Navigation autonome et commande référencée capteurs de robots d'assistance à la personne

Hela Ben Said

Thèse de doctorat

Université de Limoges

Université de Limoges

ED 610 - Sciences et Ingénierie des Systèmes, Mathématiques, Informatique (SISMI)

XLIM-UMR CNRS/ Université de Limoges 7252

Thèse pour obtenir le grade de
Docteur de l'Université de Limoges
Science et Ingénierie pour l'Information, Mathématiques

Présentée et soutenue par
Hela Ben Said

Le 23 mars 2018

Navigation autonome et commande référencée capteurs de robots d'assistance à la personne

Thèse dirigée par Ouiddad LABBANI-IGBIDA et Joanny STEPHANT

JURY :

Président du jury
M.Youcef Mezouar, Professeur des universités, Sigma-Clermont

Rapporteurs
M. Patrick Rives, Directeur de recherche, INRIA Sophia Antipolis
M. Pierre Melchior, Maître de conférences HDR, Université Bordeaux 1

Examinateurs
M. Emmanuel Moulay, Chargé de recherche, XLIM

À mes parents et mes sœurs,

La suprême récompense du travail n'est pas ce qu'il vous permet de gagner,
mais ce qu'il vous permet de devenir
John Ruskin (1819-1900)

Remerciements

Ce travail a été réalisé au sein du laboratoire XLIM de l'Université de Limoges dirigé par Dominique Baillargeat. Je le remercie pour son accueil.

Mon travail a été effectué au sein de l'équipe de Robotique et Mécatronique (Remix) de l'axe Système et Réseaux Intelligents (SRI) sous la direction de Ouiddad Labbani-Igbida et Joanny Stéphant. Je tiens à leur exprimer ma reconnaissance. Ils m'ont proposé un sujet de thèse extrêmement enrichissant. Je les remercie de m'avoir accueillie et soutenue durant ce travail. Cette expérience m'a appris beaucoup de choses sur le plan scientifique et personnel. Cela n'aurait pas été possible sans les conseils, la disponibilité de mes encadrants, leur patience et surtout leur confiance.

Je remercie Messieurs Patrick Rives et Pierre Melchior d'avoir accepté d'être les rapporteurs de ce travail de thèse. Je remercie également Messieurs Youcef Mezouar et Emmanuel Moulay d'avoir été présents en tant qu'examinateurs de ce travail.

Je remercie Romain Marie pour sa patience et son aide tant sur le côté personnel que professionnel.

Je remercie mes collègues les doctorants Bilel, Karima, Fatima, Brice, Guillaume avec qui j'ai passé toutes ses années.

Je tiens enfin à remercier les personnes proches qui m'ont toujours soutenue, supportée et encouragée. Je n'oublierai jamais leur présence surtout lors des journées difficiles. Merci à ma famille, mes parents, mes deux sœurs Dorra et Ibtissem, mon neveu Yassine et ma nièce Kenza qui peuvent me donner le sourire même dans les moments difficiles.

Hela BEN SAID

Droits d’auteurs

Table des matières

Notations

Modélisation et calculs

$\left[\frac{d}{dt}\mathbf{X}\right]_{\mathcal{B}_k}$ dérivée du vecteur $\mathbf{X}$ par rapport au temps et par rapport à la base de dérivation $\mathcal{B}_k$

O_W	:	Origine du repère monde
O_R	:	Origine du repère robot
O_C	:	Origine du repère caméra
O_I	:	Origine du repère image
$\mathcal{B}_W$	:	Base associée au repère du monde
$\mathcal{B}_R$	:	Base associée au repère du robot
$\mathcal{B}_C$	:	Base associée au repère de la caméra
$\mathcal{B}_I$	:	Base du repère associé à l'image
M_W	:	Un point de la ligne médiane d'un couloir
M	:	Projection du point M_W sur le plan image
v	:	Consigne de vitesse linéaire du robot $[m/s]$
ω	:	Consigne de vitesse angulaire du robot $[rad/s]$
$\begin{pmatrix} - \\ - \\ - \end{pmatrix}_{\mathcal{B}_k}$	:	Vecteur projeté dans la base $\mathcal{B}_k$
$\mathcal{T}_{\mathcal{F}_b \leftarrow \mathcal{F}_a}$	:	Matrice de passage du repère $\mathcal{F}_a$ au repère $\mathcal{F}_b$ définie en coordonnées généralisées : $$[\mathcal{T}_{\mathcal{F}_b \leftarrow \mathcal{F}_a}] = \begin{pmatrix} \mathcal{R}_{\mathcal{F}_b \leftarrow \mathcal{F}_a} & \mathbf{t}_{\mathcal{F}_b \leftarrow \mathcal{F}_a} \\ \mathbf{0}_{(1\times 3)} & 1 \end{pmatrix}$$ avec $\mathcal{R}_{\mathcal{F}_b \leftarrow \mathcal{F}_a}$ la matrice de rotation du repère $\mathcal{F}_a$ vers le repère $\mathcal{F}_b$ et $\mathbf{t}_{R_b \leftarrow R_a}$ est le vecteur de translation qui ramène l'origine de $\mathcal{F}_a$ vers $\mathcal{F}_b$, $\mathbf{t}_{R_b \leftarrow R_a} = \overrightarrow{O_a O_b}$.

Commande référencée capteur

$\mathbf{L_{sr}}$	:	Matrice d'interaction
$\mathbf{s}$	:	Signal du capteur appelée signature.

$\mathbf{s}^*$	:	Signature désirée.
ε	:	Écart entre signature mesurée et signature désirée.
$\mathbf{T}_C$	:	Torseur cinématique du capteur
$\mathbf{P}_{\mathbf{T} \leftarrow \mathbf{U}}$	:	Matrice de passage entre la commande et la vitesse de la caméra dans le repère monde
$\mathbf{J}_s$	:	Matrice jacobienne qui relie l'évolution de la signature et les mouvements du robot (vitesses linéaire et angulaire), $\mathbf{J}_s = (\mathbf{J}_v, \mathbf{J}_\omega)$.

Abréviations

DMA	:	Delta Medial Axis, définit l'axe médian en utilisant un paramètre δ relatif à la dimension du robot.
DVG	:	Diagramme de Voronoï Généralisé
EKF	:	Extended Kalman Filter, Filtre de Kalman étendu.
IBVS	:	Image-Based Visual Servoing : asservissement visuel basé sur une image 2D.
PBVS	:	Position Based Visual Servoing : asservissement visuel basé sur une position 3D.

Prédiction du déplacement

$\mathbf{X}$	:	État
$\hat{\mathbf{X}}$	:	État estimé
$\mathbf{U}$	:	Commande
$\mathbf{Y}$	:	Mesure
$\hat{\mathbf{Y}}$	:	Mesure estimée par le modèle
$\mathbf{l}$	:	Gain de l'observateur
$\mathbf{P}_s$	:	Point du squelette
$\mathbf{F}_k\,, \mathbf{G}_k\,, \mathbf{C}_k$	:	Matrices de la représentation d'état liées à la linéarisation autour du point estimé.
ε_k	:	Écart entre la mesure et la mesure estimée, appelé innovation.
$Cov_{\mathbf{P}}$	:	Matrice de variance/covariance, elle définit l'ellipse d'incertitude sur la position du point $\mathbf{P}_s$
$Cov_{\theta\rho}$	:	Matrice de variance/covariance pour l'approximation linéaire (θ, ρ) sur le point $\mathbf{P}_s$
$Cov_{\mathbf{U}}$	:	Matrice de variance/covariance associée à la commande.
$\mathbf{S}_k$	:	Matrice de variance/covariance de l'innovation.
$\mathbf{K}_k$	:	Gain de Kalman.
$\mathbf{R}_k$	:	Matrice de variance/covariance sur l'erreur de mesure.

Introduction générale

Présentation du contexte

Actuellement plus d'un milliard de personnes vivent avec un handicap. C'est environ 15% de la population mondiale selon la base des estimations démographiques pour 2010. Selon le rapport mondial sur le handicap, le taux des personnes handicapées est en augmentation. La cause principale est liée au vieillissement de la population [Organisation Mondiale de la Santé, 2011].

En conséquence, il y a de nombreuses demandes en matière de robotique et d'automatisation. De nos jours les robots sont présents avec de nouveaux services d'assistance : aide à domicile, de la distribution de médicaments et d'aides aux médecins pour être en contact avec des patients à distance.

Pour les personnes à mobilité réduite, une simple assistance peut représenter une liberté de mouvements. La question de l'assistance est vaste et complexe et nous avons choisi de concentrer nos efforts sur l'assistance de personnes à mobilité réduite. Le mode de déplacement choisi leur donne la possibilité d'une autonomie de mouvements en extérieur ou en intérieur, il s'agit d'un fauteuil roulant. Notre contribution s'est portée en particulier sur l'instrumentation et l'augmentation de l'autonomie. Un fauteuil intelligent se compose généralement d'une base roulante électrique standard à laquelle un ordinateur et un ensemble de capteurs ont été ajoutés.

Les fauteuils roulants intelligents ont été conçus pour fournir une assistance de navigation à l'utilisateur de différentes façons. Par exemple, il peut assurer un déplacement sans collision et transporter de manière autonome l'utilisateur entre différents lieux. Les développements de fauteuils roulants intelligents a commencé depuis le début des années 1980.

Le premier fauteuil roulant a été conçu en 1933 par les ingénieurs en mécanique Harry Jennings et Herbert Everest. Depuis, cette machine a évolué pour être déplacée par un moteur électrique et des commandes de navigation, généralement, un joystick monté sur l'accoudoir. Ce type de fauteuil est commandé par l'utilisateur par une autre personne si l'utilisateur ne peut pas le commander lui-même. Un fauteuil roulant peut être autonome (intelligent) pour aider un utilisateur ayant un handicap physique. Généralement, un fauteuil roulant intelligent, contrôlé par un ordinateur, dispose de plusieurs capteurs et applique des techniques de robotique mobile pour se déplacer. L'objectif est de réduire ou d'éliminer la tâche de conduite par des utilisateurs d'un fauteuil roulant motorisé. Dans ce contexte, notre travail participe à chercher des solutions à la navigation autonome en temps réel applicables à un fauteuil roulant électrique.

Problématique

Si l'utilisateur d'un fauteuil roulant n'a pas la capacité de le commander (paralysie des membres supérieurs, cécité, ...), il faut rendre ce fauteuil roulant autonome, c'est-à-dire lui donner la capacité de se déplacer sans intervention humaine. Pour cela le système doit être capable de percevoir lui-même son environnement, détecter tout ce qui l'entoure et qui peut gêner son mouvement (murs, personnes,...), puis déterminer une trajectoire sûre et la suivre en toute sécurité et d'une manière confortable pour l'utilisateur. C'est pour cela que nous avons traité dans cette thèse le sujet du déplacement autonome d'un robot mobile d'assistance à la personne.

Pour être totalement autonome, un agent mobile a besoin d'une stratégie sûre et efficiente pour naviguer dans la scène. Lorsque l'environnement est initialement inconnu, c'est-à-dire lorsque la connaissance du robot est limitée à sa perception, la tâche peut être difficile. Le but de cette thèse est de concevoir une commande pour rendre un robot mobile non holonome totalement autonome pour le déplacement dans un environnement inconnu. La commande des robots mobiles est l'objet d'étude de beaucoup de travaux de recherche depuis de nombreuses années.

La construction d'une loi de commande dépend de la tâche à réaliser. S'il s'agit d'atteindre une cible, il faut avoir une commande fonction de la position de la cible et de la position du robot. S'il s'agit d'un suivi de trajectoire il faut construire une commande en fonction des paramètres de la trajectoire et de la position du robot. Cependant, cette dernière ne peut pas toujours être extraite facilement. Il faut des systèmes physiques ou des algorithmes complexes pour avoir une bonne précision sur la position du robot. Un autre type de commande existe, il s'agit de la commande référencée capteur (CRC). Des informations provenant de capteurs extéroceptifs sont utilisées pour concevoir la CRC. Sans avoir une modélisation de l'environnement, la CRC peut asservir la position de robot qui sera basée directement sur une information extraite ou bien calculée à partir de données brutes de capteurs. La CRC était initialement conçue pour les robots manipulateurs. Dans [Samson et al., 1991], le formalisme général de la CRC destinée à des bras manipulateurs est détaillé.

La CRC pour les robots mobiles était peu étudiée auparavant mais maintenant, les travaux de recherche s'intéressent de plus en plus à utiliser ce type de commande pour la navigation d'un robot mobile. Les capteurs visuels ont une grande utilité dans les applications robotiques, ils ont d'ailleurs fait l'objet de nombreux travaux liés à la CRC. Ces approches sont qualifiées d'approches d'asservissement visuel. Des informations sont évaluées dans l'image qui servent à asservir la position du robot. Ces informations visuelles doivent être convenablement choisies et doivent rester visibles parce qu'elles servent à calculer les vitesses de consignes envoyées au robot. En revanche le champ de vue est limité et ces caractéristiques visuelles risquent de ne pas être perçues dans l'image. En outre, la CRC est sujette au bruit ou à la défaillance du capteur ce qui impacte directement le calcul de la loi de commande. Par conséquent la sécurité de la navigation n'est pas garantie.

Contributions

Cette thèse traite le problème de la navigation autonome dans des environnement inconnus et dynamiques. Les contributions de cette thèse sont principalement :

- la conception d'un cadre virtuel unifié qui permet de regrouper toutes les informations d'un

ensemble capteurs physiques embarqués sur le robot. Ce cadre permet, dans un repère unique de définir la signature et de concevoir la commande.

- l'utilisation d'algorithmes de squelettisation de l'image, adaptés à la représentation du cadre virtuel pour des applications d'asservissement visuel et embarquables pour des applications de robotique mobile (figure 1).

FIGURE 1 – Résultat de l'algorithme de squelettisation : estimation du squelette (trait blanc) pour une forme 2D d'avion

- la conception d'une stratégie d'asservissement visuel basée observateur pour un déplacement autonome et sûr dans un environnement peu encombré (figure 2).

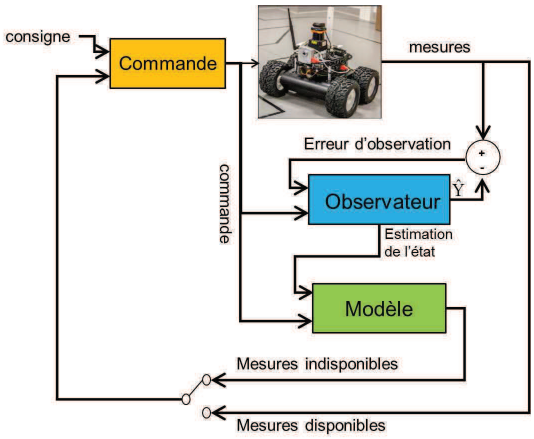


FIGURE 2 – Stratégie de la commande. Lorsque les mesures sont disponibles, la commande est calculée à partir l'état observé. Sinon, la commande est calculée à partir du modèle en utilisant comme conditions initiales la dernière observation délivrée par l'observateur.

- la conception d'une nouvelle stratégie de déplacement sur le diagramme de Voronoï en temps réel dans un environnement inconnu. Cette stratégie se base sur un asservissement visuel dont la consigne est une linéarisation du diagramme de Voronoï généralisé.
- une analyse des propriétés de stabilité du système commandé.
- la conception d'un filtre de Kalman adapté à la prédiction du futur DVG.
- une stratégie de planification dans l'image pour le déplacement sur une prédiction du DVG.

Structuration du mémoire

L'organisation du manuscrit est illustrée sur la figure 3.

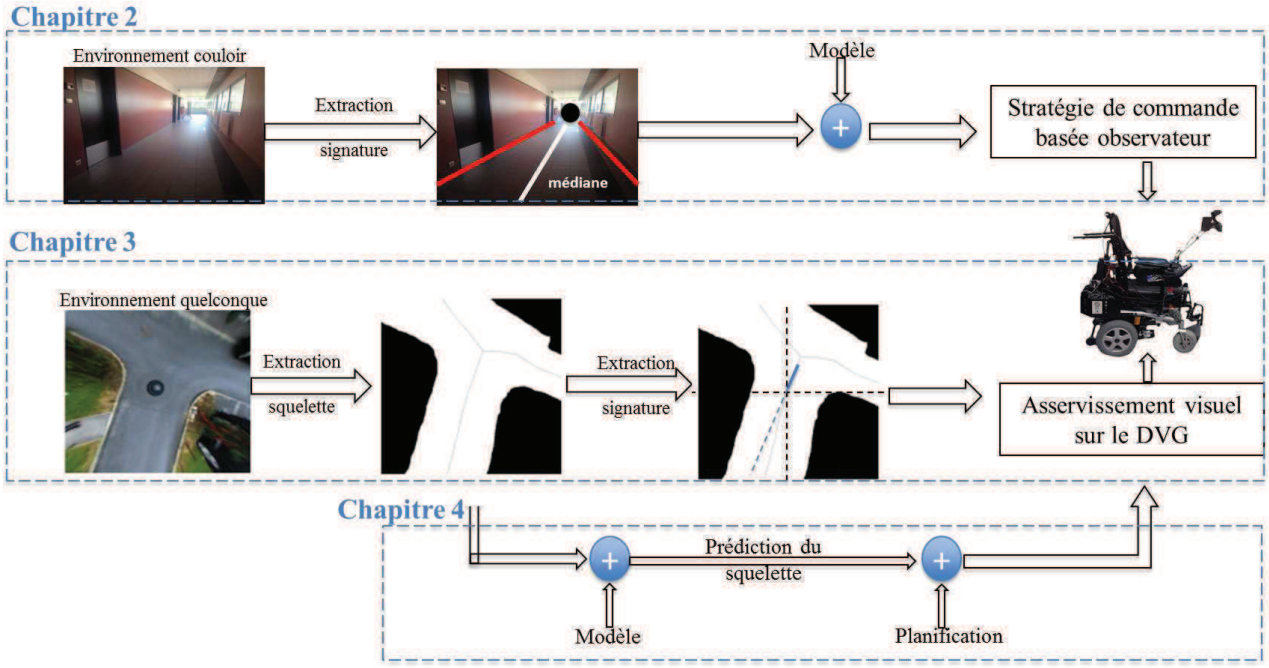


FIGURE 3 – Organisation générale des différents chapitres du manuscrit

Chapitre 1 - Navigation autonome et sûre d'un robot mobile

Le premier chapitre présente un état de l'art sur la navigation autonome des robots mobiles. L'objectif de ce chapitre est de présenter quelques outils à la base de ce travail. Avec ce chapitre nous aurons les sous-ensembles nécessaires à la conception d'une commande d'un robot mobile.

Chapitre 2 - Asservissement visuel dans un environnement idéalisé non encombré

Ce chapitre sera consacré à l'étude de la perception en temps réel d'un robot mobile, pour un déplacement autonome dans un environnement non-encombré (couloir). Un asservissement visuel dans un couloir, étudié dans plusieurs travaux, sera présenté. Ces travaux utilisent deux informations visuelles pour pouvoir naviguer sur la médiane du couloir. Nous présenterons dans ce chapitre une nouvelle stratégie de mesure et de commande basée observateur et nous prouverons que l'utilisation d'une seule information visuelle est suffisante pour assurer le suivi de couloir.

Chapitre 3 - Asservissement visuel sur le diagramme de Voronoï

Ce chapitre s'intéressera à la navigation autonome dans n'importe quel environnement. Il présentera toutes les étapes et les outils nécessaires que nous utiliserons pour commander un robot mobile dans un environnement inconnu et dynamique. La stratégie présentée est un asservissement visuel sur un DVG. Une application d'automatisation d'un fauteuil roulant sera introduite.

Chapitre 4 - Prédiction sur une linéarisation du diagramme de Voronoï généralisé

Dans ce chapitre, les limites de l'approche du chapitre précédent seront exposées et une nouvelle stratégie sera présentée pour le déplacement autonome en utilisant le DVG. Un asservissement visuel sur une prédiction d'une linéarisation du DVG sera développée en utilisant un filtre de Kalman.

Projet de recherche

Les travaux présentés dans ce manuscrit ont été réalisés au sein du laboratoire XLIM de l'Université de Limoges grâce à un financement de la région Limousin (bourse régionale). Ces travaux s'inscrivent dans la continuité des travaux menés dans le cadre du projet Européen Interreg COALAS (Cognitive Assistive Living Ambiant System) clôturé en juin 2015, portant sur l'aide à la mobilité des personnes en situation de handicap.

Navigation autonome et sûre d'un robot mobile

Résumé du chapitre

Afin de réaliser une navigation autonome et sûre, le système robotisé doit réaliser un certain nombre de missions. Cette première partie présente les systèmes autonomes et l'utilité de rendre une machine autonome. Pour rendre un système autonome il faut qu'il perçoive son environnement. La perception peut se faire en utilisant des capteurs physiques ou des capteurs logiciels. Une fois la perception opérationnelle, une étape d'extraction de données et de calcul se fait pour avoir l'information qui nous intéresse. Pour les robots mobiles, l'information utile est une trajectoire de référence pour se déplacer. Cette étape est réalisée à l'aide de la planification. Vient ensuite l'action qui consiste en le fait d'envoyer aux actionneurs les commandes nécessaires pour faire déplacer le robot. Pour réaliser toutes ces tâches, une architecture de contrôle englobe et gère toutes ces actions. Ce chapitre présente l'ensemble des concepts manipulés dans ce mémoire.

Sommaire

1.1 Introduction

Depuis plusieurs décennies, les robots peuvent être contrôlés à distance par un opérateur humain, c'est la téléopération. L'interface de contrôle peut être un joystick, un système de réalité virtuelle, ou n'importe quelle autre interface. Dans la téléopération, un contrôle semi-autonome peut exister. Dans ce cas, le robot est en charge d'une partie de sa mission. L'autonomie des robots devient de plus en plus répandue dans le monde car certains robots peuvent remplacer l'être humain lorsqu'il s'agit de faire une tâche à risque comme les applications nucléaires, spatiales ou militaires.

L'objectif de la navigation d'un robot mobile est de permettre, à partir de la perception de l'environnement, de rechercher une trajectoire dans un espace libre entre un point initial et un point final (point cible), puis de se déplacer sur cette trajectoire. Si la perception se fait au niveau du robot, la décision de mouvement peut se prendre de façon partagée entre l'opérateur et le robot. L'autonomie peut se définir comme la capacité d'un système à réagir par lui-même à ses propres observations de l'environnement, sans intervention humaine.

Les trois actions majeures de la navigation autonome sont : la perception, la décision et l'action.

- La perception représente l'aptitude du système à extraire, étudier et mettre en œuvre des données acquises par des capteurs pour permettre au robot de prendre les décisions et agir en fonction de l'environnement qui l'entoure. Il existe deux types de perception : une globale et une locale. La perception locale désigne l'utilisation d'une information centrée sur le robot. Par exemple pour une perception visuelle, une seule image peut être prise. La perception globale désigne l'utilisation d'une séquence d'informations (une séquence d'images) et un repère de référence extérieur au robot.
- La décision permet au robot de planifier la séquence d'actions pour réaliser sa mission (planification de trajectoire libre d'obstacles par exemple). Cela se fait à partir de l'analyse de la perception.
- L'action consiste à mettre en œuvre des commandes ou des consignes aux actionneurs. Cette étape se fait généralement par des asservissements de bas niveau.

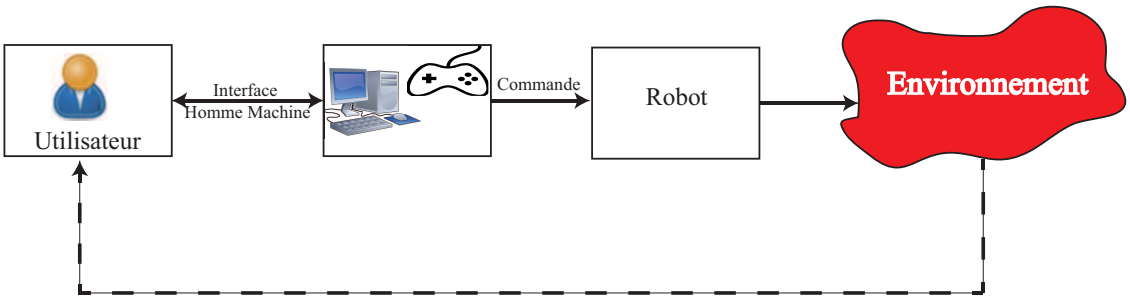


FIGURE 1.1 – Le schéma de la téléopération

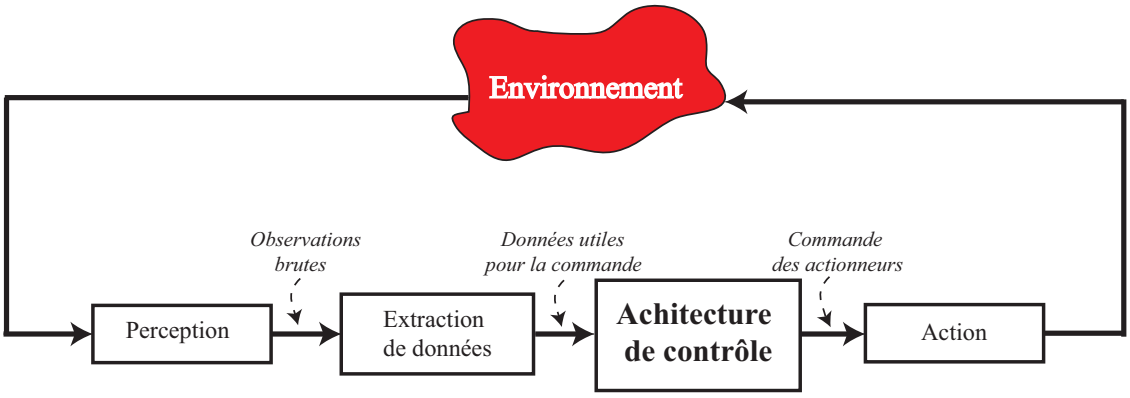


FIGURE 1.2 – Le schéma général pour la commande d'un robot autonome

1.2 De la téléopération à l'autonomie

1.2.1 Téléopération

Chaque système robotisé qui est commandé par un système externe, est dit téléopéré c'est-à-dire piloté à distance (figure 1.1). Dans ce cas, l'opérateur et le robot ont un certain type de relation maître-esclave. Dans la plupart des cas, l'opérateur humain se trouve à un poste de travail et dirige un robot à travers une interface. Celle-ci peut être un joystick, un ordinateur, un dispositif de réalité virtuelle, une interface haptique, un smartphone, etc. L'opérateur humain, est souvent désigné comme le local (en raison d'être à la station de travail locale) et le robot comme le système robotisé qui fonctionne à distance (il fonctionne à un endroit éloigné du téléopérateur). Le local doit avoir des retours d'informations et des mécanismes de contrôle, tandis que le système doit avoir des capteurs et des effecteurs. Le téléopérateur n'observe pas directement l'action du système à distance, soit parce que le robot est physiquement éloigné (par exemple sur Mars), soit parce que la personne doit être protégée (par exemple, dans une cellule chaude d'une usine de traitement nucléaire ou pharmaceutique) [Murphy, 2000].

1.2.2 Semi-autonomie

La méthode de téléopération peut être semi-autonome. Dans ce cas, elle est souvent appelée contrôle de supervision, où le robot reçoit une instruction ou une tâche partielle qu'il peut faire en toute sécurité. Deux classes de contrôle semi-autonome existent : l'assistance continue ou le contrôle partagé [Murphy, 2000]. Des exemples de robots semi-autonomes sont présentés dans [Ali, 2011], [Skrabánek et al., 2016], [Sequeira et Ribeiro, 2004]. Ces travaux traitent du contrôle des robots ayant des tâches qui peuvent être spécifiquement définies par des opérateurs externes d'où l'appellation de "semi-autonome".

1.2.3 L'autonomie

L'autonomie peut se définir par l'aptitude du robot à réaliser seul sa mission. Le robot autonome peut réagir à n'importe quel environnement (statique ou dynamique) qu'il doit percevoir à l'aide de capteurs et agir à l'aide d'un ensemble d'actionneurs (figure 1.2). Depuis le début des années 1960,

les recherches de la navigation sur les robots mobiles ont progressivement augmenté. Aujourd'hui, les robots mobiles autonomes effectuent des tâches diverses avec une vitesse élevée et une précision avec sécurité. L'article [Pol et Murugan, 2015] résume les différentes techniques utilisées pour la navigation des robots mobiles autonomes. Parmi ces techniques, nous trouvons les méthodes de commande avec la logique floue qui peuvent être utilisées pour faire face à l'incertitude de l'environnement lorsqu'il est dynamique. Le robot est ainsi réactif pour agir rapidement face aux changements de l'environnement. Il est également capable de prédire l'état futur de son environnement [Ayari et al., 2010]. Il y a aussi les méthodes de navigation par radio dont l'efficacité a été prouvée pour la navigation autonome en utilisant le système RFID [Park et Hashimoto, 2009]. Chaque robot autonome a une certaine capacité d'adaptation à un environnement inconnu et pour la réalisation de tâches prédéfinies. Ses capacités ont permis de s'intégrer dans le monde industriel, où l'être humain est remplacé dans les tâches difficiles, répétitives ou à risque. L'autonomie d'un robot peut être résumée par la connaissance du robot de la commande envoyée aux actionneurs à chaque pas de temps. Cette connaissance est dépendante de la mission à accomplir et les informations données par les différents capteurs. Il s'agit donc de construire une architecture de contrôle qui permet de définir un lien entre la perception et l'action connaissant l'objectif final.

1.3 Architectures de contrôle

Une fois que les trois actions majeures à réaliser (perception, décision et action) sont définies, il faut établir les relations qui existent entre elles. C'est le rôle de l'architecture de contrôle. Elle permet de faire la coordination de différents processus internes d'un système robotisé. Les architectures de contrôle sont classées en trois catégories : délibérative, réactive et hybride. La figure 1.3 schématise ces différentes approches. Chacune présente de nouveaux concepts et solutions pour résoudre le problème de navigation. Les sections qui suivent présentent chaque architecture. Pour plus de détails sur les architectures de commande, une synthèse complète a été réalisée dans [Nakhaeinia et al., 2011].

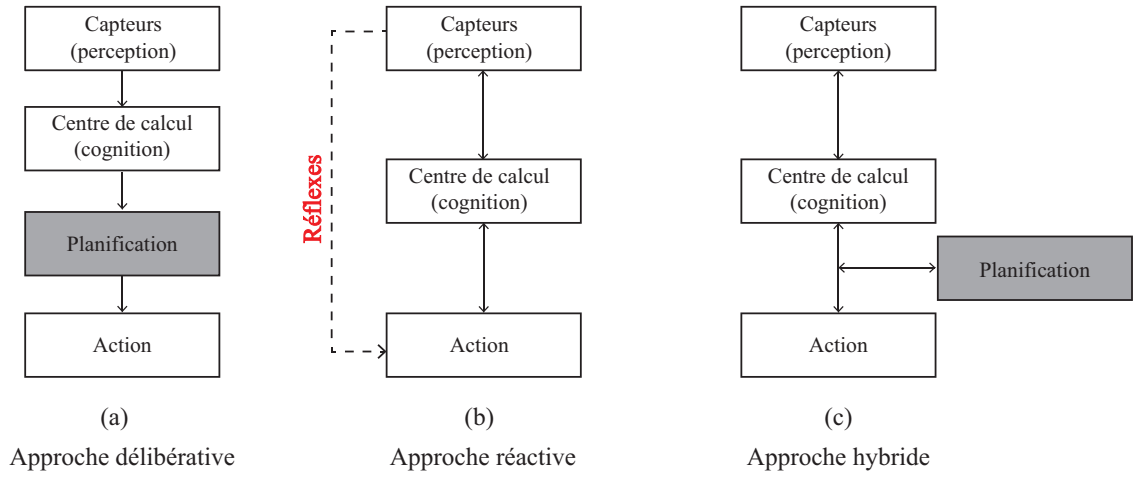


FIGURE 1.3 – Les différentes architectures de contrôle

1.3.1 Approche délibérative

L'approche délibérative (figure 1.3.a) dite aussi "hiérarchique" consiste à relier d'une manière séquentielle les trois actions : percevoir, planifier et agir [Chatila et Laumond, 1985]. Après avoir perçu son environnement local, le système robotique doit toujours mettre à jour sa perception globale (une carte de l'environnement par exemple) et puis passer à la planification. **La planification** pour la navigation est une étape très importante pour les robots autonomes. Elle permet à un robot mobile de calculer une trajectoire pour atteindre une cible tout en respectant des contraintes qui peuvent être liées à l'environnement (évitement de collision) ou la dynamique du robot. La planification délibérative recherche un chemin optimal et génère une trajectoire pour atteindre une cible, cette étape sera détaillée dans la section 1.4. Ensuite, le système d'exécution est appliqué pour effectuer une action dans le contexte du modèle statique de l'environnement et du système de planification pour accomplir une tâche donnée [Huq et al., 2008]. Cette architecture nécessite un modèle de l'environnement précis.

1.3.2 Approche réactive

L'approche réactive a été développée à partir des années 1980 pour faire face aux limites de l'approche délibérative dans des environnements dynamiques et inconnus [Seraji et Howard, 2002]. C'est une approche généralement utilisée en temps réel. Elle ne nécessite pas de modéliser l'environnement ni de planifier une trajectoire à l'avance. Il s'agit d'agir directement après la perception [Ye et Wang, 2000]. Cette architecture génère des commandes de contrôle basées sur l'environnement actuellement perçu.
Le contrôle robotique basé sur une approche réactive produit des performances robustes dans des environnements complexes et dynamiques. Les hypothèses prises pour déployer les systèmes purement réactifs sont que :

- l'environnement manque de cohérence temporelle et de stabilité,
- il est difficile de localiser un robot par rapport à un repère fixe tout au long de la mission,
- la connaissance symbolique du monde représentatif a peu ou pas de valeur.

1.3.3 Approche hybride

L'architecture hybride a été initialement développée par [Arkin, 1998] et [Murphy, 2000]. Cette approche combine l'aspect de temps réel et la rapidité des réponses retrouvées dans l'approche réactive dans des environnements dynamiques ou inconnus, et l'efficacité de la planification de l'approche délibérative. L'architecture de contrôle hybride a été développée pour présenter de nouvelles approches pour obtenir des systèmes de contrôle de supervision qui utilisent des architectures de contrôle réactives et délibératives [Minguez et Montano, 2005], [Du et al., 2007]. Ces travaux ont utilisé l'architecture hybride avec les trois étapes (modélisation, planification et réaction). Il s'agit d'un système de commande basé capteur pour la navigation des robots mobiles sans collision dans des environnements inconnus, encombrés et dynamiques.

1.4 Planification de chemin

La planification de chemin se définit par la recherche d'un mouvement libre de toute collision entre un robot possédant un nombre variable de degrés de liberté et son environnement. Ce chemin se définit par un point de départ et un point d'arrivée [Halperin, 1994]. La planification de chemin dépend en grande partie de l'environnement dans lequel le robot va se déplacer. L'environnement pour un robot mobile est constitué d'un espace libre (navigable) et un espace occupé (non navigable) par des obstacles. Le robot doit détecter avec sa perception ces deux espaces pour pouvoir bien naviguer. Avant d'entamer la recherche d'un chemin approprié pour le robot il faut modéliser son environnement pour distinguer les espaces navigable et non navigable. C'est-à-dire qu'il faut construire une carte de l'environnement. Il existe plusieurs types de cartes (topologique, métrique, hybride...). Une fois les cartes acquises, elles permettent diverses fonctions clés nécessaires à la navigation de robot mobile, telles que la localisation, la planification des chemins, l'évitement des collisions et la détermination de cibles d'intérêt [Thrun, 2003].

Dans cette partie il faut distinguer ces trois termes : mouvement, chemin et trajectoire.

Un mouvement est la manière de relier un point A à un instant initial à un point B à un instant final.

Un chemin est le support de mouvement, il peut, par exemple, être représenté comme une courbe paramétrée en fonction d'une abscisse curviligne.

Une trajectoire est le chemin défini en fonction de temps, c'est-à-dire l'évolution du paramètre de l'abscisse curviligne en fonction du temps.

L'étape de planification nécessite une modélisation de l'environnement. Différentes représentations existent pour modéliser un environnement et seront présentés dans la section suivante. Plusieurs méthodes de planification de trajectoires seront détaillées dans la partie (1.4.2).

