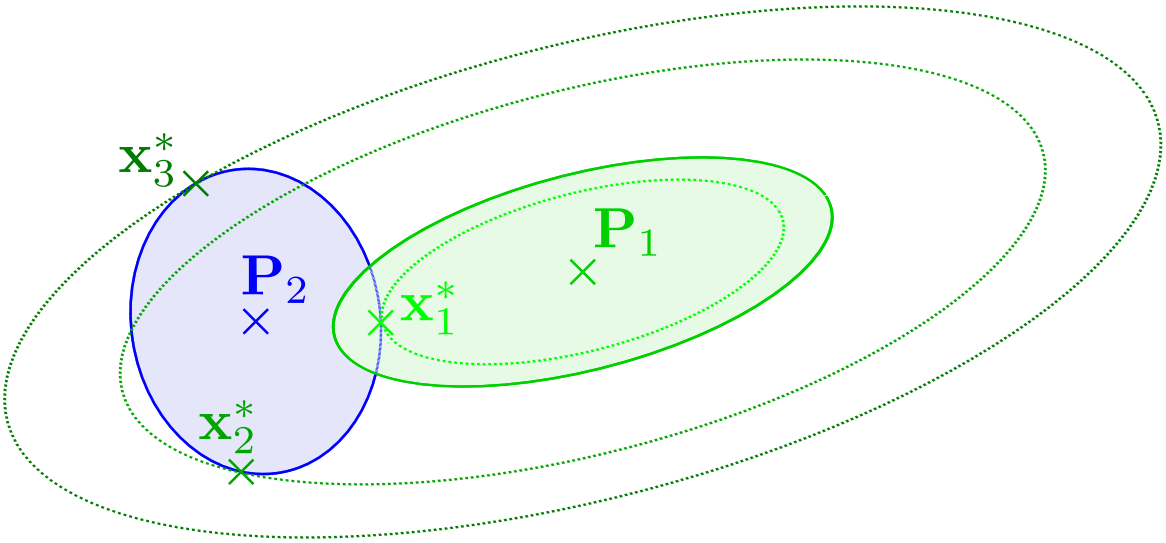


Figure 5.9 Illustration des points critiques dans le problème d'optimisation (5.24).

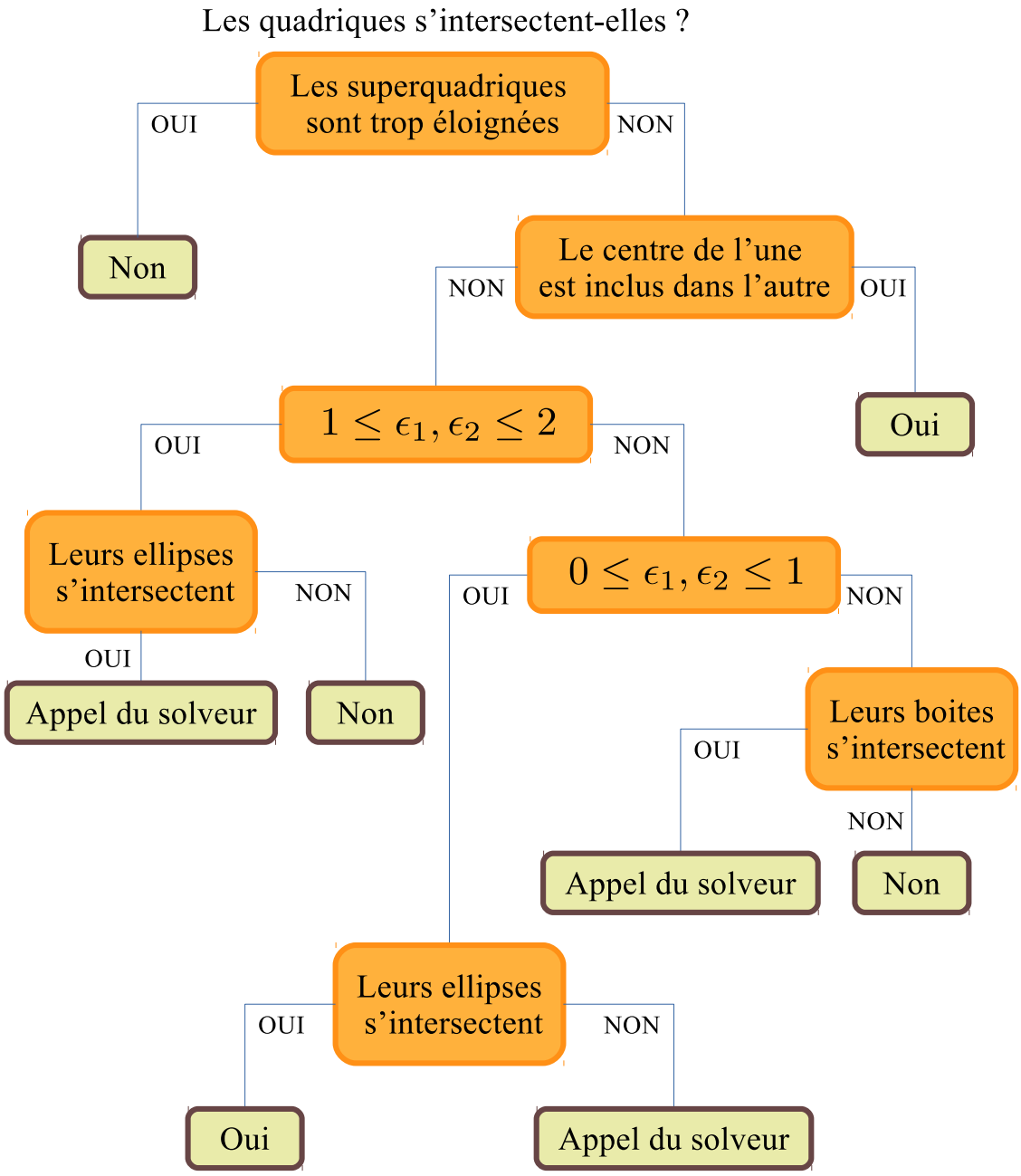


Figure 5.10 Algorithme permettant de tester si deux SQs s'intersectent.

5.4.2 Fusion de superquadriques

Une fois le critère d'intersection validé entre les SQs, nous entamons le processus de fusion. La méthode décrite est simple et peut fusionner autant de SQs que voulu simultanément (méthode non réduite à deux SQs). Le principe part de la génération de points autour de chaque volume. Le nuage de points créé est homogène selon les coordonnées sphériques, centré sur le centre de chaque SQ. Ensuite nous calculons l'enveloppe convexe de ces points. Tous les points strictement à l'intérieur de cette enveloppe ne sont plus considérés, et nous effectuons finalement une regression par moindre carré sur les points restant (voir section 3). Le volume résultant devient alors la fusion des SQs précédentes. La figure 5.11 montre l'intérêt de calculer l'enveloppe convexe. En effet, le volume obtenu après fusion en magenta correspond mieux à nos attentes lorsque l'on considère l'enveloppe (en bas à droite) plutôt que lorsqu'on ne la considère pas (en haut à droite).

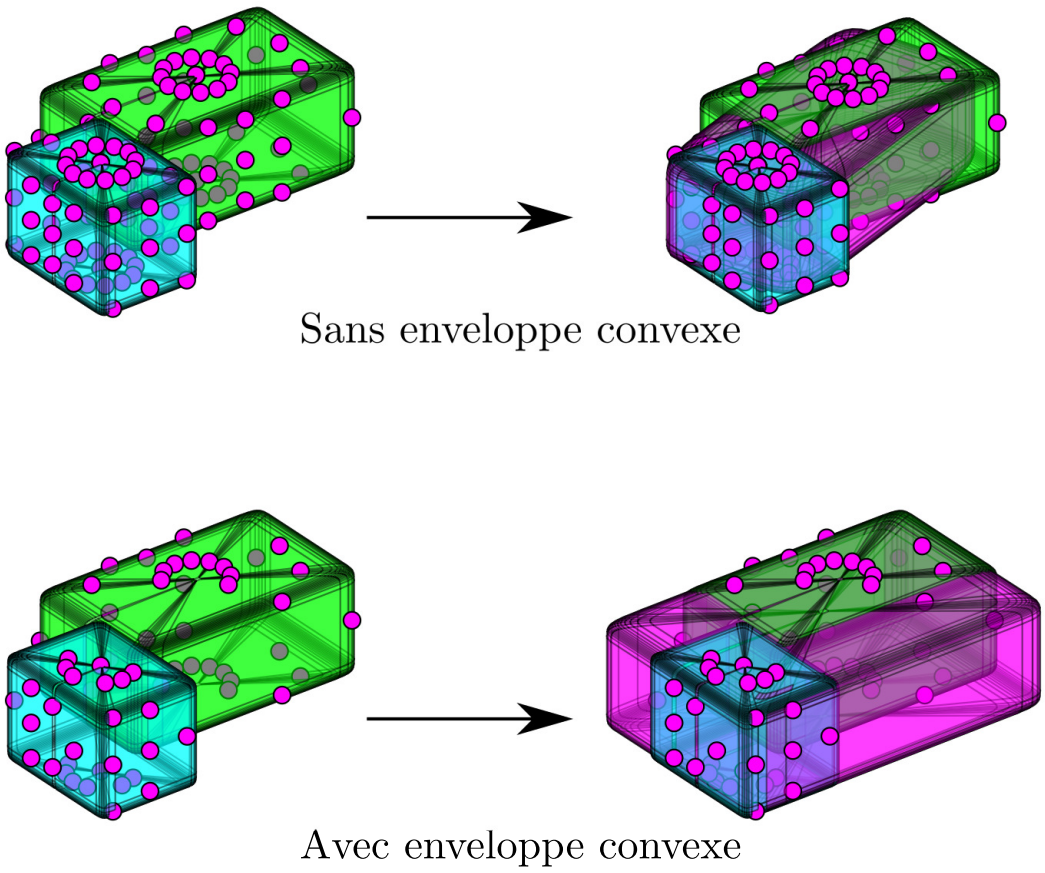


Figure 5.11 Fusion des deux SQs bleue et verte. En haut nous utilisons tous les points (en magenta), en bas nous calculons l'enveloppe convexe. Le résultat de la fusion est la figure magenta à droite.

5.5 Simulation

L'algorithme de cartographie est schématisé figure 5.12. Nous simulons l'algorithme dans `Matlab`.

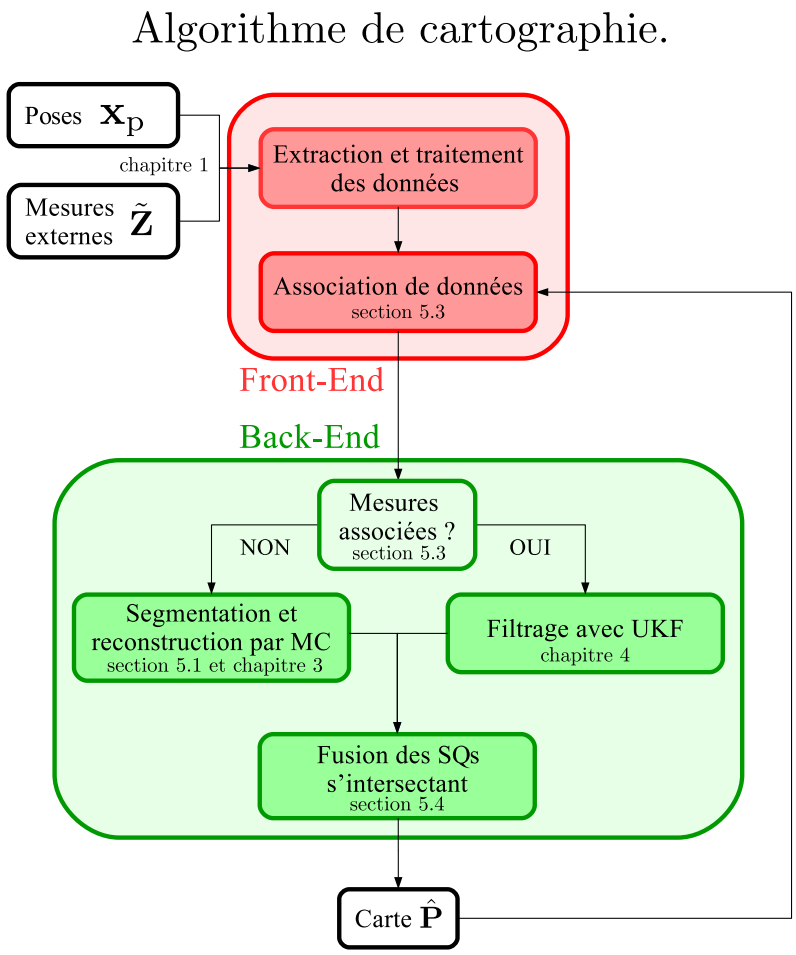


Figure 5.12 Algorithme de cartographie.