1.4.1 Les représentations de l'environnement

Pour permettre à un robot mobile de naviguer dans un environnement inconnu à priori, il faut construire une représentation de l'environnement à partir de la perception. Il est aussi très important dans une navigation autonome que le robot se localise par rapport à son environnement. La modélisation de ce dernier est donc indispensable.

Les cartes topologiques

Une carte topologique mémorise un ensemble de régions (bureaux, couloirs, intersections, rues, champs...) ainsi que les liens qui existent entre ces régions (figure 1.4). Cette carte est représentée comme un graphe. Ce dernier est constitué de nœuds qui sont les régions (lieux) que le robot peut atteindre et d'arêtes qui permettent de lier les nœuds (chemins) [Marinho et al., 2017], [Gerstmayr-Hillen et al., 2013]. L'avantage de cette représentation est qu'elle ne requière pas un modèle métrique issu de la fusion de données de capteurs proprioceptifs.

Les cartes métriques

La carte métrique représente les caractéristiques d'objets perçus (position, longueur, distance à un autre objet) dans un espace métrique (figure 1.4). Ce dernier est généralement exprimé en trois dimensions et définit un référentiel unique pour toutes les informations perçues. Les objets détectés dans cette carte correspondent aux obstacles que le robot pourrait rencontrer dans son environnement et représentent l'espace occupé. Le complémentaire correspond alors à l'espace navigable, c'est-à-dire, à l'espace dans lequel le robot peut se déplacer. Cette vision objective de l'environnement, plutôt indépendante de tout robot donné, facilite également la réutilisation de ces cartes par différents robots. Pour l'être humain, ces cartes sont plus faciles à comprendre car elles correspondent à sa propre perception [Ghazouani, 2012]. Les cartes métriques sont plus faciles à construire que les cartes topologiques en raison de la définition non ambigüe des lieux fournie par leurs coordonnées [Filliat et Meyer, 2003].

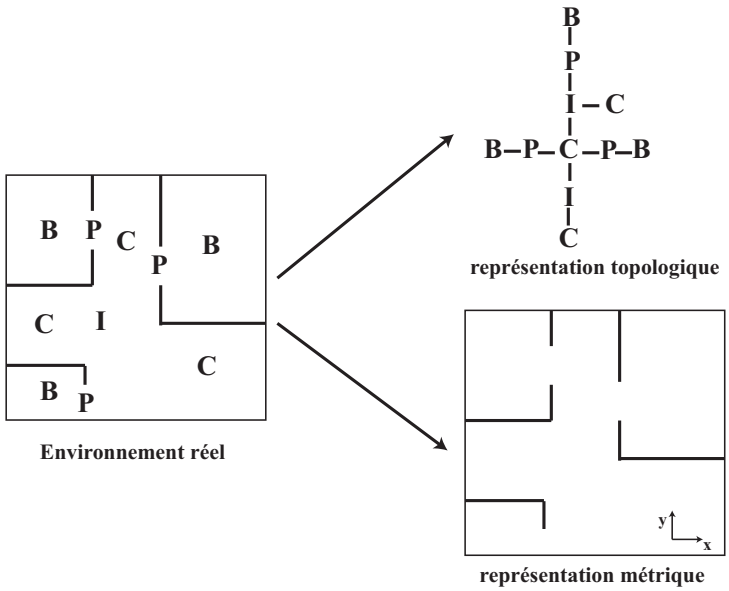


FIGURE 1.4 – Les représentations de l'environnement : la représentation topologique mémorise des lieux et des liens de déplacement, dans cet exemple **B**=Bureau, **P**=Porte, **C**=Couloir, **I**=Intersection. La représentation métrique mémorise des objets (murs) avec une position dans un repère global.

Les grilles d'occupation

Les cartes basées sur les grilles d'occupation sont des représentations métriques discrétisées de l'environnement du robot. Elles représentent des environnements par des cellules métriques qui contiennent des informations relatives à l'occupation de l'environnement. Chaque cellule contient un indice qui indique si l'espace correspondant est plutôt libre ou occupé. Cette représentation requiert très peu de temps de calcul, ce qui revient à une mise à jour d'une carte rapide et facile [Ghazouani, 2012]. Le principe d'inférence bayésienne [Elfes, 1989] et les fonctions de croyance [Moras et al., 2015] peuvent être utilisé pour estimer la probabilité d'occupation de chaque cellule.

Les cartes hybrides

La représentation basée sur les grilles produit des cartes métriques précises et peut être utilisée pour une localisation plus précise et un évitement d'obstacles. Cependant, si une représentation métrique est utilisée dans des environnements de grandes tailles, la complexité de l'espace et du calcul deviendra non négligeable. La représentation topologique peut être utilisée beaucoup plus efficacement pour la planification de chemins. La précision et la cohérence des données sont difficiles pour la carte topologique dans un environnement à grande échelle. Ainsi, la carte hybride qui combine les deux approches devient plus appropriée pour représenter l'environnement. Cela va permettre de surpasser les limitations de chaque représentation et d'exploiter les avantages de chacune. Les cartes hybrides sont des cartes topologiques qui contiennent des informations métriques sur les arêtes et les nœuds du graphe [Schmuck et al., 2016], [Jia et al., 2012].

1.4.2 Méthodes de planification de trajectoire

Après avoir modélisé l'environnement à partir de la perception du robot, il faut se déplacer en évitant les obstacles. La détermination de ce déplacement se traduit par la recherche d'un chemin qui assure d'être le plus loin possible des collisions.

Graphe de Voronoï

Le graphe de Voronoï (GV) est une méthode très utilisée pour la représentation d'une configuration de l'espace navigable. L'idée de base est de générer une ligne, appelée arête de Voronoï, équidistante de tous les points (figure 1.5b). Le point où plusieurs arêtes de Voronoï se rencontrent est connu comme un sommet de Voronoï. Ce dernier représente généralement une correspondance physique aux configurations qui peuvent être détectées dans l'environnement. Cela rend la tâche de suivi de chemin beaucoup plus facile. S'il existe une stratégie implicite de contrôle local pour rester équidistant de tous les obstacles (le robot suit l'arête de Voronoï) alors il n'entrera pas en collision avec les obstacles, car il restera au "milieu" de l'espace libre. Lorsque les obstacles sont des polygones, le diagramme de Voronoï se compose de segments droits et paraboliques [Murphy, 2000],[Aurenhammer et al., 2013], [Iwaszko, 2012]. Dans cette thèse, nous nous intéressons à cette approche. Cette méthode sera plus détaillée dans la section.1.4.3.1.

Graphe de Visibilité

La méthode du graphe de visibilité est l'une des premières méthodes de planification de chemin. Elle utilise les extrémités des obstacles que le robot doit éviter. Elle s'applique principalement aux espaces de configuration bidimensionnels avec une région polygonale $C_{obstacle}$. Le graphe de visibilité est un graphe G dont les nœuds sont les configurations initiale et finale, ainsi que tous les sommets des obstacles. Les liens de G sont les lignes qui relient deux nœuds qui ne coupent pas l'intérieur de la région $C_{obstacle}$ [Rekleitis et al., 2004](figure 1.5a).

Décomposition cellulaire

La méthode de décomposition cellulaire consiste à décomposer l'espace des configurations libres du robot en un certain nombre d'ensembles disjoints appelés cellules (figure 1.5). Le graphe de connectivité est un élément important pour cette décomposition, il capture la structure de l'espace de configuration. Chaque cellule est représentée comme un nœud dans ce graphique. Deux nœuds sont connectés par un bord si et seulement si les deux cellules correspondantes sont adjacentes. La planification du chemin entre deux configurations est réalisée en deux étapes : 1) recherche du graphe de connectivité et du chemin qui relie les deux configurations choisies, 2) détermination de la résolution du problème de planification en utilisant les cellules adjacentes trouvées dans la première étape.

Cette approche a été introduite par [Keil et Sack, 1985] puis utilisée dans plusieurs travaux [Lingelbach, 2004], [Cai et Ferrari, 2009]. Elle peut être utilisée pour la planification de plusieurs trajectoires de plusieurs robots comme il a été présenté dans [Swingler et Ferrari, 2010].

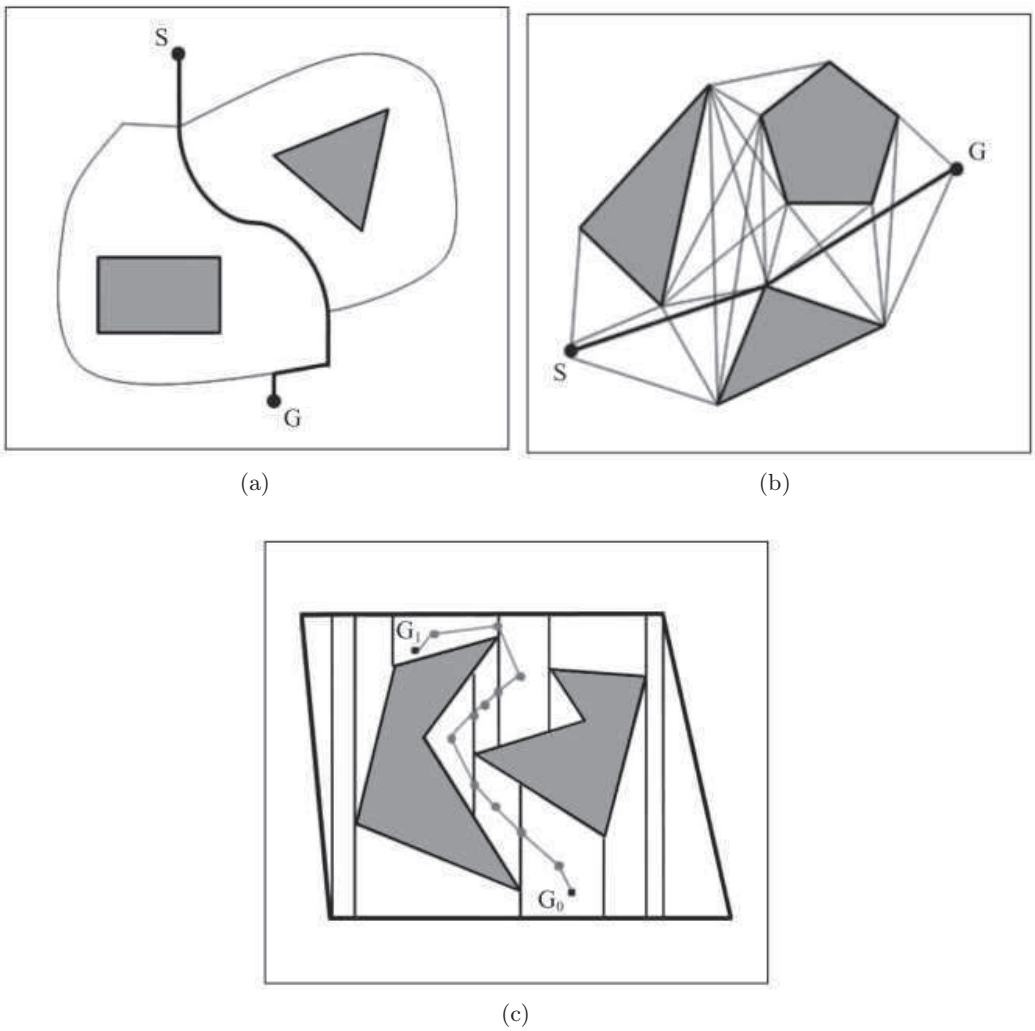


FIGURE 1.5 – Les différents graphes pour la planification de trajectoire extraits de [Tang et al., 2012] (a) Graphe de Vononoï, (b) Graphe de visibilité et (c) Décomposition cellulaire.

Champs de potentiel

La méthode de planification par les champs potentiels a été introduite par [Khatib, 1986]. Il s'agit de définir une fonction potentiel dans l'espace libre qui présente un minimum global à la configuration de l'objectif et de suivre sa plus forte pente. La fonction de potentiel peut être définie sur l'espace libre comme la somme d'un potentiel attractif tirant le robot vers la configuration d'une cible et d'un potentiel répulsif éloignant le robot des obstacles. Pour réaliser cela, l'espace dans lequel le robot doit naviguer est traité comme un champ de potentiel. La position cible pour le robot est considérée comme un minimum global et les obstacles sont considérés comme des maxima globaux [Choset, 2005].

1.4.3 Diagramme de Voronoï Généralisé

Les diagrammes de Voronoï ont été définis et étudiés depuis 1908 par le mathématicien russe Georgy Fedoseevich Voronoy. Cette représentation a été utilisée dans plusieurs applications mais a été conçue initialement pour la géophysique et la météorologie. De nos jours, elle est utilisée dans presque tous les domaines : médecine (diagnostic de cellules cancéreuses), robotique (planification de trajectoire pour les robots mobiles), étude de matériaux (modélisation de structures)...

1.4.3.1 Diagramme de Voronoï

Les diagrammes de Voronoï sont des structures de données fondamentales qui ont fait l'objet d'études approfondies en géométrie algorithmique [Iwaszko, 2012].

Ce diagramme est défini pour un ensemble fini de n points appartenant à $\mathbb{R}^d$, $P = \{p_1, p_2, ..., p_n\}$. Il s'agit de subdiviser l'espace en plusieurs régions (n-régions) appelées cellules, telles que chaque cellule $V(p_i)$ est constituée de points qui sont plus proches de p_i que de p_j, avec $i \neq j$

$$V(p_i) = \left\{ x \in \mathbb{R}^d : || \, x - p_i \, || \leq || \, x - p_j \, || \;\; \forall j \leq n \right\}, \tag{1.1}$$

Les approches géométriques par modèle de Voronoï sont largement utilisées dans le traitement d'image. Les exemples de différentes utilisations de diagramme de Voronoï dans l'image sont :

- la segmentation pour l'identification des objets dans une image binaire [Dan, 2016] ou pour l'identification de la couleur [Itoh et Matsuda, 1996],
- l'effet de mosaïque dans l'image [Martínez et al., 2007],
- la squelettisation dans l'image (l'extraction d'un squelette, appelée aussi axe médian à partir d'une image binaire) [Brandt et Algazi, 1992], [Garrido et al., 2006], [Marie et al., 2016].

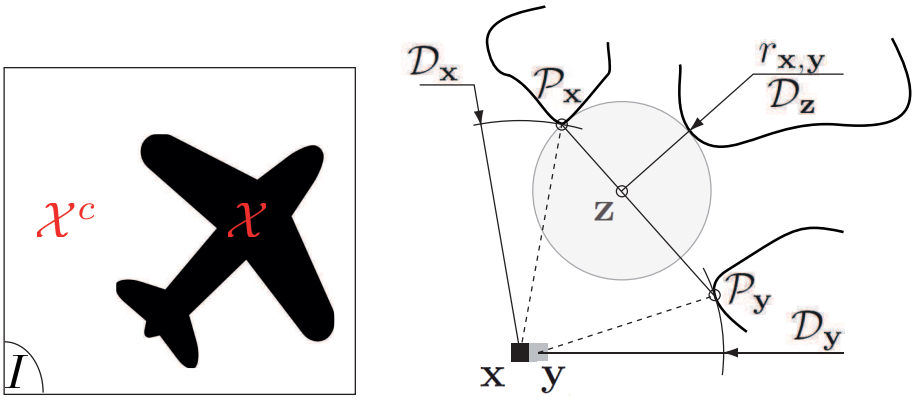
1.4.3.2 Méthodes d'extraction

À partir des mesures, une segmentation approximative de l'espace navigable local doit être disponible pour appliquer une extraction du diagramme de Voronoï. Généralement, le calcul explicite du DVG nécessite un temps de calcul important. Il est donc souvent extrait hors ligne, en utilisant une carte globale de l'environnement [Choset et Burdick, 1995], [Wilmarth et al., 1999]. Seuls quelques travaux [Mahkovic et Slivnik, 2000], [Garrido et al., 2011] proposent des solutions d'extraction en ligne. Dans [Victorino et al., 2004b], à l'aide d'une nappe laser, le robot navigue sur le DVG sans fabriquer de carte

globale. Dans [Mahkovic et Slivnik, 2000], le même capteur est utilisé pour regrouper les obstacles visibles localement et en déduire le DVG correspondant. Dans [Garrido et al., 2011], un algorithme de squelettisation hors ligne est appliqué sur une carte globale pour définir un planificateur de chemin basé sur le DVG. Ensuite, un laser est utilisé en ligne pour détecter des obstacles inconnus et éliminer les branches incorrectes de DVG. Ce qui nous intéresse pour notre travail, est une extraction précise en temps réel du DVG local. Une solution a été donnée par [Marie et al., 2016]. Un algorithme de squelettisation d'image, appelée "*Delta Medial Axis*" (DMA) est proposé. Cet algorithme fournit une extraction en temps réel et un filtrage de squelette pour toute forme binaire discrète. Avec de petites adaptations de l'espace libre et aux contraintes de dimension du robot, il produit une représentation locale du DVG, approximée par un squelette de forme. La section suivante détaille cet algorithme.

1.4.3.3 Delta Medial Axis

Les éléments suivants présentent les concepts fondamentaux à la compréhension de la méthode. Les détails relatifs à l'implémentation et aux performances de cette méthode sont présentées dans [Marie, 2014]. On désigne par $I = \mathcal{X} \cup \mathcal{X}^c$ un masque binaire. $\mathcal{X}$ désigne une forme dans I, et $\mathcal{X}^c$ désigne le complémentaire (figure 1.6a). L'axe médian (ou squelette) de $\mathcal{X}$ correspond à l'ensemble des points dans $\mathcal{X}$ à distance égale de leurs deux points les plus proches dans $\mathcal{X}^c$.



(a) (b)

FIGURE 1.6 – Les éléments utilisés pour le calcul de l'axe médian dans une image.

Les squelettes sont connus pour être très sensibles aux petites déformations de forme et ont tendance à produire des branches parasites qui ne correspondent pas aux arêtes DVG réelles. Le DMA a été spécifiquement conçu pour les identifier et les éliminer en temps réel.

Pour $\mathbf{x}$ appartenant à $\mathcal{X}$, on peut définir :

- $\mathcal{P}_\mathbf{x} = \underset{\mathbf{y}\in\mathcal{X}^c}{argmin}\, d(\mathbf{x},\mathbf{y})$ la projection de $\mathbf{x}$ ceci correspond au point le plus proche de $\mathcal{X}^c$,
- $\mathcal{D}_\mathbf{x} = d(\mathbf{x},\mathcal{P}_\mathbf{x})$ la distance euclidienne entre $\mathbf{x}$ et $\mathcal{P}_\mathbf{x}$ dans l'image,
- $N_\mathbf{x} = \{\mathbf{y} \in \mathcal{X} \mid d(\mathbf{x},\mathbf{y}) = 1\}$ l'ensemble des pixels voisins directs de $\mathbf{x}$ dans $\mathcal{X}$.

En utilisant $\mathbf{x} \in \mathcal{X}$, $\mathbf{y} \in N_\mathbf{x}$, $r_{\mathbf{x},\mathbf{y}} = \sup\{\mathcal{D}_\mathbf{z} \mid \mathbf{z} \in [\mathcal{P}_\mathbf{x}, \mathcal{P}_\mathbf{y}]\}$ désignera le rayon du plus grand disque disjoint de $\mathcal{X}^c$ centré sur un point le long $[\mathcal{P}_\mathbf{x}, \mathcal{P}_\mathbf{y}]$.
La valeur maximale de delta $\delta_M(\mathbf{x})$ de $\mathbf{x}$ est définie comme la valeur maximale de $r_{\mathbf{x},\mathbf{y}}$ pour chacun de

ses voisins directs $\mathbf{y} \in N_{\mathbf{x}}$:

$$\delta_M(\mathbf{x}) = \sup\{r_{\mathbf{x},\mathbf{y}} \mid \mathbf{y} \in N_{\mathbf{x}},\ d(\mathbf{m}, \mathcal{P}_{\mathbf{x}}) \geq d(\mathbf{m}, \mathcal{P}_{\mathbf{y}})\} \tag{1.2}$$

où $\mathbf{m} = (\mathbf{x} + \mathbf{y})/2$ est le point médian du segment continu $[\mathbf{x}, \mathbf{y}]$ (figure 1.6b). La deuxième condition est utilisée dans de nombreux algorithmes de squelettisation et contribue à générer un squelette fin. Le Delta Medial Axis est alors défini de la manière suivante :

Définition 1. ***Delta Medial Axis***
Soit $\mathcal{X} \cup \mathcal{X}^c$ un masque binaire, et $\delta \in \mathbb{R}_+^*$. Le delta medial axis $MA_\delta(\mathcal{X})$ de $\mathcal{X}$ est l'ensemble des points $\mathbf{x} \in \mathcal{X}$ avec une valeur delta maximale d'au moins δ :

$$MA_\delta(\mathcal{X}) = \{\mathbf{x} \in \mathcal{X} \mid \delta_M(\mathbf{x}) \geq \delta\} \tag{1.3}$$

Intuitivement, $\delta_M(\mathbf{x})$ peut être considéré comme le rayon de la déformation de la forme à l'origine d'une branche de squelette reliée à $\mathbf{x}$. Si nous considérons la plate-forme mobile comme un objet circulaire, il est en effet possible de définir δ de telle sorte que le squelette généré ne comporte que des branches traversables de DVG, c'est-à-dire celles qui autorisent le passage du robot avec son paramètre de sécurité. L'algorithme ci-dessous présente le processus qui conduit d'un masque binaire brut au squelette.

Algorithme : Extraction du DMA
Entrées : Distance au bord $\mathcal{D}$ et projection $\mathcal{P}$ pour chaque point de $\mathcal{X} \cup \mathcal{X}^c$; Paramètre δ
Sorties : δ-Medial Axis
$\mathbf{x} \in \mathcal{X}, \mathbf{y}$ voisinage de $\mathbf{x}$
Pour $\mathbf{x}$, $\mathbf{y}$ **faire**
  Si (*testSegment*$(\mathcal{P}(\mathbf{x}), \mathcal{P}(\mathbf{y}), \delta)$ est vrai) **Alors**
    Si $\mathcal{D}(\mathbf{x}) \geq \mathcal{D}(\mathbf{y})$ Alors $DMA(\mathcal{X}) = DMA(\mathcal{X}) \cup \mathbf{x}$
    Si $\mathcal{D}(\mathbf{x}) \leq \mathcal{D}(\mathbf{y})$ Alors $DMA(\mathcal{X}) = DMA(\mathcal{X}) \cup \mathbf{y}$
  Fin Si
Fin Pour
Procédure testSegment$(\mathcal{P}(\mathbf{x}), \mathcal{P}(\mathbf{y}), \delta))$
  Entrées : Points extrémités du segment étudié ; Paramètre d'élagage δ
  Sortie : *Vrai* si $\exists \mathbf{z} \in [\mathcal{P}(\mathbf{x}), \mathcal{P}(\mathbf{y})] \mid \mathcal{D}(\mathbf{z}) > \delta$ Faux sinon
  $L = d\left(\mathcal{P}(\mathbf{x}), \mathcal{P}(\mathbf{y})\right)$
  Si $L \leq 2\,\delta$ **Alors** renvoyer Faux
  $\alpha^- = \alpha^+ = 0.5$
  Tant que $(d\left(z^-, \mathcal{P}(\mathbf{x})\right) > \delta$ **ou** $d\left(z^+, \mathcal{P}(\mathbf{y})\right) > \delta)$ **faire**
    $z^- = \alpha^-\,\mathcal{P}(\mathbf{x}) + (1 - \alpha^-)\mathcal{P}(\mathbf{y}), \quad z^+ = \alpha^+\,\mathcal{P}(\mathbf{x}) + (1 - \alpha^+)\mathcal{P}(\mathbf{y})$
    Si $\mathcal{D}(\mathbf{z}^-) > \delta$ **ou** $\mathcal{D}(\mathbf{z}^+) > \delta$ **Alors** renvoyer Vrai
    $\alpha^- = \alpha^- - \frac{\delta - \mathcal{D}(\mathbf{z})}{L}$
    $\alpha^+ = \alpha^+ + \frac{\delta - \mathcal{D}(\mathbf{z})}{L}$
  Fait
  Renvoyer *Faux*
Fin

Cette méthode du calcul de l'axe médian (DMA) a été initialement développée dans [Marie, 2014] sur la base d'une caméra omnidirectionnelle. La figure 1.7 illustre les résultats de ce travail. Cet algorithme constitue une brique élémentaire de notre travail. Nous allons utilisé cette approche de squelettisation qui permet d'extraire un Diagramme de Voronoï Généralisé (DVG) local, pour ensuite planifier les déplacements autonomes du système.

Les déplacements du robot seront effectués en envoyant des vitesses de consigne aux actionneurs. Il s'agit de l'étape de l'action. La section suivante détaille cette étape en exposant quelques solutions existantes dans la littérature.

FIGURE 1.7 – Extraction du squelette en utilisant une caméra omnidirectionnelle : image brute (gauche), extraction espace libre en bleu (milieu), extraction du squelette en blanc (droite) [Marie, 2014]

1.5 Commande d'un robot mobile non holonome

La dernière étape pour une navigation autonome consiste en la réalisation du déplacement. Une fois la trajectoire déterminée par l'étape de la planification, la commande des robots mobiles permet de résoudre différents problèmes :

- le suivi de chemin, consiste en la définition d'un point lié au robot et de le faire suivre une courbe prédéterminée,
- la stabilisation sur une trajectoire en tenant compte du temps. La vitesse du robot est calculée à partir de cette trajectoire,
- la stabilisation sur une configuration fixe en utilisant une régulation à zéro d'un écart entre la configuration fixe et un repère mobile.

Dans notre cas, nous nous intéressons au premier point et au dernier point. Il s'agit de la poursuite de trajectoire en stabilisant sur une configuration fixe.

1.5.1 Modélisation d'un robot non-holonome

Un robot mobile non-holonome est un système mécanique soumis à un ensemble de contraintes cinématiques [Guechi, 2010] :

- le robot est pris comme un véhicule rigide se déplaçant dans un plan horizontal,
- les roues sont indéformables et roulent sans glisser sur le sol,
- le contact roue/sol est considéré ponctuel.

Pour un déplacement sur un plan, le système possède deux degrés de liberté : une translation et une rotation autour d'un axe normal au plan. La translation est pilotée par la vitesse linéaire v et la rotation par la vitesse du lacet ω. Le modèle cinématique d'un robot de type unicycle ne tenant pas compte des contraintes de roulement sans glissement de ses roues est décrit par le système non linéaire suivant :

$$\begin{cases} \dot{x} &= v\cos(\theta) \\ \dot{y} &= v\sin(\theta) \\ \dot{\theta} &= \omega \end{cases} \tag{1.4}$$

avec (x, y, θ) la position et l'orientation du robot dans un référentiel global.
En notant $q = (x, y, \theta)$ la configuration du robot et $u = (v, \omega)$ la commande, le modèle non linéaire simplifié peut être écrit comme suit :

$$\dot{q} = f(q, u) \tag{1.5}$$

1.5.2 Poursuite de trajectoire

Si nous disposons d'une trajectoire de référence q_{ref}, avec un ensemble de points $\mathbf{x}_{ref} = (x_{ref}, y_{ref})$, le but de la poursuite de trajectoire est de chercher une commande u en fonction de q et q_{ref} pour faire converger l'écart $\varepsilon_q = q - q_{ref}$ vers 0.

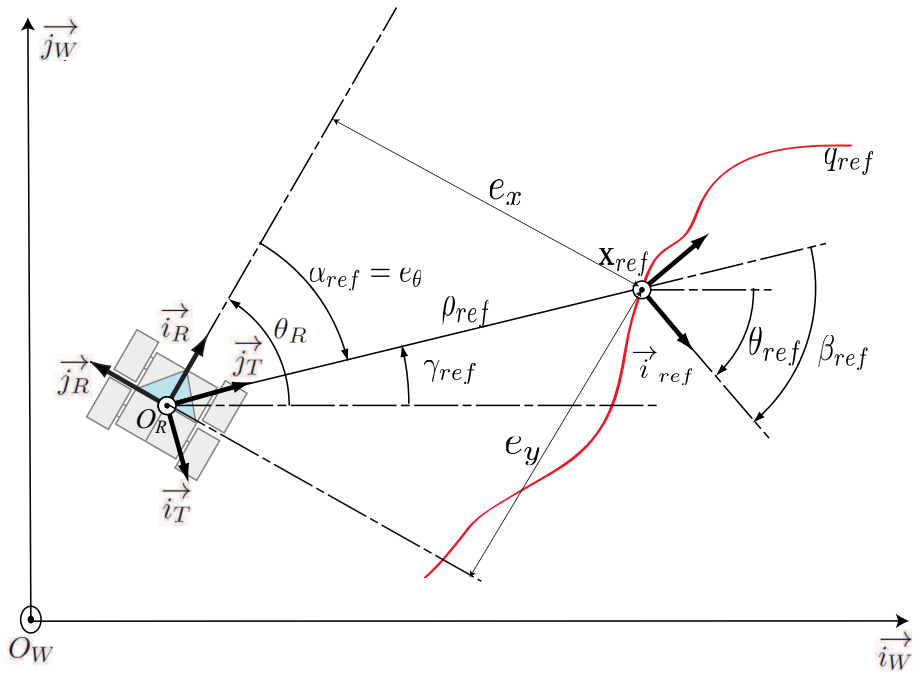


FIGURE 1.8 – Les paramètres utilisés pour un ralliement de cible

Beaucoup de travaux ont résolu ce problème par une commande avec un retour d'état non stationnaire [Kanayama et al., 1990], [Guechi, 2010], [Maalouf et al., 2006]. Certains travaux de poursuite de trajectoire se basent sur une planification utilisant des courbes paramétrées comme les courbes de Béziers [Chen et al., 2012] ou les catmull-rom splines [Yuksel et al., 2011].

Quelques commandes de poursuite de trajectoire sont détaillées dans les sections suivantes.

1.5.2.1 Commande pour une trajectoire paramétrée

Si la trajectoire est paramétrée, la tâche de poursuite se résout par :

- faire converger la vitesse du robot vers une vitesse de référence (v_{ref}, ω_{ref}) (1.6) de la trajectoire (cible en mouvement par rapport au robot) [Kanayama et al., 1990],
- faire converger la position du robot vers une cible fixe dans la trajectoire en utilisant un PID [Wung Choi et Elkaim, 2008].

Les vitesses linéaire v_{ref} et angulaire ω_{ref} de référence s'écrivent :

$$\begin{cases} v_{ref} & = & \sqrt{\dot{x}_{ref} + \dot{y}_{ref}} \\ \omega_{ref} & = & \dfrac{\dot{x}\ddot{y}_{ref} - \ddot{x}_{ref}\dot{y}_{ref}}{\dot{x}_{ref}^2 + \dot{y}_{ref}^2} \end{cases} \tag{1.6}$$

Dans [Kanayama et al., 1990] une loi de commande stabilisante a été conçue suivant un modèle d'erreur (e_x, e_y, e_θ) entre la pose du robot et la pose d'une trajectoire de référence (figure 1.8).

$$\begin{pmatrix} \dot{e}_x \\ \dot{e}_y \\ \dot{e}_\theta \end{pmatrix} = \begin{pmatrix} \cos(e_\theta) & 0 \\ \sin(e_\theta) & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} v_{ref}(t) \\ \omega_{ref}(t) \end{pmatrix} + \begin{pmatrix} -1 & e_y \\ 0 & -e_x \\ 0 & -1 \end{pmatrix} u \tag{1.7}$$

Les commandes proposées dans [Kanayama et al., 1990] servent à faire converger l'erreur (e_x, e_y, e_θ) vers 0 :

$$\begin{pmatrix} v \\ \omega \end{pmatrix} = \begin{pmatrix} v_{ref}\ \cos(e_\theta) + k_x\, e_x \\ \omega_{ref} + v_{ref}\,(k_y\, e_y + k_\theta\, e_\theta) \end{pmatrix} \begin{pmatrix} v_{ref} \\ \omega_{ref} \end{pmatrix} \tag{1.8}$$

avec k_x, k_y et k_θ sont des scalaires positifs.

Cette loi de commande propose un suivi stable pour les véhicules non-holonomes. Elle est efficace si la trajectoire de référence est paramétrée pour bien calculer les paramètres v_{ref} et ω_{ref}. Elle nécessite également l'estimation précise de la position du robot. Cependant la trajectoire de référence parfois ne peut pas être paramétrée et la position du robot ne donne pas une valeur précise suite au glissement des roues.

1.5.2.2 Commande pour atteindre une cible

Si la trajectoire n'est pas paramétrée, le problème peut être ramené à un suivi de cible [Park et Kuipers, 2011] et [Siegwart et al., 2011]. La poursuite de trajectoire peut être considérée comme un ralliement de points fixes sur la trajectoire. A cause des problèmes de commandabilité, au lieu d'asservir la position absolue du robot, c'est la position du robot par rapport à la cible qui sera asservie $\mathbf{X} = (\rho_{ref}, \alpha_{ref}, \beta_{ref})$. Cette dernière est définie par les paramètres suivants (figure 1.8) :

- ρ_{ref} est la distance entre le robot et la cible

$$\rho_{ref} = \sqrt{(x_{ref} - x)^2 + (y_{ref} - y)^2} \tag{1.9}$$

- α_{ref} est l'angle entre la direction du robot et la direction de la droite reliant le robot et la cible.
- β_{ref} est l'angle entre la direction de la droite reliant le robot et la cible et l'orientation de la

cible.

Le modèle d'évolution de ces paramètres s'écrit en fonction des variables de commande du robot :

$$\begin{pmatrix} \dot{\rho}_{ref} \\ \dot{\alpha}_{ref} \\ \dot{\beta}_{ref} \end{pmatrix} = \begin{pmatrix} -\cos(\alpha_{ref}) & 0 \\ \frac{\sin(\alpha_{ref})}{\rho_{ref}} & -1 \\ -\frac{\sin(\alpha_{ref})}{\rho_{ref}} & 0 \end{pmatrix} \begin{pmatrix} v \\ \omega \end{pmatrix} \tag{1.10}$$

La loi de commande, dans ce cas, est définie par un retour d'état $u = \mathbf{K}_u\mathbf{X}$. La matrice du gain $\mathbf{K}_u$ est définie pour que v pilote la distance ρ_{ref} et ω l'écart angulaire :

$$\begin{pmatrix} v \\ \omega \end{pmatrix} = \begin{pmatrix} k_\rho & 0 & 0 \\ 0 & k_\alpha & k_\beta \end{pmatrix} \begin{pmatrix} \rho_{ref} \\ \alpha_{ref} \\ \beta_{ref} \end{pmatrix} \tag{1.11}$$

avec k_ρ, k_α et k_β sont des scalaires positifs.

Ces lois de commande nécessitent une connaissance a priori de la position du robot pour pouvoir faire le suivi de trajectoire. Cette information n'est pas toujours facile à extraire ou à obtenir avec des imprécisions importantes. Ainsi les limites de ces approches sont :

- la difficulté de localiser un robot non holonome avec précision,
- ces commandes ne sont pas conçues pour des environnement dynamiques et pour des approches réactives,
- la robustesse de la loi de commande n'est pas toujours assurée suite aux erreurs de modélisation de la cible ou de la trajectoire de référence.

Une autre approche robuste de commande existe, qui s'appuie sur le formalisme de la commande référencée capteur.

1.5.3 Commande référencée capteur

Pour le contrôle réactif, la commande référencée capteur est la plus utilisée. Elle a débuté avec les robots manipulateurs. Dans [Samson et al., 1991], les synthèses de commande référencée capteur est faite en introduisant le formalisme de "fonction tâche".

La commande référencée capteurs nécessite l'utilisation de capteurs extéroceptifs (vision, capteurs télémétriques, capteurs de force). Ce type de commande traite plusieurs problématiques :

- le choix, la modélisation et le calibrage des capteurs pour déterminer la relation qui relie le robot et les capteurs,
- le choix des informations à utiliser pour la commande qui dépendent de la tâche à réaliser,
- la fusion des informations de différents capteurs,
- la conception d'une loi de commande en se basant sur des informations capteurs extéroceptifs.