La figure 5.13 montre l'allure d'une carte reconstituée pour différentes étapes. En un tour la reconstitution se passe bien quand les obstacles restent cantonnés à l'intérieur de la trajectoire du robot.

Des difficultés apparaissent quand les objets sont proches les uns des autres (deux objets peuvent être considérés comme un seul), ou lorsqu'ils sont trop partiellement observés. La figure 5.14 montre une reconstitution qui a mal fonctionnée, à cause des deux carrés trop proches et qui ont été "fusionnés" par l'algorithme.

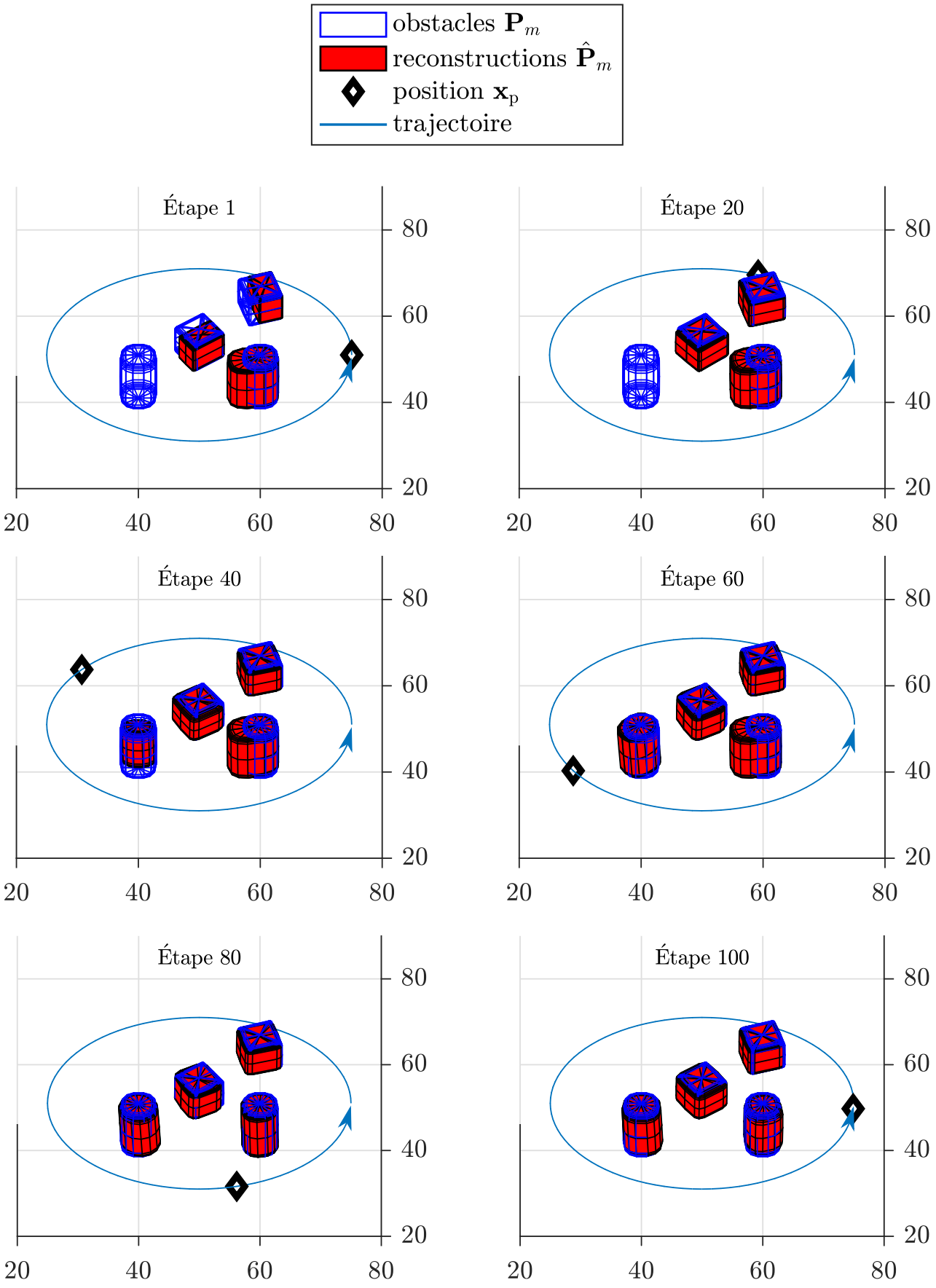


Figure 5.13 Cartographie d'obstacles en forme de superquadrique.

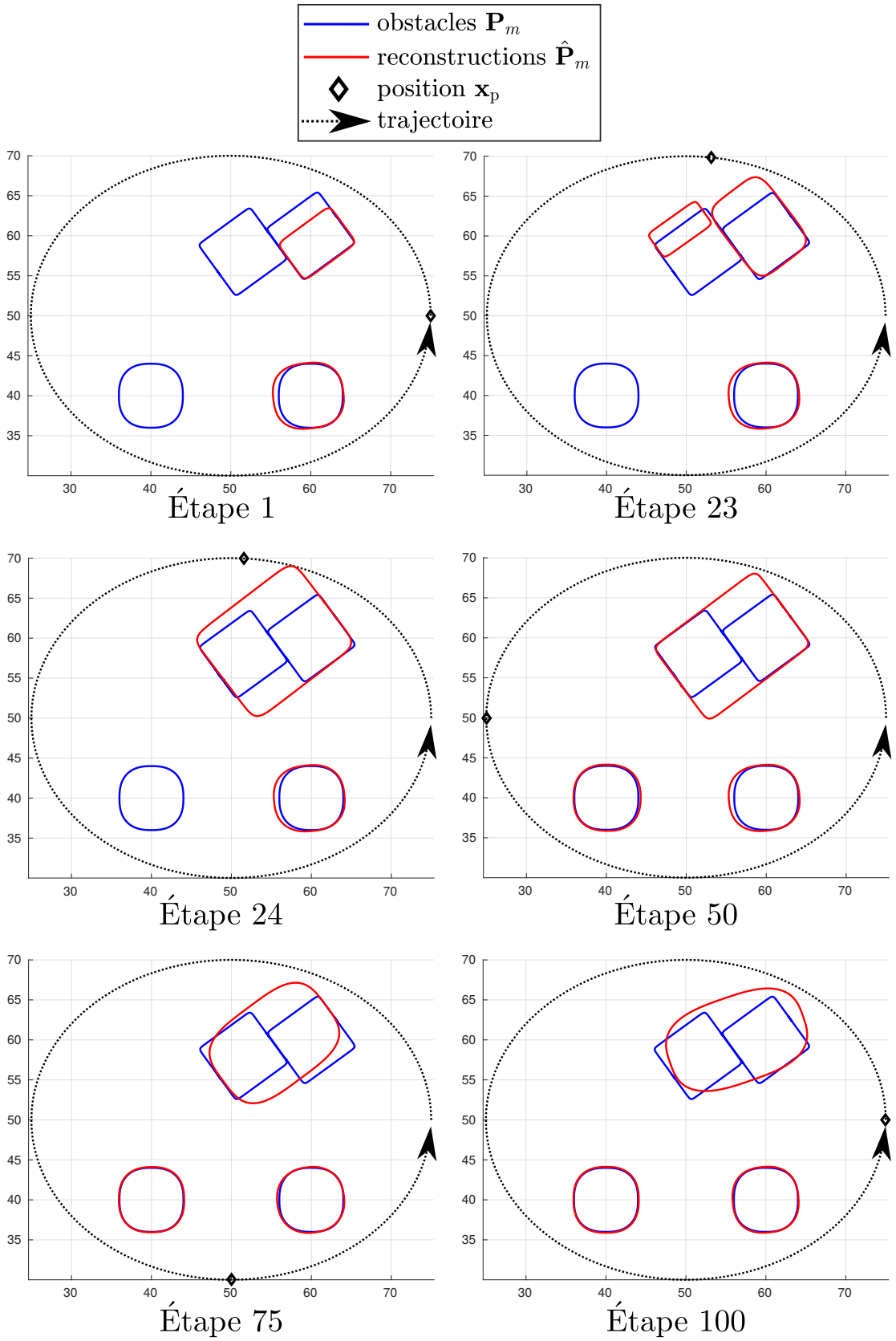


Figure 5.14 Cartographie d'obstacles rapprochés en forme de superellipse.

CHAPITRE 6 POSITIONNEMENT

Dans ce chapitre nous abordons le problème de localisation du robot. Nous assumons alors connaître la carte de notre environnement. Dans la première partie section 6.1 nous traitons de la localisation à partir de marqueurs ponctuels connus dans l'environnement. Les techniques abordées serviront pour la localisation à partir de SQs, que nous verrons dans la seconde partie section 6.2.

6.1 Localisation avec des points

Nous avons un environnement composé de marqueurs ponctuels différenciables. La carte de cet environnement est connue par le robot. Le but est de localiser le robot à partir de ces marqueurs à l'aide d'observations faites par le LRF. De plus le robot dispose d'encodeurs sur ses roues. Nous allons utiliser un algorithme de filtrage pour localiser notre robot.

6.1.1 Modèle de propagation

L'état qui nous intéresse $\mathbf{x}_\mathrm{p}$ concerne la position du robot, ainsi que son orientation dans le référentiel de navigation. Dans cette section nous notons la pose du robot $\mathbf{X} = \mathbf{x}_\mathrm{p} = [x_\mathrm{p}\ y_\mathrm{p}\ \psi_\mathrm{p}]^T$. À chaque étape $t \in [\![0, T-1]\!]$, nous utilisons l'odométrie $\mathbf{U}_t = [u_{\mathrm{g},t}\ u_{\mathrm{d},t}]^T$ comme mesure et commande interne discrète du robot, qui représente les distances parcourues par la roue gauche et la roue droite entre les étapes t et $t+1$. La variable L représente la distance entre les centres des deux roues du robot.

$$\mathbf{X}_{t+1} = \mathbf{f}_t(\mathbf{X}_t, \tilde{\mathbf{U}}_t, \mathbf{\Xi}_t), \tag{6.1}$$

$$\begin{bmatrix} x_{\mathrm{p},t+1} \\ y_{\mathrm{p},t+1} \\ \psi_{\mathrm{p},t+1} \end{bmatrix} = \begin{bmatrix} x_{\mathrm{p},t} \\ y_{\mathrm{p},t} \\ \psi_{\mathrm{p},t} \end{bmatrix} + \begin{bmatrix} \frac{\tilde{u}_{\mathrm{g},t} + \tilde{u}_{\mathrm{d},t}}{2} \cos\left(\psi_{\mathrm{p},t} + \frac{\tilde{u}_{\mathrm{d},t} - \tilde{u}_{\mathrm{g},t}}{L}\right) \\ \frac{\tilde{u}_{\mathrm{g},t} + \tilde{u}_{\mathrm{d},t}}{2} \sin\left(\psi_{\mathrm{p},t} + \frac{\tilde{u}_{\mathrm{d},t} - \tilde{u}_{\mathrm{g},t}}{L}\right) \\ \frac{\tilde{u}_{\mathrm{d},t} - \tilde{u}_{\mathrm{g},t}}{L} \end{bmatrix} + \begin{bmatrix} \xi_{x,t} \\ \xi_{y,t} \\ \xi_{\psi,t} \end{bmatrix}. \tag{6.2}$$

Nous définissons :

$$\tilde{\mathbf{U}}_t = \begin{bmatrix} \tilde{u}_{\mathrm{g},t} \\ \tilde{u}_{\mathrm{d},t} \end{bmatrix} = \mathbf{U}_t + \mathbf{W}_t = \begin{bmatrix} u_{\mathrm{g},t} \\ u_{\mathrm{d},t} \end{bmatrix} + \begin{bmatrix} w_{\mathrm{g},t} \\ w_{\mathrm{d},t} \end{bmatrix}, \tag{6.3}$$

avec les processus à temps discret $\{w_{\mathrm{g},t}\}$ et $\{w_{\mathrm{d},t}\}$, $t \in [\![0, T-1]\!]$ blancs, gaussiens et

indépendants entre eux, de même variance σ_w^2. L'introduction de ces processus nous permet de modéliser les bruits et imprécisions des encodeurs sur les deux roues, ils rendent compte notamment de l'hypothèse de roulement sans glissement qui n'est jamais respectée en réalité, tout comme les incertitudes sur L et le rayon des roues.

De même nous introduisons $\mathbf{\Xi}_t = \begin{bmatrix} \xi_{x,t} \\ \xi_{y,t} \\ \xi_{\psi,t} \end{bmatrix}$, de variances respectives $\begin{bmatrix} \sigma_x^2 \\ \sigma_y^2 \\ \sigma_\psi^2 \end{bmatrix}$. Ces bruits sont utiles pour rendre compte de l'erreur de modélisation de la dynamique. En effet, notre modèle d'odométrie n'est qu'approximatif, car il prend pour hypothèse qu'entre deux étapes successives la vitesse de rotation du robot est constante, ce qui ne sera pas le cas dans notre simulation.

Par ailleurs, nous remarquons dans la dynamique que l'orientation ψ_p du robot peut sortir de l'intervalle $[-\pi, \pi]$ si notre robot fait plusieurs tours sur lui-même. Ce n'est pas un problème ici, mais dans l'étape de mise à jour où nous allons comparer des angles, nous devons conserver l'unicité de la représentation. Ainsi, nous faisons attention et nous utilisons l'opération modulo (mod) : $\psi \leftarrow \mathrm{mod}(\psi + \pi, 2\pi) - \pi$ afin de conserver nos angles dans l'intervalle $[-\pi, \pi]$.

6.1.2 Modèle des mesures externes

Les mesures externes proviennent du LRF déjà utilisé précédemment pour la cartographie. Les mesures se présentent sous cette forme : $\mathbf{Y}_t = [\tilde{d}_{1,t}\ \tilde{\lambda}_{1,t}\ \tilde{\phi}_{1,t}\ \tilde{d}_{2,t}\ \tilde{\lambda}_{2,t}\ \tilde{\phi}_{2,t}\ \ldots\ \tilde{d}_{K_t,t}\ \tilde{\lambda}_{K_t,t}\ \tilde{\phi}_{K_t,t}]^T$, avec K_t le nombre de détections faites à l'étape t par le capteur (lors d'un scan complet).

$$\mathbf{Y}_t = \mathbf{g}_t(\mathbf{X}_t, \mathcal{V}_t), \tag{6.4}$$

$$\begin{bmatrix} \tilde{d}_{1,t} \\ \tilde{\lambda}_{1,t} \\ \tilde{\phi}_{1,t} \\ \vdots \\ \tilde{d}_{K_t,t} \\ \tilde{\lambda}_{K_t,t} \\ \tilde{\phi}_{1,K_t} \end{bmatrix} = \begin{bmatrix} \sqrt{(x_{\mathrm{p},t} - x_{1,t})^2 + (y_{\mathrm{p},t} - y_{1,t})^2} \\ \mathrm{atan2}(y_{1,t} - y_{\mathrm{p},t}, x_{1,t} - x_{\mathrm{p},t}) - \psi_{\mathrm{p},t} \\ \mathrm{asin}\left(\dfrac{z_{1,t}}{\sqrt{(x_{\mathrm{p},t} - x_{1,t})^2 + (y_{\mathrm{p},t} - y_{1,t})^2}} \right) \\ \vdots \\ \sqrt{(x_{\mathrm{p},t} - x_{K_t,t})^2 + (y_{\mathrm{p},t} - y_{K_t,t})^2} \\ \mathrm{atan2}(y_{K_t,t} - y_{\mathrm{p},t}, x_{K_t,t} - x_{\mathrm{p},t}) - \psi_{\mathrm{p},t} \\ \mathrm{asin}\left(\dfrac{z_{K_t,t}}{\sqrt{(x_{\mathrm{p},t} - x_{K_t,t})^2 + (y_{\mathrm{p},t} - y_{K_t,t})^2}} \right) \end{bmatrix} + \begin{bmatrix} \nu_{d,1,t} \\ \nu_{\lambda,1,t} \\ \nu_{\phi,1,t} \\ \vdots \\ \nu_{d,K_t,t} \\ \nu_{\lambda,K_t,t} \\ \nu_{\phi,K_t,t} \end{bmatrix}. \tag{6.5}$$

Le vecteur $[x_{k,t}\ y_{k,t}\ z_{k,t}], k \in [\![1, K_t]\!]$ représente la position du marqueur associé à la détection

k lors de l'étape t. Un marqueur peut ne pas avoir été repéré (hors de portée), repéré une fois, ou même plusieurs fois par le LRF.

Les processus à temps discret $\{\mathcal{V}_t\} = \left\{ \begin{bmatrix} \nu_{d,k,t} \\ \nu_{\lambda,k,t} \\ \nu_{\phi,k,t} \end{bmatrix} \right\}$, $k \in [\![1, K_t]\!]$, $t \in [\![0, T]\!]$, sont blancs, gaussiens et indépendants entre eux, de variances respectives σ_d^2, σ_λ^2 et σ_ϕ^2. Ces processus permettent de rendre compte des bruits de mesure du LRF.

6.1.3 Linéarisation des équations d'état

Le modèle de propagation traduit les informations provenant des odomètres, tandis que notre modèle de mesures externes rend compte des informations de l'environnement. Afin de pouvoir combiner aux mieux ces deux sources d'informations, nous décidons d'utiliser un filtrage de Kalman qui va se charger de fusionner les données des capteurs. Nous utilisons un Filtre de Kalman Étendu (EKF), et il faut alors en premier linéariser le système d'équations :

$$\begin{cases} \mathbf{X}_{t+1} = \mathbf{f}_t(\mathbf{X}_t, \tilde{\mathbf{U}}_t, \mathbf{\Xi}_t), \\ \mathbf{Y}_t = \mathbf{g}_t(\mathbf{X_t}, \mathcal{V}_t). \end{cases} \tag{6.6}$$

Nous transformons ce système avec les notations suivantes :

$$\begin{cases} \mathbf{X}_{t+1} = \mathbf{A}_t\mathbf{X}_t + \mathbf{B}_t\tilde{\mathbf{U}}_t + \mathbf{L}_t\mathbf{\Xi}_t, \\ \mathbf{Y}_t = \mathbf{C}_t\mathbf{X}_t + \mathbf{S}_t\mathcal{V}_t. \end{cases} \tag{6.7}$$

La linéarisation se fait autour de l'état estimé du système $\hat{\mathbf{X}}_t$, tel que $\mathbf{X}_t = \hat{\mathbf{X}}_t + \mathrm{d}\mathbf{X}$, en supposant, $\|\mathrm{d}\mathbf{X}\|$ suffisamment petit. La linéarisation est une simplification de nos équations d'état, rendant le modèle un peu plus inexacte. Le bruit additif représenté par $\mathbf{\Xi}_t$ tente de rendre compte aussi de ces erreurs de linéarisation.

Nous notons : $\mathbf{A}_t = \dfrac{\partial \mathbf{f}_t}{\partial \mathbf{X}_t}\left(\hat{\mathbf{X}}_t, \tilde{\mathbf{U}}_t, \mathbf{0}\right)$, $\mathbf{B}_t = \dfrac{\partial \mathbf{f}_t}{\partial \mathbf{U}_t}\left(\hat{\mathbf{X}}_t, \tilde{\mathbf{U}}_t, \mathbf{0}\right)$, $\mathbf{C}_t = \dfrac{\partial \mathbf{g}_t}{\partial \mathbf{Y}_t}\left(\hat{\mathbf{X}}_t, \mathbf{0}\right)$, $\mathbf{L}_t = \dfrac{\partial \mathbf{f}_t}{\partial \mathbf{\Xi}_t}\left(\hat{\mathbf{X}}_t, \tilde{\mathbf{U}}_t, \mathbf{0}\right) = \mathbf{I}_3$, $\mathbf{S}_t = \dfrac{\partial \mathbf{g}_t}{\partial \mathcal{V}_t}\left(\hat{\mathbf{X}}_t, \mathbf{0}\right) = \mathbf{I}_{3K_t}$, $\hat{\Psi} = \hat{\psi}_{\mathrm{p},t} + \dfrac{\tilde{u}_{\mathrm{d},t} - \tilde{u}_{\mathrm{g},t}}{L}$.

$$\mathbf{A}_t = \begin{bmatrix} 1 & 0 & -\dfrac{\tilde{u}_{\mathrm{g},t} + \tilde{u}_{\mathrm{d},t}}{2}\sin\hat{\Psi} \\ 0 & 1 & \dfrac{\tilde{u}_{\mathrm{g},t} + \tilde{u}_{\mathrm{d},t}}{2}\cos\hat{\Psi} \\ 0 & 0 & 1 \end{bmatrix},$$

$$\mathbf{B}_t = \frac{1}{2}\begin{bmatrix} \cos\hat{\Psi} + \dfrac{\tilde{u}_{\mathrm{g},t} + \tilde{u}_{\mathrm{d},t}}{L}\sin\hat{\Psi} & \cos\hat{\Psi} - \dfrac{\tilde{u}_{\mathrm{g},t} + \tilde{u}_{\mathrm{d},t}}{L}\sin\hat{\Psi} \\ \sin\hat{\Psi} - \dfrac{\tilde{u}_{\mathrm{g},t} + \tilde{u}_{\mathrm{d},t}}{L}\cos\hat{\Psi} & \sin\hat{\Psi} + \dfrac{\tilde{u}_{\mathrm{g},t} + \tilde{u}_{\mathrm{d},t}}{L}\cos\hat{\Psi} \\ -2/L & 2/L \end{bmatrix},$$