Dans la commande référencée capteur, le capteur de vision est très utilisé dans la littérature parce qu'il fournit une information très riche de l'environnement. Cependant, effectuer une tâche de navigation dans un environnement complexe nécessite d'utiliser plus d'un type de capteurs ou bien d'autres types de données. Par exemple, dans [Cadenat et al., 2000] est proposé une approche de commande référencée capteurs qui combine les données d'une caméra perspective et les données d'un

laser pour construire une commande de navigation d'un robot mobile. Dans [Victorino et al., 2004a] et [Victorino et al., 2004b] une approche robuste de commande référencée laser a été développée. Il s'agit d'extraire des informations issues de mesures télémétriques avec un laser à balayage horizontal et de les utiliser pour construire une loi de commande. Des fonctions de navigation sont proposées, basées sur les caractéristiques du diagramme de Voronoï (rallier le diagramme, suivre le diagramme et se stabiliser sur un point de bifurcation). Cette approche permet à un robot de naviguer dans un environnement inconnu sans avoir besoin de le reconstruire.

1.5.3.1 Formalisme de la commande référencée capteur

Soit $\mathbf{s} = (s_1, .., s_n)^T$ un vecteur de signatures de taille n judicieusement choisi, fourni par un capteur $\mathcal{S}$ placé sur un robot. Soit $\mathbf{s}^* = (s_1^*, .., s_n^*)^T$ un vecteur de valeurs désirées de la signature. L'idée globale de la commande référencée capteur est la conception d'une loi de commande qui utilise la variation de la signature par rapport au temps, $\dot{\mathbf{s}}$, pour induire une convergence de $\mathbf{s}$ à $\mathbf{s}^*$, et donc une convergence implicite du robot à son comportement désiré. L'évolution de $\mathbf{s}$ dans le temps peut être exprimée par rapport à la cinématique du capteur par :

$$\dot{\mathbf{s}} = \frac{\partial \mathbf{s}}{\partial \mathbf{r}} \frac{\partial \mathbf{r}}{\partial \mathbf{t}} = \mathbf{L_{sr}} \mathbf{T}_C \tag{1.12}$$

où $\mathbf{r}$ définit la position du capteur dans le repère monde et $\mathbf{T}_C = (\mathbf{v}_C^T, \boldsymbol{\omega}_C^T)^T$ le torseur cinématique du capteur par rapport au monde, $\mathbf{v}_C = (v_x, v_y, v_z)^T$ et la vitesse angulaire $\boldsymbol{\omega}_C = (\omega_x, \omega_y, \omega_z)^T$. $\mathbf{L_{sr}} \in \mathbb{R}^{n \times 6}$ est la matrice d'interaction, qui lie la variation de la signature au mouvement du capteur.

Soit l'erreur de la signature ε telle que

$$\varepsilon = \mathbf{s} - \mathbf{s}^* \tag{1.13}$$

L'objectif de la commande référencée capteur est d'annuler ε. Cela peut se faire en imposant une décroissance exponentielle de ε. En prenant un gain positif λ, nous avons :

$$\dot{\varepsilon} = -\lambda \, \varepsilon \tag{1.14}$$

En notant $\mathbf{L_{sr}}^+ \in \mathbb{R}^{6 \times n}$ la pseudo-inverse de la matrice d'interaction, la loi de commande $\mathbf{u}$ s'écrit :

$$\mathbf{u} = -\lambda \, \mathbf{L_{sr}}^+ \varepsilon \tag{1.15}$$

La matrice $\mathbf{L_{sr}}^+$ est choisie comme étant la pseudo-inverse de Moore-Penrose telle que : $\mathbf{L_{sr}}^+ = (\mathbf{L_{sr}}^T \mathbf{L_{sr}})^{-1} \mathbf{L_{sr}}^T$ quand $\mathbf{L_{sr}}$ est de rang plein 6. Si $n = 6$ et $det(\mathbf{L_{sr}}) \neq 0$ il est simple d'inverser $\mathbf{L_{sr}}$, par conséquent la commande s'écrira : $\mathbf{u} = -\lambda \, \mathbf{L_{sr}}^{-1} \varepsilon$

1.5.3.2 Les systèmes de mesures physiques

Ces dispositifs de mesure permettent au robot de détecter ce qui se trouve autour de lui ou d'acquérir des informations sur lui-même. Les capteurs peuvent être classés suivant le principe de fonctionnement : un capteur proprioceptif ou extéroceptif. Nous présentons dans cette partie les capteurs les plus utilisés en robotique mobile.

L'odométrie est un capteur proprioceptif. La méthode la plus classique de mesure se fait à partir de l'utilisation d'un codeur installé sur chacune des roues. Cette méthode consiste à mesurer le déplacement angulaire élémentaire et, connaissant le rayon des roues la position du robot peut être déduite. Pour un codeur optique incrémental l'estimation du déplacement se calcule à partir des incréments δ_{θ_g} et δ_{θ_d} des roues (gauches et droites). Puis l'estimation de la position se détermine par intégration en calculant le déplacement curviligne δ_s et le déplacement angulaire élémentaire δ_θ

$$\begin{aligned} \delta_s &= \frac{1}{2}(R_g\delta_{\theta_g} + R_d\delta_{\theta_d}) \\ \delta_\theta &= \frac{1}{2}(R_g\delta_{\theta_g} - R_d\delta_{\theta_d}) \end{aligned} \tag{1.16}$$

L'avantage de l'utilisation de ce capteur est la simplicité de la mesure et de l'implémentation. Mais, l'inconvénient réside dans le bruit de mesure que peut engendrer ce type de capteurs, lié en particulier à la résolution angulaire. Enfin, l'intégration se fait selon une hypothèse de roulement sans glissement rarement vérifiée.

La centrale inertielle est un capteur proprioceptif constitué d'au moins un gyromètre mesurant une vitesse angulaire par rapport à un axe de référence, un accéléromètre mesurant l'accélération inertielle et elle peut être équipée d'un magnétomètre mesurant la direction du champ magnétique. Par intégrations successives, les vitesses linéaires, positions et orientations peuvent être déterminées. L'avantage de ce capteur est qu'il perçoit la gravité donc il permet d'avoir une référence extérieure fixe. En revanche il subit des accélérations autres que celle induite par la gravité, et des champs magnétiques autres que ceux de la terre.

La caméra est utilisée pour la vision par ordinateur. Cette dernière consiste en l'extraction de données à partir des images afin de percevoir, d'analyser et de comprendre la scène observée. On peut considérer des caméras perspectives, des caméras stéréoscopiques ou des caméras catadioptriques. Ces dernières sont constituées d'une caméra simple associée à un miroir de révolution (figure 1.9). La projection de la scène dans le miroir est l'image fournie par ce type de capteurs. Elles fournissent une vue globale de la scène en une seule prise de vue, permettant un champs d'observation de $360\,deg$ autour du capteur. Ce type de capteurs est très utilisé dans la commande référencée capteur pour, par exemple, la détection de piétons [Dollár et al., 2009], "pick and place" de robots manipulateurs [Sbnchez et Martinez, 2000] et bien d'autres tâches robotique. Lorsqu'il s'agit de l'utilisation des informations visuelles cette méthode s'appelle "Asservissement Visuel" [Chaumette, 1990].

FIGURE 1.9 – Une caméra catadioptriques est constituée d'une caméra simple et d'un miroir (a). Un exemple de capteur est illustré dans (b) avec (c) l'image qu'il peut fournir

Le télémètre est un capteur de mesure de distance. Le principe de fonctionnement de ce capteur se base sur l'émission et la réception d'une onde (rayon lumineux, onde sonore, un faisceau infrarouge ...) (figure 1.10). Plusieurs technologies peuvent être utilisées :

- Les télémètres laser envoient un rayon laser avec une longueur d'onde de l'ordre de 600 nm. Ces télémètres sont utilisés pour de courtes distances et permettent de détecter la distance au point d'impact. Les avantages de ce type de capteurs sont qu'ils sont peu sensibles aux conditions atmosphériques. Il y a aussi les télémètres lasers à balayage pour lesquels le faisceau laser balaye sur un plan.
- Les télémètres à ultrasons émettent des ondes sonores avec une longueur d'ondes de l'ordre de 9 mm à intervalles réguliers avec une haute fréquence.
- Les télémètres à infrarouges utilisent un faisceau infrarouge modulé avec une longueur d'ondes de l'ordre de 900 nm. Ce capteur est peu coûteux, peu encombrant et nécessite peu d'énergie. L'inconvénient réside dans la limitation de distance qu'il peut détecter et il est utilisé généralement sur les petits robots.
- Le Radar (RAdio Detection And Ranging) pour la détection et la télémétrie radio. Comme les autres capteurs, il comporte un émetteur et un récepteur d'onde électromagnétique. Le radar Doppler est le plus connu dans le domaine de l'automobile. Pour les Radars de haute fréquence, la longueur d'onde peut être comprise entre 10 m et 100 m.

Le GNSS (Global Navigation Satellite System) est le terme générique pour les systèmes de localisation par satellite. Le principe de fonctionnement se base sur l'envoi de signaux codés d'une onde porteuse, suivant deux modes : un mode précis de positionnement (précision de l'ordre de 10 m) et un mode standard de positionnement (précision de l'ordre de 100 m). Un récepteur reçoit au même moment les signaux codés en provenance de plusieurs satellites situés à des distances différentes. Le décodage de ces signaux permet de déterminer ces distances et par la suite de calculer la position du récepteur dans un référentiel géodésique connu. Les systèmes de navigation par satellites existants sont : le GPS (US), GLONASS (Russie), GALILEO (Union Européenne), BEIDOU (Chine).
La précision peut être améliorée par l'utilisation d'un système différentiel comme le GPS différentiel (DGPS).

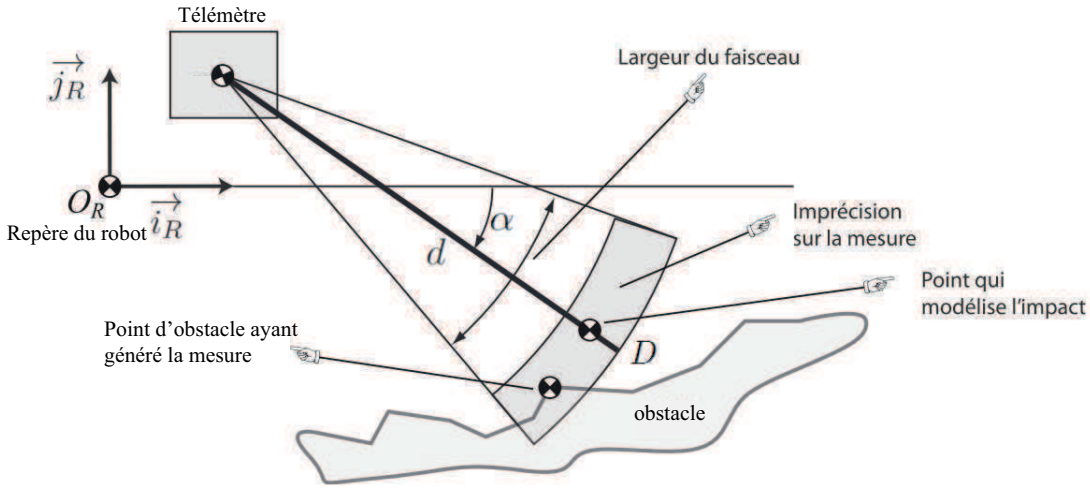


FIGURE 1.10 – Principe de fonctionnement d'un télémètre. d correspond à la distance mesurée par le capteur, α est la direction que le télémètre vise par rapport à la direction du robot $\vec{i}_R$

1.5.3.3 Les systèmes de mesures par logiciel

La technologie des capteurs logiciels offre une méthode efficace pour fournir des informations sur un processus. En pratique, la réalisation de la mesure par un capteur physique n'est pas toujours possible. Les capteurs logiciels utilisent généralement un modèle de connaissance du processus et une boucle de rétroaction à partir de mesures. Les capteurs logiciels peuvent également être présentés comme capteurs virtuels, observateurs, estimateurs d'état et observateurs d'état. Le modèle peut être de différente nature : modèle linéaire, non-linéaire, déterministe, stochastique ou ensembliste.

Le reconstructeur d'état a été abordé dès les années 1960 par David Gilbert Luenberger avec les observateurs d'état pour des systèmes déterministes [Luenberger, 1964], [Luenberger, 1966]. Rudolf Kalman a proposé la solution pour les systèmes probabilistes [Kalman et al., 1960]. Ces deux techniques s'appliquent aux systèmes linéaires stationnaires. Il n'y a pas une méthode exacte pour les systèmes non linéaires, il s'agit juste d'adapter les techniques des systèmes linéaires au non linéaire autour d'un point de fonctionnement.

Une étude comparative détaillée entre les différents types d'observateurs a été développée dans [Zhang et al., 2009] pour une application de moteur à induction. Généralement, dans tous les systèmes, les seules grandeurs disponibles sont les variables d'entrée et de sortie. Dans un algorithme d'estimation d'état, une connaissance a priori du système est utilisée sous la forme d'un modèle d'état (1.17) (figure 1.11). L'information requise est reconstruite par les variables d'état définies dans un vecteur d'état. L'observateur peut déterminer une estimation du système dynamique pour l'état réel du modèle considéré à chaque instant.

Notons $\mathbf{x}(t)$ le vecteur d'état, $\mathbf{u}(t)$ l'entrée du système et $\mathbf{y}(t)$ la mesure, nous rappelons ici le modèle d'état :

$$\begin{cases} \dot{\mathbf{x}}(t) &= f(\mathbf{x}(t), \mathbf{u}(t)) \\ \mathbf{y}(t) &= h(\mathbf{x}(t), \mathbf{u}(t)) \\ \mathbf{x}(t = t_0) &= \mathbf{x}_0 \end{cases} \tag{1.17}$$

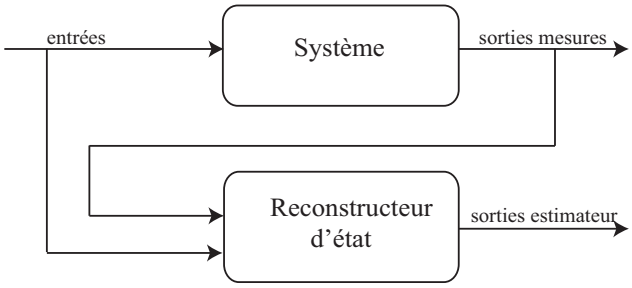


FIGURE 1.11 – Schéma d'un algorithme d'estimation d'état

Les capteurs logiciels sont utilisés :

- Pour estimer des variables qui ne sont pas directement mesurées comme pour mesurer la profondeur d'une image lorsque une seule caméra perspective est utilisée [Shahriari et al., 2013], [De Luca et al., 2008], [Sassano et al., 2010].
- Lorsque les vrais capteurs donnent des mesures erronées [Chen et al., 2015].
- Lorsque les variables à mesurer requièrent des systèmes de mesures très coûteux.
- Lorsqu'il existe des contraintes technologiques pour avoir accès à ces signaux.

La commande de robots mobiles dépend en partie de la qualité du signal à réguler, l'utilisation de capteurs logiciels est tout à fait adaptée.

Pour les systèmes linéaires, on pourra distinguer :

Les observateurs de Luenberger qui sont utilisés pour l'estimation des états connus ou inconnus d'un système dynamique linéaire. Généralement, ces observateurs se comportent comme un filtrage passe-bas pour les systèmes soumis à des perturbations [Liu et al., 2002], [Godler et al., 2002]. Ils peuvent également être utilisés pour la détection de défauts de capteurs [Gaddouna et al., 1994] [Tarantino et al., 2000].

Le filtre de Kalman peut être décrit comme une solution récursive pour estimer l'état instantané d'un système et la précision de l'estimation, de sorte qu'une erreur entre le modèle et le réel pourrait être minimisée. Le filtrage de Kalman est implémenté initialement pour les systèmes de dynamique linéaire perturbés par du bruit blanc. L'avantage de ce type de filtrage est qu'il fournit non seulement l'instant passé, actuel et futur, mais fonctionne aussi bien lorsque la nature du système n'est pas assez précise [Grewal et Andrews, 1993], [Welch et Bishop, 1997].

Pour les systèmes non linéaires, il y a :

Les observateurs de Luenberger étendu produisent de meilleures performances relativement aux techniques classiques d'observateurs linéaires [Zeitz, 1987]. Ils sont aussi capable d'estimer de manière précise la dynamique d'un système non-paramétré dans de nombreux cas [Palli et Melchiorri, 2008], [Lampaert et al., 2004], [Roset et al., 2006]. Plusieurs travaux récents ont étudié et développé une nouvelle approche d'estimation basée sur l'observateur de Luenberger étendu [Xiong et al., 2015], [Zhao et Guo, 2017]. Dans [Zhao et Guo, 2017], une estimation d'état d'un système multi-entrées

multi-sorties avec une incertitude incohérente a été développée pour une application d'un suivi de trajectoire avec un modèle de véhicule sous-marin autonome.

Le filtre de Kalman étendu est réalisé en procédant à un développement de Taylor des fonctions du système. Une linéarisation appropriée de l'état non linéaire et des équations d'observation peut être trouvée autour d'un point de fonctionnement [Durrant-Whyte et al., 2001a], [Fujii, 2013]. Le filtre de Kalman étendu (EKF) peut fournir une estimation extrêmement précise des états d'un système non linéaire, c'est la raison pour laquelle de nombreuses applications sont développées avec plusieurs problèmes d'estimation, comme les paramètres de la dynamique d'un véhicule automobile [Wang, 2013, Stéphant, 2004, Wenzel et al., 2006].

Nous avons défini différents types de mesure qui peuvent être utilisés pour calculer une signature, **s** du formalisme de la commande référencée capteur. Plusieurs types de commandes référencées capteur existent et se différencient par rapport à la nature de la signature. Ce qui nous intéresse dans cette thèse est l'asservissement visuel, il s'agit de la commande basée sur un capteur de vision.

1.5.3.4 Asservissement visuel

Dans la littérature, un nombre important de travaux s'intéressent à l'asservissement visuel depuis une vingtaine d'années [Chaumette, 1990, Espiau et al., 1992]. Les caractéristiques visuelles qu'on peut extraire d'une image sont diverses et variées et permettent de réaliser différents objectifs. Les systèmes d'asservissement visuel diffèrent principalement par la façon dont les caractéristiques visuelles sont conçues. Les deux approches principales sont l'asservissement basé sur l'image (Image Based Visual Servoing : IBVS), qui est constitué d'un ensemble de paramètres 2D directement exprimés dans l'image, et l'asservissement visuel basé sur une pose 3D (Position Based Visual Servoing : PBVS), dans lequel la signature se compose d'un ensemble de paramètres 3D liés à la position relative entre la caméra et la cible [Chaumette et al., 2014, Chaumette, 1990]

Dans les approches IBVS et PBVS, de nombreuses possibilités existent selon le choix des fonctionnalités. Chaque choix entraînera un comportement particulier du système. Il existe également des approches hybrides, qui combinent les paramètres 2D et 3D de la signature pour bénéficier des avantages de IBVS et PBVS tout en évitant leurs inconvénients respectifs [Buonocore et al., 2015, Shen et al., 2010, Hadj-Abdelkader et al., 2006].

L'asservissement visuel 3D (PBVS) est l'approche d'asservissement visuel qui a été implémentée en premier par [Shirai et Inoue, 1973]. Le principe de cette approche est d'estimer des paramètres 3D à partir d'images par un processus d'estimation de la pose relative entre la caméra et un objet dans la scène. Cette approche nécessite une connaissance du modèle de l'objet 3D. Le processus d'estimation de la pose partielle utilise les propriétés de la géométrie épipolaire entre les images désirées et les images courantes si on dispose d'une seule caméra, sinon on utilise un processus de triangulation pour un système de stéréovision [Rizzi et Koditschek, 1996, Chaumette et al., 2014, Ansar et Daniilidis, 2003]. Pour l'asservissement visuel avec une stéréo-vision, la reconstruction 3D peut être réalisée sans connaître a priori la géométrie de la scène [Rizzi et Koditschek, 1996]. La figure 1.12 définit le schéma

bloc d'un asservissement visuel 3D. La consigne 3D , $\mathbf{s}^*(t)$, désigne la position désirée en 3D de l'objet par rapport à la caméra. Cette variable est comparée à une mesure $\mathbf{s}(t)$ extraite du bloc reconstruction 3D. Le bloc loi de commande envoie au robot les commandes adéquates pour faire converger $\mathbf{s}(t)$ vers $\mathbf{s}^*(t)$ en respectant une certaine dynamique.

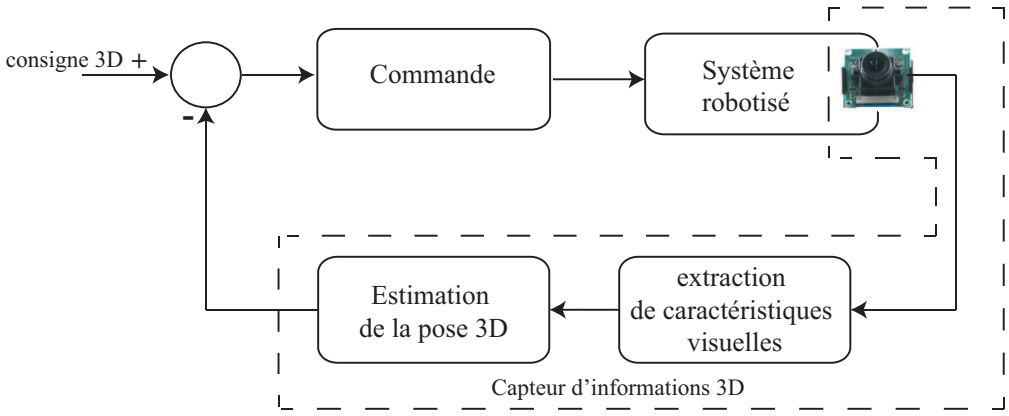


FIGURE 1.12 – Asservissement visuel 3D

L'asservissement visuel 2D (IBVS) a été introduit dès les années 80 par Weiss [Weiss, 1984, Weiss et al., 1987]. Cette méthode consiste à exprimer la commande dans l'espace de l'image comme illustré dans la figure 1.13, c'est-à-dire, contrôler le système robotisé en utilisant directement les informations visuelles dans la boucle de commande. Les caractéristiques visuelles dans cette méthode sont exprimées soit en pixels, soit dans le plan normalisé de l'image. Différentes signatures dans l'image peuvent être mesurées : un point, un segment, une droite, une ellipse [Chaumette, 1990] ou des moments dans l'image [Chaumette, 2004]. Le choix de la signature dépend de la tâche à réaliser. Cette méthode n'a pas besoin de l'estimation 3D de la pose du robot mais elle a besoin d'une bonne estimation de la matrice d'interaction. Par rapport à l'asservissement 3D, cette méthode est beaucoup plus précise parce qu'elle ne requiert pas une estimation de la position 3D de la cible [Palmieri et al., 2012, Janabi-Sharifi, 2002].

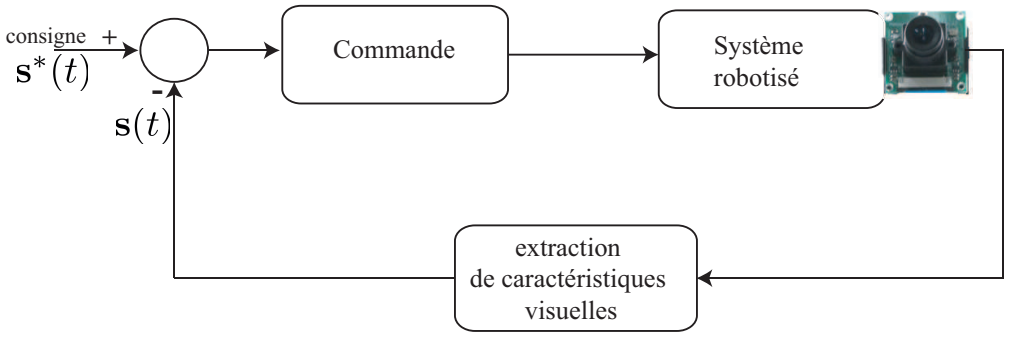


FIGURE 1.13 – Asservissement visuel 2D

L'asservissement visuel Hybride est appelé aussi $2D$ et demi et a été initialement développé par [Deguchi, 1998, Malis et al., 1998]. Il regroupe les techniques de l'IBVS avec celles du

PBVS, [Malis et al., 1999, Hafez et al., 2008]. Cette approche permet de garantir le maintien des caractéristiques dans la zone visible pendant le processus d'asservissement dans l'espace image [Chesi et al., 2004], ou de garder le bras dans son espace libre pendant l'asservissement dans l'espace cartésien/articulaire [Mezouar et Chaumette, 2002]. La méthode d'asservissement visuel 2D et demi permet également de découpler complètement les mouvements de rotation et les mouvements de translation [Deguchi, 1998]. Les vitesses de translation sont calculées à partir d'informations 3D et les vitesses de rotation sont, quant-à-elles, déterminées à partir d'informations 2D. Avec ce type d'asservissement la géométrie de la cible n'est pas nécessaire. Dans [Hadj-Abdelkader et al., 2006] un asservissement visuel hybride basé sur les images omnidirectionnelles a été réalisé.

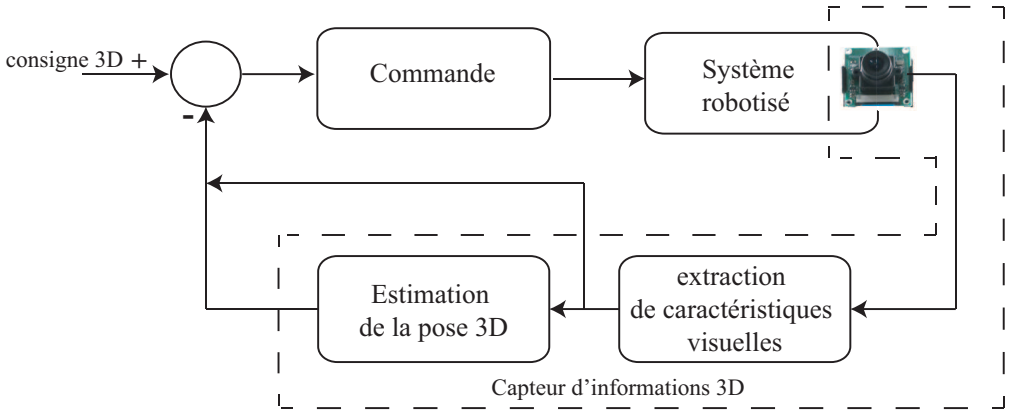


FIGURE 1.14 – Asservissement visuel Hybride 2D 1/2

1.6 Conclusion

La navigation autonome des robots mobiles est un domaine de recherche actif depuis les quatre dernières décennies. Une navigation sûre et autonome nécessite une bonne perception de l'environnement. Pour cela, un robot mobile doit être équipé d'un ou plusieurs capteurs. Une fois son environnement perçu, il peut extraire des informations nécessaires pour calculer son déplacement. Dans ce chapitre toutes les étapes fondamentales à la navigation autonome ont été présentées (la perception, la planification et l'action).

Plusieurs travaux ont traité ce problème avec différents types de capteurs. [Hadj-Abdelkader et al., 2008] propose une approche de navigation d'asservissement visuel en utilisant une caméra omnidirectionnelle. Les travaux présentés dans [Pasteau et al., 2013] et [Pasteau et al., 2014] utilisent une caméra perspective pour le déplacement d'un fauteuil roulant électrique. Un télémètre laser est utilisé dans [Victorino et al., 2004a, Luan Bui et al., 2013]. [Ohya et al., 1998] propose une stratégie de commande en combinant les données d'une caméra et des ultrasons.

Malgré la robustesse de la commande référencée capteur, elle présente des limites :

- limitation des champs de vue des capteurs,
- possibilité de défaillance des capteurs,
- erreurs de mesures ou mesures bruitées.

Ce chapitre a succinctement exposé la littérature et les méthodes utilisés pour la navigation d'un robot mobile non-holonome. Il a également présenté les outils qui représentent la clé de voûte des contributions importantes de ce travail (la commande référencée capteur (section.1.5.3), un algorithme de squelettisation : le DMA (section.1.4.3.3) et les reconstructeurs d'état (section.1.5.3.3)).

Dans notre travail nous proposons une méthodologie de commande référencée capteurs d'un robot mobile où la construction de l'espace libre, les trajectoires que le robot doit suivre, la loi de commande, sont exprimées dans un référentiel virtuel unifié qui regroupe toutes les mesures de capteurs. Nous traitons aussi le cas de défaut de capteur et bruit de mesures en utilisant un système de prédiction basé sur un modèle d'évolution.

Asservissement visuel dans un environnement non encombré

Résumé du chapitre

Dans la littérature, on peut trouver une solution de déplacement dans un couloir utilisant un asservissement visuel basé sur une signature constituée de deux informations extraites dans l'image. La signature choisie était constituée de la position du point de fuite et de l'orientation de la médiane dans l'image. Utilisant les mêmes concepts et en se basant sur ce travail, nous allons proposer ici une nouvelle stratégie de mesure et de commande. La proposition consiste en l'utilisation d'un repère unifié commun de projection pour exprimer la signature et la commande. Ce cadre sera appelé "capteur virtuel". Nous prouverons que l'utilisation d'une seule information de la signature est suffisante pour assurer un suivi de couloir en utilisant les informations extraites de ce capteur virtuel. L'étude de la stabilité du système commandé est notamment détaillée. Nous avons de plus pris en considération le cas où du bruit de mesure serait présent et les cas où les périodes pendant lesquelles l'extraction de la mesure serait impossible. Ceci a été réalisé par une stratégie basée observateur/modèle.

Sommaire

2.1 Introduction

Pour permettre à un robot mobile de naviguer dans un environnement inconnu (sans avoir une carte a priori), il faut qu'il possède d'un ensemble de données pertinentes de son espace libre prélevées en temps réel. Ces données lui permettent de réaliser différentes tâches (éviter les obstacles, suivre une cible, passer les portes, ...).

Ce chapitre est consacré à la perception et la navigation en temps réel d'un robot mobile, pour un déplacement dans un environnement non-encombré (couloir). Nous nous intéressons à la navigation qui est réalisée à l'aide du formalisme de l'asservissement visuel. La solution de navigation dans un couloir avec un asservissement visuel utilisant une caméra perspective présente des limitations. En effet, dans le cas où les mesures des caractéristiques visuelles ne peuvent pas être extraites ou donnent des valeurs erronées, l'impact sur la loi de commande est important. Par la suite, une navigation sûre n'est plus assurée. Pour surmonter cette difficulté, nous proposons une solution pour la navigation le long des couloirs en utilisant un observateur qui estime les caractéristiques visuelles (le point de fuite et l'orientation de la ligne médiane du couloir). Même sans mesure, la loi de commande peut alors être maintenue sur la base d'une estimation de la signature. Pour la construction de la commande nous montrerons que l'asservissement sur le milieu du couloir peut se faire en utilisant uniquement l'information de l'orientation de la médiane du couloir. Notre étude se base généralement dans un environnement dont la structure peut être assimilée à un couloir (lignes au sol, murs, délimitation herbes, routes, etc...

Nous avons défini un référentiel unifié virtuel là où toutes les observations sont projetées (capteur virtuel). Nous calculons des informations visuelles exprimées dans ce repère virtuel pour construire la loi de commande. Dans la première partie du chapitre, un état de l'art de la navigation dans un couloir est présenté avec l'extraction de la signature et les limites de l'utilisation d'une caméra. Dans la deuxième partie une construction d'une autre mesure est présentée pour pallier les limites. La dernière partie présente la stratégie de commande.

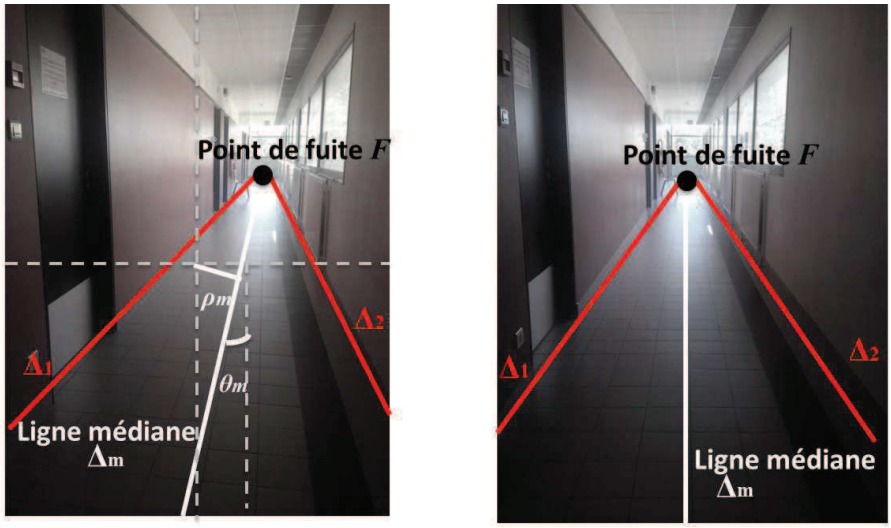
2.2 Commande référencée capteur dans un couloir : état de l'art

Des solutions à ce problème de commande dans un couloir ont déjà été proposées dans la littérature. Pour différents types de capteurs, dans [Victorino et al., 2004a] et [Victorino et al., 2004b], une méthodologie de navigation sûre est conçue avec un laser 2D dans un environnement inconnu. Dans [Carelli et Freire, 2003], le suivi du couloir est basé sur les mesures extraites d'un sonar et de codeurs pour l'odométrie. Dans d'autres travaux, [Vassallo et al., 2000], [Yang et Tsai, 1999] et [Pasteau et al., 2013], une navigation basée asservissement visuel est réalisée en utilisant une caméra perspective montée sur le robot.

Le choix de la signature visuelle dépend fortement de la tâche souhaitée par le robot mobile. Par exemple, pour se déplacer le long d'un couloir, [Pasteau et al., 2013] et [Vassallo et al., 2000] considèrent les bords du couloir dans l'image et déduire la ligne médiane et le point de fuite du couloir (figure 2.1). Ce dernier représente l'intersection des lignes d'image liées aux plinthes du couloir. Plusieurs études ont relevé la pertinence de ces caractéristiques. Dans [Vassallo et al., 2000] une loi de commande a été conçue à l'aide d'un correcteur Proportionnel Intégral Dérivée. Dans

[Yang et Tsai, 1999], le problème du déplacement autonome dans un couloir est résolu en utilisant le point de fuite des lignes du plafond en tant que caractéristique visuelle. Une autre méthode récente d'asservissement visuel est proposée dans [Pasteau et al., 2013], où l'environnement et la pose du robot sont inconnus. Dans ce travail, la tâche consiste à suivre d'une manière autonome un couloir en utilisant un asservissement visuel basé image (IBVS). Seules les caractéristiques extraites dans l'image sont utilisées et la commande est calculée pour assurer la convergence de chaque caractéristique à sa valeur désirée [Pasteau et al., 2013], [Narayanan et al., 2014].

Dans notre cas, nous nous intéressons à l'approche de l'asservissement visuel de type IBVS pour éviter d'estimer la pose du robot. La commande basée vision repose sur deux informations visuelles, à savoir la position du point de fuite dans l'image perspective et l'orientation de la ligne médiane [Pasteau et al., 2013], [Pasteau et al., 2014].



(a) position actuelle de la caméra (b) position désirée de la caméra

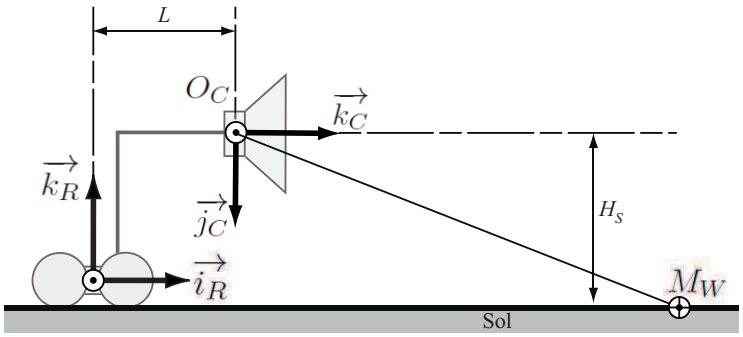
FIGURE 2.1 – Les paramètres visuels utilisés pour l'asservissement dans un couloir. Δ_1 et Δ_2 correspondent aux droites parallèles des murs, $\Delta_m(\rho_m, \theta_m)$ correspond à la médiane dans l'image. Le point de fuite F présente l'intersection des deux droites Δ_1 et Δ_2.