$$\mathbf{C}_t = \begin{bmatrix} \dfrac{\hat{x}_{\mathrm{p},t} - x_{1,t}}{\hat{d}_{1,t}} & \dfrac{\hat{y}_{\mathrm{p},t} - y_{1,t}}{\hat{d}_{1,t}} & 0 \\ -\dfrac{\hat{y}_{\mathrm{p},t} - y_{1,t}}{\hat{d}_{1,t}^2} & \dfrac{\hat{x}_{\mathrm{p},t} - x_{1,t}}{\hat{d}_{1,t}^2} & -1 \\ -\dfrac{\sqrt{z_{1,t}}(\hat{x}_{\mathrm{p},t} - x_{1,t})}{\hat{d}_{1,t}^2\sqrt{\hat{d}_{1,t} - z_{1,t}}} & -\dfrac{\sqrt{z_{1,t}}(\hat{y}_{\mathrm{p},t} - y_{1,t})}{\hat{d}_{1,t}^2\sqrt{\hat{d}_{1,t} - z_{1,t}}} & 0 \\ \vdots & \vdots & \vdots \\ \dfrac{\hat{x}_{\mathrm{p},t} - x_{K_t,t}}{\hat{d}_{K_t,t}} & \dfrac{\hat{y}_{\mathrm{p},t} - y_{K_t,t}}{\hat{d}_{K_t,t}} & 0 \\ -\dfrac{\hat{y}_{\mathrm{p},t} - y_{K_t,t}}{\hat{d}_{K_t,t}^2} & \dfrac{\hat{x}_{\mathrm{p},t} - x_{K_t,t}}{\hat{d}_{K_t,t}^2} & -1 \\ -\dfrac{\sqrt{z_{K_t,t}}(\hat{x}_{\mathrm{p},t} - x_{K_t,t})}{\hat{d}_{K_t,t}^2\sqrt{\hat{d}_{K_t,t} - z_{K_t,t}}} & -\dfrac{\sqrt{z_{K_t,t}}(\hat{y}_{\mathrm{p},t} - y_{K_t,t})}{\hat{d}_{K_t,t}^2\sqrt{\hat{d}_{K_t,t} - z_{K_t,t}}} & 0 \end{bmatrix}, \ \hat{d}_{k,t} = \sqrt{(\hat{x}_{\mathrm{p},t} - x_{k,t})^2 + (\hat{y}_{\mathrm{p},t} - y_{k,t})^2 + z_{k,t}^2},$$

$$\mathbf{H} = \sigma_w^2\mathcal{I}_2, \ \mathbf{Q} = \mathrm{diag}(\sigma_x^2, \sigma_y^2, \sigma_\psi^2), \ \mathbf{R}_t = \mathrm{diag}(\left\{(\sigma_d^2, \sigma_\lambda^2, \sigma_\phi^2)\right\} k \in [\![1, K_t]\!]). \tag{6.8}$$

6.1.4 Fusion des capteurs

Nous utilisons maintenant les données fournies par le LRF et l'odomètre pour localiser le robot et déterminer l'état $\mathbf{X}$. Nous utilisons les équations de l'EKF détaillées dans [48].

Initialisation :

- $\hat{\mathbf{X}}_0 \sim \mathcal{N}(\mathbf{X_0}, \mathbf{\Sigma_0})$,
- $\mathbf{\Sigma_0} = \mathrm{diag}(\sigma_{x,0}, \sigma_{y,0}, \sigma_{\psi,0})$.

Étape de propagation, $t > 0$:

- $\hat{\mathbf{X}}_t^- = \mathbf{f}(\hat{\mathbf{X}}_{t-1}^+, \tilde{\mathbf{U}}_{t-1}, \mathbf{0})$,
- $\mathbf{\Sigma}_t^- = \mathbf{A}_{t-1}\mathbf{\Sigma}_{t-1}^+\mathbf{A}_{t-1}^T + \mathbf{B}_{t-1}\mathbf{H}\mathbf{B}_{t-1}^T + \mathbf{Q}$.

Ajout des mesures externes :

- $\hat{\mathbf{Y}}_t^- = \mathbf{g}_t(\hat{\mathbf{X}}_t^-, \mathbf{0})$,
- $\mathbf{K}_t = \mathbf{\Sigma}_t^-\mathbf{C}_t^T(\mathbf{C}_t\mathbf{\Sigma}_t^-\mathbf{C}_t^T + \mathbf{R}_t)^{-1}$,
- $\hat{\mathbf{X}}_t^+ = \hat{\mathbf{X}}_t^- + \mathbf{K}_t(\mathbf{Y}_t - \hat{\mathbf{Y}}_t^-)$,
- $\mathbf{\Sigma}_t^+ = (\mathbf{I}_3 - \mathbf{K}_t\mathbf{C}_t)\mathbf{\Sigma}_t^-$.

6.1.5 Simulation

Nous montrons le résultat du filtrage de Kalman (en 2D) figure 6.1. À gauche nous supposons que les points sont reconnaissables, et à droite on calcule l'association de données. Celle-ci est calculée en prenant le point de la carte qui est le plus proche de la mesure effectuée (distance euclidienne ou maximum de vraisemblance).

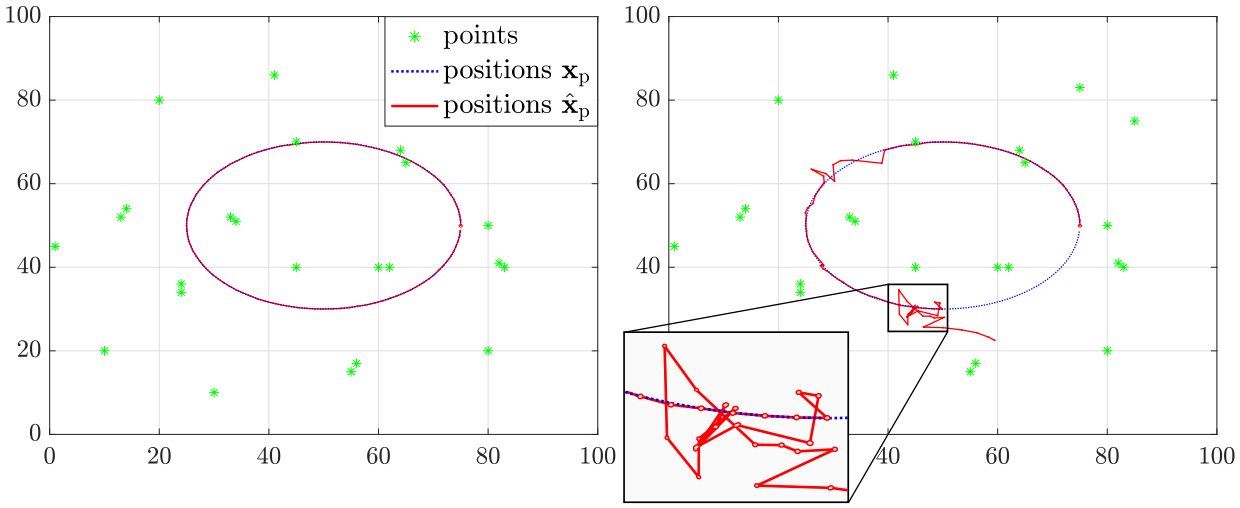


Figure 6.1 Positionnement du robot à partir de points localisés dans la carte. À gauche avec l'association de données connue, à droite elle est calculée.

Dans le premier cas (association de données connue), le filtrage se passe très bien et la position estimée s'aligne bien sur la position réelle.

Dans le second cas (association de données calculée), le filtrage est plus compliqué. Nous avons volontairement placé certains marqueurs proches les uns des autres pour apporter de l'ambiguïté dans l'association de données. Ainsi, lors du calcul de l'association de données, certains points peuvent être confondus. À l'étape de mise à jour, l'innovation calculée n'est pas correcte, ce qui résulte en un mauvais positionnement final (figure 6.1 à droite). Nous représentons les ellipses d'erreur à 3σ soit 98.9%, et nous voyons alors que l'erreur est largement sous-estimée. Ceci a pour conséquence la mauvaise estimation de la pose, et risque de détériorer encore plus les associations de données futures, puisque celle-ci sont calculées à partir de la pose estimée elle-même. Nous avons évoqué comme solution le filtre à particule qui envisage différentes associations de données probables. D'autres approches existent, comme ICP [11], qui tente d'aligner l'ensemble des détections sur la carte, en appariant chaque détection à un marqueur de l'environnement. Si l'algorithme converge correctement, on peut utiliser cet appariement comme association de donnée.

De notre côté, nous avons choisi de représenter notre carte avec des SQs. Cette représentation

de plus haut niveau facilite l'association de données, car reconnaître des objets est plus aisé que de reconnaître de simples points. Nous proposons alors de garder ce filtre de Kalman en l'adaptant aux SQs.

6.2 Localisation avec des superquadriques

Notre environnement est maintenant composé d'obstacles en superquadriques $\mathbf{P}_m$. Les paramètres de ces obstacles sont connus, et nous cherchons à localiser notre robot grâce à l'observation de ces obstacles par le LRF. Pour ceci, nous utilisons le filtre de Kalman détaillé précédemment section 6.1. Le problème ici s'agit d'associer nos mesures à des marqueurs de la carte, sachant que nous n'avons pas connaissance desdits marqueurs, mais seulement de leur répartition autour de SQs.

6.2.1 Association de données

La première chose que nous faisons est d'associer chaque mesure $(\tilde{d}, \tilde{\lambda}, \tilde{\phi})$ à un obstacle connu de la carte. Tout d'abord, nous mettons nos mesures dans le référentiel de navigation :

$$\hat{\mathbf{x}} = \begin{bmatrix} \hat{x}_\mathrm{p} \\ \hat{y}_\mathrm{p} \\ 0 \end{bmatrix} + \tilde{d} \begin{bmatrix} \cos(\tilde{\lambda} + \hat{\psi}_\mathrm{p}) \cos\tilde{\phi} \\ \sin(\tilde{\lambda} + \hat{\psi}_\mathrm{p}) \cos\tilde{\phi} \\ \sin\tilde{\phi} \end{bmatrix}. \tag{6.9}$$

L'association du point $\hat{\mathbf{x}}$ à un obstacle m se fait comme suit :

$$m = \underset{m}{\operatorname{argmin}} \; d_r(\hat{\mathbf{x}}; \mathbf{P}_m)^2, \tag{6.10}$$

et nous notons alors $\hat{\mathbf{x}}_{m_k}$, le k-ième point attribué à l'obstacle m.

6.2.2 Calcul des marqueurs

Il s'agit maintenant de trouver les marqueurs $\mathbf{x}_k$ de notre environnement qui correspondent aux points $\hat{\mathbf{x}}_k$ estimés. Pour ceci, nous cherchons à trouver l'erreur $\mathrm{d}\mathbf{x}_\mathrm{p} = [\mathrm{d}x_\mathrm{p}\ \mathrm{d}y_\mathrm{p}\ \mathrm{d}\psi_\mathrm{p}]^T$ entre la pose à priori $\hat{\mathbf{x}}_\mathrm{p}^-$, et la pose réelle $\mathbf{x}_\mathrm{p}$. Cette erreur est en fait une transformation rigide bi-dimensionnelle avec trois degrés de liberté.

Nous faisons ici le parallèle avec le problème de Procruste [59], où nous cherchons la matrice $\mathbf{T} \in SE(3)$ qui minimise la distance entre les deux nuages de points $\mathbf{x}_k$ et $\mathbf{T}\hat{\mathbf{x}}_k$, c'est à dire qui va permettre d'aligner les points $\hat{\mathbf{x}}_k$ sur la carte composée de $\mathbf{x}_k$. La matrice $\mathbf{T}$ représente

la transformation rigide (une translation et une rotation), et dans notre cas où le robot se déplace sur un plan, cette transformation n'a que 3 degrés de liberté [dx dy dψ].

$$\mathbf{T} = \begin{bmatrix} \cos \mathrm{d}\psi & -\sin \mathrm{d}\psi & 0 & \mathrm{d}x \\ \sin \mathrm{d}\psi & \cos \mathrm{d}\psi & 0 & \mathrm{d}y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{6.11}$$

Le problème de Procruste s'écrit donc :

$$\begin{cases} \underset{\mathbf{T}}{\text{argmin}} & \|\mathbf{T}\hat{\mathbf{x}}_k - \mathbf{x}_k\|_F, \\ \text{s.l.c.} & \mathbf{T} \in SE(3), \end{cases} \tag{6.12}$$

avec $\|.\|_F$ la norme de Frobenius : $\|\mathbf{M}\|_F = \sqrt{\text{Trace}\left(\mathbf{M}\mathbf{M}^T\right)}$. La résolution de ce problème est bien connue, et [60] fournit une résolution possible.

La difficulté dans notre cas est que nous cherchons aussi les points $\mathbf{x}_k$ dans la carte. Or, notre carte est composée de SQs, mais nous pourrions échantillonner celle-ci en un nuage de points, comme fait figure 6.2. Il s'agit maintenant de choisir dans ce nuage de points les K marqueurs $\mathbf{x}_k$ adéquats qui correspondraient au mieux aux mesures $\hat{\mathbf{x}}_k$.

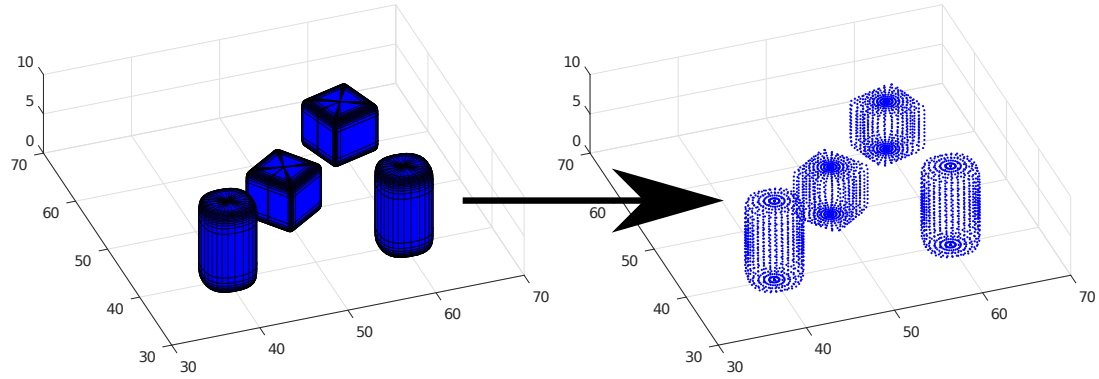


Figure 6.2 Échantillonage de la carte en nuage de points.

Les algorithmes ICPs permettent de résoudre ce problème de choix de marqueurs. Une approche simple d'ICP est de choisir naïvement pour chaque $\hat{\mathbf{x}}_k$ le point $\mathbf{x}_k$ de la carte qui lui est le plus proche au sens de la norme euclidienne, mais d'autres approches plus sophistiquées existent. Ensuite, il faut résoudre le problème de Procruste, replacer les points selon la transformation trouvée, et ré-itérer jusqu'à convergence.

Nous proposons une autre solution, n'utilisant pas d'échantillonnage, mais en travaillant

directement avec la carte de SQs et l'association de données calculée précédemment section 6.2.1. Nous cherchons dans notre cas à aligner les points mesurés $\hat{\mathbf{x}}_k$ avec les volumes. Pour ceci nous utilisons le travail effectué section 3, et nous posons le problème de moindres carrés :

$$\underset{\mathrm{d}\mathbf{x}_\mathrm{p}}{\operatorname{argmin}} \sum_{m=1}^{M} \sum_{k=1}^{K_M} F_S(\hat{\mathbf{x}}_{m_k}; \mathbf{P}_m)^2, \tag{6.13}$$

avec

$$\hat{\mathbf{x}}_{m_k} = \begin{bmatrix} \hat{x}_\mathrm{p}^- + \mathrm{d}x_\mathrm{p} \\ \hat{y}_\mathrm{p}^- + \mathrm{d}y_\mathrm{p} \\ 0 \end{bmatrix} + \tilde{d}_{m_k} \begin{bmatrix} \cos(\hat{\psi}_\mathrm{p}^- + \mathrm{d}\psi_\mathrm{p} + \tilde{\lambda}_{m_k}) \cos\tilde{\phi}_{m_k} \\ \sin(\hat{\psi}_\mathrm{p}^- + \mathrm{d}\psi_\mathrm{p} + \tilde{\lambda}_{m_k}) \cos\tilde{\phi}_{m_k} \\ \sin\tilde{\phi}_{m_k} \end{bmatrix}. \tag{6.14}$$

La fonction F_S est définie équation 3.27. La valeur $F_S(\hat{\mathbf{x}}_{m_k}; \mathbf{P}_m)$ est nulle lorsque le point $\hat{\mathbf{x}}_{m_k}$ est sur la SQ $\mathbf{P}_m$. La sommation sur M correspond aux M SQs de l'environnement, tandis que celle sur K_M correspond aux K_M marqueurs associés à l'obstacle M. Il s'agit ici de trouver la transformation rigide $\mathrm{d}\mathbf{x}_\mathrm{p}$ qui aligne aux mieux les mesures sur les SQs. Nos marqueurs sont alors les $\hat{\mathbf{x}}_{m_k}$.

Lorsque $\mathrm{d}\mathbf{x}_\mathrm{p}$ est grand, l'association de données est souvent mal calculée, et les points s'alignent alors mal sur les volumes. Nous proposons une amélioration plus robuste dans la suite, avec une approche itérative à l'instar de ICP.

6.2.3 Calcul des marqueurs par Espérance-Maximisation

Afin de trouver nos marqueurs, nous avons d'abord associé les mesures à des obstacles, puis ensuite calculé nos marqueurs par moindres carrés en utilisant cette association de données. Cette approche est très sensible aux mesures ambiguës, car une mauvaise association aura de lourdes conséquences lors du calcul des marqueurs. Nous proposons ici une approche plus robuste à ces associations incertaines. Plutôt que d'associer une mesure à un obstacle en ne considérant que la distance minimale, nous considérons la probabilité d'une mesure à provenir d'un obstacle. Cette probabilité est définie comme suit :

$$\mathbb{P}(\hat{\mathbf{x}} \in m) = \frac{\dfrac{1}{d_r(\hat{\mathbf{x}}; \mathbf{P}_m)^2}}{\displaystyle\sum_{m=1}^{M} \frac{1}{d_r(\hat{\mathbf{x}}; \mathbf{P}_m)^2}}. \tag{6.15}$$

Par la suite nous introduisons ces probabilités sous la forme de poids lors de la minimisation

par moindres carrés :

$$\underset{\mathrm{d}\mathbf{x}_\mathrm{p}}{\operatorname{argmin}} \sum_{m=1}^{M} \sum_{k=1}^{K} \left(F_S(\hat{\mathbf{x}}_k; \mathbf{P}_m) \mathbb{P}(\hat{\mathbf{x}}_k \in m)^2 \right)^2 . \tag{6.16}$$

De plus, nous procédons itérativement, en replaçant les marqueurs selon l'erreur sur l'état $\mathrm{d}\mathbf{x}_\mathrm{p}$ trouvée. Notre démarche s'appuie sur l'espérance maximisation [61], et place les marqueurs de la façon suivante :

. Placer les marqueurs selon la position $\hat{\mathbf{x}}_\mathrm{p}^{(i)}$

$$\hat{\mathbf{x}}^{(i)} = \begin{bmatrix} \hat{x}_\mathrm{p}^{(i)} \\ \hat{y}_\mathrm{p}^{(i)} \\ 0 \end{bmatrix} + \tilde{d} \begin{bmatrix} \cos(\hat{\psi}_\mathrm{p}^{(i)} + \tilde{\lambda}) \cos\tilde{\phi} \\ \sin(\hat{\psi}_\mathrm{p}^{(i)} + \tilde{\lambda}) \cos\tilde{\phi} \\ \sin\tilde{\phi} \end{bmatrix} . \tag{6.17}$$

. Calculer les distances aux obstacles $d_r(\hat{\mathbf{x}}^{(i)}; \mathbf{P}_m)$.

. Calculer les poids probabilistes $\mathbb{P}(\hat{\mathbf{x}}^{(i)} \in m)$.

. Réaliser la minimisation équation (6.16) pour obtenir $\mathrm{d}\mathbf{x}_\mathrm{p}^{(i)}$.

. Mettre à jour $\hat{\mathbf{x}}_\mathrm{p}^{(i+1)} = \hat{\mathbf{x}}_\mathrm{p}^{(i)} + \mathrm{d}\mathbf{x}_\mathrm{p}^{(i)}$, et recommencer jusqu'à convergence.

La position est initialisée avec l'état propagé par le filtre $\hat{\mathbf{x}}_\mathrm{p}^{(0)} = \hat{\mathbf{x}}_\mathrm{p}^{-}$. La convergence est détectée quand l'évolution des poids ne varie plus significativement d'une itération à l'autre. Pour ceci nous utilisons l'entropie de Shannon [62] appliquée à nos poids probabilistes.

$$H^{(i)} = -\frac{1}{K} \sum_{k=1}^{K} \sum_{m=1}^{M} \mathbb{P}(\hat{\mathbf{x}}_k^{(i)} \in m) \log_M(\mathbb{P}(\hat{\mathbf{x}}_k^{(i)} \in m)). \tag{6.18}$$

Cette définition nous assure $H^{(i)} \in [0, 1]$, et l'algorithme se termine lorsque $H^{(i-1)} - H^{(i)} < \tau$, avec τ un seuil fixé petit. La figure 6.3 à gauche montre le fonctionnement de l'algorithme itératif (en 2D). En entrée nous partons avec les données en rouge, qui sont fortement décalées par rapport à notre carte en bleu. L'ajustement étape par étape de la pose et des marqueurs est schématisée en noir. Les données en vert sont obtenues après convergence. Notre algorithme a pu converger correctement malgré le fort décalage initial, avec beaucoup d'associations de données ambiguës. Dans cette simulation, la qualité de la convergence peut-être estimée avec l'entropie. Celle-ci est représentée figure 6.3 à droite et plus sa valeur est faible, moins l'association de données est ambiguë.