L'objectif de ce travail était de faire un suivi de couloir autonome. Pour que le robot soit au milieu du couloir, la caméra doit être positionnée au milieu du couloir. Dans l'image, l'abscisse du point de fuite doit être au centre et l'orientation de la ligne doit être nulle. Une vitesse d'avance constante est choisie et seule la vitesse de rotation est à asservir.

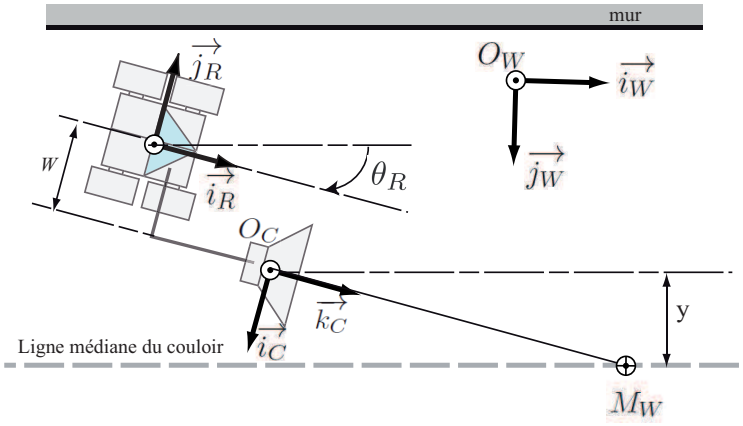
2.2.1 Modélisation

2.2.1.1 Définition des repères

Pour la commande référencée capteur, il est nécessaire d'exprimer les mouvements du robot et de l'environnement par rapport au repère capteur. De ce fait, les lois de commande seront directement calculées à partir des expressions trouvées dans ce repère. Quelques repères et variables doivent être



(a) Vue de côté du problème



(b) Vue de dessus du problème

FIGURE 2.2 – Le robot équipé de sa caméra ainsi que le plan image. M_W est un point du sol appartenant à la ligne médiane du couloir

définis :

- le repère monde $\mathcal{F}_W\left(\mathcal{O}_W, \mathcal{B}_W\right)$, $\mathcal{B}_w = \left(\vec{i}_w, \vec{j}_w, \vec{k}_w\right)$
- le repère robot $\mathcal{F}_R\left(\mathcal{O}_R, \mathcal{B}_R\right)$, $\mathcal{B}_R = \left(\vec{i}_R, \vec{j}_R, \vec{k}_R\right)$
- le repère caméra $\mathcal{F}_C\left(\mathcal{O}_C, \mathcal{B}_C\right)$, $\mathcal{B}_C = \left(\vec{i}_C, \vec{j}_C, \vec{k}_C\right)$
- le repère image $\mathcal{F}_I\left(\mathcal{O}_I, \mathcal{B}_I\right)$, $\mathcal{B}_I = \left(\vec{i}_I, \vec{j}_I, \vec{k}_I\right)$
- la pose du robot est $\mathbf{X}_R = (x_R, y_R, z_R, \theta_R)$
- la pose de la caméra est $\mathbf{X}_C = (x_C, y_C, z_C, \theta_R)$
- la vitesse de la caméra est : $\mathbf{T} = (v_x, v_y, v_z, \omega_x, \omega_y, \omega_z)^T$

La transformation entre le repère du robot et le repère de la caméra peut être décrite en coordonnées homogènes :

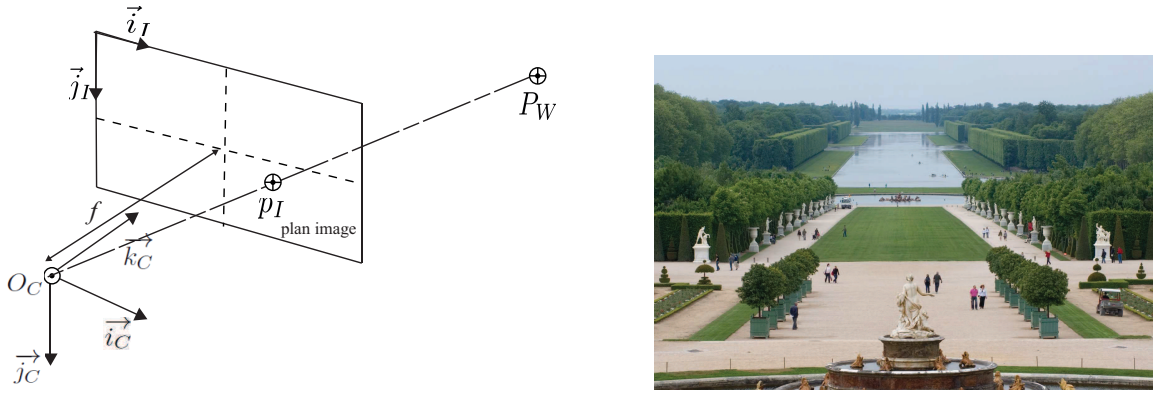
$$\begin{pmatrix} x_C \\ y_C \\ z_C \\ 1 \end{pmatrix}_{\mathcal{F}_C} = \begin{pmatrix} 0 & -1 & 0 & -W \\ 0 & 0 & -1 & H \\ 1 & 0 & 0 & -L \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_R \\ y_R \\ z_R \\ 1 \end{pmatrix}_{\mathcal{F}_R} \tag{2.1}$$

Pour un robot mobile non holonome, deux variables de commande servent à contrôler le robot. Elles correspondent à la vitesse linéaire et à la vitesse de rotation du robot. $\mathbf{U} = \begin{pmatrix} v & \omega \end{pmatrix}^T$. v est définie

positive dans la direction de $\overrightarrow{i_R}$ et ω est définie positive autour de $\overrightarrow{k_R}$.

2.2.1.2 Projection Perspective

Nous rappelons dans cette partie la projection d'un point 3D d'une scène dans une image perspective.



(a) (b)

FIGURE 2.3 – (a) La projection d'un point $3D$ du monde P_W dans une image $2D$, (b) La projection perspective est bien illustrée dans cette image puisque les segments parallèles du jardin du château de Versailles deviennent des droites qui s'intersectent dans l'image

La modélisation de la génération d'image perspective se base sur un modèle sténopé. Ce modèle s'appelle aussi *pin-hole* car tous les rayons lumineux passent par le centre optique (figure 2.3a). La distance entre le capteur et le centre optique est la distance focale f. Pour ce modèle, la projection d'une scène 3D dans une image 2D est une projection perspective. Ainsi, un point 3D $P_W = (X_P, Y_P, Z_P, 1)$ exprimé dans le repère caméra $\mathcal{F}_C$ est projeté dans le plan image normalisé en $p_I = (x_P, y_P, 1)$. Les coordonnées métriques dans l'image sont telles que :

$$x_P = f\frac{X_P}{Z_P}, \;\; y_P = f\frac{Y_P}{Z_P} \tag{2.2}$$

Le point p_I est exprimé dans le plan image normalisé. Il s'agit d'un repère métrique défini par le centre de projection de la caméra comme origine, et qui constitue le centre de l'image. Du fait de l'implémentation informatique classique, l'image est stockée comme un tableau dont l'origine se situe dans le coin en haut à gauche et pas au centre de l'image. Une transformation est donc nécessaire pour ramener un point de coordonnées métriques normalisées vers un point exprimé dans l'image avec des coordonnées pixelliques. Les paramètres permettant cette transformation sont les paramètres intrinsèques de la caméra : $(u_0, v_0, \alpha_u, \alpha_v)$, où u_0 et v_0 sont les coordonnées du centre principal dans l'image. α_u et α_v sont des paramètres de changement d'échelle, qui s'écrivent en fonction de la distance focale f et de la densité de pixels en chaque direction de l'image k_u et k_v : $\alpha_u = f\,k_u$ et $\alpha_v = f\,k_v$. Avec ces paramètres, l'équation (2.3) devient :

$$x_P = \frac{X_P}{Z_P}, \;\; y_P = \frac{Y_P}{Z_P} \tag{2.3}$$

Enfin les coordonnées pixelliques du point p_I sont :

$$p_I = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \mathbf{K} \begin{pmatrix} x_p \\ y_p \\ 1 \end{pmatrix} \tag{2.4}$$

avec la matrice intrinsèque :

$$\mathbf{K} = \begin{pmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.5}$$

2.2.2 Extraction de la signature visuelle

Dans un environnement intérieur, la brique de navigation dans les couloirs est indispensable. Pour cette raison, la capacité d'extraction de l'information visuelle nécessaire est importante. La trajectoire de référence généralement utilisée pour le déplacement dans un couloir est la médiane. Par conséquent, la position du robot est asservie sur la médiane du couloir. Le but de cet asservissement est d'annuler l'écart angulaire ϕ et l'écart latéral y par rapport à la médiane Δ pour que la position de la caméra soit sur Δ et orientée parallèlement par rapport aux murs.

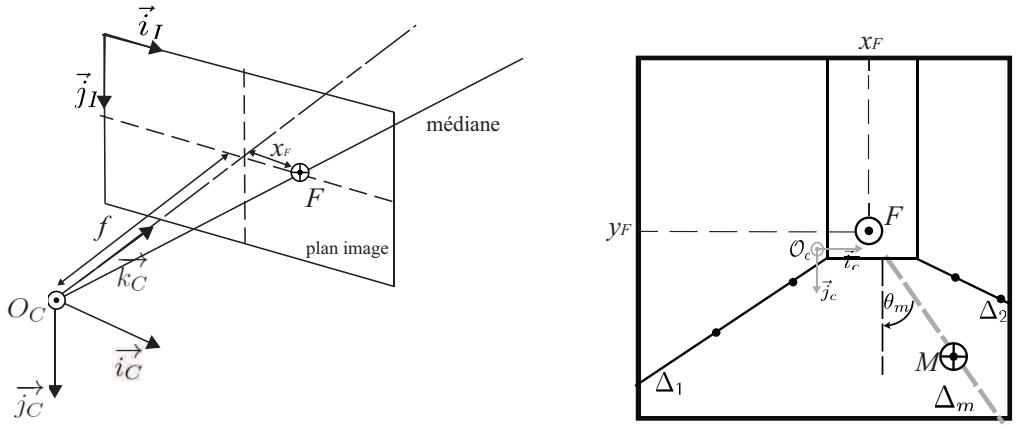


FIGURE 2.4 – Vue embarquée du problème, projection du monde sur le plan image.

Si l'axe $\vec{i}_w$ est pris colinéaire avec la ligne médiane, alors l'écart angulaire correspond aussi à l'orientation du robot : $\phi = \theta_R$. Par conséquent, il y a une relation entre cette orientation et l'abscisse x_F dans l'image du point de fuite $\mathbf{F} = (x_F, y_F)$, en fonction de la distance focale de la caméra f (figure 2.4a) :

$$\phi = arctan(x_F/f) \tag{2.6}$$

L'écart latéral y s'écrit en fonction de l'orientation de la ligne médiane θ_m :

$$y = H_s \tan(\theta_m) \tag{2.7}$$

Le fait d'annuler l'écart angulaire et l'écart latéral par rapport à la médiane revient à annuler

l'abscisse du point de fuite et l'orientation de la ligne médiane dans l'image. Par la suite, il est nécessaire de les extraire pour faire de l'asservissement visuel. La méthode d'extraction de la médiane du couloir s'appuie sur l'extraction des lignes Δ_1 et Δ_2 d'intersections des murs (supposés verticaux) avec le sol supposé horizontal. Cette méthode peut être déduite de l'image présentée sur la figure 2.4.
L'extraction des droites Δ_1 et Δ_2 peut se faire en utilisant plusieurs techniques de traitement d'images :

- dans [Vassallo et al., 2000] un filtre de Sobel de détection des contours dans une image [Vincent et Folorunso, 2009], .
- [Herout et al., 2013] utilise une transformée de Hough
- dans [Delage et al., 2006] un modèle de réseau bayésien dynamique est appliqué sur chaque colonne de l'image pour estimer la limite du sol
- dans [Pasteau et al., 2013] une méthode de Line Segment Detection (LSD) est utilisée.

La sortie des algorithmes de détection de lignes dans un couloir sont les paramètres ρ_i et θ_i avec $i = 1; 2$. Ces données sont identifiées dans le repère image $(O_I, \mathcal{B}_I)$. L'angle θ_i est défini et signé entre $\overrightarrow{i_I}$ et la droite normale à Δ_i (figure 2.10).

Les deux droites sont alors d'équation $x\cos(\theta_i) + y\sin(\theta_i) - \rho_i = 0$

La droite médiane du couloir peut se calculer en utilisant les paramètres des droites Δ_1 et Δ_2 et le point de fuite F qui correspond à l'intersection de ces deux droites :

$$\begin{aligned} \theta_m &= \arctan\left[\frac{1}{2}\left(\frac{\tan(\theta_1)+\tan(\theta_2)}{\tan(\theta_1)\tan(\theta_2)}\right)\right] + \frac{3\pi}{2} \\ \rho_m &= y_F\sin(\theta_m) + x_F\cos(\theta_m) \end{aligned} \tag{2.8}$$

2.2.3 Asservissement visuel appliqué au couloir

Nous rappelons et analysons ici la commande proposée dans [Pasteau et al., 2013] qui permet au robot d'être au milieu du couloir avec une direction parallèle aux murs. La signature visuelle $\mathbf{s}$ est constituée des deux variables $\mathbf{s} = (x_F, \theta_m)^T$. La signature de consigne est $\mathbf{s}^* = (x_F^*, \theta_m^*)^T = (0,\, 0)^T$. L'erreur d'asservissement s'écrit donc : $\varepsilon_{\mathbf{s}} = (\mathbf{s} - \mathbf{s}^*) = (\varepsilon_{\mathbf{x}}, \varepsilon_\theta)^T$

Pour réaliser l'asservissement visuel, il faut calculer la matrice d'interaction. Cette matrice correspond dans ce cas à l'évolution de la signature $\mathbf{s} = (x_F, \theta_m)^T$ en fonction des mouvements de la caméra.

$$\dot{\mathbf{s}} = \mathbf{L_{sr}}\,\mathbf{T} \tag{2.9}$$

Pour relier l'évolution de la signature avec la commande $\mathbf{U} = (v, \omega)^T$ et en notant $[\mathbf{P_{T\leftarrow U}}]$ la matrice de passage entre la commande $\mathbf{U}$ et la vitesse de la caméra $\mathbf{T}$ dans le repère monde nous avons :

$$\dot{\mathbf{s}} \;=\; [\mathbf{L}_{sr}]\,.\,[\mathbf{P_{T\leftarrow U}}]\,.\mathbf{U} \tag{2.10}$$

avec,

$$\mathbf{L_{sr}} = \begin{pmatrix} L_{x_F} \\ L_{\theta_m} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & x_F.y_F & -(x_F^2+1) & y_F \\ \dfrac{cos(\theta_m)^2}{H} & \dfrac{sin(\theta_m)cos(\theta_m)}{H} & \dfrac{-\rho_m\,cos(\theta_m)}{H} & -\rho_m\,cos(\theta_m) & -\rho_m\,sin(\theta_m) & -1 \end{pmatrix} \tag{2.11}$$

$$\mathbf{P}_{\mathbf{T}\leftarrow\mathbf{U}} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ -L & 0 & -W & 0 & -1 & 0 \end{pmatrix}^T \tag{2.12}$$

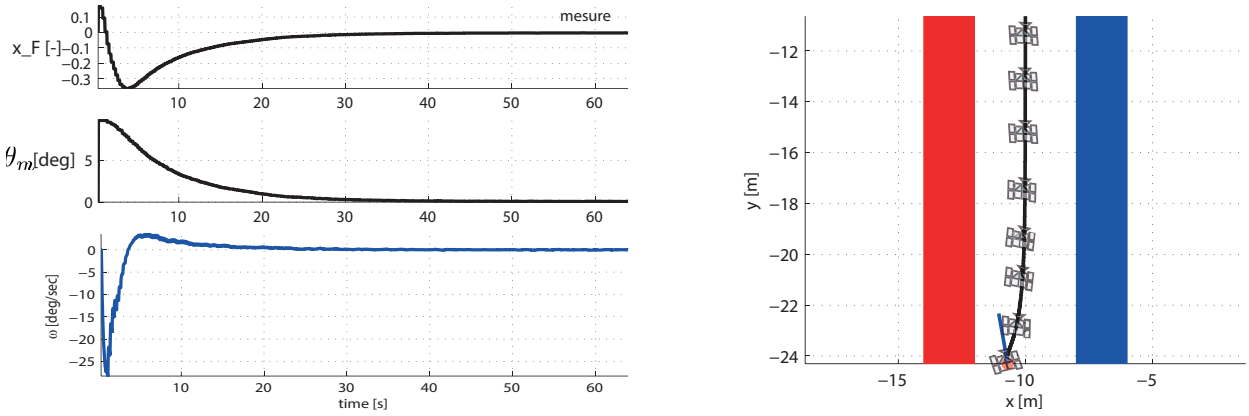
Pour faire converger la signature $\mathbf{s} = (x_F, \theta_m)^T$ vers la signature de consigne $\mathbf{s}^* = (0, 0)^T$, une vitesse linéaire est prise comme constante et seule la vitesse angulaire est commandée.

La commande proposée dans [Pasteau et al., 2013] est

$$\omega = \begin{pmatrix} J_{x\omega} \\ J_{\theta\omega} \end{pmatrix}^{\dagger} \left(\mathbf{\Lambda} \begin{pmatrix} \varepsilon_x \\ \varepsilon_\theta \end{pmatrix} - \begin{pmatrix} 0 \\ J_{\theta\vartheta}\, v \end{pmatrix} \right) \tag{2.13}$$

avec $\mathbf{\Lambda}$ est une matrice de gain, $J_{x\omega} = x_F^2 + 1$,
$J_{\theta\omega} = \left(-\frac{L}{H_s}\cos^2(\theta_m) + \frac{W}{H_s}\rho_m \cos(\theta_m) + \rho_m \sin(\theta_m) \right)$ et $J_{\theta v} = -\frac{\rho_m}{H_s}\cos(\theta_m)$

Nous avons implémenté cette approche en utilisant l'environnement de simulation Morse/Blender/ROS (F). La figure 2.5 présente l'évolution de la signature au cours du temps ainsi que la commande ω. La signature visuelle converge vers la signature désirée en appliquant la loi de commande (2.13). Ainsi le robot arrive à rallier puis à suivre la médiane du couloir (figure 2.5b) en assurant la convergence du point de fuite et de l'orientation de la médiane dans l'image vers les valeurs de consigne.



(a) Evolution de la signature $\mathbf{s} = (x_F, \theta_m)^T$ et la commande ω appliquée au robot

(b) La trajectoire du robot

FIGURE 2.5 – Résultats de simulation lorsque la commande (2.13) est appliquée

2.2.4 Limites d'utilisation

Cette méthode est efficace lorsque l'environnement est maîtrisé et correspond à la description du couloir. Cependant elle présente des inconvénients. En effet, il y a des perturbations en milieu intérieur pour l'information visuelle à cause de la variation de luminosité, de la spécularité et du scintillement des lampes.

La commande référencée capteurs est également soumise aux bruits de mesures ou à des défauts du capteur ou de calibrage. Dans ce cas, les mesures de la signature visuelle ne peuvent pas être

(a) image réelle

(b) Les droites détectées dans l'image (segments blancs)

FIGURE 2.6 – La détection de droites dans une image réelle

extraites ou donneront des valeurs erronées. Les mesures brutes influent directement sur la loi de commande car elle est proportionnelle à l'erreur, de sorte qu'une navigation sûre et confortable n'est pas garantie lorsque la méthode est conçue pour piloter de façon autonome un robot ou un fauteuil roulant [Pasteau et al., 2013].

Il y a également les situations où les mesures ne sont pas disponibles. En se déplaçant le long d'un couloir en utilisant une commande d'asservissement visuel, il existe de nombreux cas qui provoquent l'absence de mesures, comme :

- lorsque les deux bords du couloir ne sont pas visibles dans l'image à cause du champ de vision restreint de la caméra,
- lorsque l'intersection des deux lignes n'est pas sur l'image,
- la défaillance du capteur.

Pour l'asservissement visuel dans un couloir, la détection de lignes des plinthes dans l'image n'est pas une tâche facile. Dans une image réelle plusieurs segments peuvent être détectés (figure 2.6). Pour extraire la signature visuelle une sélection de lignes est nécessaire. Ceci engendre plus de temps de calcul pour une tâche destinée à des applications embarquées avec des contraintes temps réel.

Un autre inconvénient de cette approche est la limitation de l'environnement. Cet asservissement visuel fonctionne juste dans un couloir (présence de deux murs pour extraire la signature). Si l'environnement présente des obstacles ou des virages de grande courbure, la loi de commande risque de ne plus être fonctionnelle. Pour surmonter ces limites, nous proposons une autre stratégie de mesure et de commande qui sera présentée dans la section suivante.

2.3 Asservissement visuel virtuel dans un couloir

La stratégie d'asservissement visuel dans un couloir détaillée dans la section précédente (2.2) est une méthode efficace pour une navigation autonome dans un couloir. En revanche, vu qu'un capteur de vision est utilisé, la tâche d'extraction de la signature est parfois difficile à réaliser suite à la perte de vue des lignes caractéristiques, le manque de luminosité ou les problèmes de calibrage de la caméra

(pose et paramètres intrinsèques). Pour limiter ces problèmes, nous proposons une nouvelle stratégie d'asservissement visuel basé sur un capteur virtuel. Nous conservons le formalisme d'asservissement du point de fuite et de la médiane dans un couloir, mais l'outil de mesure et la stratégie de commande seront changés. Un repère virtuel unique et unifié est introduit, la signature et la commande sont exprimées dans ce référentiel. Ce cadre virtuel sera construit à partir des observations réelles issues des capteurs physiquement embarqués sur le robot. L'utilisation de ce capteur virtuel peut permettre de :

- fusionner l'information de plusieurs capteurs en utilisant des observateurs d'état, un filtrage de Kalman...,
- définir un repère unique pour concevoir la commande,
- définir un positionnement du capteur adapté à la tâche,
- paramétrer le capteur virtuel pour mettre en avant certaine paramètres.

Pour limiter les bruits de mesure ou les défauts de capteurs, la stratégie de commande se base sur une estimation de l'évolution de l'état du capteur par un observateur en temps fini en utilisant l'entrée et la sortie du système et son modèle local.
Nous rappelons que la commande utilisée pour les travaux précédents sert à asservir deux variables extraites dans l'image : le point de fuite qui sert à rendre la direction du robot parallèle aux murs, et l'orientation de la médiane dans l'image qui permet au robot d'être sur la médiane. Pour notre cas, nous montrons qu'en utilisant un capteur virtuel avec un positionnement spécifique comme nous le souhaitons, l'asservissement visuel dans un couloir peut se faire avec une seule variable visuelle. Il s'agit juste d'asservir l'orientation de la médiane dans l'image pour avoir un robot qui rallie et suit la médiane dans le monde.

2.3.1 Modélisation

La construction d'une loi de commande nécessite avant tout la modélisation du système à commander. Dans cette partie, nous présentons le modèle du système utilisé avec tous les repères associés et les matrices de passage associées.

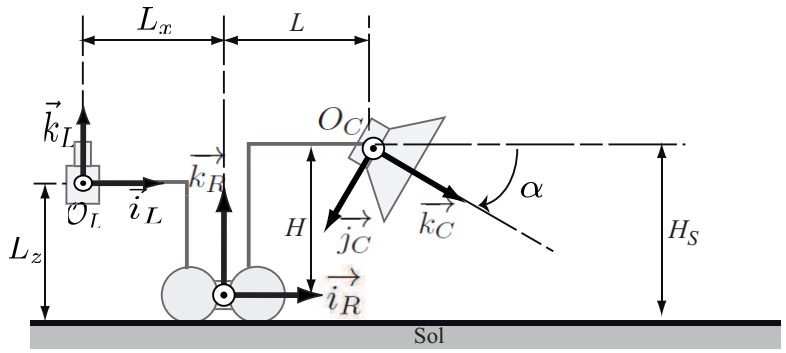
Pour reprendre et étendre le travail de [Pasteau et al., 2013], nous prenons comme capteur virtuel une caméra perspective ayant comme repère $\mathcal{F}_C$. Cette caméra virtuelle sera embarquée sur le robot avec $\mathbf{X}_C = \begin{pmatrix} L & W & H \end{pmatrix}_{\mathcal{F}_R}$. Elle est orientable avec un angle $(\vec{i}_R \rightarrow \vec{k}_C) = \alpha$ avec $\alpha \in [-\pi/2\ ;\ 0]$ (figure 2.7) avec une hauteur H_s par rapport au sol.

La matrice de passage entre le repère monde et le repère capteur virtuel est :

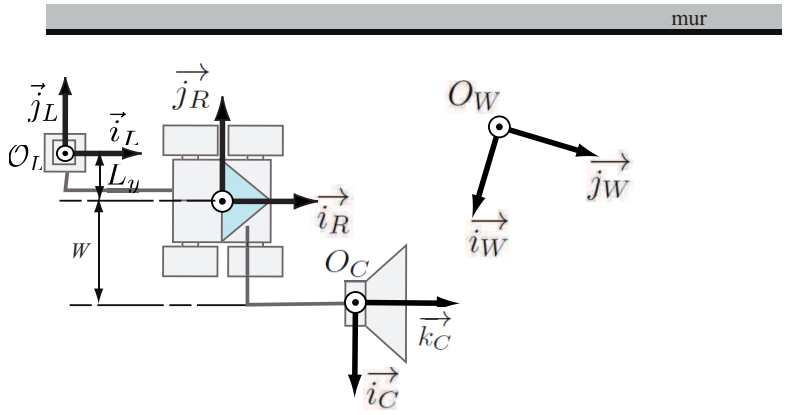
$$[\mathcal{T}_{\mathcal{F}_W \leftarrow \mathcal{F}_C}] = [\mathcal{T}_{\mathcal{F}_W \leftarrow \mathcal{F}_R}].[\mathcal{T}_{\mathcal{F}_R \leftarrow \mathcal{F}_C}] \tag{2.14}$$

avec la matrice de passage entre le repère monde et le repère robot :

$$[\mathcal{T}_{\mathcal{F}_W \leftarrow \mathcal{F}_R}] = \begin{pmatrix} \cos(\theta_R) & -\sin(\theta_R) & 0 & x_R \\ \sin(\theta_R) & \cos(\theta_R) & 0 & y_R \\ 0 & 0 & 1 & z_R \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.15}$$



(a) Vue de côté du problème



(b) Vue de dessus du problème

FIGURE 2.7 – La représentation du repère du robot, repère de l'observation réelle $\mathcal{F}_L$ et de l'observation virtuelle $\mathcal{F}_C$

la matrice de passage entre le repère robot et le repère caméra est :

$$[\mathcal{T}_{\mathcal{F}_R \leftarrow \mathcal{F}_C}] = \begin{pmatrix} 0 & -1 & 0 & -W \\ sin(\alpha) & 0 & -cos(\alpha) & L\,sin(\alpha) - H\,cos(\alpha) \\ cos(\alpha) & 0 & sin(\alpha) & L\,cos(\alpha) + H\,sin(\alpha) \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.16}$$

2.3.1.1 Modèle cinématique du capteur virtuel

Le torseur cinématique du système caméra virtuelle S_C auquel on associe un repère $\mathcal{F}_C = (\mathcal{O}_C; \mathcal{B}_C)$, par rapport au repère du monde :

$$\{\mathcal{V}_{S_C/\mathcal{F}_W}\} \equiv \left\{ \begin{array}{c} \vec{\Omega}_{\mathcal{B}_C/\mathcal{B}_W} \\ \\ \vec{V}_{O_C \in S_C/\mathcal{F}_W} \end{array} \right\} \equiv \left\{ \begin{array}{cc} \omega_x & v_x \\ \omega_y & v_y \\ \omega_z & v_z \end{array} \right\} \tag{2.17}$$

La vitesse linéaire de la caméra dans le repère monde est

$$\begin{aligned}\overrightarrow{V}_{O_C\in S_C/\mathcal{F}_W} &= \left[\frac{d}{dt}\overrightarrow{O_WO_C}\right]_{\mathcal{B}_W} = \left[\frac{d}{dt}\overrightarrow{O_WO_R}\right]_{\mathcal{B}_W} + \left[\frac{d}{dt}\overrightarrow{O_RO_C}\right]_{\mathcal{B}_R} + \overrightarrow{\Omega}_{\mathcal{B}_R/\mathcal{B}_W} \wedge \overrightarrow{O_RO_C} \\ &= \begin{pmatrix} v - W\dot{\theta}_R \\ L\dot{\theta}_R \\ 0 \end{pmatrix}_{\mathcal{B}_R} = \begin{pmatrix} -L\dot{\theta}_R \\ \left(v - W\dot{\theta}_R\right) sin(\alpha) \\ \left(v - W\dot{\theta}_R\right) cos(\alpha) \end{pmatrix}_{\mathcal{B}_C}\end{aligned} \tag{2.18}$$

La vitesse de la caméra dans le repère monde est exprimée dans la base du robot pour la première partie de l'équation (2.18) et dans la base associée à la caméra dans la seconde partie.

La vitesse de rotation de la base de la caméra par rapport à la base associée au monde est identique à celle du robot.

$$\overrightarrow{\Omega}_{\mathcal{B}_C/\mathcal{B}_W} = \overrightarrow{\Omega}_{\mathcal{B}_R/\mathcal{B}_W} = \begin{pmatrix} 0 \\ 0 \\ \dot{\theta}_R \end{pmatrix}_{\mathcal{B}_R} = \begin{pmatrix} 0 \\ -cos(\alpha)\dot{\theta}_R \\ sin(\alpha)\dot{\theta}_R \end{pmatrix}_{\mathcal{B}_C} \tag{2.19}$$

Les déplacements en translation et en rotation de la caméra s'écrivent alors en fonction des commandes :

$$\overrightarrow{V}_{O_C\in S_C/\mathcal{F}_W} = \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix}_{\mathcal{B}_C} = \begin{pmatrix} -L\omega \\ (v - W\omega)\, sin(\alpha) \\ (v - W\omega)\, cos(\alpha) \end{pmatrix}_{\mathcal{B}_C} \text{ et } \overrightarrow{\Omega}_{\mathcal{B}_C/\mathcal{B}_W} = \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix}_{\mathcal{B}_C} = \begin{pmatrix} 0 \\ -cos(\alpha)\omega \\ sin(\alpha)\omega \end{pmatrix}_{\mathcal{B}_C} \tag{2.20}$$

Le robot peut être équipé d'un ou de plusieurs capteurs physiques (un lidar, un sonar, une caméra perspective ou une caméra omnidirectionnelle). L'idée est de projeter toutes les mesures disponibles de ces capteurs vers le repère du capteur virtuel puis dans l'image associée.

Nous allons faire l'étude de projection avec un capteur lidar comme illustré dans la figure 2.7. La base du repère lidar $\mathcal{B}_L$ est translatée de la base du repère robot telle que :

$$\overrightarrow{\mathcal{O}_R\mathcal{O}_L} = \begin{pmatrix} L_x \\ L_y \\ L_z \end{pmatrix}_{\mathcal{B}_\mathcal{R}} \tag{2.21}$$

La matrice de passage entre repère robot $\mathcal{F}_R$ et le repère de l'observation réelle $\mathcal{F}_L$

$$[\mathcal{T}_{\mathcal{F}_R\leftarrow\mathcal{F}_L}] = \begin{pmatrix} 1 & 0 & 0 & L_x \\ 0 & 1 & 0 & L_y \\ 0 & 0 & 1 & L_z \end{pmatrix} \tag{2.22}$$

2.3.2 Génération d'image virtuelle et construction de la signature

Dans ce chapitre, le capteur virtuel sera orienté vers l'avant. Il s'agit d'une caméra perspective frontale. Étant donné que le point de fuite doit être extrait pour construire l'observateur et par la suite la loi de commande, nous avons gardé la représentation de [Pasteau et al., 2013]. La caméra perspective

frontale permet de déformer les lignes parallèles des plinthes dans une image perspective pour qu'elles s'intersectent et qu'on puisse déduire le point de fuite. Dans ce cas, l'orientation de la caméra α est choisie nulle.

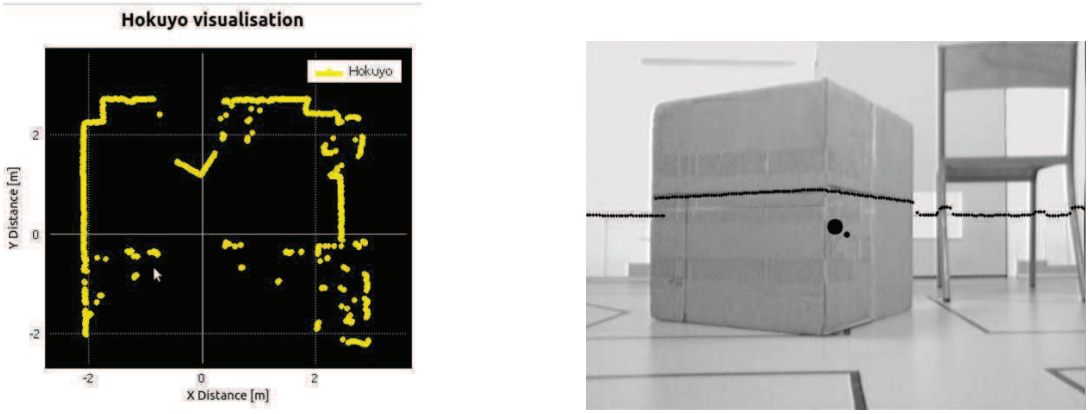
2.3.2.1 Génération d'une image virtuelle

Nous disposons comme données des informations réelles du capteur lidar. L'information brute donne les coordonnées polaires dans le repère lidar : l'angle a et la distance d de chaque impact détecté P_I. Il est ensuite assez simple de transformer ces informations en coordonnées cartésiennes $X_{P_I} = d\sin(a)$ et $Y_{P_I} = d\cos(a)$. En connaissant la mesure du lidar et la position relative entre le capteur virtuel et le lidar, nous avons les coordonnées de l'impact dans le repère du capteur virtuel :

$$\overrightarrow{\mathcal{O}_C P_I}_{/\mathcal{B}_\mathcal{C}} = \overrightarrow{\mathcal{O}_C \mathcal{O}_L}_{/\mathcal{B}_\mathcal{C}} + \overrightarrow{\mathcal{O}_L P_I}_{/\mathcal{B}_\mathcal{C}} = \begin{pmatrix} X^c_{P_I} \\ Y^c_{P_I} \\ Z^c_{P_I} \end{pmatrix}_{\mathcal{B}_\mathcal{C}} \tag{2.23}$$

En appliquant le modèle de projection d'une caméra perspective, la nappe laser est exprimée dans la base de la caméra de coordonnées normalisées $(x_{Li},\, y_{Li})^T = \left(\frac{X^c_{P_I}}{Z^c_{P_I}},\, \frac{Y^c_{P_I}}{Z^c_{P_I}}\right)^T$, puis les coordonnées pixelliques sont déterminées en connaissant les paramètres intrinsèques de la caméra virtuelle $(u_0, v_0, \alpha_u, \alpha_v)$ avec $(u_{L1}\, v_{L1})^T = (\alpha_u.x_{Li} + u_0,\, \alpha_v.y_{Li} + v_0)^T$.

Nous pouvons avoir plusieurs observations projetées donc plusieurs projections issues de différents capteurs $\{(u_{L1}, v_{L1}), (u_{L2}, v_{L2}), (u_{L3}, v_{L3}), ..., (u_{Ln}, v_{Ln}))\}$



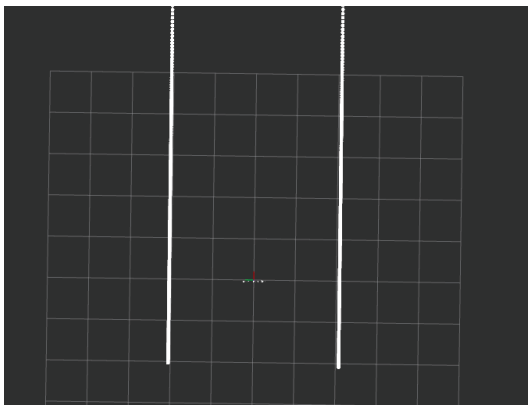
(a) Visualisation réelle du lidar

(b) Projection dans l'image

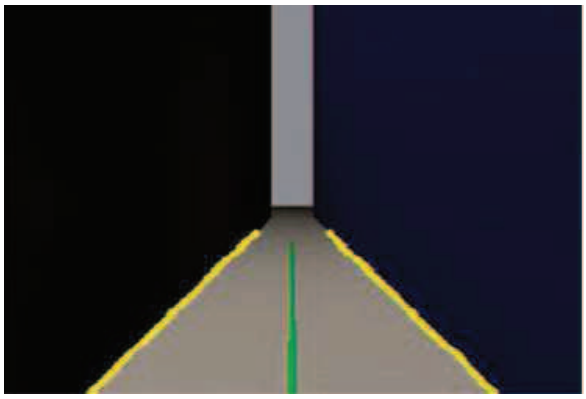
FIGURE 2.8 – La projection des données d'un capteur lidar dans une image perspective.