Par ailleurs, nous avons remarqué que la prise en compte de la distance des mesures apporte un plus dans la convergence de l'algorithme. Les mesures de cap réalisées par le LRF sont bruitées. Ainsi, pour deux mesures avec une même erreur de cap, celle qui est la plus éloignée

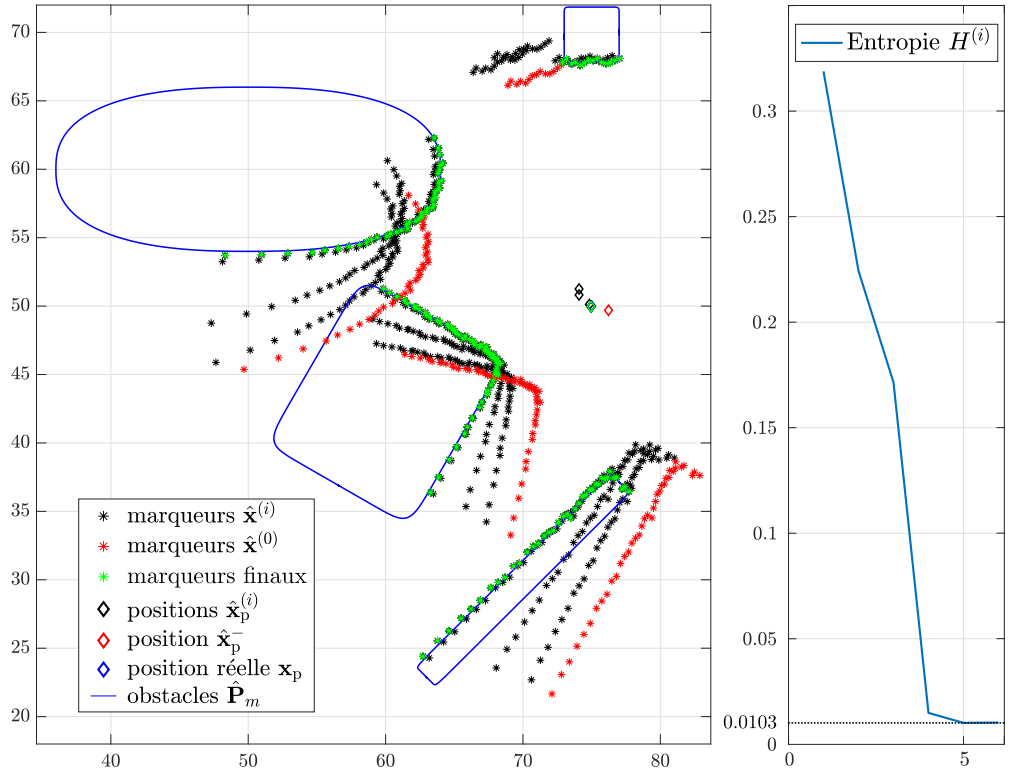


Figure 6.3 Alignement des marqueurs sur la carte.

du robot sera plus incertaine que l'autre. Nous quantifions les erreurs sur la position des marqueurs en accord avec les erreurs de mesure à la manière de la section 5 avec les équations (5.14) et (5.15) :

$$\mathbf{\Sigma}_{\hat{\mathbf{x}}_k^{(i)}} = \mathbf{J}_{\mathbf{z}_k}^{(i)} \mathbf{R} \mathbf{J}_{\mathbf{z}_k}^{(i)^T}, \tag{6.19}$$

et nous créons les poids $w_k^{(i)} = \mathrm{Trace}\left(\mathbf{\Sigma}_{\hat{\mathbf{x}}_k^{(i)}}\right)$. Lors de la minimisation i, nous avons :

$$\mathrm{d}\mathbf{x}_{\mathrm{p}}^{(i)} = \underset{\mathrm{d}\mathbf{x}_{\mathrm{p}}}{\mathrm{argmin}} \sum_{m=1}^{M} \sum_{k=1}^{K} \left(\frac{\mathbb{P}(\hat{\mathbf{x}}_k^{(i)} \in m)^2}{w_k^{(i)}} F_S(\hat{\mathbf{x}}_k^{(i)}; \mathbf{P}_m) \right)^2. \tag{6.20}$$

Désormais, notre problème de minimisation prend moins en compte les mesures lointaines.

6.3 Simulation

Nous simulons notre algorithme avec l'EKF sous `Matlab`. La figure 6.4 montre comment se passe le positionnement le long d'une trajectoire en ellipse. La position et la forme des obstacles en bleu est connue, et le robot utilise ces informations pour se localiser. L'odométrie

représentée en vert est aussi utilisée.

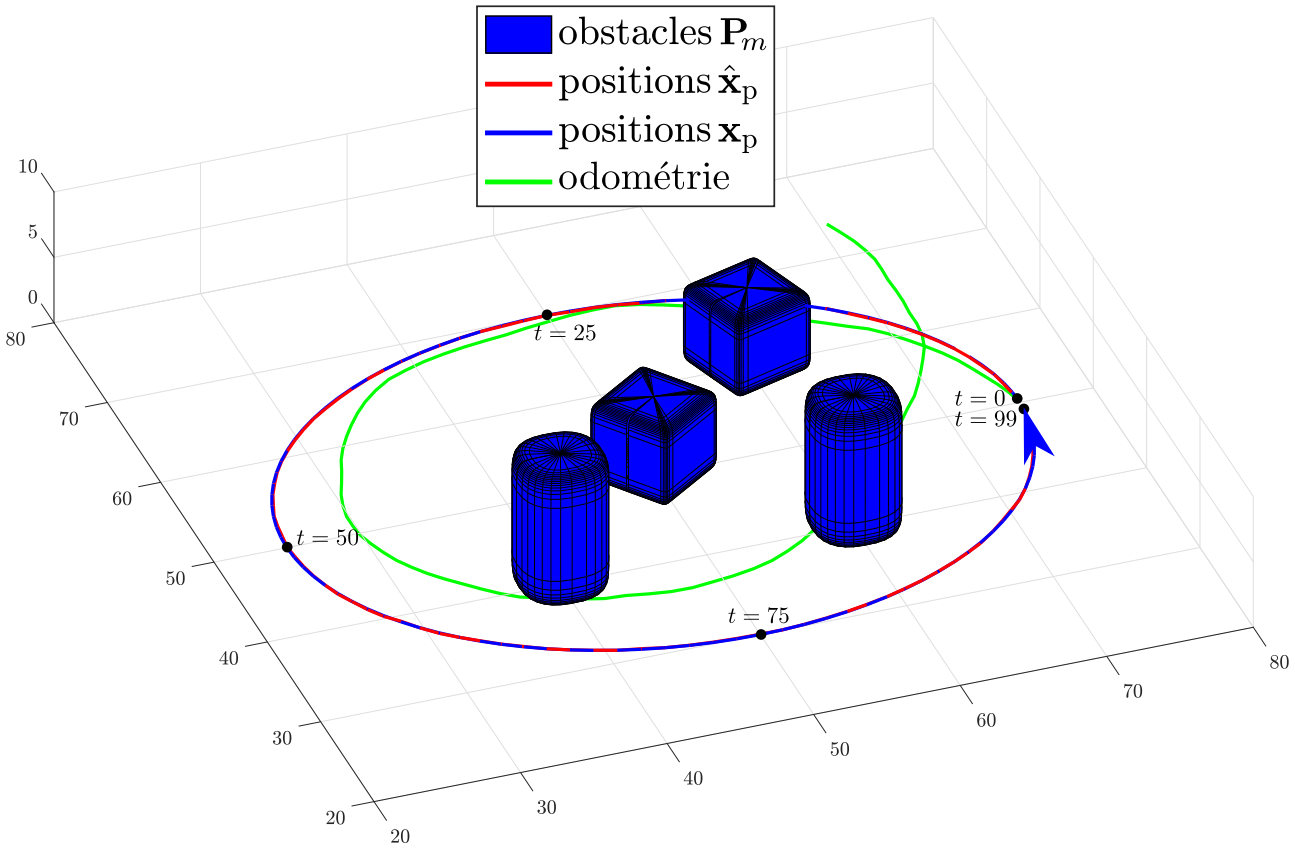


Figure 6.4 Positionnement à partir d'obstacles en forme de superquadrique.

Nous voyons figure 6.5 les erreurs entre l'état estimé $\hat{\mathbf{x}}_\mathrm{p}$ et l'état réel $\mathbf{x}_\mathrm{p}$ à chaque étape t. Dans notre simulation, l'erreur en position tourne autour des 15cm, et l'erreur en cap 1°.

Le filtre de Kalman nous renvoie la matrice $\mathbf{\Sigma}$ qui permet d'évaluer la précision de l'estimation de la pose. Les difficultés du robot à se localiser peuvent aussi être quantifiées figure 6.6 en utilisant la Dilution de Précision Horizontale (HDOP) [51]. La HDOP est calculée à partir de la position estimée des marqueurs $\hat{\mathbf{x}}_k$ repérés dans l'environnement, en calculant la Jacobienne suivante :

$$\mathbf{J}_{\mathbf{x}_\mathrm{p}} = \begin{bmatrix} \dfrac{\hat{x}_\mathrm{p} - \hat{x}_1}{\|\hat{\mathbf{x}}_\mathrm{p} - \hat{\mathbf{x}}_1\|_2} & \dfrac{\hat{y}_\mathrm{p} - \hat{y}_1}{\|\hat{\mathbf{x}}_\mathrm{p} - \hat{\mathbf{x}}_1\|_2} \\ \vdots & \vdots \\ \dfrac{\hat{x}_\mathrm{p} - \hat{x}_K}{\|\hat{\mathbf{x}}_\mathrm{p} - \hat{\mathbf{x}}_K\|_2} & \dfrac{\hat{y}_\mathrm{p} - \hat{y}_K}{\|\hat{\mathbf{x}}_\mathrm{p} - \hat{\mathbf{x}}_K\|_2} \end{bmatrix}. \tag{6.21}$$

On obtient notre matrice de DOP $\mathbf{G} = \mathfrak{A}(\mathbf{J}_{\mathbf{x}_\mathrm{p}})$ en prenant l'inverse généralisée définie équation (4.25). La HDOP affichée en bleu figure 6.6 est égale à $\sqrt{\mathbf{G}_{1,1} + \mathbf{G}_{2,2}}$.

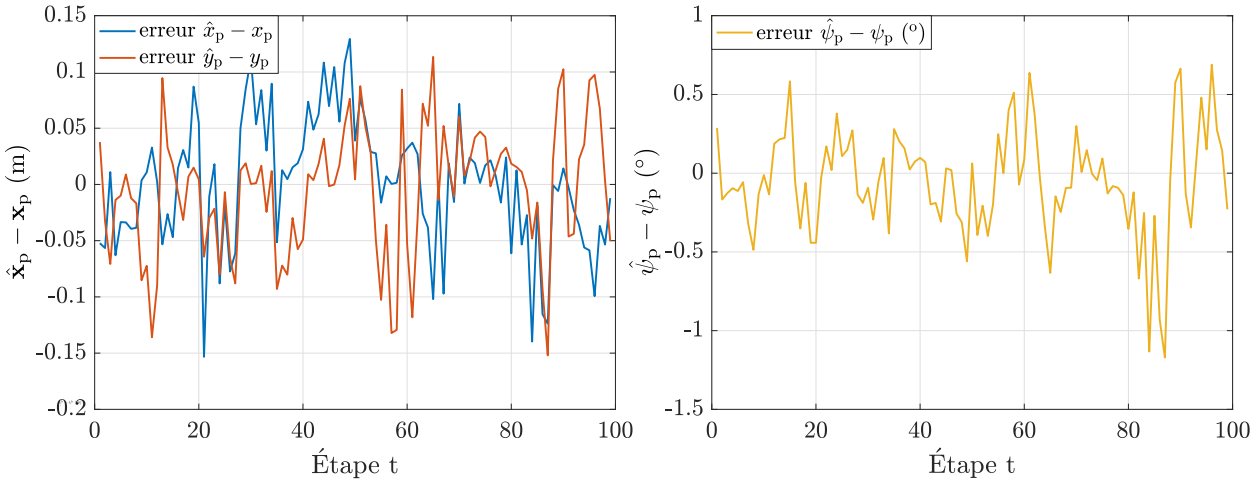


Figure 6.5 Erreur sur l'estimation de l'état $\mathbf{x}_\mathrm{p}$.

Autour des étapes 15, 60 et 85, celle-ci est plus importante à cause des phénomènes d'occlusion par les obstacles de la carte. En effet, à ces positions, seul un obstacle est visible, et il est alors plus dur pour le robot de se localiser. Au contraire, autour de l'étape $t = 75$, les quatre obstacles peuvent être observés, et une meilleure localisation est possible.

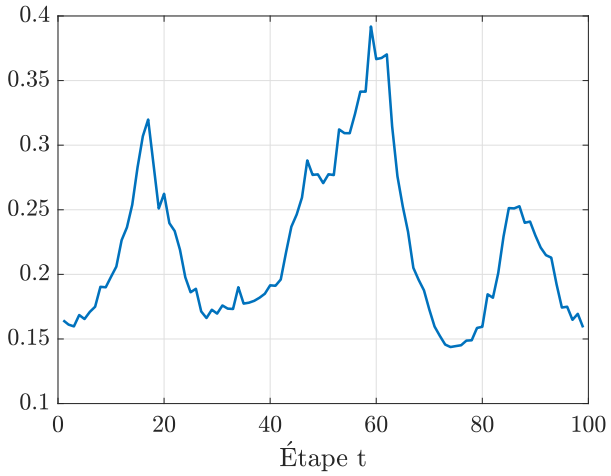


Figure 6.6 Dilutions de précision horizontales.