2.3.2.2 Extraction de la signature

La projection des données lidar d'un couloir dans une image permet d'extraire les deux droites correspondant aux intersections du plan lidar avec les murs du couloir (figure 2.9). Pour la détection de lignes dans une image virtuelle cette tâche s'avère facile par rapport à un traitement d'image classique. Seules deux droites peuvent être détectées dans un couloir non encombré par un lidar. L'algorithme



(a) Visualisation réelle des données lidar dans un couloir



(b) Les points jaunes sont la projection des données lidar dans l'image

FIGURE 2.9 – Pour un environnement couloir : la projection des données réelles d'un capteur lidar dans une image perspective.

que nous avons utilisé pour la détection des deux lignes du bas des murs, est une méthode adaptée de la transformée de Hough. Il s'agit de prendre tous les points projetés dans l'image virtuelle. Si les points sont proches et condensés il s'agit de l'un des murs. Une régression linéaire est réalisée pour chercher les paramètres des droites. Des points parasités peuvent être présents dans l'image à cause de la présence des objets collés aux murs comme des radiateurs, des portes que le lidar peut détecter. Pour cela une sélection est faite et nous ne gardons que les plus longs morceaux de lignes.

Par construction de la perspective, le point de fuite de la vision embarquée est à l'intersection de ces deux droites. Les coordonnées sont alors $\overrightarrow{O_I F} = \begin{pmatrix} x_F^I & y_F^I & 1 \end{pmatrix}^T_{\mathcal{B}_I}$ avec

$$x_F^I = -\frac{\rho_1 \sin(\theta_2) - \rho_2 \sin(\theta_1)}{\sin(\theta_1 - \theta_2)} \tag{2.24}$$

$$y_F^I = \frac{\rho_1 \cos(\theta_2) - \rho_2 \cos(\theta_1)}{\sin(\theta_1 - \theta_2)} \tag{2.25}$$

Par construction également, la droite médiane du couloir est calculée à partir des droites Δ_1 et Δ_2. Ainsi, on peut identifier les paramètres de la médiane du couloir par

$$\theta_3 = \arctan\left[\frac{1}{2}\left(\frac{\tan(\theta_1) + \tan(\theta_2)}{\tan(\theta_1)\tan(\theta_2)}\right)\right] \tag{2.26}$$

et ρ_3, la distance à O_I

$$\rho_3 = y_F^I \sin(\theta_3) + x_F^I \cos(\theta_3) \tag{2.27}$$

Ce paramétrage est écrit dans le repère image (figure 2.10). Il est maintenant nécessaire de le replacer dans le repère caméra.

Ayant identifié θ_3 dans le repère lié à l'image, on peut en déduire l'angle θ_m que nous cherchons à asservir dans le cas de notre asservissement visuel (figure 2.11).

$$\theta_m = \theta_3 + \frac{3\pi}{2} \tag{2.28}$$

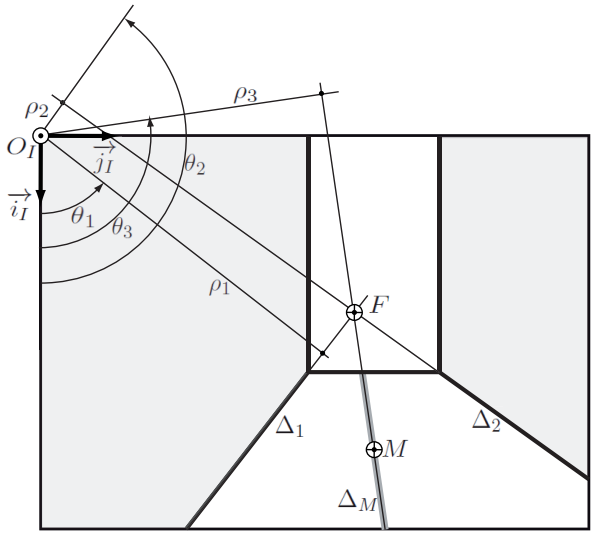


FIGURE 2.10 – Données utiles à l'extraction de la médiane du couloir. Dans le repère image, plan image.

En revanche, pour la variable ρ_m, un calcul supplémentaire est nécessaire. En effet, lors de l'acquisition de données et de l'extraction de ρ_3 dans le repère image, l'ensemble des mesures était en pixels. Or, pour notre asservissement visuel, les données doivent être au format de la perspective mais avant la transformation sur le plan image.

Pour un point P quelconque de coordonnées $\begin{pmatrix} X_P & Y_P & Z_P \end{pmatrix}^T_{\mathcal{B}_C}$, on a :

$$\overrightarrow{O_I P} = \begin{pmatrix} k_x f & 0 & v_0 \\ 0 & k_y f & u_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{X_P}{Z_P} \\ \frac{Y_P}{Z_P} \\ 1 \end{pmatrix}_{\mathcal{B}_C} = \begin{pmatrix} k_x f x_P + v_0 \\ k_y f y_P + u_0 \\ 1 \end{pmatrix}_{\mathcal{B}_I} \tag{2.29}$$

Ainsi, pour obtenir les coordonnées du point F dans le repère de la caméra à partir des données identifiées dans le repère image, on applique

$$\overrightarrow{O_C F} = \begin{pmatrix} x_F \\ y_F \\ 1 \end{pmatrix}_{\mathcal{B}_C} = \begin{pmatrix} \frac{1}{k_x f} & 0 & \frac{-v_0}{k_x f} \\ 0 & \frac{1}{k_y f} & \frac{-u_0}{k_y f} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_F^I \\ y_F^I \\ 1 \end{pmatrix}_{\mathcal{B}_I} \tag{2.30}$$

Ce qui permet de déduire :

$$\rho_m = y_F \sin(\theta_m) + x_F \cos(\theta_m) \tag{2.31}$$

Pour une navigation dans un couloir, au lieu d'asservir une signature visuelle complète (ρ_m, θ_m), nous allons montrer que l'utilisation d'une seule partie de la signature est suffisante en utilisant une caméra virtuelle.

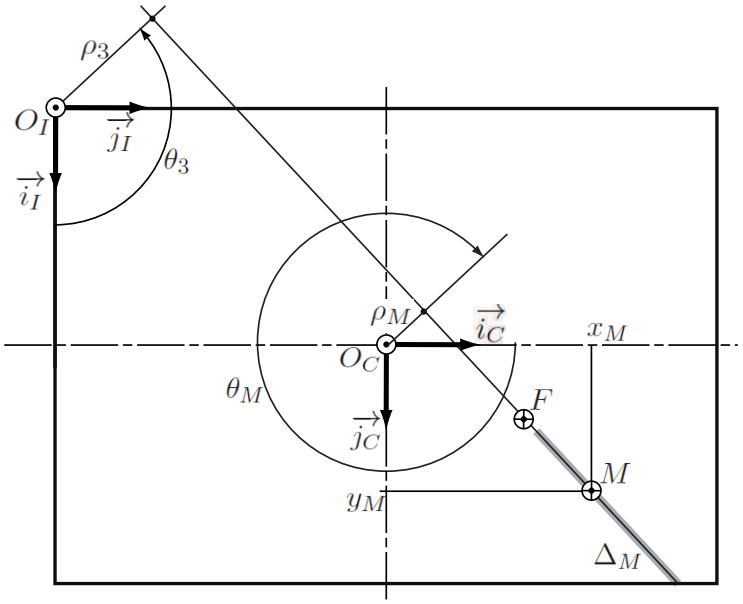


FIGURE 2.11 – Variables entre le repère image et le repère caméra.

Le capteur virtuel suit le milieu du couloir lorsque seule l'orientation de la médiane du couloir est asservie. D'après la modélisation utilisée, la variable θ_m dont nous réalisons l'asservissement représente le fait que la caméra virtuelle se déplace sur la médiane du couloir. Nous montrons ici qu'elle sera également orientée parallèlement aux murs.

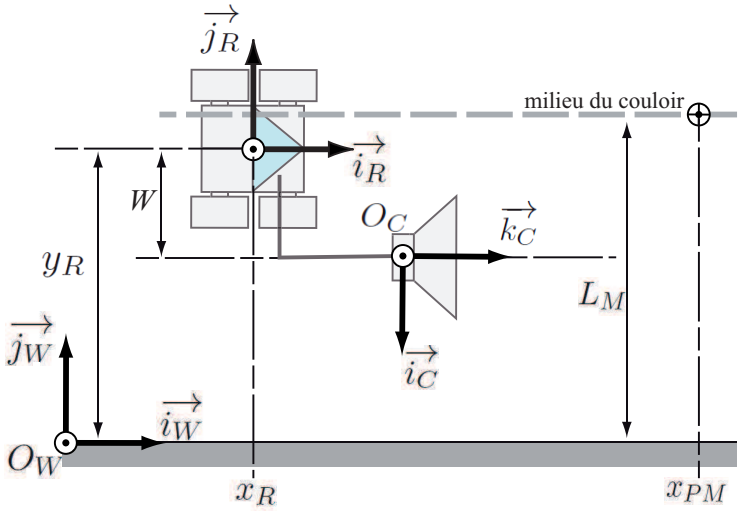


FIGURE 2.12 – Illustration du fait que lorsque $\theta_m = 0$, la caméra suit le milieu du couloir.

Posons le repère monde tel que $\vec{i}_W$ soit parallèle au couloir. Posons une ligne au sol constituée des points P_M représentant la médiane du couloir

$$\overrightarrow{O_W P_M} = \begin{pmatrix} x_{PM} & y_{PM} & z_{PM} \end{pmatrix}^T_{\mathcal{B}_W} = \begin{pmatrix} x_{PM} & L_M & 0 \end{pmatrix}^T_{\mathcal{B}_W} \tag{2.32}$$

où L_M est la demi largeur du couloir. Vu du capteur virtuel, nous noterons

$$\overrightarrow{O_C P_M} = \begin{pmatrix} X_M & Y_M & Z_M \end{pmatrix}^T_{\mathcal{B}_C} \tag{2.33}$$

En appliquant les différentes transformations, nous pouvons écrire le lien entre les coordonnées d'un point dans le repère monde et dans le repère du capteur virtuel.

$$\begin{aligned}
\overrightarrow{O_C P_M} &= \overrightarrow{O_C O_R} + \overrightarrow{O_R O_W} + \overrightarrow{O_W P_M} \\
\overrightarrow{O_C P_M}\Big|_{\mathcal{B}_C} &= \begin{pmatrix} -W \\ -H_s \\ L \end{pmatrix}_{\mathcal{B}_C} + [T_{C\leftarrow W}] \begin{pmatrix} -x_R \\ -y_R \\ 0 \end{pmatrix}_{\mathcal{B}_W} + [T_{C\leftarrow W}] \begin{pmatrix} x_{PM} \\ L_M \\ 0 \end{pmatrix}_{\mathcal{B}_W} \\
&= \begin{pmatrix} -W + (x_{PM} - x_R)\sin(\theta_R) + (y_R - L_M)\cos(\theta_R) \\ -H_s \\ L - (x_{PM} + x_R)\cos(\theta_R) + (L_M - y_R)\sin(\theta_R) \end{pmatrix}_{\mathcal{B}_C}
\end{aligned}$$

Appliquant la transformation induite par une caméra perspective, nous obtenons :

$$\begin{aligned}
x_M &= \frac{X_M}{Z_M} \\
&= \frac{-W + (x_{PM} - x_R)\sin(\theta_R) + (y_R - L_M)\cos(\theta_R)}{L - (x_{PM} + x_R)\cos(\theta_R) + (L_M - y_R)\sin(\theta_R)}
\end{aligned} \tag{2.34}$$

$$\begin{aligned}
y_M &= \frac{Y_M}{Z_M} \\
&= \frac{-H_s}{L - (x_{PM} + x_R)\cos(\theta_R) + (L_M - y_R)\sin(\theta_R)}
\end{aligned} \tag{2.35}$$

Ces points forment une droite dans l'image telle que

$$x_M \cos(\theta_m) + y_M \sin(\theta_m) = \rho_m$$

En appliquant la commande précédente, on garantit que, asymptotiquement θ_m tend vers 0. Lorsque la médiane est verticale dans l'image, alors

$$x_M = \rho_m$$

En utilisant l'évolution de la variable x_F (A.11) et en considérant que $\theta_m = 0$, alors

$$\begin{aligned}
\dot{\rho}_m &= \dot{x}_F \\
&= \omega(1 + \rho_m^2)
\end{aligned} \tag{2.36}$$

Lorsque $\varepsilon_\theta = 0$, la commande proposée devient $\omega = \dfrac{\rho_m v^c}{-L + W\rho_m}$ ce qui implique

$$\dot{\rho}_m = \frac{\rho_m v^c}{-L + W\rho_m}(1 + \rho_m^2) \tag{2.37}$$

Procédant à l'analyse de cette fonction,

- Si $\rho_m = 0$ alors $\dot{\rho}_m = 0$. C'est un point fixe.
- Si le robot est orienté vers le mur droit du couloir alors $\rho_m \in]-\infty; 0[$, le signe de $\dot{\rho}_m$ dépend du signe de $(W - L/\rho_m)$
  si $L > W\rho_m$, alors ρ_m est une fonction décroissante et $\lim_{t\to+\infty} \rho_m = -\infty$.
  si $L < W\rho_m$, alors ρ_m est une fonction croissante $\lim_{t\to+\infty} \rho_m = 0$
- Si le robot est orienté vers le mur gauche du couloir alors $\rho_m \in]0; +\infty[$
  si $L > l\rho_m$, alors ρ_m est une fonction décroissante et $\lim_{t\to+\infty} \rho_m = 0$.
  si $L < W\rho_m$, alors ρ_m est une fonction croissante et $\lim_{t\to+\infty} \rho_m = +\infty$.

Dans le cas où $\lim_{t\to+\infty} \rho_M = 0$, on a alors $x_M = 0 = constante$,

$$x_M = \frac{-W + (x_{PM} - x_R)\sin(\theta_R) + (y_R - L_M)\cos(\theta_R)}{L - (x_{PM} + x_R)\cos(\theta_R) + (L_M - y_R)\sin(\theta_R)} = 0$$

Cette équation implique que

$$-W + (x_{PM} - x_R)\sin(\theta_R) + (y_R - L_M)\cos(\theta_R) = 0$$

Pour que cela soit vrai quelle que soit l'abscisse du point du monde regardé ($\forall x_{PM}$), il est nécessaire que $\sin(\theta_R) = 0$ et donc que $\theta_R = 0 + k\pi$.

On en déduit alors que la position du robot est $y_R = L_M + W$ et donc que la caméra suit le milieu du couloir avec une orientation parallèle à la médiane.

2.3.3 Commande utilisant l'orientation de la médiane du couloir

2.3.3.1 Asservissement sur θ_m

La loi de commande que nous proposons s'intéresse uniquement à la dynamique de la variable θ_m représentant la position relative du robot par rapport à la médiane du couloir. Faire converger le paramètre θ_m permet de rallier le milieu du couloir. Nous allons montrer qu'avec une représentation virtuelle, en assurant la convergence de cette variable vers 0, cela permet non seulement de rallier le milieu du couloir mais aussi d'obtenir une orientation parallèle aux murs du couloir. Nous allons utiliser uniquement la matrice d'interaction L_{θ_m} (2.11) pour construire la commande. Nous supposons également que le robot est en déplacement avec une vitesse linéaire non nulle, notée v^c.

En utilisant l'évolution d'une droite dans une image décrite dans la partie Annexe (B), la dynamique de θ_m s'écrit :

$$\dot{\theta}_m = -\frac{\rho_m}{H_s}\cos(\theta_m)v^c + \left(-\frac{L}{H_s}\cos^2(\theta_m) + \frac{W}{H_s}\rho_m\cos(\theta_m) + \rho_m\sin(\theta_m)\right)\omega \tag{2.38}$$

avec (ρ_m, θ_m) la mesure effectuée dans l'image et (L, W, H_s) le positionnement du capteur. Il est toutefois nécessaire de connaître la valeur de ρ_m pour évaluer cette expression. Une solution consisterait à utiliser l'équation (2.8) et à procéder à son intégration (sous réserve de connaissance de la condition initiale ou de son observabilité).

Considérons maintenant l'erreur de positionnement par rapport au milieu du couloir

$$\varepsilon_\theta = (\theta_m - \theta_m^*) \tag{2.39}$$

L'angle de référence de la médiane du couloir dans le plan image a toujours la même position de référence tant qu'on se trouve dans le même couloir. Du fait que θ_m^* est constant, la dynamique de l'erreur est : $\dot{\varepsilon}_\theta = \dot{\theta}_m$. Afin de rallier le milieu du couloir avec une convergence exponentielle de l'erreur et une constante de temps τ_θ ($\tau_\theta > 0$), il faudra que :

$$\dot{\varepsilon}_\theta = -\frac{1}{\tau_\theta}\varepsilon_\theta \tag{2.40}$$

En utilisant (B.24) et (2.40), nous pouvons déduire une commande angulaire à appliquer au robot pour rallier le milieu du couloir (lorsqu'il se déplace à vitesse constante v^c) :

$$\omega = -\frac{-\frac{H_s}{\tau_\theta}\varepsilon_\theta + \rho_m \cos(\theta_m) v^c}{-L\cos^2(\theta_m) + W\rho_m \cos(\theta_m) + H_s \rho_m \sin(\theta_m)} \tag{2.41}$$

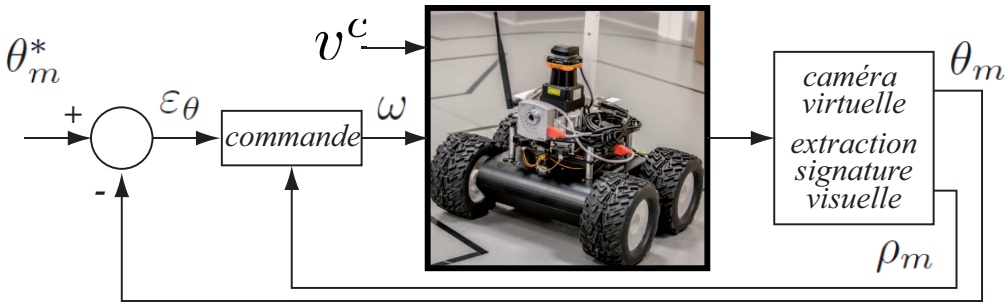


FIGURE 2.13 – Structure de la commande implémentée.

En appliquant la loi de commande (2.41) il faut s'assurer que cette commande n'admet pas de singularités. Certaines conditions doivent être respectées pour obtenir une commande opérationnelle.

- Le capteur virtuel ne doit pas être placée au niveau du sol $H_s \neq 0$. Dans le cas contraire, l'erreur d'alignement ε_θ ne fera plus varier la commande.
- Lorsque le robot se déplace parallèlement au couloir nous rappelons que $x_F = y_F = 0$, nous avons alors $\rho_m = y_F \sin(\theta_m) + x_F \cos(\theta_m) = 0$. Dans ce cas, il est nécessaire que la caméra soit placée à l'avant ou l'arrière de l'axe de rotation du robot $L \neq 0$. Dans le cas contraire, au moment de l'alignement du robot avec le couloir, la commande est infinie (division par 0).
- Cette commande présente une autre singularité. Lorsque le dénominateur est nul. Ce qui implique que la commande devient infinie puis change de signe. Pour éviter ce problème, il faut avoir, (2.42) vérifiée si $\theta_m = 0$:

$$-L + W\rho_m \neq 0 \tag{2.42}$$

Donc il faut choisir un bon emplacement du capteur. Nous avons choisi de travailler avec un capteur virtuel, donc nous avons la possibilité de le placer là où la commande ne présente pas de

singularité. C'est-à-dire la position longitudinale L et la position latérale W ne peuvent pas être toute les deux égales à 0. Il ne faut pas placer le capteur virtuel sur le point caractéristique du robot.

- Une commande nulle est obtenue lorsque :

$$\varepsilon_\theta \, H_s/\tau_\theta - \rho_m \cos(\theta_m) \, v^c = 0 \tag{2.43}$$

Ceci est une position particulière du système tel que

$$\rho_m = \varepsilon_\theta \, H_s/(\tau_\theta v^c \cos(\theta_m)) \tag{2.44}$$

Cette expression est en particulier vraie quand $\varepsilon_\theta = \theta_m = 0$ et $\rho_m = 0$ pour une vitesse $v \neq 0$.

Dans la section suivante, la stabilité de la commande proposée est évaluée localement et le domaine d'attraction est défini par une analyse dans le plan de phase du système. De plus, des conclusions sont données sur le positionnement du capteur pour obtenir une commande stabilisante.

2.3.3.2 Stabilité de la loi de commande proposée

Dans les approches d'asservissement visuel, la robustesse de commande peut être étudiée à partir des erreurs de calibrage de la caméra, des erreurs de mesures [Chaumette, 1998], ou des erreurs de modélisation de la cible [Deng et al., 2002].

Dans notre cas, seule l'orientation de la ligne médiane dans l'image est asservie. Pour suivre le milieu du couloir, la position relative de la ligne médiane par rapport au centre de l'image doit atteindre son propre point de consigne sans aucune commande. Selon certaines hypothèses, nous allons montrer que cette loi de commande est stable. Une fois la commande conçue, plusieurs questions se manifestent et doivent êtres résolues :

- Y-a-il des singularités dans la commande ?
- La commande est-elle robuste par rapport à la position de la caméra et au bruit de mesure ?
- Quel est le domaine d'attraction de commande ?
- Est-il possible de spécifier la trajectoire du robot ?

Nous allons montrer dans ce qui suit que la position du capteur virtuel a un impact sur la stabilité de la loi de commande (2.41) et doit être placée judicieusement.

La stabilité de la commande proposée (2.41) est étudiée en procédant à une linéarisation [Slotine et Li, 1991]. Ensuite, le domaine d'attraction de la commande est évalué à l'aide d'une analyse dans le plan de phase et d'un portrait de phase du système bouclé.

La commande étant basée uniquement sur la variable θ_m, la question à aborder ici est l'atteignabilité et la stabilité du point d'équilibre $(\theta_m^*, \rho_m^*) = (0, 0)$. En injectant la commande (2.41) et en développant les équations (B.24) et (B.25), on obtient

$$\dot{X} = f(X) \Leftrightarrow \begin{cases} \dot{\theta}_m & = f_1(\theta_m, \rho_m) \\ \dot{\rho}_m & = f_2(\theta_m, \rho_m) \end{cases} \tag{2.45}$$

Pour étudier la stabilité de ce système nous allons utiliser la deuxième théorème de Lyapunov

autour du point d'équilibre :

Définition 2. ***Point d'équilibre***
Soit un système non linéaire avec la variable d'état $X(t) \in \mathbb{R}^n$ et la variable de commande $U(t) \in \mathbb{R}^p$

$$\dot{X}(t) = f(X(t), U(t))$$

un point d'équilibre X_e peut être défini comme un état où le système n'évolue plus lorsqu'aucun signal externe n'est présent $U(t) = U_e(t) = 0$. Ce point assure les conditions suivantes :

- $\dot{X}_e = 0$
- *pour $X(t) = X_e$ et $U = U_e$ alors $f(X_e, 0) = 0$ pour tout $t \geq 0$*
- *X_e est une solution de $f(X_e, 0) = 0$*

Théorème 1. ***Deuxième théorème de Lyapunov***
Si le système linéarisé (approximation linéaire autour d'un point d'un système non linéaire) est asymptotiquement stable, alors le système original est aussi asymptotiquement stable autour du point de linéarisation sous des perturbations suffisamment petites.

La jacobienne du système (2.45) au point d'équilibre, $X_e = \left(\theta_m^*, \rho_m^*\right)^T = \left(0, 0\right)^T$ est :

$$\nabla f|_{X=X_e} = \left.\begin{pmatrix} \frac{\partial f_1}{\partial \theta_m} & \frac{\partial f_1}{\partial \rho_m} \\ \frac{\partial f_2}{\partial \theta_m} & \frac{\partial f_2}{\partial \rho_m} \end{pmatrix}\right|_{\begin{pmatrix} \theta_m = 0 \\ \rho_m = 0 \end{pmatrix}} = \begin{pmatrix} \frac{1}{\tau_\theta} & \frac{-2v^c}{H_S} \\ \frac{-H_S}{L\tau_\theta} & \frac{v^c}{L} \end{pmatrix} \tag{2.46}$$

On peut tout d'abord remarquer que la position latérale de la caméra n'apparaît pas dans la linéarisation. La stabilité locale est donc indépendante de la position latérale de la caméra.

Le polynôme caractéristique de la matrice jacobienne autour du point d'équilibre est :

$$p^2 - (\frac{v^c}{L} + \frac{1}{\tau_\theta})p - \frac{v^c}{L\tau_\theta} = 0 \tag{2.47}$$

En utilisant le critère de Routh-Hurwitz, pour que le système soit stable, il faut que

$$-(\frac{v^c}{L} + \frac{1}{\tau_\theta}) > 0 \quad \text{et} \quad -\frac{v^c}{L\tau_\theta} > 0 \tag{2.48}$$

Trois paramètres peuvent jouer sur le placement des pôles : la vitesse linéaire v^c, la constante du temps τ_θ et la position longitudinale L de la caméra virtuelle.

En choisissant une vitesse linéaire v^c positive et une constante de temps $\tau_\theta > 0$, une longueur de relaxation est définie et elle sera notée λ_r (Définition 3).

Définition 3. ***Longueur de relaxation***
Une longueur de relaxation λ_r est définie par la distance parcourue par le robot pour assurer la convergence souhaitée

$$\lambda_r = v\,\tau \tag{2.49}$$

avec v la vitesse linéaire appliquée au robot et τ un scalaire positif qui représente le gain (constante de temps) appliqué à la loi de commande.

A partir de (2.48) et (2.49), la condition sur le positionnement de la caméra est la suivante :

$$-\lambda_r < L < 0. \tag{2.50}$$

Ainsi, pour avoir une commande stable, la caméra virtuelle doit être placée derrière le point caractéristique du robot avec une position longitudinale liée à la vitesse linéaire et à la constante de temps choisies. Si la caméra est placée devant le point caractéristique, le système bouclé est instable. La trajectoire du robot peut être spécifiée en utilisant le placement de la caméra virtuelle à l'aide d'un placement de pôles.

Le discriminant du polynôme caractéristique (2.47) est :

$$\Delta_f = (L^2 + 6L\lambda_r + \lambda_r^2)/(L^2\tau_\theta^2) \tag{2.51}$$

En considérant $L \neq 0$ et $\tau_\theta \neq 0$, et en notant $\Delta = L^2 + 6L\lambda_r + \lambda_r^2$ le numérateur de Δ_f, les pôles du système linéarisé dépendent du signe du discriminant Δ_f donc du signe du Δ :

- avec un discriminant positif $\Delta > 0$, les pôles sont réels

$$(L + \lambda_r \pm \sqrt{\Delta})/(2L\tau_\theta) \tag{2.52}$$

- avec un discriminant négatif $\Delta < 0$, les pôles sont complexes

$$(L + \lambda_r \pm i\sqrt{|\Delta|})/(2L\tau_\theta) \tag{2.53}$$

- avec un discriminant nul $\Delta = 0$, un seul pôle réel double

$$(L + \lambda_r)/(2L\tau_\theta) \tag{2.54}$$

Pour que le système soit stable, les pôles doivent avoir des valeurs propres à partie réelles négatives. Le placement L de la caméra influence la position des pôles. Les solutions du discriminant Δ sont :

$$\begin{aligned} L_1 &= (-3 - 2\sqrt{2})\lambda_r \simeq -5.83\lambda_r \\ L_2 &= (-3 + 2\sqrt{2})\lambda_r \simeq -0.17\lambda_r \end{aligned} \tag{2.55}$$

Comme λ_r est positif, une relation d'ordre est établie par rapport à la position de la caméra virtuelle

$$L_1 < L_2 < 0. \tag{2.56}$$

Δ est négatif pour $L \in]L_1; L_2[$, nul pour $L = \{L_1; L_2\}$ et positif ailleurs.

Nous allons montrer par la suite, l'influence du choix des valeurs propres sur le système non linéaire bouclé.

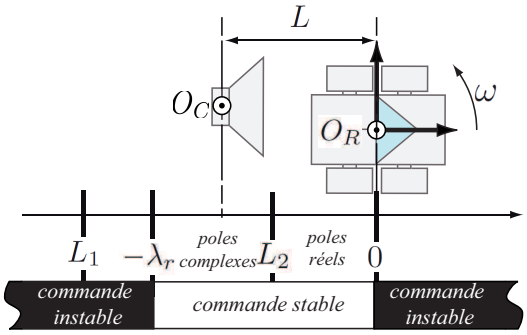


FIGURE 2.14 – Contraintes pour la position de la caméra lorsque la vitesse linéaire et la constante de temps de la commande proposée sont choisies.

Des valeurs propres complexes : Pour obtenir un système oscillant et stable avec une partie imaginaire non nulle, la partie réelle des valeurs propres doit être négative.

$$(L + \lambda_r)/(2L\tau_\theta) < 0 \tag{2.57}$$

et par la suite la contrainte sur la position de la caméra devient

$$L_1 < -\lambda_r < L < L_2 < 0 \tag{2.58}$$

La caméra virtuelle doit donc être placée derrière le point caractéristique du robot et derrière le seuil L_2 avec une limite correspondant à la longueur de relaxation. Au delà, un comportement oscillatoire est obtenu.

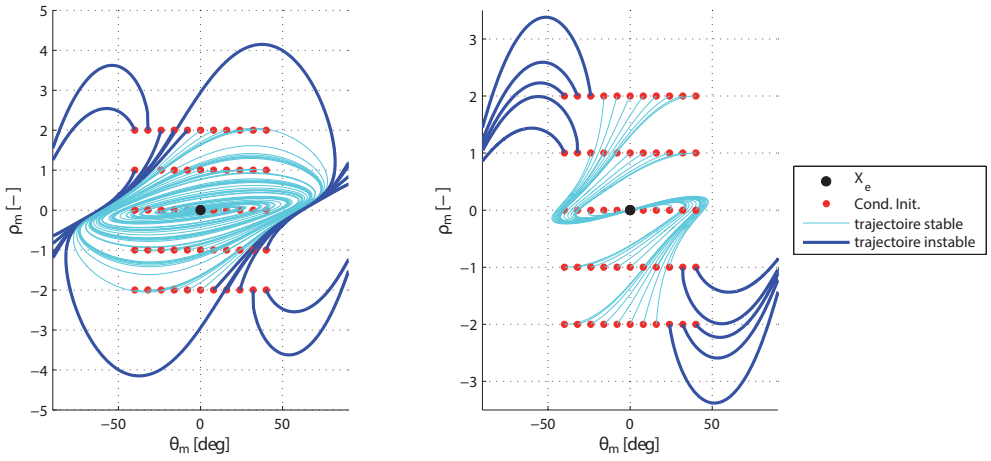
Ayant résolu le problème de la stabilité locale du point d'équilibre, le problème du domaine d'attraction de ce point doit être abordé. Pour ce faire une analyse dans le plan de phase [Slotine et Li, 1991] du système non linéaire (2.45) est réalisée. Les conditions initiales sont liées aux positions latérale et angulaire du robot dans le couloir. Les simulations sont obtenues en utilisant un solveur de type Euler d'ordre 1 avec un pas d'échantillonnage fixe de $10^{-2}\,s$ pendant $30\,s$.

Supposons les paramètres suivants $L = -1\,m$; $W = 0\,m$; $H = 0,7\,m$; ($\tau_\theta = 3\,s$, $v = 1\,m/s$), la longueur de relaxation est $\lambda_r = 3\,m$ et le système est localement stable. On obtient $\Delta = -8$ et les seuils $L_1 = -17.4853m$ et $L_2 = -0.5147m$. Les pôles sont $\{p_1, p_2 = -0.3333 \pm 0.4714i\}$.

Les trajectoires dans le plan de phase (figure 2.15a) montrent un comportement d'un cycle limite instable. Il y a quelques conditions initiales où le point d'équilibre n'est pas attractif (les trajectoires en bleu foncé). Pour ces conditions initiales, le système bouclé n'est pas stable.

Des valeurs propres réelles : En plaçant la caméra avec $L \in]-\infty; L_1[\bigcup]L_2; +\infty[$, le discriminant est positif et la variable $\sqrt{\Delta}$ affecte la valeur de la partie réelle des valeurs propres. Pour obtenir un comportement stable, la condition précédente devient $L \in]L_2; 0[$. En effet, le système sera instable si le capteur est placé devant le point caractéristique du robot ou en abscisse $-\lambda_r$. Notant aussi que cette relation $L_1 < -\lambda_r$ doit être vérifiée. Les deux valeurs propres réelles sont :

$$(L + \lambda_r \pm \sqrt{\Delta})/(2L\tau_\theta) < 0. \tag{2.59}$$



(a) (b)

FIGURE 2.15 – Le portrait de phase du système bouclé (2.45) : (a) Plan de phase pour le 1er cas - deux pôles complexes (b) Plan de phase pour le 2^{eme} cas - deux pôles réels et stables

Puisque λ_r et Δ sont positifs, le placement de la caméra doit respecter la relation suivante

$$L + \lambda_r \pm \sqrt{L^2 + 6L\lambda_r + \lambda_r} > 0 \tag{2.60}$$

Ces conditions pour obtenir un contrôle stable sont respectées pour tous $L \in]L_2; 0[$.

Le portrait de phase lorsque les pôles sont réels est illustré dans la figure 2.15b. Les trajectoires sont stables et le point d'équilibre est attractif. Cependant, il existe des valeurs de conditions initiales pour ρ_m et θ_m, au delà de ses valeurs le système devient instable.

La position latérale n'apparaît pas dans la linéarisation locale du système. Toutefois, cela influencera la trajectoire du robot. Le portrait de phase a été tracé avec une caméra placée sur le côté droit du robot.

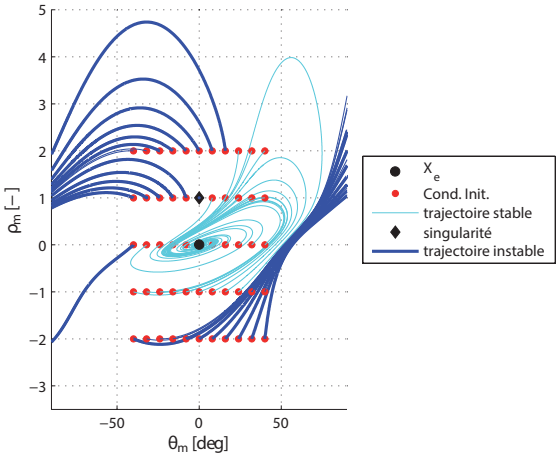


FIGURE 2.16 – Effet de la position latérale du capteur virtuel.

Les paramètres sont $L = -0.5m\,; W = -0.5m\,; H = 0.7m\,; (\tau_\theta = 3s\,; v = 0.3m/s) \Rightarrow \lambda_r = 3$. Δ est -1.6400 et les seuils de la position longitudinale sont $L_1 = -5.2456m$ et $L_2 = -0.1544m$.

Les valeurs propres sont : $\{p_1, p_2 = -0.1333 \pm 4269i\}$. Dans ce cas, une singularité apparaît. En effet, avec une condition initiale $\rho_m = 1$ et $\theta_m = 0$, le dénominateur de la loi de commande est nul. La figure 2.16 montre que la position latérale de la caméra peut entraîner une déstabilisation du système pour quelques conditions initiales. Pour évaluer l'influence de la commande sur la trajectoire du robot, certains résultats de simulation sont présentés ici.