Nous simulons notre algorithme de localisation sur un autre exemple figure 6.7 avec des obstacles qui sont proches et d'autres qui s'intersectent. Nous affichons aussi les marqueurs $\hat{\mathbf{x}}_k$ observés tout au long de la trajectoire par le robot, placés dans le référentiel global en accord avec la position estimée $\mathbf{x}_\mathrm{p}$ par le filtre. Nous voyons que malgré que les SQs se touchent ou s'intersectent, les marqueurs sont correctement placés sur la carte, et le robot arrive à se localiser correctement.

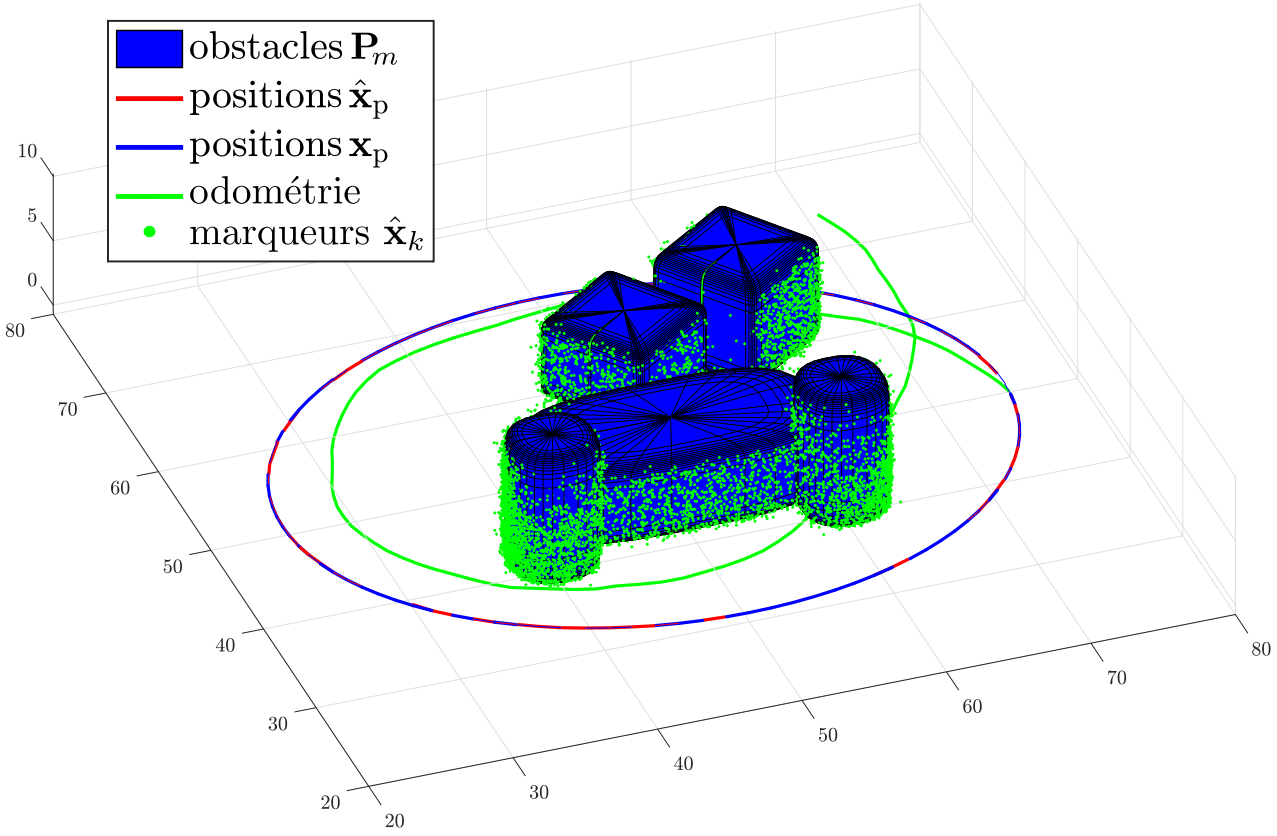


Figure 6.7 Positionnement à partir d'obstacles proches et se recoupant.

CHAPITRE 7 CONCLUSION

7.1 Synthèse des travaux

Dans ce mémoire, nous avons proposé des algorithmes permettant de cartographier et de se localiser dans un environnement composé de SQs. Toutes les simulations expliquées ont été menées en deux puis trois dimensions. Le développement de ces algorithmes nous a d'abord amené à travailler sur les méthodes de reconstruction de SQs. Retrouver une SQ à partir d'un nuage de points n'est pas trivial, et plusieurs phénomènes comme l'auto occlusion compliquent cette tâche. Par ailleurs, un enjeu majeur était de ne pas conserver toutes les données fournies par le capteur, et nous avons alors développé une technique de filtrage permettant de reconstruire une SQ sans devoir sauvegarder exhaustivement les mesures du LRF. Ceci permet un grand gain de mémoire. Il arrive que le filtrage ne converge pas ou alors lentement vers le modèle attendu, et nous avons proposé un moyen de détection et de correction pour accélérer les cas de convergence lente.

Concernant la cartographie nous avons décrit deux approches permettant de reconstruire la carte d'un environnement composé de SQs. Ces approches requièrent notamment de devoir segmenter les mesures obtenues par le capteur. Nous avons décrit et testé plusieurs techniques de segmentation de nuages de points. Toujours dans l'esprit d'utiliser notre représentation de carte en SQs, nous avons développé un algorithme permettant d'associer une SQ à un point mesuré, en utilisant les données reçues du filtre. Enfin, nous avons détaillé une méthode pour détecter lorsque deux SQs s'intersectent, et comment faire pour les fusionner correctement.

Concernant le positionnement, nous avons d'abord expliqué comment localiser un robot mobile à partir d'une carte composée de marqueurs. Ensuite, nous avons montré comment appliquer cette technique à une carte composée de SQs. Pour se faire, nous avons développé une méthode d'association de données itérative robuste permettant de calculer dans l'environnement des marqueurs à partir des mesures effectuées par laser.

7.2 Limitations de la solution proposée

La plus grosse limitation ici concerne l'algorithme de cartographie, qui n'est efficace que dans des environnements idéaux composés de SQs séparées les unes des autres. Notre algorithme échoue quand nous rapprochons trop entre eux certains obstacles de la carte, et il ne parvient alors plus à distinguer les obstacles séparément en reconstruisant une SQ qui les englobe tous. L'algorithme de localisation est plus robuste à ce phénomène, et parvient à différencier des

obstacles qui se touchent.

Par ailleurs, les algorithmes de cartographie et de localisation n'ont été testés qu'en simulation dans des environnements parfaitement contrôlés composés uniquement de SQs. Lors de la cartographie, la position du robot est connue exactement ce qui n'est jamais le cas, de même pour la localisation où la représentation de la carte est connue parfaitement. Par ailleurs, nos algorithmes n'ont pas été testés dans des environnements plus complexes, et il persiste des objets volumiques qui ont des formes plus sophistiquées pouvant difficilement être approchées par une SQ. La représentation de murs, plafonds, sols, façades ne parait pas idéale avec des SQs. Il est plus pertinent de représenter ces derniers avec des plans (ou des superellipses), et donc d'utiliser les primitives en SQs qu'en parallèle d'autres représentations.

7.3 Améliorations futures

Tout d'abord, il faudrait valider ces algorithmes dans un environnement réel. Ensuite, nous voudrions rendre la représentation plus robuste. Développer des algorithmes de segmentation plus poussés pouvant décomposer un objet en union de SQs permettrait une extension importante pour la représentation d'objets plus complexes. Certains algorithmes existent déjà en traitement d'image [41], et il ne reste qu'à les intégrer dans le domaine de la cartographie et d'appliquer nos techniques de filtrage. Par ailleurs, nos algorithmes sont basés sur le filtrage de Kalman, mais d'autres filtres ont été évoqués sans être testés. Nous pensons notamment aux filtres à particules, plus demandant en calcul et en mémoire, mais supportant les modèles de mesures non linéaires et non gaussiens plus aisément. Enfin, nos algorithmes de cartographie et localisation ont aussi été pensés dans le cadre de la localisation et cartographie simultanées (SLAM), et répondre à ce problème en utilisant des SQs n'est pas encore résolu.

RÉFÉRENCES

[1] "Velodyne ultra puck lidar," https://velodynelidar.com/vlp-32c.html.

[2] R. Pascoal *et al.*, "Simultaneous segmentation and superquadrics fitting in laser-range data," *IEEE Transactions on Vehicular Technology*, vol. 64, n°. 2, p. 441–452, 2015.

[3] iRobot, "Roomba," *Website : https://www.irobot.co.uk/roomba*, 2002.

[4] P. D. Groves, *Principles of GNSS, inertial, and multisensor integrated navigation systems*, 2e éd. Artech House, 2013.

[5] B. Hofmann-Wellenhof, H. Lichtenegger et E. Wasle, *GNSS–global navigation satellite systems : GPS, GLONASS, Galileo, and more*. Springer Science & Business Media, 2007.

[6] J. L. Crassidis, "Sigma-point Kalman filtering for integrated GPS and inertial navigation," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 42, n°. 2, p. 750–756, 2006.

[7] X. Kong, E. M. Nebot et H. Durrant-Whyte, "Development of a nonlinear psi-angle model for large misalignment errors and its application in INS alignment and calibration," dans *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C)*, vol. 2. IEEE, 1999, p. 1430–1435.

[8] V. Indelman *et al.*, "Factor graph based incremental smoothing in inertial navigation systems," dans *2012 15th International Conference on Information Fusion*. IEEE, 2012, p. 2154–2161.

[9] P. Corke, *Robotics, vision and control : fundamental algorithms in MATLAB® second edition*. Springer, 2017, vol. 118.

[10] J.-L. Blanco, J. González et J.-A. Fernández-Madrigal, "An optimal filtering algorithm for non-parametric observation models in robot localization," dans *2008 IEEE International Conference on Robotics and Automation*. IEEE, 2008, p. 461–466.

[11] P. J. Besl et N. D. McKay, "Method for registration of 3-D shapes," dans *Sensor fusion IV : control paradigms and data structures*, vol. 1611. International Society for Optics and Photonics, 1992, p. 586–606.

[12] F. Moosmann et C. Stiller, "Velodyne slam," dans *2011 ieee intelligent vehicles symposium (iv)*. IEEE, 2011, p. 393–398.

[13] I. J. Cox, "Blanche an experiment in guidance and navigation of an autonomous robot vehicle," *IEEE Transactions on robotics and automation*, vol. 7, n°. 2, p. 193–204, 1991.

[14] J. Aulinas *et al.*, "The SLAM problem : a survey." *CCIA*, vol. 184, n°. 1, p. 363–371, 2008.

[15] C. Cadena *et al.*, "Past, Present, and Future of Simultaneous Localization and Mapping : Toward the Robust-Perception Age," *IEEE Transactions on Robotics*, vol. 32, n°. 6, déc. 2016.

[16] T. Bailey *et al.*, "Consistency of the EKF-SLAM algorithm," dans *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2006, p. 3562–3568.

[17] M. Montemerlo *et al.*, "FastSLAM : A factored solution to the simultaneous localization and mapping problem," *Aaai/iaai*, vol. 593598, 2002.

[18] ——, "FastSLAM 2.0 : An improved particle filtering algorithm for simultaneous localization and mapping that provably converges," dans *IJCAI*, 2003, p. 1151–1156.