La trajectoire du robot est évaluée pour deux cas :

(1) : lorsque les pôles sont complexes et cas (2) : lorsque les pôles sont réels.

Un bruit a été injecté sur la mesure de θ_m pour évaluer la robustesse de la commande et préparer la mise en œuvre expérimentale. Un bruit gaussien a été choisi avec un écart type de 2 .

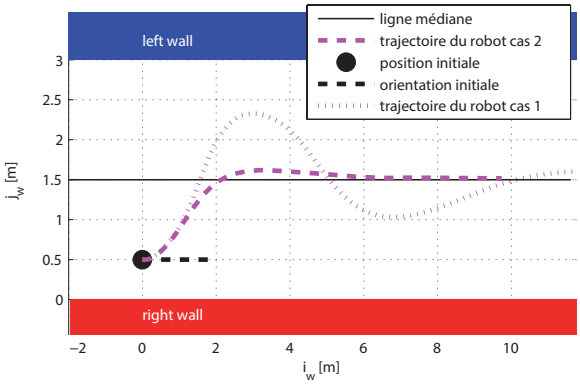
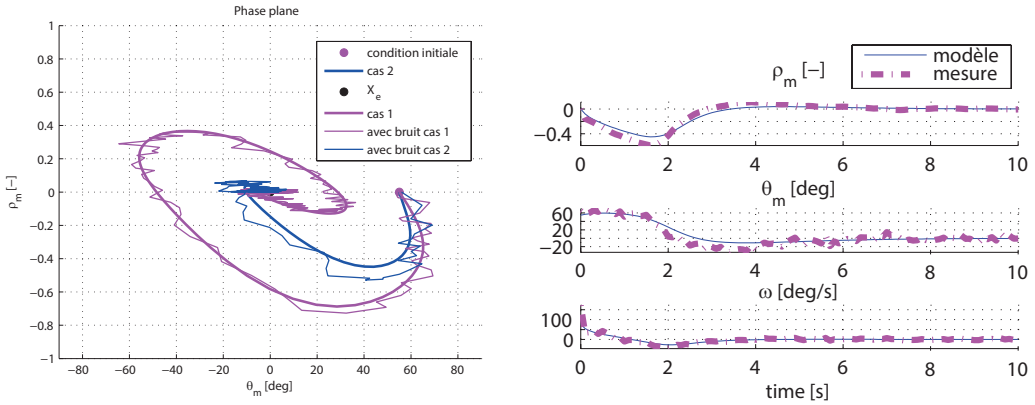


FIGURE 2.17 – Vue de dessus des trajectoires du robot pour les cas 1 (pôles complexes) et le cas 2 (pôles réels).

Sur la figure 2.17, lorsque la commande est construite avec deux pôles complexes (cas (1)), la trajectoire présente plus d'oscillations par rapport au cas (2) avec deux pôles réels, le robot oscille autour de la ligne médiane avant de l'atteindre. Cet effet d'oscillation se voit aussi dans le plan de phase de la signature visuelle dans la figure 2.18b. Les deux trajectoires dans le plan de phase sont stables puisque elles convergent vers le point d'équilibre $(0, 0)$ (Foyer stable).



(a) (b)

FIGURE 2.18 – (a) Plan de phase pour le cas 1 (pôles complexes) et le cas 2 (pôles réels). Avec et sans le bruit additif. (b) Comparaison entre le modèle (2.45) et le système réel. Les deux variables d'état/ mesure et la commande appliquée

La signature visuelle (modèle et mesure) et la commande sont visualisées pour le cas (2) (figure 2.18b). Le modèle représente l'intégration de (2.45) et la mesure représente l'extraction de la signature dans l'image, avec un bruit additif. Malgré la présence de ce dernier, la commande et l'évolution de la signature sont stables et suivent le comportement du système. Les variables θ_m et ρ_m convergent vers les valeurs désirées.

Dans cette section la loi de commande a été définie et une analyse des propriétés de stabilité a été réalisée. Dans la section qui suit, nous allons définir la stratégie de commande que nous avons utilisé pour assurer la robustesse et la sûreté d'un déplacement autonome dans un couloir.

2.4 Stratégie de commande basée observateur

L'observateur reconstruit un état en utilisant un modèle du système et des mesures des entrées et des sorties. Dans le cas étudié, le problème d'observabilité est résolu car l'état complet est mesuré. Plusieurs techniques de conception de l'observateur peuvent être utilisées. Ici, nous concevons un observateur de convergence en temps fini avec l'avantage principal qu'aucune linéarisation n'est nécessaire. Cet observateur fournit simultanément un filtre passe-bas capable de lisser les perturbations induites par l'estimation de la position du point de fuite. Par rapport à un filtre simple, la conception de l'observateur garantit la convergence à la valeur réelle de la position du point de fuite. Lorsque les mesures directes de la position du point de fuite et de l'orientation de la ligne médiane ne sont pas disponibles, il est possible d'utiliser l'état fourni par l'observateur. La commande est alors basée sur un modèle en boucle ouverte de la dynamique des fonctionnalités précédemment initialisées par l'observation de l'état.

L'utilité de de l'observateur est la prédiction de l'état lors de mesures erronées et lors de la perte de la mesure surtout pour la détection du point de fuite. Si une des lignes du bas des murs n'apparaît pas dans l'image (limitation du champs de vision ou le robot est trop orienté vers l'un des murs) le point de fuite ne peut pas être calculé. Il est également possible d'ignorer certaines mesures pour diminuer le temps de calcul afin de réaliser d'autres opérations.

2.4.1 Observateur de la signature

On considère un vecteur d'état constitué de la signature visuelle :

$$\mathbf{X} = \begin{pmatrix} x_F & y_F & \theta_m \end{pmatrix}^T = \begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix}^T$$

L'état est mesuré grâce aux principes explicités à la section 2.3.2.2 où x_F et y_F sont issus de l'équation (2.30) et θ_m de (2.28) après une extraction des droites supports des intersections murs/sol.

La mesure est :

$$\mathbf{Y} = \begin{pmatrix} x_F & y_F & \theta_m \end{pmatrix}^T = \begin{pmatrix} y_1 & y_2 & y_3 \end{pmatrix}^T$$

La commande est :

$$\mathbf{U} = \begin{pmatrix} v & \omega \end{pmatrix}^T = \begin{pmatrix} u_1 & u_2 \end{pmatrix}^T$$

La dynamique du système et l'équation d'observation s'écrivent

$$\Sigma \begin{cases} \dot{\mathbf{X}} &= f(\mathbf{X}, \mathbf{U}) \\ \mathbf{Y} &= \mathbf{X} \end{cases} \tag{2.61}$$

où

$$f(\mathbf{X}, \mathbf{U}) = \begin{pmatrix} f_1(\mathbf{X}, \mathbf{U}) \\ f_2(\mathbf{X}, \mathbf{U}) \\ f_3(\mathbf{X}, \mathbf{U}) \end{pmatrix}$$

explicitées par l'évolution du point de fuite dans l'image, détaillée dans l'annexe (A.17), (A.18) et l'évolution de la droite médiane dans l'image (B.24) (dans laquelle ρ_m est remplacé par l'expression (2.31))

$$f_1(\mathbf{X}, \mathbf{U}) = -u_2 \left(1 + x_1^2\right) \tag{2.62}$$

$$f_2(\mathbf{X}, \mathbf{U}) = u_2 x_1 x_2 \tag{2.63}$$

$$\begin{aligned} f_3(\mathbf{X}, \mathbf{U}) &= \cos^2(x_3)\frac{1}{H}\left(-Lu_2 - u_1 x_1 + x_1 u_2 W\right) + \sin^2(x_3) x_2 u_2 \\ &\quad + \cos(x_3) sin(x_3) \left(-\frac{u_1 x_2}{H} + u_2 \left(x_2 \frac{W}{H} + x_1\right)\right) \end{aligned} \tag{2.64}$$

Nous proposons l'observateur

$$\Sigma \begin{cases} \dot{\hat{\mathbf{X}}} &= f(\hat{\mathbf{X}}, \mathbf{U}) + l(\tilde{\mathbf{Y}}, \mathbf{U}, \hat{\mathbf{X}}) \\ \hat{\mathbf{Y}} &= \hat{\mathbf{X}} \end{cases} \tag{2.65}$$

avec l'erreur d'observation

$$\tilde{\mathbf{Y}} = \mathbf{Y} - \hat{\mathbf{Y}} = \begin{pmatrix} \tilde{y}_1 = y_1 - \hat{x}_1 \\ \tilde{y}_2 = y_2 - \hat{x}_2 \\ \tilde{y}_3 = y_3 - \hat{x}_3 \end{pmatrix}$$

et le gain de l'observateur

$$l(\tilde{\mathbf{Y}}, \mathbf{U}, \hat{\mathbf{X}}) = \begin{pmatrix} l_1(\tilde{\mathbf{Y}}, \mathbf{U}, \hat{\mathbf{X}}) & l_2(\tilde{\mathbf{Y}}, \mathbf{U}, \hat{\mathbf{X}}) & l_3(\tilde{\mathbf{Y}}, \mathbf{U}, \hat{\mathbf{X}}) \end{pmatrix}^T$$

En notant K_1, K_2 et K_3 des scalaires positifs, nous proposons les fonctions de corrections suivantes :

$$l_1\left(\tilde{\mathbf{Y}}, \mathbf{U}, \hat{\mathbf{X}}\right) = u_2 \tilde{y}_1^2 + 2u_2 \tilde{y}_1 \hat{x}_1 + K_1 sign(\tilde{y}_1) \tag{2.66}$$

$$l_2(\tilde{\mathbf{Y}}, \mathbf{U}, \hat{\mathbf{X}}) = u_2 \left((\tilde{y}_1 + \hat{x}_1)(\tilde{y}_2 + \hat{x}_2) - \hat{x}_1 \hat{x}_2\right) + K_2 sign(\tilde{y}_2) \tag{2.67}$$

$$l_3(\tilde{\mathbf{Y}}, \mathbf{U}, \hat{\mathbf{X}}) = \tilde{y}_1 \left(u_2(\frac{W}{H} + 1) - \frac{u_1}{H}\right) + \tilde{y}_2 \left(u_2(1 + \frac{W}{H}) - \frac{u_1}{H}\right) + K_3 sign(\tilde{y}_3) \tag{2.68}$$

Le choix de ces gains est expliqué dans l'annexe (D).

2.4.2 Organisation de la stratégie de commande

Habituellement, les lois de commande sont basées sur une comparaison entre une consigne et une mesure. Les performances de la commande dépendent en effet de la qualité des mesures. Nous allons étudier ici l'apport de l'observateur. La stratégie de commande que nous avons utilisée est illustrée sur la figure 2.19. Lorsque les mesures sont disponibles, nous proposons de calculer la commande à partir de l'état observé (D.5), ceci utilisera les propriétés de filtrage de l'observateur. Nous proposons également de considérer les situations où les mesures ne sont pas disponibles. En effet, dans certaines situations, l'extraction d'une des lignes directrices du couloir est impossible, ce qui rend la localisation du point du fuite impossible.

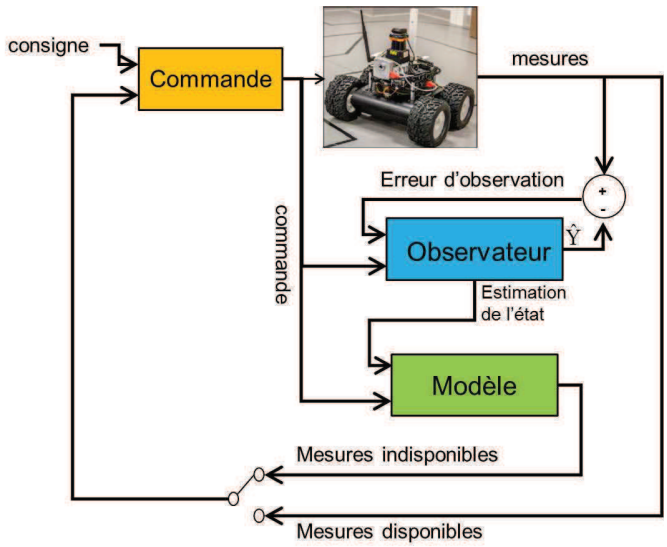


FIGURE 2.19 – Stratégie de la commande. Lorsque les mesures sont disponibles, la commande est calculée à partir de l'état observé. Sinon, la commande est calculée à partir du modèle en utilisant comme conditions initiales la dernière observation estimée par l'observateur.

Dans ce cas, nous travaillons avec l'estimation issue du modèle suivant. Une estimation de la signature visuelle est ensuite obtenue pour calculer la loi de commande par

$$\Sigma \begin{cases} \dot{\hat{\mathbf{X}}} & = \mathbf{f}(\hat{\mathbf{X}}, \omega) \\ \hat{\mathbf{Y}} & = h(\hat{\mathbf{X}}) \\ \hat{\mathbf{X}}(T_0) & = \hat{\mathbf{X}}_0 \end{cases} \tag{2.69}$$

Comme l'observateur garantit l'estimation de l'état réel, donc la condition initiale $\hat{\mathbf{X}}_0$ du modèle en boucle ouverte est bien positionnée. Il faudra s'assurer qu'une première phase de convergence de l'observateur a eu lieu avant de commander le système sur la base de l'estimation.

2.4.3 Simulations et résultats expérimentaux

2.4.3.1 Validation du modèle et d'observateur d'état

Nous proposons tout d'abord de valider la stratégie à l'aide de l'environnement de simulation Morse-Blender-ROS (Annexe F). Pour valider l'observateur proposé et la stratégie de commande deux

scénarios ont été testés.

2.4.3.1.1 Scénario I : Ce premier scénario sert à valider le bon fonctionnement de l'observateur. Dans un même essai, nous introduisons différentes phases pour mettre en valeur la robustesse de l'observateur proposé :

- Phase 1 : nous appliquons la commande en boucle fermée en utilisant les variables estimées entre 0 et 25 s. L'observateur rejoint la mesure et la loi de commande annule l'erreur.
- Phase 2 : nous injectons une perturbation de 11 deg/s au niveau de la commande pour modéliser par exemple le glissement des roues de l'instant 25 s jusqu'à l'instant 51 s.
- Phase 3 : nous faisons disparaître la perturbation au niveau de la commande et nous ajoutons un bruit gaussien à la mesure θ_m de l'instant 60 s jusqu'à la fin de l'essai.

Nous remarquons dans la figure 2.21 que l'estimation rejoint toujours la position réelle du système et la suit. Malgré la perturbation de 11 deg ajoutée à l'instant 25 s au niveau de l'entrée du système, la loi de commande compense cette perturbation et le robot s'éloigne d'une erreur de 10 cm par rapport à la médiane. L'observateur filtre le bruit ajouté au niveau de la mesure θ_m.

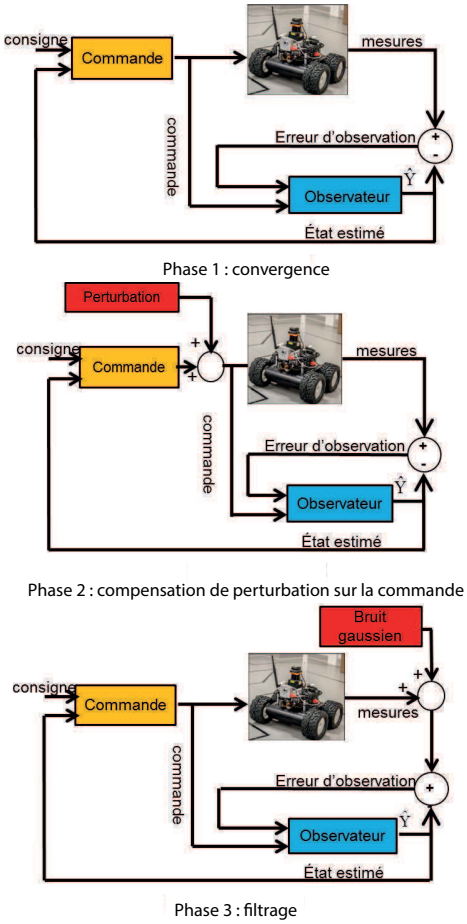


FIGURE 2.20 – Explication des étapes utilisés pour le scénario I

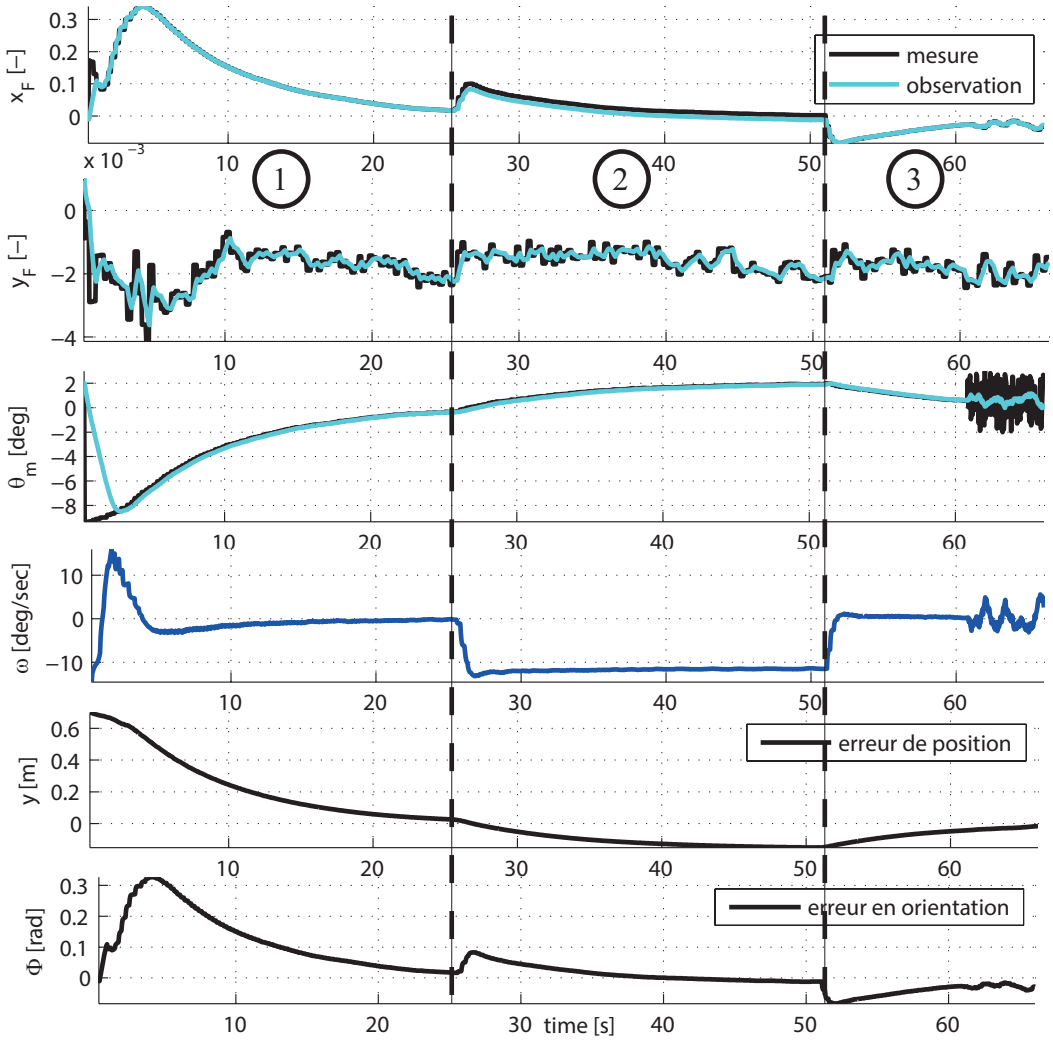


FIGURE 2.21 – Résultats pour le scénario I, les trois premières courbes correspondent aux valeurs estimées et les valeurs réelles de x_F, y_F et θ_m. La quatrième courbe correspond à la commande appliquée au robot durant ce scénario et les deux dernières courbes représentent respectivement l'erreur de la position y en [m], et Φ en [rad] l'erreur en orientation du robot par rapport à la médiane.

2.4.3.1.2 Scénario II : Ce scénario sert à valider le bon fonctionnement de la stratégie de commande "observateur + modèle".

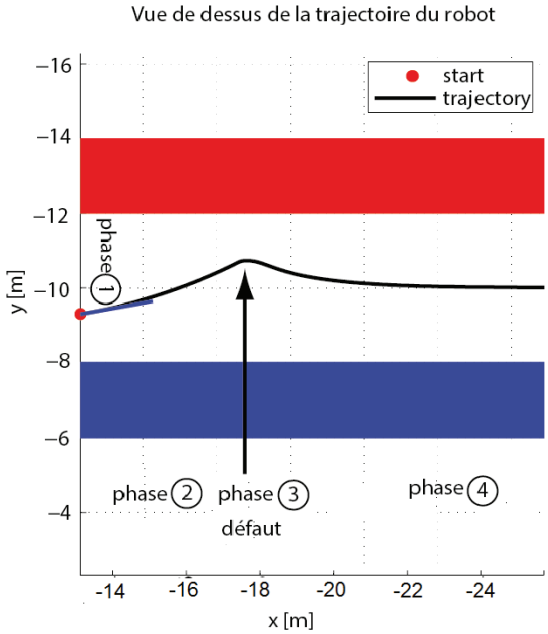


FIGURE 2.22 – La trajectoire du robot dans l'environnement de simulation. Phase 1 : convergence de l'observateur. Phase 2 : vitesses linéaires et angulaires constantes pour créer indisponibilité de mesure. Phase 3 : commande en boucle fermée en fonction de l'état estimé. Phase 4 : commande en boucle fermée basée sur l'état observé.

Nous générons les tests suivants au cours du scénario :

- Phase 1 : nous devons attendre la convergence de l'observateur. L'objectif est d'obtenir une bonne estimation de l'état. Les modèles sont arbitrairement initialisés. Nous imposons un contrôle oscillant en boucle ouverte sur ω. Ce contrôle est facultatif, il permet de mieux visualiser les étapes de l'expérience.
- Phase 2 : le but de cette phase est de commander le robot pour l'amener à une situation de défaillance de mesure. Une vitesse linéaire constante $v \neq 0$ et une vitesse angulaire constante $\omega \neq 0$ sont appliquées. Le robot se déplace vers le mur. Cela provoque la disparition de la visibilité du mur droit dans l'image et l'impossibilité de mesurer la position du point de fuite. Cette situation représente un défaut de mesure. Pendant cette phase, l'observateur est toujours activé et l'observation est utilisée ici pour initialiser l'état initial du modèle (2.69) qui fonctionnera en boucle ouverte.
- Phase 3 : lorsque le défaut est détecté, l'observateur est désactivé (la mesure n'est pas disponible). La commande est calculée à partir de l'estimation fournie par le modèle (2.69) dont les conditions initiales ont été précédemment fournies par l'observateur (phases 1 et 2).
- Phase 4 : lorsque la défaillance de mesure disparaît, l'observateur est réactivé et la commande est réalisée en boucle fermée en se basant sur l'observation (D.5).

Nous choisissons d'initialiser le modèle d'observateur avec une position arbitraire $\hat{x}_F = -0.2$, $\hat{y}_F = -0.1$, $\hat{\theta}_m = 0$. Les gains de l'observateur sont choisis comme $K_1 = 0.5$, $K_2 = 0.1$, $K_3 = 0.1$. Pour

la commande, le temps de réponse est défini comme $\tau_\theta = 3\,s$. La trajectoire de cette simulation est représentée dans la figure 2.22. De l'instant $10\,s$ à $30\,s$ le robot se déplace vers le mur. Lorsqu'il ne peut pas détecter les deux lignes des bords du couloir, il y a une défaillance de mesure, l'algorithme ne peut pas calculer les paramètres de la ligne médiane (les coordonnées du point de fuite (x_F, y_F) et l'orientation θ_m). Ainsi, la commande en boucle fermée est déduite des estimations construites à partir du modèle (2.69). Lorsque la signature est à nouveau récupérée, la commande est calculée avec l'état observé. Nous pouvons constater le bon fonctionnement de l'observateur et de la loi de commande proposés. Le robot suit la ligne médiane du couloir même lorsque les mesures sont indisponibles. Toutefois, le fait de travailler sur le modèle en boucle ouverte n'est pas robuste à des changements d'environnement. L'évolution de la commande est représentée dans la figure 2.23. De l'instant $10\,s$ à l'apparition du défaut de mesure (phase 2), nous appliquons une vitesse linéaire et angulaire constante. Lorsque le défaut apparaît à l'instant $29\,s$, la commande, basée sur le modèle, est activée. L'observateur rejoint la mesure à l'instant $32\,s$ et la commande devient presque nulle dès l'instant $50\,s$.

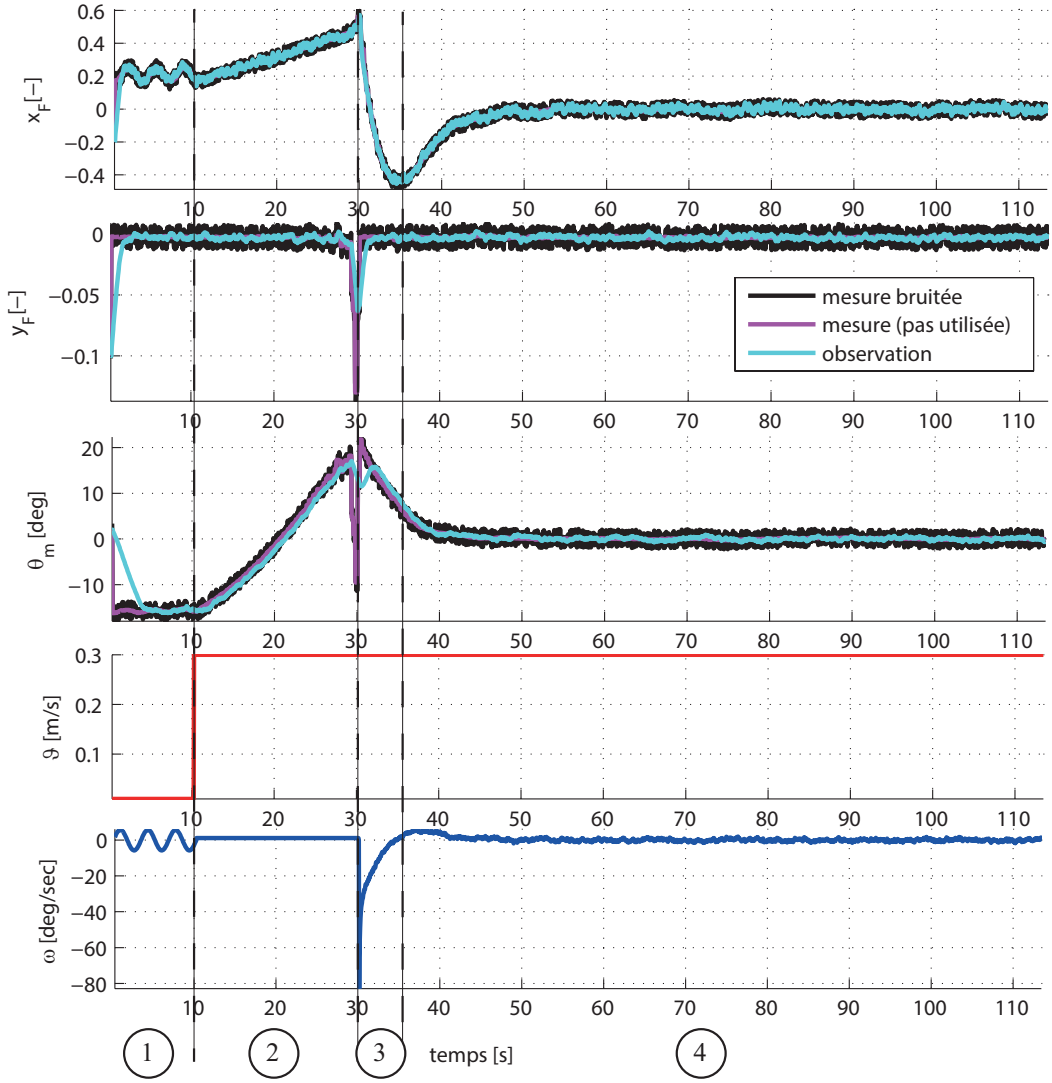


FIGURE 2.23 – Les variables estimées (respectivement $\hat{x}_F$, $\hat{y}_F$ et $\hat{\theta}_m$) durant le scénario II avec des mesures bruitées utilisées pour l'observateur et les valeurs réelles pour la comparaison.

La figure 2.23 présente aussi l'évolution de l'état observé $(\hat{x}_F, \hat{y}_F, \theta_m)$ le long de la simulation avec la mesure de référence (non utilisée pour la commande) et une mesure bruitée utilisée comme mesure pour l'observateur. Les écarts-types du bruit sont $\sigma_{\theta_m} = 0.6\,deg$, $\sigma_{x_F} = 0,016m$ et $\sigma_{y_F} = 0,0033m$. Les trois premières secondes consistent en la convergence de l'observateur de sa condition initiale à la valeur d'état réel. La croissance de la variable x_F indique le déplacement du robot vers le côté gauche du couloir. Lorsque le défaut de mesure apparaît (disparition du point de fuite dans l'image), un dépassement apparaît sur la mesure, qui a été filtré par l'observateur. Puis, la commande en boucle ouverte est basée sur le modèle (2.69) et l'observateur est désactivé. Sa condition initiale est mise à jour à partir du modèle. Lorsque la mesure est récupérée, la convergence de l'observateur est à nouveau visible et le commande est basée sur la valeur observée.

2.4.3.2 Résultats expérimentaux

Pour l'évaluation expérimentale, un robot Wifibot est utilisé (figure 2.24). Dans ce cas, seul le capteur laser Hokuyo est utilisé. La caméra virtuelle est placée à $(-1\,m, 0\,m, 0.7\,m)_{\mathcal{F}_R}$. Les résultats

FIGURE 2.24 – Le robot et l'environnement utilisés pour l'expérimentation.

sont présentés à la figure 2.25. Le haut de la figure présente la trajectoire du robot le long du couloir. Le robot est initialement positionné à environ $30\,cm$ du milieu du couloir et n'est pas aligné par rapport aux murs. Pour ce test, les mesures sont considérées comme indisponibles pendant les phases 2 et 4. Au cours de ces phases, le robot a été supposé volontairement aveugle et seule l'estimation est utilisée pour la commande. Cependant, nous utilisons les signatures visuelles extraites comme référence pour évaluer l'écart entre les sorties réelles et prédites. Les discontinuités du contrôle au moment $25\,s$ et $38\,s$ sont la conséquence de la réactivation de l'observateur car le modèle en boucle ouverte dérive (surestimation de θ_m). Après la période de convergence, la valeur d'observation θ_m est égale à la mesure. Nous pouvons également voir que le robot a presque atteint le milieu du couloir et θ_m est presque nul.

2.5 Conclusion

Nous avons présenté dans ce chapitre une loi de commande pour le déplacement d'un robot mobile dans un couloir. Nous avons utilisé l'évolution de la signature visuelle et construit un asservissement visuel en utilisant les propriétés du point de fuite et en manipulant une caméra virtuelle placée et orientée comme on le souhaite.

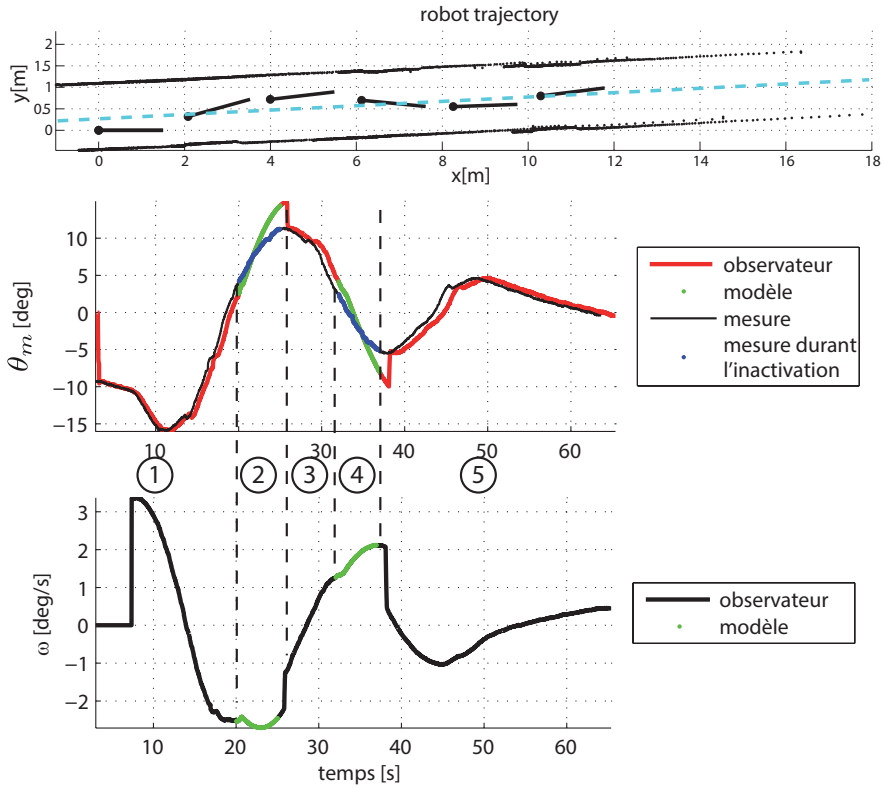


FIGURE 2.25 – Haut : Trajectoire du robot (une position chaque 7s). Milieu : θ_m observé pendant les phases 1, 3 et 5 et estimé pendant 2 et 4. Les mesures sont également indiquées comme référence. Bas : commande ω le long du test.

La stabilité dans différentes conditions de positionnement du capteur a été évaluée. Avec la commande et les paramètres proposés, le robot se déplace sur la ligne médiane du couloir. Nous avons montré que la stabilité de la trajectoire dépend de la position du capteur et est liée à la longueur de relaxation souhaitée de la commande. La figue 2.14 résume la stabilité de la loi de commande par rapport à l'emplacement du capteur. Une attention particulière doit être accordée aux conditions initiales avant l'activation du contrôle en boucle fermée. Le problème des singularités de la commande a également été discuté. La stabilité et l'attractivité du point d'équilibre ont été présentées localement en utilisant la linéarisation du système et en utilisant une analyse du plan de phase pour évaluer le domaine d'attraction pour une application sur le système non linéaire. La conclusion est que le positionnement du capteur virtuel doit satisfaire certaines contraintes. Pour avoir un comportement stable, la position du capteur et la longueur de relaxation doivent être choisis ensemble. Lorsque la vitesse linéaire du robot augmente pour diminuer le temps de convergence, le capteur peut être déplacé vers l'arrière. Mettre un capteur en avant du robot entraînerait une instabilité du système en boucle fermée. Nous avons aussi mis en œuvre un observateur en temps fini pour estimer la signature dans un environnement couloir. Pour éviter les mesures erronées, le filtrage peut être appliqué, mais lorsque la mesure n'est pas disponible, la commande ne peut pas être calculée. La technique de l'observateur garantit à la fois l'opération de filtrage passe-bas et la convergence de l'observation vers l'état réel. Avec l'observateur proposé, aucune linéarisation ni inversion de matrice ne doit être faite, toutes les non-linéarités sont implicitement manipulées dans le modèle. L'utilisation de l'estimation issue d'un modèle en boucle

ouverte est une autre possibilité de contrôler le robot lors de l'indisponibilité de la mesure. Cette solution nécessite un modèle correctement paramétré et une condition initiale parfaitement connue. L'utilisation de l'observateur combine les avantages du filtre passe-bas et la garantie de connaître l'état initial du modèle. La solution proposée peut également être utilisée pour économiser du temps de calcul en remplaçant tout le processus d'extraction de caractéristiques par l'intégration du modèle le long du temps avec une condition initiale déduite de l'observateur. Cela pourrait être particulièrement utile dans des environnements faiblement perturbés. La commutation entre observateur et modèle doit être gérée par un processus spécifique dans la stratégie de contrôle du robot. Cette méthode est adaptée aux environnements intérieurs délimités par deux murs comme des couloirs non encombrés. Le prochain chapitre traite le cas de la navigation autonome dans tout type d'environnement : en calculant un diagramme de Voronoï local de l'espace libre extrait à partir de la perception du robot. Nous proposerons ensuite une nouvelle méthode de déplacement sur le diagramme de Voronoï en utilisant l'asservissement visuel.

Asservissement visuel sur le diagramme de Voronoï

Résumé du chapitre

Une approche de suivi du squelette avec un asservissement visuel calculé dans un cadre centralisé virtuel en temps réel est proposée dans ce chapitre. La commande est basée sur une approximation du diagramme de Voronoï local. Ce diagramme est extrait à l'aide de l'algorithme de squelettisation dans l'image qui prend en compte la dimension du robot *Delta Medial Axis*. Une signature visuelle a été définie et adaptée à ce cadre virtuel afin d'assurer le suivi du diagramme de Voronoï local. L'analyse de la loi de commande proposée est réalisée en utilisant les portraits de phase du système et en donnant des préconisations quant à la position du capteur virtuel. La démarche est validée en utilisant une structure de simulation permettant de tester directement le code informatique qui sera implanté dans les expérimentations. Finalement, la loi de commande proposée a été testée sur un fauteuil roulant robotisé.

Sommaire

3.1 Introduction

Ce chapitre étend la contribution du chapitre précédent à des environnements plus généraux et dynamiques en construisant une approche d'asservissement visuel sur une signature visuelle de la structure explicite du diagramme de Voronoï généralisé (DVG) local. Les techniques habituelles de navigation sur le DVG tiennent compte d'une approche de contrôle réactif. Dans [Victorino et al., 2004a] et [Victorino et al., 2004a] une approche de commande référencée laser a été étudiée dans des environnements inconnus en utilisant implicitement le DVG.

Dans notre cas, nous conservons le cadre du capteur virtuel générique qui a été introduit dans le chapitre 2, dans lequel toutes les observations des capteurs disponibles sont projetées. Ce cadre unifié de référence sera le support de l'extraction du diagramme de Voronoï Généralisé (DVG) explicite. Ensuite, une approximation du DVG est localement construite en utilisant une extraction en temps réel du squelette de l'espace libre, puis sera utilisée comme une signature visuelle. Enfin, une commande d'asservissement visuel est conçue pour naviguer de manière autonome le long de la structure du DVG.

Cette approche comporte deux intérêts majeurs. Tout d'abord, contrairement à la plupart des approches existantes d'asservissement visuel, il ne nécessite aucune connaissance préalable de l'environnement [Pasteau et al., 2013] et aucune phase d'apprentissage [Caron et al., 2013]. Deuxièmement, étant défini dans un cadre sensoriel virtuel, il peut être utilisé avec n'importe quel type de capteurs, à condition que l'espace navigable soit défini. Un autre avantage est que l'utilisateur peut définir un positionnement approprié du capteur virtuel afin d'obtenir un comportement souhaité spécifique, d'exprimer des propriétés intéressantes comme l'observabilité ou d'adapter la taille de l'espace de travail à un horizon spécifique. L'approche est d'abord testée à l'aide d'un environnement de simulation. Ensuite, une validation expérimentale est réalisée sur deux plateformes expérimentales (un robot Pioneer et un fauteuil roulant différentiel) permettant au système de naviguer de manière sûre et autonome sans aucune intervention humaine. Les scénarios d'environnements statiques et dynamiques sont étudiés. La méthode est immédiatement généralisable à tout robot différentiel. Ce chapitre présentera dans un premier temps le cadre sensoriel unifié. Puis l'asservissement visuel basé sur le squelette sera défini en introduisant la signature visuelle et la loi de commande. Les propriétés de la loi de commande seront discutées et enfin, la validation expérimentale est réalisée en utilisant à la fois des scénarios de simulation et du monde réel, dans des environnements statiques et dynamiques.

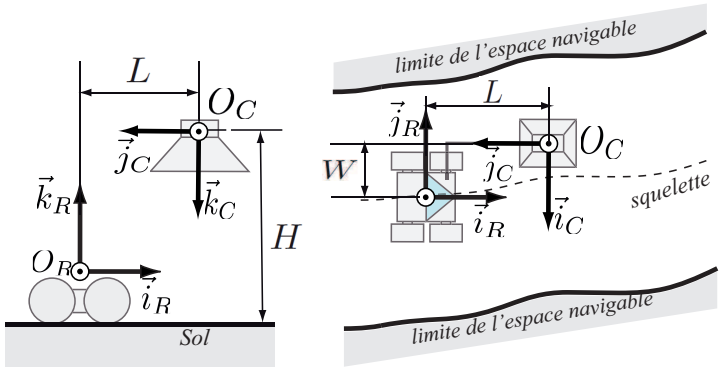
3.2 Modélisation

Dans le chapitre précédent, le capteur virtuel a été introduit et modélisé (2.3.1). Dans ce chapitre, nous allons calculer un squelette local de l'espace libre. Pour avoir une vue la plus complète de l'espace navigable, nous choisissons d'orienter le capteur (figure 2.7) avec un angle d'observation $\alpha = -\frac{\pi}{2}$.

Cela permet d'obtenir une vue aérienne (Bird Eye View (BEV)) (figure 3.1) représentant une vue globale de l'environnement. Ce cadre virtuel sera utilisé pour la construction explicite en temps réel du DVG local et le support de la commande référencée capteurs. Nous rappelons que $\mathcal{F}_R$ et $\mathcal{F}_C$ désignent respectivement le repère du robot et le repère du capteur virtuel. La matrice de transformation entre

les deux repères est définie dans (2.16) en remplaçant α nous avons

$$\mathcal{R}_{\mathcal{F}_R \leftarrow \mathcal{F}_C} = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix}, \ \mathbf{t}_{\mathcal{F}_R \leftarrow \mathcal{F}_C} = \begin{pmatrix} W \\ L \\ H \end{pmatrix}_{\mathcal{F}_\mathbf{R}} \tag{3.1}$$



(a) Vue de coté (b) Vue de dessus

FIGURE 3.1 – Représentation de la transformation rigide entre le repère du robot $\mathcal{F}_\mathbf{R}$ et le repère de la caméra virtuelle $\mathcal{F}_\mathbf{C}$.

Pour garantir une bonne détection du DVG local, le champ de vision de la caméra virtuelle (en particulier H) doit inclure suffisamment d'observations de l'environnement. Les paramètres intrinsèques $(u_0, v_0, \alpha_u, \alpha_v)$ de la caméra virtuelle sont arbitrairement choisis. Le modèle cinématique de la caméra virtuelle a été introduit à la section (2.3.1.1).

Soit $B \subseteq \mathbb{Z}^2$ la représentation de l'image BEV virtuelle construite à partir de capteurs disponibles, le centre de l'image est $c = (u_0, v_0)$, il correspond à la projection de O_c dans B. Nous choisissons $\alpha_u = \alpha_v = \alpha$. Pour chaque point $\mathbf{p} = (u, v)$ dans B, le point du sol correspondant de coordonnées 3D est $P = (X, Y, Z)$:

$$\begin{cases} X = (u - u_0)/\alpha \\ Y = (v - v_0)/\alpha \\ Z = H \end{cases} \tag{3.2}$$

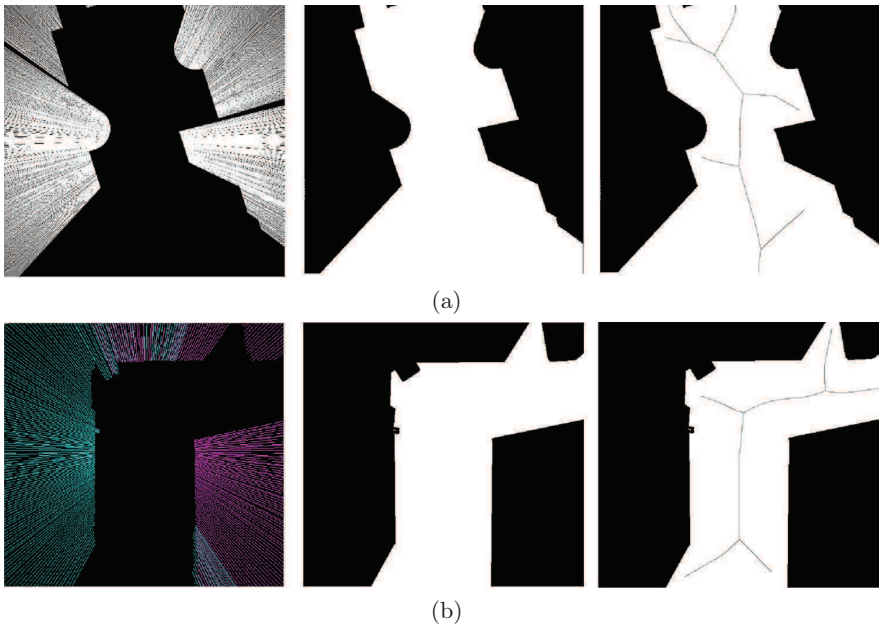
3.3 Génération de la représentation virtuelle et extraction du squelette

Toutes les mesures/capteurs sont projetés dans le capteur virtuel BEV. La transformation entre le capteur et l'image virtuelle diffère d'un capteur à un autre. Dans la figure 3.2 l'image virtuelle est construite à partir d'un télémètre laser et dans la figure 3.3 l'image virtuelle est construite à partir d'une caméra omnidirectionnelle.

Après avoir construit l'image virtuelle, une étape de binarisation est effectuée pour distinguer l'espace libre de l'espace occupé. Une image binaire est obtenue : l'espace blanc représente l'espace libre et l'espace noir représente l'espace occupé. Un algorithme de traitement d'images de squelettisation [Marie et al., 2016](détaillé dans le premier chapitre dans la section (1.4.3.3) et l'Annexe (E)) est

appliqué sur l'image binaire. Il se trouve que le squelette de la zone blanche représente aussi le diagramme de Voronoï de l'espace libre.

Le masque binaire peut être aussi construit par une fusion de données de capteurs pour obtenir une vue globale de l'environnement. Cette fusion peut être résolu par les techniques de réseau bayésien [Coué et al., 2006] ou des techniques de Dempster-Shafer [Yu et al., 2014].



(a)

(b)

FIGURE 3.2 – Observations construites à partir de télémètres : (a) un seul laser avec 270° de champs de vue et (b) la fusion de données de deux lasers. De gauche à droite : projection des données capteurs dans la BEV (référentiel virtuel), construction d'un masque binaire délimitant l'espace navigable (blanc), des obstacles (noir), et squelettisation de l'espace libre

Utilisation d'un télémètre

Nous allons effectuer les mêmes transformations que celles proposées à (2.3.2.1) pour un point issu d'une observation réelle vers un point dans l'image virtuelle. Dans cette partie, nous représenterons chaque point issu de l'observation réelle par une droite radiale dans l'image (pour distinguer l'espace navigable et non navigable) (figure 3.2). Cette dernière représente une projection d'une droite verticale dans le monde (chaque droite est constituée d'un point détecté par le lidar ramené au sol et un point d'altitude arbitraire) (figure 3.2).

Utilisation d'une caméra

Dans les environnements extérieurs le capteur le mieux adapté est une caméra. La caméra perspective contient un champs de vue limité. Par contre l'observation de la caméra omnidirectionnelle peut englober une perception de l'environnement à 360^o. La figure 3.3 illustre la construction de la représentation virtuelle à partir d'une image omnidirectionnelle.

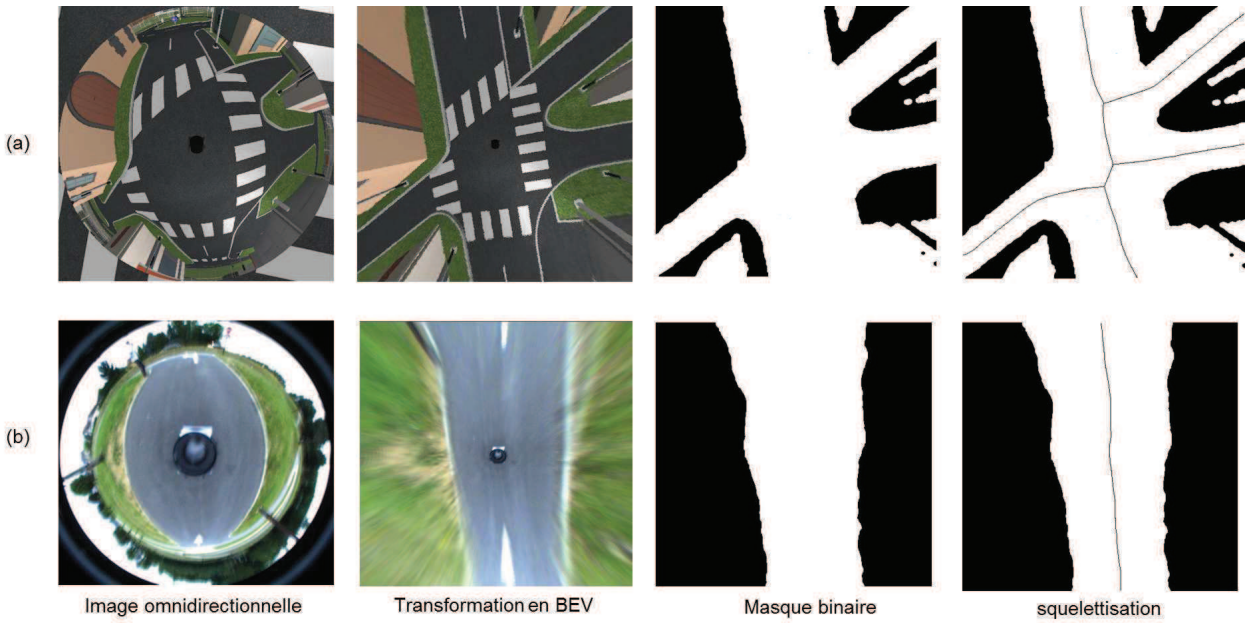


FIGURE 3.3 – Construction du capteur virtuel et du masque binaire à partir d'une caméra omnidirectionnelle. (a) environnement du simulateur 4D-virtualiz (b) environnement extérieur (une route). Une étape de segmentation de couleur est réalisée pour extraire le masque binaire.

La transformation réalisée pour obtenir l'image virtuelle à partir de la caméra omnidirectionnelle est :

$$B(u,\, v) = I(u',\, v') \text{ telle que } \begin{cases} \begin{bmatrix} u' & v' & 1 \end{bmatrix}^T = \mathbf{K}_{omni}. \begin{bmatrix} x & y & 1 \end{bmatrix}^T \\ x = (u - u_0)/(\alpha H + \alpha\xi\rho_{omni}) \\ y = (v - v_0)/(\alpha H + \alpha\xi\rho_{omni}) \end{cases} \tag{3.3}$$

avec $\rho_{omni} = \sqrt{(u-u_0)^2/\alpha^2 + (v-v_0)^2/\alpha^2 + H^2}$, K_{omni} est la matrice de paramètres intrinsèques de la caméra omnidirectionnelle et ξ un paramètre qui dépend de la géométrie du miroir de la caméra.

3.4 Asservissement visuel sur le squelette de l'espace libre

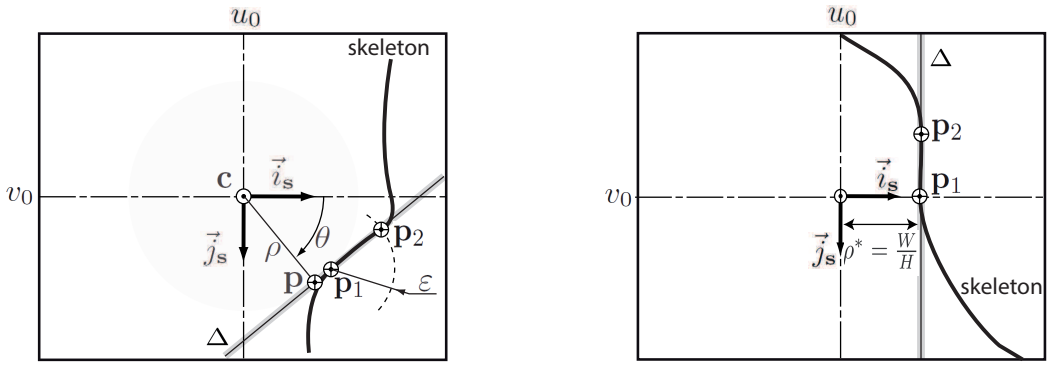
3.4.1 Définition et extraction de la signature

Le squelette de l'espace libre permet de définir de bonnes caractéristiques visuelles pour l'asservissement. Nous considérons ici une approximation linéaire de la branche du squelette la plus proche de la position du robot. La tâche d'asservissement résultante est alors exprimée dans l'image en tant que suivi du squelette.

Dans le repère du capteur virtuel $\mathcal{F}_C$, le comportement souhaité se traduit par la convergence de la projection de O_R dans l'image vers le squelette avec un alignement de $\vec{j}_R$ avec une approximation linéaire locale Δ du squelette indépendamment de la position de la caméra virtuelle. Comme illustrée sur la figure 3.4 lorsque la caméra virtuelle n'est pas placée sur l'axe longitudinal du robot ($W \neq 0$), pour avoir le robot sur le squelette, le point O_C doit être éloigné du squelette de W/H dans l'image et la ligne Δ doit être verticale (parallèle à l'axe $\vec{j}_C$). Lorsque $W = 0$, le point principal de l'image

$\mathbf{c} = (u_0, v_0)$ doit être sur le squelette et l'approximation linéaire Δ alignée verticalement.

En nous appuyant sur la figure 3.4, nous considérons une signature visuelle de la forme $\mathbf{s} = (\rho, \theta)$ définie sur un paramétrage polaire de la projection de Δ dans l'image d'un point $\mathbf{p}_1$ sur le squelette, avec $\mathbf{p}_1$ le point de projection le plus proche du squelette au point principal $\mathbf{c}$. Comme l'illustre la figure, la ligne projetée Δ est définie à l'aide d'un deuxième point $\mathbf{p}_2$ sur le squelette de telle sorte que la distance euclidienne entre les deux points soit $d(\mathbf{p}_1, \mathbf{p}_2) = \varepsilon$ où ε est un paramètre positif choisi. Si plusieurs solutions sont possibles pour le point $\mathbf{p}_2$, en particulier des branches du squelette, le point le plus proche au sens de l'alignement du robot est sélectionné.



(a) La signature visuelle dans l'image virtuelle.

(b) La position désirée de la signature visuelle durant le suivi du DVG.

FIGURE 3.4 – Une approximation linéaire de la projection DVG dans l'image virtuelle est réalisée. La signature visuelle est calculée à partir de la ligne Δ définie par les paramètres (ρ, θ). Le suivi du squelette consiste à maintenir la ligne Δ verticale dans l'image.

Pour déterminer $(\rho,\ \theta)$, on définit le point $\mathbf{p} = (\mathbf{u_p}, \mathbf{v_p})^T$ dans l'image qui correspond à la projection de $\mathbf{c}$ sur Δ. La signature visuelle $\mathbf{s} = (\rho, \theta)^T$ peut être calculée par :

$$\mathbf{s} = \begin{pmatrix} \rho \\ \theta \end{pmatrix} = \begin{pmatrix} \sqrt{(u-u_0)^2 + (v-v_0)^2} \\ atan2(v-v_0, u-u_0) \end{pmatrix} \tag{3.4}$$

Afin de rallier et suivre le squelette, la signature de consigne doit correspondre à une droite verticale qui passe par la projection du centre du robot dans l'image, donc :

$$\begin{cases} \rho^* & = \frac{W}{H} \\ \theta^* & = 0[\pi] \end{cases} \tag{3.5}$$

$$\text{avec} \begin{cases} \text{si} & \mid \theta - \pi \mid < \frac{\pi}{2} \ \text{ alors } \ \theta^* = \pi \ \text{ (Le robot est à gauche du squelette)} \\ \text{si} & \theta^* < \frac{\pi}{2} \ \text{ alors } \ \theta^* = 0 \\ \text{sinon} & \theta^* = 0 \ \text{ (le robot est à droite du squelette)} \end{cases} \tag{3.6}$$

3.4.2 Élaboration de la loi de commande

En notant $\mathbf{T}_C$ le torseur cinématique du capteur, et $\mathbf{L_{sr}}$ la matrice d'interaction, nous rappelons que l'évolution de $\mathbf{s}$ dans le temps peut être exprimée par rapport à la cinématique du capteur par :

$$\dot{\mathbf{s}} = \mathbf{L_{sr}}\mathbf{T}_C \tag{3.7}$$

Pour calculer la loi de commande, il faut chercher l'évolution de la signature en fonction de la vitesse du robot $\mathbf{U} = (v, \omega)^T$. Selon (3.7) pour modéliser les variations de la signature visuelle $\mathbf{s}$ par rapport à $\mathbf{U}$, il est d'abord nécessaire de définir la matrice de passage entre la commande et le torseur cinématique de la caméra dans le repère monde $\mathbf{P_{T_C \leftarrow U}}$ telle que $\mathbf{T_C} = \mathbf{P_{T_C \leftarrow U}}\mathbf{U}$.

La vitesse du robot est définie par $\mathbf{T_R} = (\mathbf{v}, 0, 0, 0, 0, \omega)^T$ dans le repère du robot $\mathcal{F}_R$. La relation entre $\mathbf{T}_R$ et $\mathbf{T_C}$ est alors donnée par :

$$\mathbf{T_C} = \begin{pmatrix} \mathcal{R}_{\mathcal{F}_C \leftarrow \mathcal{F}_R} & [\mathbf{t}_{\mathcal{F}_C \leftarrow \mathcal{F}_R}]_\times \mathcal{R}_{\mathcal{F}_C \leftarrow \mathcal{F}_R} \\ 0 & \mathcal{R}_{\mathcal{F}_C \leftarrow \mathcal{F}_R} \end{pmatrix} \mathbf{T_R} \tag{3.8}$$

avec $[\mathbf{t}_{\mathcal{F}_C \leftarrow \mathcal{F}_R}]_\times$ est une matrice antisymétrique associée à $\mathbf{t}_{\mathcal{F}_C \leftarrow \mathcal{F}_R}$. En conséquence, la relation entre le mouvement du capteur virtuel et la commande du robot est donnée par

$$\mathbf{P_{T_C \leftarrow U}} = \begin{pmatrix} 0 & -1 & 0 & 0 & 0 & 0 \\ -L & W & 0 & 0 & 0 & -1 \end{pmatrix}^T \tag{3.9}$$

L'équation (3.7) se réécrit comme suit : $\dot{\mathbf{s}} = \mathbf{L_{sr}}\mathbf{P_{T_S \leftarrow U}}\mathbf{U}$. Considérant la ligne Δ comme projection dans le capteur virtuel d'une ligne 3D appartenant au plan du sol ($Z = H$ dans le repère du capteur virtuel), $\mathbf{L_{sr}}$ est :

$$\mathbf{L_{sr}} = \begin{pmatrix} \frac{-cos(\theta)}{H} & \frac{-sin(\theta)}{H} & \frac{\rho}{H} & \sin(\theta)\left(1+\rho^2\right) & -\cos(\theta)\left(1+\rho^2\right) & 0 \\ 0 & 0 & 0 & -\rho\cos(\theta) & -\rho\sin(\theta) & -1 \end{pmatrix} \tag{3.10}$$

En combinant (3.7) et (3.10), l'évolution de $\mathbf{s}$ par rapport à l'entrée de commande $\mathbf{U}$ est :

$$\dot{\mathbf{s}} = \mathbf{J_s}\mathbf{U} = \begin{pmatrix} \frac{1}{H}\sin\theta & \frac{1}{H}\left(L\cos\theta - W\sin\theta\right) \\ 0 & 1 \end{pmatrix} \mathbf{U} \tag{3.11}$$

Rappelons que, pour avoir le robot sur le DVG, la consigne de la signature visuelle est $\mathbf{s}^* = (\frac{W}{H}, 0)^T$. L'erreur de l'asservissement est $\varepsilon_\mathbf{s} = (\varepsilon_\rho, \varepsilon_\theta)^T = \mathbf{s} - \mathbf{s}^*$. Sa dérivée temporelle peut donc être écrite en utilisant (3.11) :

$$\dot{\varepsilon}_\mathbf{s} = \dot{\mathbf{s}} = \mathbf{J_s}\mathbf{U} = \mathbf{J_v}v^c + \mathbf{J_\omega}\omega \tag{3.12}$$

avec :

$$\mathbf{J_v} = \begin{pmatrix} \frac{\sin\theta}{H} \\ 0 \end{pmatrix} \text{ et } \mathbf{J}_\omega = \begin{pmatrix} \frac{L\cos\theta - W\sin\theta}{H} \\ 1 \end{pmatrix} \tag{3.13}$$

Pour garantir la convergence de la signature $\mathbf{s}$ vers $\mathbf{s}^*$, une décroissance exponentielle de l'erreur

est imposée sous la forme :

$$\dot{\varepsilon}_{\mathbf{s}} = \mathbf{\Lambda}\varepsilon_{\mathbf{s}} \text{ avec } \mathbf{\Lambda} = \begin{pmatrix} -1/\tau_\rho & 0 \\ 0 & -1/\tau_\theta \end{pmatrix} \tag{3.14}$$

avec τ_ρ et τ_θ sont des gains positifs représentant le temps de convergence en tant que paramètres de la commande.

En combinant (3.12) et (3.14), nous avons :

$$\omega = -\mathbf{J}_{\boldsymbol{\omega}}^{+}\left[\mathbf{\Lambda}\varepsilon_{\mathbf{s}} - \mathbf{J}_{\mathbf{v}}v^c\right] \tag{3.15}$$

avec $\mathbf{J}_{\boldsymbol{\omega}}^{+}$ la matrice pseudo-inverse de Moore-Penrose :

$$\mathbf{J}_{\boldsymbol{\omega}}^{+} = \begin{cases} 0 & \text{si } \mathbf{J}_{\boldsymbol{\omega}} \text{ est nulle (ce ne sera pas le cas ici)} \\ \mathbf{J}_{\boldsymbol{\omega}}^{T}/(\mathbf{J}_{\boldsymbol{\omega}}^{T}\mathbf{J}_{\boldsymbol{\omega}}) & \text{ailleurs} \end{cases} \tag{3.16}$$

En remplaçant tous les termes, la loi de commande proposée est obtenue :

$$\omega = \frac{\left(\dfrac{W\sin\theta}{H} - \dfrac{L\cos\theta}{H}\right)\left(\dfrac{\varepsilon_\rho}{\tau_\rho} + \left(\dfrac{\sin\theta}{H}\right)v^c\right) - \dfrac{\varepsilon_\theta}{\tau_\theta}}{1 + \left(\dfrac{L}{H}\cos\theta - \dfrac{W}{H}\sin\theta\right)^2} \tag{3.17}$$

3.5 Étude de la stabilité de la commande

3.5.1 Singularités de la loi de commande

Certaines conditions doivent être respectées afin d'éviter les singularités dans la commande proposée (3.17). Tout d'abord, la commande ω est définie lorsque $H \neq 0$. Deuxièmement, lorsque la droite Δ est verticale dans l'image, c'est-à-dire lorsque $\theta \equiv 0$ (*modulo* π), si $L = 0$, il n'y a pas de correction possible de l'erreur latérale ε_ρ. En d'autres termes, lorsque le robot est parallèle à la tangente locale du DVG avec un décalage, l'erreur ne peut pas diminuer.

De plus, pour éviter les singularités du dénominateur dans (3.17), en particulier à $\theta \equiv 0$, nous devons avoir : $L^2 + H^2 \neq 0$. Cela signifie que L et H ne peuvent pas être tous les deux égaux à 0.

3.5.2 Convergence du système commandé

L'évolution de θ et ρ est décrite dans (3.11) par le système non linéaire suivant :

$$\begin{cases} \dot{\rho} = \dfrac{\sin\theta}{H}\,v^c + \left(\dfrac{L}{H}\,\cos\theta - \dfrac{W}{H}\,\sin\theta\right)\omega \\ \dot{\theta} = \omega \end{cases} \tag{3.18}$$

Les positions $\mathbf{s_{eq}} = (\frac{W}{H}, k\pi)^T$ sont des points d'équilibre avec $k \in \mathbb{N}$. En effet, avec une commande nulle ($\omega = 0$) et un système placé à ces points d'équilibre, les deux dérivées par rapport au temps du système (3.18) sont nulles et le système reste dans cette position.

En raison du processus de fabrication de la mesure (3.4), l'angle extrait θ de l'image virtuelle appartient à l'intervalle $[-\pi;\pi]$. Pour la même raison, la mesure de ρ est toujours positive. L'état initial du système respectera ces contraintes.

Nous avons $\varepsilon_\rho = \rho - \dfrac{W}{H}$ et $\varepsilon_\theta = \theta$ avec la signature de consigne $\mathbf{s}^* = (\frac{W}{H}, 0)^T$. En appliquant la commande (3.17) dans (3.18), le système suivant est obtenu :

$$\dot{\varepsilon}_\mathbf{s} = f(\varepsilon_\mathbf{s}) \Leftrightarrow \begin{cases} \dot{\varepsilon}_\theta = f_1(\varepsilon_\theta, \varepsilon_\rho) \\ \dot{\varepsilon}_\rho = f_2(\varepsilon_\theta, \varepsilon_\rho) \end{cases} \tag{3.19}$$

Une fois le système (3.19) développé, le point $\varepsilon_{\mathbf{s_{eq}}} = (0,0)^T$ est un point d'équilibre de la commande en boucle fermée. La stabilité, l'unicité et l'attractivité de ce point d'équilibre seront discutées ci-dessous.

3.5.3 Analyse de la stabilité locale

En appliquant le deuxième théorème de Lyapunov, si une approximation linéaire du système original autour du point d'équilibre est strictement stable (si la partie réelle de chaque valeur propre de la matrice jacobienne de $f(\varepsilon_\mathbf{s})$ par rapport à $\varepsilon_\mathbf{s} = \varepsilon_{\mathbf{seq}}$ est strictement négative), alors le point d'équilibre est asymptotiquement stable pour le système non linéaire d'origine dans un domaine autour du point d'équilibre [Slotine et Li, 1991].

La matrice jacobienne de (3.19) calculée au point d'équilibre $\mathbf{s_{eq}}$ est :

$$\nabla f|_{\varepsilon_{\mathbf{seq}}} = \left(\begin{array}{cc} \dfrac{\partial f_1}{\partial \varepsilon_\theta} & \dfrac{\partial f_1}{\partial \varepsilon_\rho} \\ \dfrac{\partial f_2}{\partial \varepsilon_\theta} & \dfrac{\partial f_2}{\partial \varepsilon_\rho} \end{array} \right) \Bigg|_{\varepsilon_\mathbf{s} = \varepsilon_{\mathbf{seq}}} \tag{3.20}$$

$$\text{avec} \begin{cases} \dfrac{\partial f_1}{\partial \varepsilon_\theta} = -\dfrac{H^2 + L\, v^c\, \tau_\theta}{\tau_\theta\,(H^2+L^2)} & ; \quad \dfrac{\partial f_1}{\partial \varepsilon_\rho} = -\dfrac{H\,L}{\tau_\rho\,(H^2+L^2)} \\ \dfrac{\partial f_2}{\partial \varepsilon_\theta} = \dfrac{H\,(v^c\,\tau_\theta - L)}{\tau_\theta\,(H^2+L^2)} & ; \quad \dfrac{\partial f_2}{\partial \varepsilon_\rho} = -\dfrac{L^2}{\tau_\rho\,(H^2+L^2)} \end{cases}$$

Les valeurs propres $\lambda_\pm$ de la matrice jacobienne sont :

$$\lambda_\pm = \frac{-H^2\tau_\rho - L\tau_\theta(L + \tau_\rho v^c) \pm \sqrt{D}}{2\tau_\theta\tau_\rho\,(H^2+L^2)} \tag{3.21}$$

avec D le discriminant :

$$D = L^4\tau_\theta^2 + \tau_\rho^2\left(H^2 + Lv^c\tau_\theta\right)^2 - 2L\tau_\theta\tau_\rho\left(L^2 v^c\tau_\theta - H^2\left(L - 2v^c\tau_\theta\right)\right)$$

Puisque les paramètres v^c, H et les gains τ_θ, τ_ρ sont choisis positifs, le seul paramètre qui influe sur le signe de la partie réelle des valeurs propres est la position longitudinale de la caméra virtuelle L. La stabilité du point d'équilibre $\varepsilon_{\mathbf{seq}}$ dépend donc de la position longitudinale L du capteur virtuel. Bien que le paramètre W n'apparaisse pas dans la matrice jacobienne, nous pouvons aussi nous attendre à une influence sur la trajectoire du robot. Le placement du capteur virtuel influence la stabilité de la

commande.

Pour conclure sur la stabilité locale du système autour du point d'équilibre, une analyse de plan de phase du système en boucle fermée (3.19) sera réalisée pour étudier à la fois la stabilité et l'attractivité des points d'équilibre lorsqu'on s'éloigne du point d'équilibre.

3.5.4 Stabilité globale et attractivité des points d'équilibre

Différentes conditions initiales ont été choisies pour évaluer l'attractivité du point d'équilibre $\mathbf{s}_{eq}$. La simulation est calculée avec la méthode d'Euler avec un pas de temps fixe $T_e = 0.01\,s$ pendant $100\,s$. L'influence de différentes positions du capteur virtuel est également illustrée.

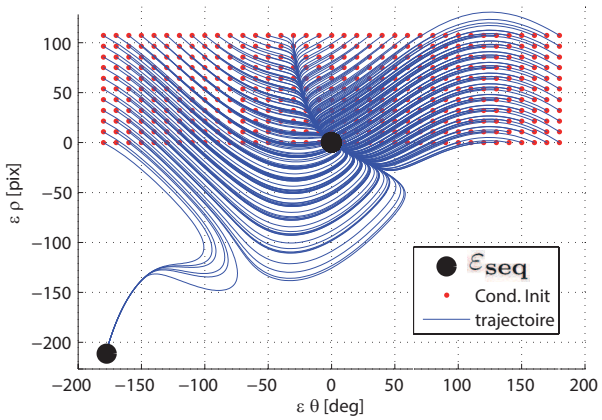


FIGURE 3.5 – Portrait de phase du système en boucle fermée pour différentes conditions initiales. Certaines trajectoires se déplacent vers un deuxième point d'équilibre ($\varepsilon_\theta = -180, \varepsilon_\rho = -220pix$)

La figure 3.5 montre les trajectoires dans le plan de phase pour différentes conditions initiales. Le capteur virtuel est placé à $L = 1\,m$, $W = -1\,m$, $H = 7\,m$. Les constantes de temps sont choisies $\tau_\theta = 1\,s$ et $\tau_\rho = 10\,s$, la vitesse linéaire est fixée $v^c = 0.5\,m/s$. Les valeurs propres sont réelles et négatives avec des valeurs autour de -1 et -10^{-3}. Le système est donc asymptotiquement stable autour du point d'équilibre.

En utilisant la loi de commande (3.17), on constate que pour certaines positions initiales un demi-tour est effectué par le robot. Pour ces cas le portrait de phase montre que certaines trajectoires convergent vers un second point d'équilibre $\left(\varepsilon_\theta = 0, \varepsilon_\rho = \dfrac{\pi}{\tau_\theta}\dfrac{H}{L}\tau_\rho\right)$ ce qui correspond à un demi-tour. Pour les autres trajectoires, le système est stable et le point d'équilibre $(\varepsilon_\theta = 0, \varepsilon_\rho = 0)$ est attractif.