[19] W. Burgard *et al.*, "Sonar-based mapping with mobile robots using EM," dans *Machine Learning - International Workshop then Conference*. Morgan Kaufmann Publishers, INC, 1999, p. 67–76.

[20] F. Lu et E. Milios, "Globally consistent range scan alignment for environment mapping," *Autonomous robots*, vol. 4, n°. 4, p. 333–349, 1997.

[21] ——, "Robot pose estimation in unknown environments by matching 2D range scans," *Journal of Intelligent and Robotic systems*, vol. 18, n°. 3, p. 249–275, 1997.

[22] M. Kaess *et al.*, "iSAM2 : Incremental smoothing and mapping using the Bayes tree," *The International Journal of Robotics Research*, vol. 31, n°. 2, p. 216–235, 2012.

[23] R. Kümmerle *et al.*, "g^{2}o : A general framework for graph optimization," dans *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, p. 3607–3613.

[24] J. Shi *et al.*, "Good features to track," dans *1994 Proceedings of IEEE conference on computer vision and pattern recognition*. IEEE, 1994, p. 593–600.

[25] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, n°. 2, p. 91–110, 2004.

[26] E. Rublee *et al.*, "ORB : An efficient alternative to SIFT or SURF." dans *ICCV*, vol. 11, n°. 1. Citeseer, 2011, p. 2.

[27] Y. Lu et D. Song, "Visual navigation using heterogeneous landmarks and unsupervised geometric constraints," *IEEE Transactions on Robotics*, vol. 31, n°. 3, p. 736–749, 2015.

[28] A. J. Davison *et al.*, "MonoSLAM : Real-time single camera SLAM," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, n°. 6, p. 1052–1067, 2007.

[29] R. Mur-Artal, J. M. M. Montiel et J. D. Tardos, "ORB-SLAM : a versatile and accurate monocular SLAM system," *IEEE transactions on robotics*, vol. 31, nº. 5, p. 1147–1163, 2015.

[30] B. Triggs *et al.*, "Bundle adjustment—a modern synthesis," dans *International workshop on vision algorithms*. Springer, 1999, p. 298–372.

[31] A. Hornung *et al.*, "OctoMap : An efficient probabilistic 3D mapping framework based on octrees," *Autonomous robots*, vol. 34, nº. 3, p. 189–206, 2013.

[32] R. A. Newcombe, S. J. Lovegrove et A. J. Davison, "DTAM : Dense tracking and mapping in real-time," dans *2011 international conference on computer vision*. IEEE, 2011, p. 2320–2327.

[33] M. Pizzoli, C. Forster et D. Scaramuzza, "Remode : Probabilistic, monocular dense reconstruction in real time," dans *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, p. 2609–2616.

[34] R. O. Castle *et al.*, "Towards simultaneous recognition, localization and mapping for hand-held and wearable cameras," dans *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, 2007, p. 4102–4107.

[35] M. Kaess, "Simultaneous localization and mapping with infinite planes," dans *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, p. 4605–4611.

[36] P. J. Schneider, "Nurb curves : a guide for the uninitiated," *Apple Tech. J*, nº. 25, 1996.

[37] R. F. Salas-Moreno *et al.*, "Slam++ : Simultaneous localisation and mapping at the level of objects," dans *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, p. 1352–1359.

[38] A. Dame *et al.*, "Dense reconstruction using 3D object shape priors," dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, p. 1288–1295.

[39] M. McCartney, "Lamé hypersurfaces and Lagrange multipliers," *International Journal of Mathematical Education in Science and Technology*, vol. 47, nº. 2, p. 315–322, 2016. [En ligne]. Disponible : https://doi.org/10.1080/0020739X.2015.1078005

[40] A. H. Barr, "Superquadrics and angle-preserving transformations," *IEEE Computer graphics and Applications*, vol. 1, nº. 1, p. 11–23, 1981.

[41] A. Jaklic, A. Leonardis et F. Solina, *Segmentation and recovery of superquadrics*. Springer Science & Business Media, 2013, vol. 20.

[42] Y. Zhan, "Least squares fitting of ellipses," 2014. [En ligne]. Disponible : http://www.hep.princeton.edu/mumu/target/Yan/ellipse_fit.pdf

[43] T. F. Coleman et Y. Li, "An interior trust region approach for nonlinear minimization subject to bounds," *SIAM Journal on optimization*, vol. 6, n°. 2, p. 418–445, 1996.

[44] J. Shlens, "A tutorial on principal component analysis," *arXiv preprint arXiv :1404.1100*, 2014.

[45] S. Lang, *Graduate Texts in Mathematics : Algebra*. Springer, 2002.

[46] J. Guérin, N. Lahrichi et S. Le Digabel, "Matrices symétriques et matrices définies positives," 2019. [En ligne]. Disponible : https://www.gerad.ca/Sebastien.Le.Digabel/MTH1007/12_matrices_symetriques_def_positives.pdf

[47] F. Solina et R. Bajcsy, "Recovery of parametric models from range images : The case for superquadrics with global deformations," *IEEE transactions on pattern analysis and machine intelligence*, vol. 12, n°. 2, p. 131–147, 1990.

[48] C. K. Chui et G. Chen, *Kalman Filtering with Real-time Applications*. New York, NY, USA : Springer-Verlag New York, Inc., 1987.

[49] S. J. Julier, "The scaled unscented transformation," dans *Proceedings of the 2002 American Control Conference*, vol. 6. IEEE, 2002, p. 4555–4559.

[50] S. Särkkä, *Bayesian filtering and smoothing*. Cambridge University Press, 2013, vol. 3.

[51] P. D. Groves, *Principles of GNSS, inertial, and multisensor integrated navigation systems*. Artech house, 2013.

[52] G. A. Borges et M.-J. Aldon, "Line extraction in 2D range images for mobile robotics," *Journal of intelligent and Robotic Systems*, vol. 40, n°. 3, p. 267–297, 2004.

[53] A. Rosenfeld, J. L. Pfaltz *et al.*, "Sequential operations in digital picture processing." *J. ACM*, vol. 13, n°. 4, p. 471–494, 1966.

[54] R. M. Haralick et L. G. Shapiro, *Computer and robot vision*. Addison-wesley Reading, 1992, vol. 1.

[55] M. Ester *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." dans *KDD*, vol. 96, n°. 34, 1996, p. 226–231.

[56] C. Robert et G. Casella, *Monte Carlo statistical methods*. Springer Science & Business Media, 2013.

[57] J. Zhu *et al.*, "Computed tomography simulation with superquadrics," *Medical physics*, vol. 32, n°. 10, p. 3136–3143, 2005.

[58] D. Eberly, "Intersection of ellipsoids," Technical report, Geometric Tools, LLC, Rapport technique, 2008.

[59] J. C. Gower, G. B. Dijksterhuis *et al.*, *Procrustes problems*. Oxford University Press on Demand, 2004, vol. 30.

[60] R. Subramanian *et al.*, “Orientation invariant gait matching algorithm based on the Kabsch alignment,” dans *IEEE International Conference on Identity, Security and Behavior Analysis (ISBA 2015)*. IEEE, 2015, p. 1–8.

[61] S. L. Bowman *et al.*, “Probabilistic data association for semantic slam,” dans *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, p. 1722–1729.

[62] C. E. Shannon, “A mathematical theory of communication,” *Bell system technical journal*, vol. 27, n°. 3, p. 379–423, 1948.

ANNEXE A SIMULATION DE LA TRAJECTOIRE DU ROBOT

Le robot a une trajectoire elliptique dans le plan $z = 0$. Figure A.1 montre le schéma `Simulink` de la trajectoire.

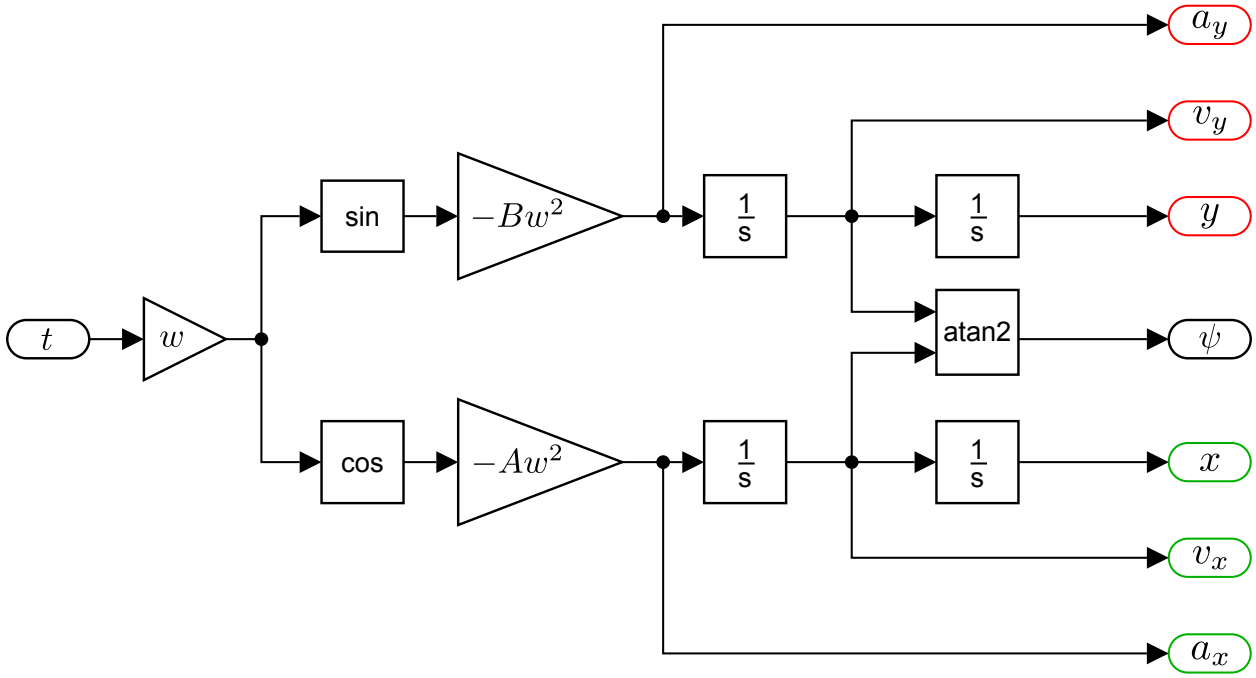


Figure A.1 Fichier Simulink générant la trajectoire elliptique du robot.

$$\begin{cases} a_x(t) = -Aw^2\cos(wt) \\ a_y(t) = -Bw^2\sin(wt) \end{cases} \quad \begin{cases} v_x(t) = -Aw\sin(wt) \\ v_y(t) = Bw\cos(wt) \end{cases} \quad \begin{cases} x(t) = A\cos(wt) \\ y(t) = B\sin(wt) \\ \psi(t) = \text{atan2}(v_y(t), v_x(t)) \end{cases}$$

Les paramètres A, B et w représentent respectivement les deux axes de l'ellipse et la pulsation (pour régler la vitesse du robot).

ANNEXE B CARACTÉRISTIQUE DU CAPTEUR LASER

Le tableau ci dessous présente les caractéristiques du capteur Velodyne LiDAR ULTRA Puck™ [1] et de notre capteur simulé.

Caractéristiques	Capteur Velodyne	Capteur simulé
Distance maximum	200m	30m
Distance écart type	3cm	10cm
Champ de vu horizontal	360°	360°
Champ de vu vertical	40°	45°
Résolution horizontale	0.1°	1°
Résolution verticale	0.33° (min)	1°
Nombre de faisceau	32	45
Vitesse de rotation	300 RPM	300 RPM
Fréquence des mesures	18kHz	1.8kHz