Pour éviter tout problème d'instabilité, la consigne peut être modifiée suivant les conditions suivantes :

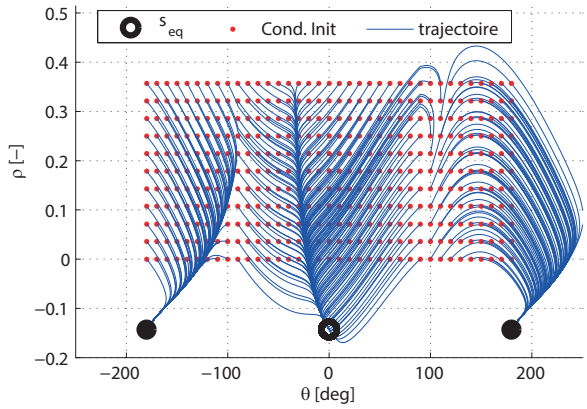
$$\begin{cases} \text{si } \theta \in [-\frac{\pi}{2}; \frac{\pi}{2}] \text{ alors } \theta^* = 0 \quad \text{et } \varepsilon_\theta = \theta \\ \text{si } \theta < -\frac{\pi}{2} \quad \text{alors } \theta^* = -\pi \text{ et } \varepsilon_\theta = \theta + \pi \\ \text{si } \theta > \frac{\pi}{2} \quad \text{alors } \theta^* = \pi \quad \text{et } \varepsilon_\theta = \theta - \pi \\ \varepsilon_\rho = \rho - \frac{W}{H} \end{cases} \tag{3.22}$$

Cette proposition équivaut à imposer un mouvement vers l'avant au robot et crée trois points d'équilibre possibles $\theta_{eq} = \{-\pi; 0; \pi\}$. Cela correspond à atteindre et à suivre la même approximation linéaire du DVG dans un sens ou l'autre. Les trois points d'équilibre sont mis en évidence sur la

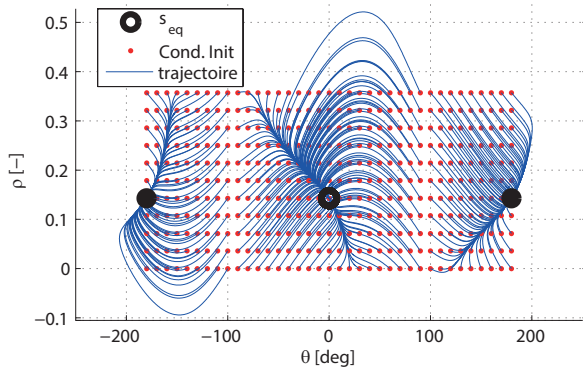
figure 3.6.

Considérons maintenant l'effet du positionnement du capteur virtuel sur la stabilité de la commande. Nous allons tester les cas ou $L > 0$ et $L < 0$.

1) $L > 0$: le capteur virtuel placé devant le point caractéristique du robot



(a) Caméra virtuelle en avant et à droite par rapport au robot $L > 0$ and $W < 0$



(b) Caméra virtuelle en avant et à gauche par rapport au robot $L > 0$ et $W > 0$

FIGURE 3.6 – Portrait de phase du système en boucle fermée pour différentes conditions initiales et différentes positions de capteur virtuel. Toutes les trajectoires sont stables.

Pour ce cas, les mêmes paramètres sont utilisés pour produire la figure 3.6a. Pour la figure 3.6b la caméra est mise de l'autre côté du robot $W = 1\,m$. La convergence est également obtenue pour toutes les conditions initiales. Toutes les trajectoires convergent directement vers l'un des points d'équilibre.

Nous pouvons remarquer à partir de la figure 3.6 que, pour chaque condition initiale, une courbe d'attraction existe. Toutes les trajectoires convergent vers cette courbe avant de rallier le point d'équilibre. Le comportement est celui d'une surface de glissement et sa forme dépend des constantes de temps τ_ρ et τ_θ et des autres paramètres (position de la caméra virtuelle et la vitesse linéaire du robot).

La position latérale n'apparaît pas dans la linéarisation du système en boucle fermée (3.20). Cependant, comme on pouvait s'y attendre, son effet est visible dans le portrait de phase comme

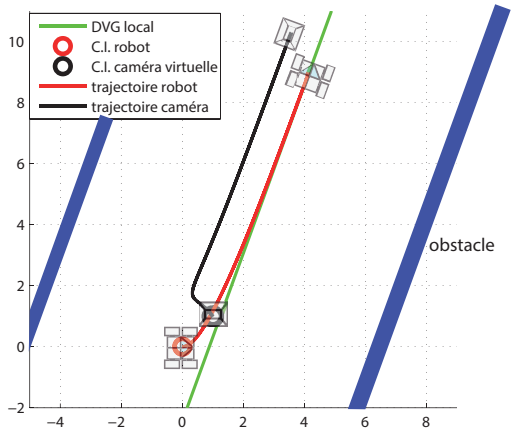


FIGURE 3.7 – Trajectoire du robot et de la caméra lorsque la commande proposée est appliquée. La caméra est placée à $L = 1\,m$, $W = 1\,m$. A l'état initial, la signature visuelle commence à ($\theta_0 = -70\,deg$ et $\rho_0 = 10\,pix$). Le robot atteint et suit le DVG.

illustré par la différence entre la figure 3.6a et la figure 3.6b.

La figure 3.7 présente les trajectoires du robot et de la caméra virtuelle dans le monde lorsque la commande proposée est appliquée. À chaque instant, une fois que le squelette de l'espace libre est calculé, une approximation linéaire se fait dans l'image virtuelle, produisant les paramètres (ρ, θ) souhaités de la ligne Δ correspondante. La trajectoire montre que le système est stable et atteint l'approximation locale du DVG avec la commande proposée. Ce résultat est vérifié pour toute condition initiale du système.

2) $L < 0$: le capteur virtuel derrière le point caractéristique du robot.

Dans ce cas, les mêmes paramètres sont choisis. Seul le paramètre L change de signe $L = -1$. Les valeurs propres deviennent environ -0.97 et $+10^{-3}$; le point d'équilibre est donc un point selle. Le portrait de phase de la figure 3.8 montre que les trajectoires s'éloignent du point d'équilibre vers l'infini. Nous pouvons conclure que $\mathbf{s}_{eq}$ est répulsif quand $L < 0$. Les trajectoires subissent une

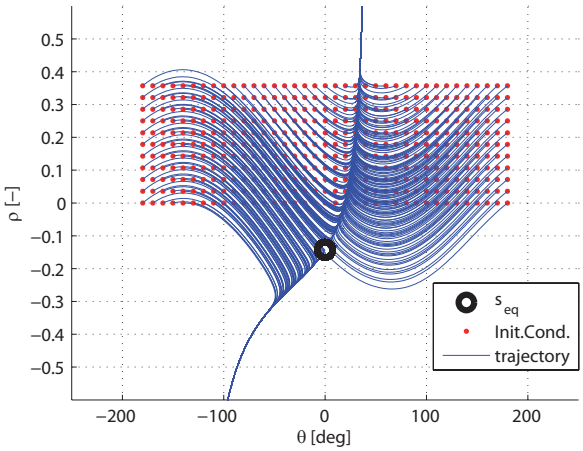


FIGURE 3.8 – Portrait de phase lorsque le capteur virtuel est placé derrière le point caractéristique du robot $L < 0$.

direction particulière qui est instable du point de vue du système bouclé. Cette direction est une valeur asymptotique de θ mais ρ augmente en raison de la vitesse linéaire du robot. Ceci correspond à une droite parallèle à la droite de consigne, $\theta = atan(L/W)$ ou $\theta = -\pi + atan(L/W)$ qui annihile l'effet de ε_ρ dans la commande.

En conclusion, la loi de contrôle n'est pas stable lorsque $L < 0$. Le capteur virtuel doit être positionné de telle sorte que $L > 0$. L'analyse des portraits de phase a montré l'effet d'attractivité des points d'équilibre lorsque $L < 0$ et le comportement du système en subissant la commande (3.17).

3.6 Simulation et résultats expérimentaux

Pour valider l'approche proposée, un ensemble de scénarios est conçu, d'abord à l'aide d'outils de simulation, puis dans des environnements réels. Ils ont été réalisés à l'aide du middleware ROS (version Kinetic). Le même code source C^{++} est utilisé pour les expériences en simulation et dans le monde réel.

3.6.1 Résultats de simulation

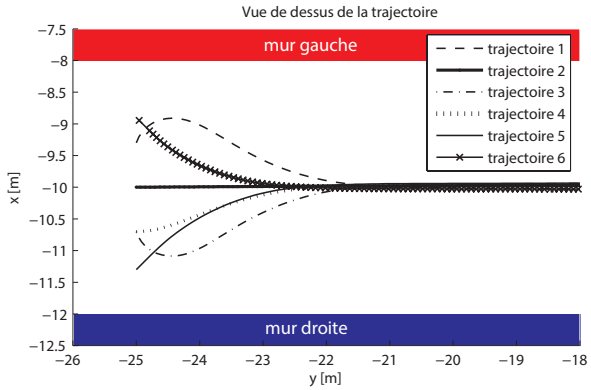
Les résultats de simulation sont obtenus à l'aide de l'environnement Morse-Blender. Un robot différentiel virtuel est équipé d'un lidar (270 degrés d'ouverture). Pour la simulation nous allons tester différents environnements (un couloir avec plusieurs poses initiales du robot et deux autres environnements avec des obstacles statiques et dynamiques). Le capteur virtuel et la caméra associée sont placés à $L = 0.2\ m$, $W = 0\ m$ et $H = 7\ m$ par rapport au repère robot. L'image générée a pour dimensions 640×480 avec des paramètres intrinsèques définis $u_0 = 320, v_0 = 240$, et $\alpha_u = \alpha_v = 200$. La vitesse linéaire est réglée à $v^c = 0.8\, m/sec$.

Scénario 1 : Le premier scénario a été principalement conçu pour valider les performances de la loi de commande à induire une convergence vers le DVG local pour différentes positions initiales du robot. Les constantes de temps sont définies telles que $\tau_\theta = 1$ et $\tau_\rho = 0.05$. Le temps de réponse global est d'environ 6 secondes.

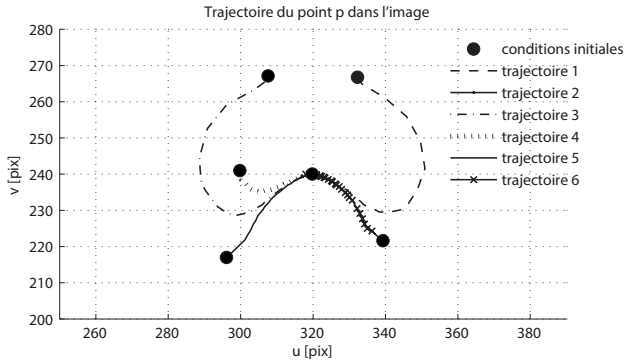
Les résultats sont présentés sur la figure 3.9. Ils incluent les trajectoires du robot pour chaque pose de départ (figure 3.9a), l'évolution du point $\mathbf{p}$ dans l'image virtuelle (figure 3.9b) (utilisée pour déterminer la signature visuelle $\mathbf{s}$). Une convergence du robot vers le DVG local est bien constatée. La caméra virtuelle étant placée sur le plan longitudinal du robot, le robot et la caméra virtuelle suivent le DVG. L'évolution de la commande (vitesse angulaire) correspondante est également visible sur la figure 3.9c.

Comme représenté à la figure 3.9a, l'environnement considéré est un couloir droit. Le robot a pour positions initiales des positions angulaires et latérales variables par rapport au DVG (la ligne médiane dans ce cas). Dans chaque cas, la loi de commande (3.17) est appliquée.

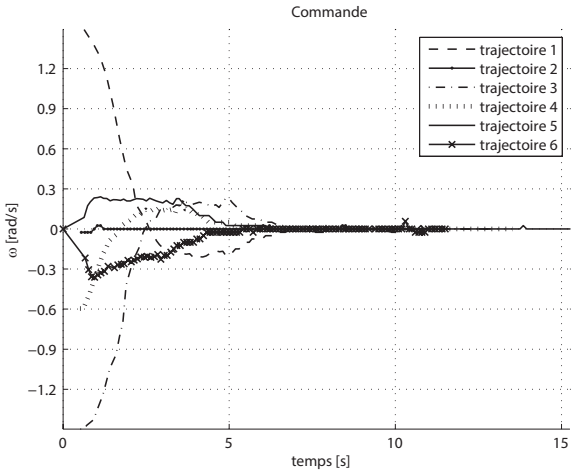
Les résultats mettent en évidence une convergence rapide du système vers le DVG local. Dans chaque cas, le point $\mathbf{p}$ converge vers le point principal de l'image virtuelle, tandis que la vitesse angulaire atteint 0 en moins de 6 secondes pour chaque cas. Notez cependant que lorsque l'erreur angulaire initiale est $\pi/2$ (trajectoires 1 et 3), le robot s'éloigne de la ligne médiane avant la convergence. Enfin, nous



(a) Trajectoires du robot avec différentes positions et orientations initiales.



(b) Convergence du point $\mathbf{p}$ vers le centre de l'image.



(c) Évolution de la vitesse angulaire ω pour les différents tests.

FIGURE 3.9 – Résultats de simulation de l'asservissement basé squelette, pour plusieurs poses initiales du robot dans un corridor.

pouvons voir sur la figure 3.9a une petite dérive à la fin des trajectoires qui ne sont pas parfaitement alignées avec la réalité terrain (ligne médiane). Ceci est dû à des erreurs de discrétisation du capteur caméra avec une faible résolution d'image. En effet, 1 pixel dans l'image virtuelle représente environ un carré de $14cm \times 14cm$ au sol. Ceci permet un temps de calcul rapide (moins de 100 ms) au détriment d'une plus grande précision.

Scénario 2 : Dans ce scénario, nous avons ajouté plusieurs obstacles dans l'environnement précédent. Cela permet d'avoir un squelette non rectiligne. La commande appliquée est celle proposée par l'équation (3.17).

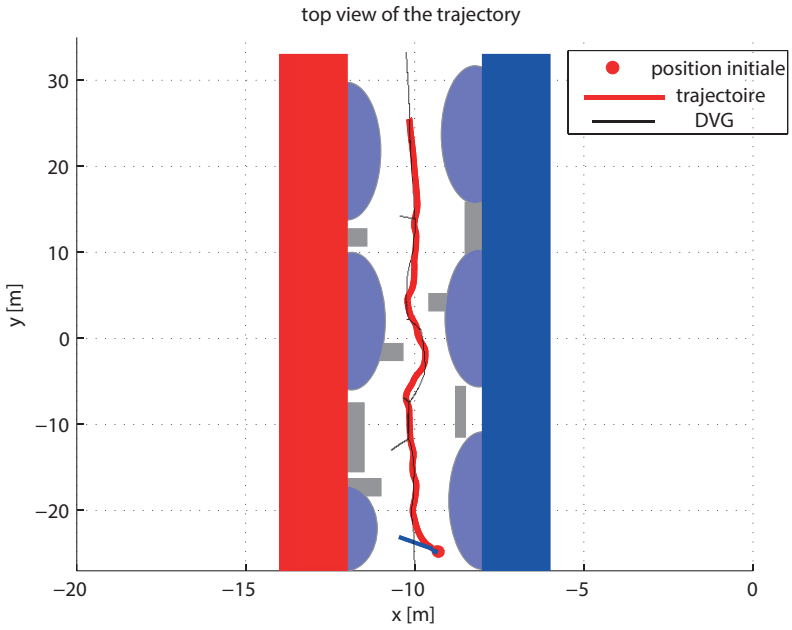


FIGURE 3.10 – Illustration du suivi du DVG pour le scénario 2, le robot (en rouge) arrive à rallier le DVG (en noir).

Le robot parcourt $54\,m$ de distance sur la durée de l'essai. La commande est appliquée en prenant :

$$\begin{cases} \tau_\theta & = 2 \\ \tau_\rho & = 0.05 \\ v^c & = 0.3m/s \end{cases} \tag{3.23}$$

Dans les premières 17 secondes (figure 3.11) l'erreur de la signature a convergé vers 0 puisque le robot a atteint le DVG. On note qu'à chaque variation locale du DVG, le robot tend à rallier le nouveau squelette. A chaque nouvelle observation, la consigne est régulièrement évaluée et elle est prise en compte dans la loi de commande. Nous pouvons observer dans la figure 3.10 que le modèle du robot-capteur évite bien les obstacles et rallie à chaque fois le squelette de l'espace navigable.

Scénario 3 : Le troisième scénario, présenté sur la figure 3.12 considère un environnement encombré avec de multiples obstacles aléatoires et dynamiques. Ce test est conçu pour mettre en évidence la

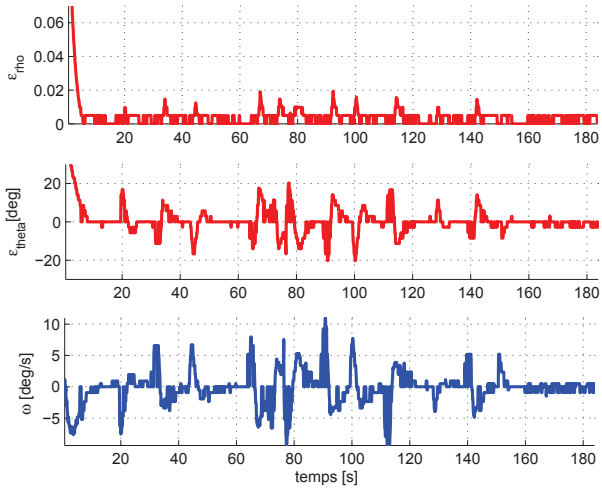


FIGURE 3.11 – Evolution de l'erreur de la signature et la commande appliquée pour le scénario 2

robustesse et l'efficacité de l'asservissement visuel basé squelette en présence d'obstacles mobiles avec un parcours de distance de 100 m. La trajectoire du robot est ici reconstruite par un algorithme SLAM [Grisetti et al., 2007] utilisant des capteurs d'odométrie intégrés.

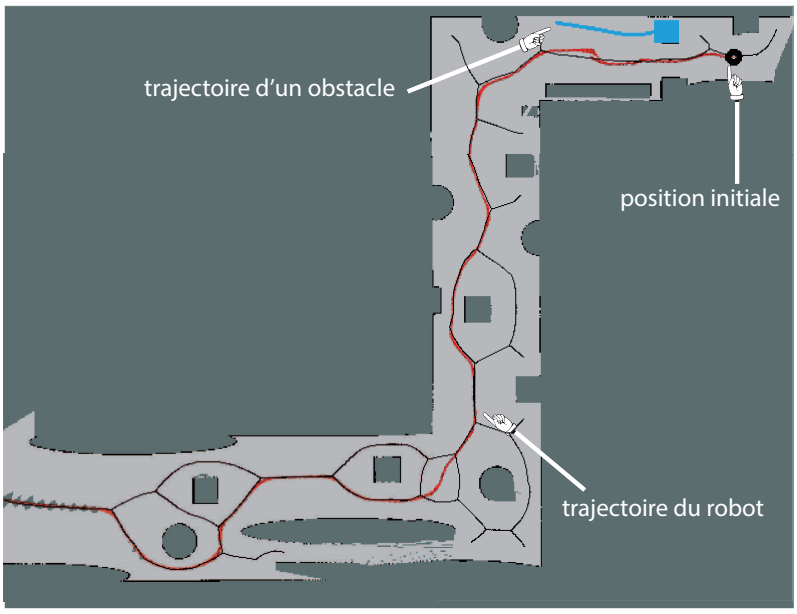


FIGURE 3.12 – Asservissement visuel basé squelette dans un environnement de simulation encombré. Cette visualisation est réalisée à l'aide de l'outil RVIZ à partir du middleware ROS. C'est une grille d'occupation créée dynamiquement. Les zones couleur gris clair représentent l'espace libre vu par le robot. En gris foncé, c'est l'environnement non exploré. Les impacts lasers apparaissent en noir. Les trajectoires du robot et de l'obstacle en mouvement sont respectivement en rouge et bleu. Le DVG de l'environnement globale est illustré en trait noir.

Un objet mobile est introduit dans l'environnement ce qui engendre une variation locale importante et fréquente du DVG. Le comportement du système est évalué par l'évolution des erreurs d'asservissement et la vitesse angulaire d'entrée. Les courbes correspondantes sont présentées à la

figure 3.13. Après une convergence rapide (les 4 premières secondes), le système suit parfaitement les variations du squelette et se rapproche du DVG. Les fortes variations visibles sur la figure 3.13 sont directement liées à la topologie complexe de l'environnement. Tous les pics correspondent à la brusque variation de DVG qui modifie dynamiquement la reconfiguration de la consigne. Néanmoins, la commande tend à maintenir les erreurs d'asservissement à zéro. Le pic, à l'instant $t = 40\,s$, coïncide avec la détection de l'objet mobile. Ici aussi, la trajectoire est automatiquement adaptée et un nouveau point de consigne est atteint en conséquence. Lorsque le DVG présente des formes localement régulières, le système converge efficacement et d'une manière lisse à la configuration souhaitée (par exemple à l'instant $250\,s$).

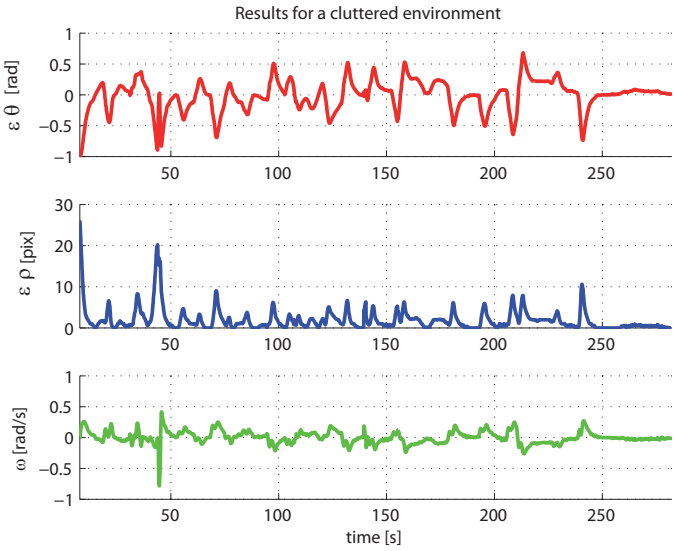
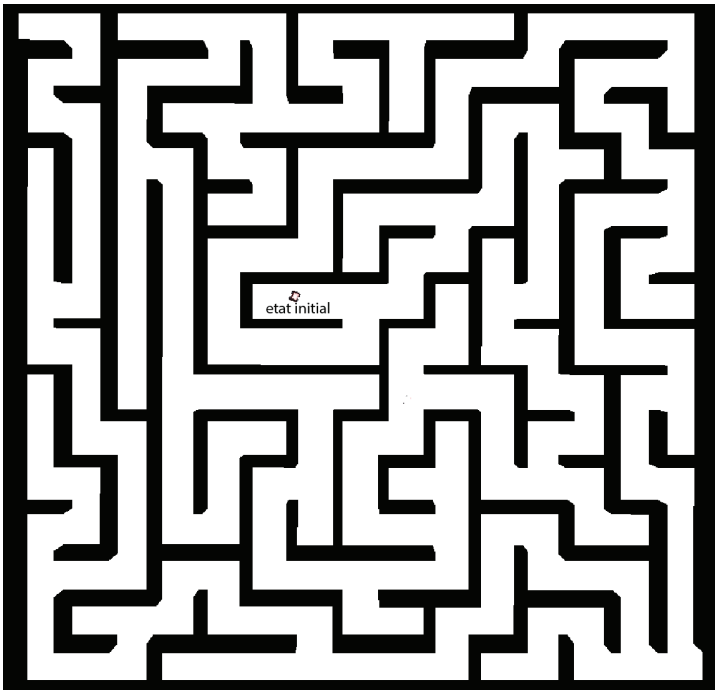
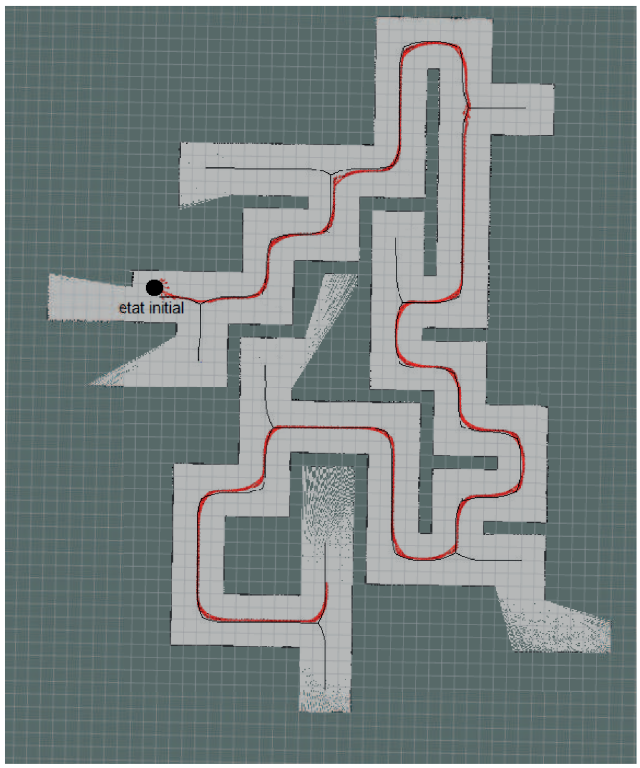


FIGURE 3.13 – Comportement du système pour le scénario 3. En haut : erreur angulaire ε_θ. Au milieu : erreur latérale exprimée dans l'image ε_ρ. En bas : commande ω.

Scénario 4 : Ce test consiste à faire déplacer le robot dans un labyrinthe, le robot est positionné aléatoirement au début. On applique la loi de commande (3.17). La trajectoire effectuée par le robot est illustrée dans la figure 3.14. Le robot parcourt une distance de $200m$ et il se déplace parfaitement au milieu des murs du labyrinthe.



(a) Environnement et position initiale du robot



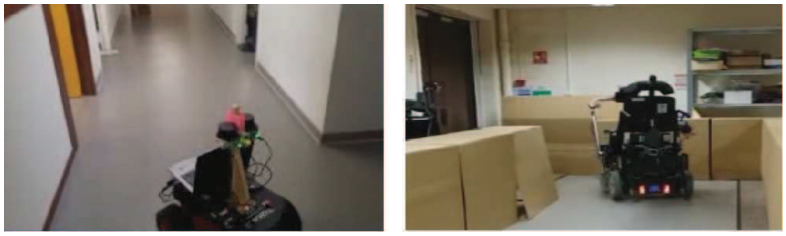
(b) Trajectoire du robot

FIGURE 3.14 – Déplacement de $200m$ du robot dans un labyrinthe : trajectoire du robot est illustrée en trait rouge et le DVG de l'environnement est en trait noir

3.6.2 Résultats expérimentaux

3.6.2.1 Résultat de l'extraction du squelette

Les figures 3.15 illustrent l'extraction de l'espace libre dans des environnements réels puis l'extraction du DVG local. Dans ces exemples, la perception est réalisée en utilisant la fusion de deux RPlidars (un laser omnidirectionnel) installés sur un robot mobile (Pioneer) qui scanne un champ de vue à 360^o pour la colonne de gauche de la figure 3.15. Sur le fauteuil roulant servant aux expériences (colonne de droite), le champ de vue est limité à 180^o par les contraintes de l'emplacement du capteur. Ces résultats présentent une prise de vue locale de chaque perception.



(a) Des robots utilisées pour extraire le squelette, un pioneer (gauche), un fauteuil roulant (droite)

(b) Masque binaire délimitant l'espace navigable construit par une vue aérienne

(c) DVG local obtenu par la squelettisation

FIGURE 3.15 – Extraction du DVG local avec différentes plateformes et capteurs : deux RPLidar (gauche), un seul Hokuyo (droite)

3.6.2.2 Tests sur un fauteuil roulant électrique

Nous avons utilisé un fauteuil roulant de l'université de Kent au sein du "School of Engineering and Digital Arts" pour la validation expérimentale de ces travaux. Ce fauteuil roulant est considéré comme un robot différentiel évoluant dans un plan horizontal. Il se comporte comme un robot unicycle. Le point caractéristique du fauteuil O_R est le centre de l'essieu arrière (les positions des capteurs sont calculé à partir de cette position). Les capteurs (qui nous intéressent) installés sur le fauteuil sont :

- Télémètre Laser : Hokuyo URG-04LX :

  - Angle d'ouverture de 240^o
  - Résolution angulaire de 0.36^o
  - Le temps du scan de 100ms/nappe
  - Position $\overrightarrow{O_R O_L} = (0.5\,m,\ -0.288\,m, 0.412\,m)$
- Codeurs rotatifs : au niveau des roues arrière du fauteuil roulant.

FIGURE 3.16 – Fauteuil roulant de l'université de Kent utilisé pour l'expérimentation.

Une carte de type Udoo est embarquée sur le fauteuil pour les traitements des capteurs et la commande. La commande envoyée au fauteuil roulant doit être comprise entre $[-1\ 1]$ rad/s. Par rapport aux simulations, nous choisissons des gains $\tau_\rho = 2.5, \tau_\theta = 4.25$ pour diminuer l'énergie demandée aux actionneurs et éviter de les saturer.

Pour visualiser la trajectoire du fauteuil roulant, les données des codeurs sont extraites et affichées dans la figure 3.17 (trajectoire en vert). L'estimation de la position du robot à partir des codeurs engendre un décalage par rapport à la position réelle du robot. La nature incrémentale de cette mesure est sujette à une dérive qui s'accumule avec de grandes erreurs dans l'estimation de pose au cours du temps. Pour éviter ce problème de dérive, un algorithme de SLAM [Santos et al., 2013] basé sur les mesures laser et un modèle odométrique est utilisé, afin de reconstruire la trajectoire réalisée.

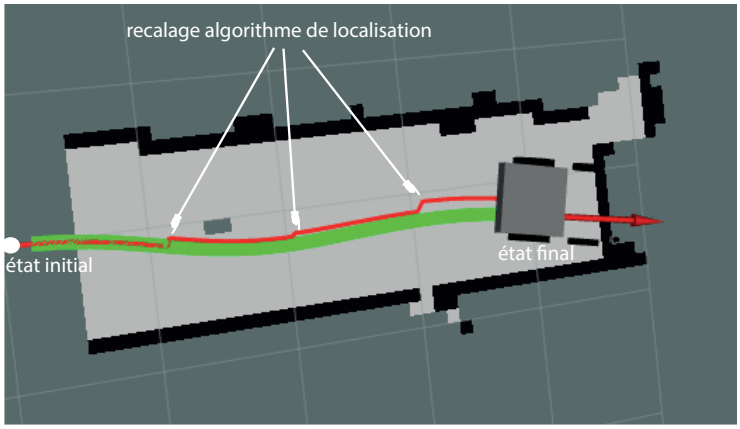


FIGURE 3.17 – Évolution dans un couloir : estimation de trajectoire avec les codeurs (vert), avec un algorithme de localisation en utilisant le laser (rouge)

Environnement non encombré L'environnement traité ici est un couloir se terminant par un petit virage (figure 3.18). Le fauteuil roulant suit correctement le squelette de l'espace libre perçu. La figure 3.19 montre l'évolution de l'erreur de la signature et la commande appliquée au système. Au cours des 15 premières secondes, on peut voir que ε_θ est presque égal à 0. Le changement brusque observé autour de l'instant $20s$, correspond une variation abrupte de la consigne en raison de la topologie de l'espace libre (un virage).

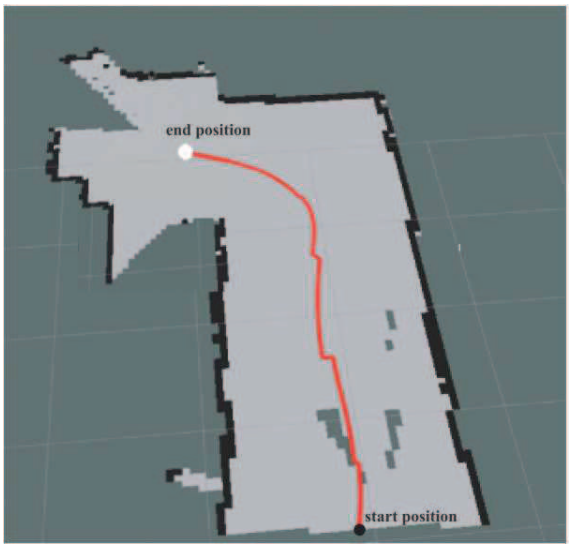


FIGURE 3.18 – Résultats expérimentaux de l'asservissement visuel basé sur le squelette utilisant un fauteuil roulant.

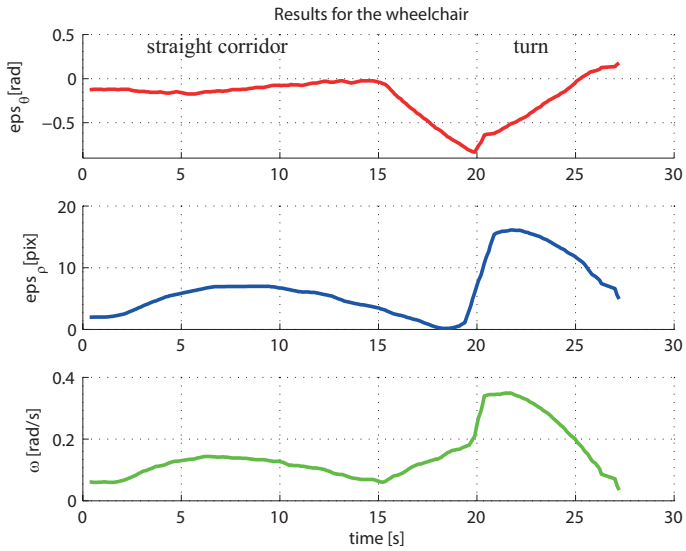


FIGURE 3.19 – Comportement du système. Haut : erreur angulaire ε_θ. Milieu : erreur latérale exprimée dans l'image ε_ρ. Bas : la commande ω appliquée au robot.

Environnement encombré La figure 3.20 montre l'évolution de l'erreur de la signature avec la commande appliquée. L'environnement au cours des premières 5 s, ressemble à un couloir, et l'erreur de la signature converge rapidement vers 0. Le robot rallie bien le DVG. La géométrie de l'environnement et donc la topologie du squelette sont fortement modifiés. On note cependant que le système converge le nouveau DVG (figure 3.21).

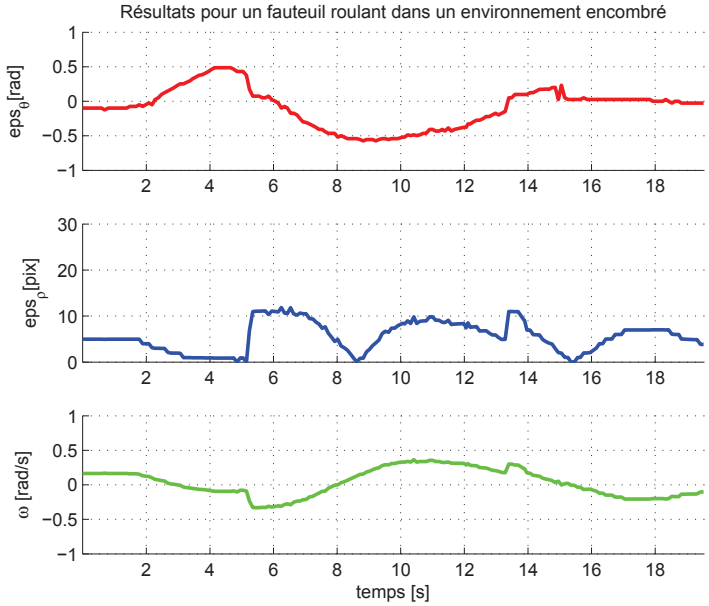


FIGURE 3.20 – Résultats pour le test sur un fauteuil roulant pour un environnement encombré.

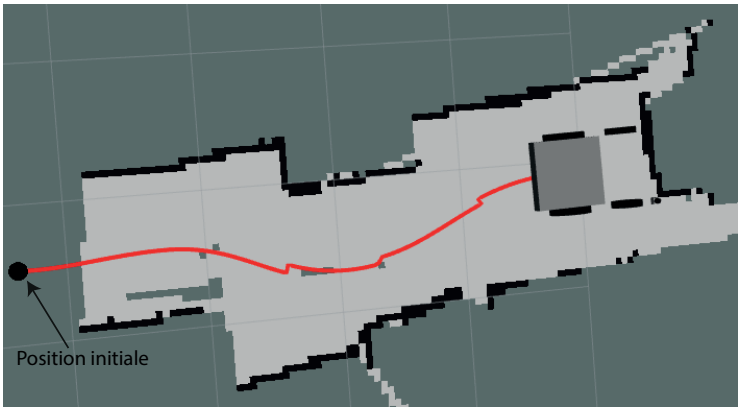


FIGURE 3.21 – Évolution du fauteuil roulant dans un environnement encombré

3.7 Conclusion

Dans ce chapitre, une nouvelle approche pour la navigation de robots mobiles dans des environnements inconnus a été introduite. Elle est basée sur une extraction en temps réel du DVG local. Un algorithme d'extraction du squelette adapté appliqué à l'espace libre local projeté sur un repère de perception virtuel unifié dans une version temps réel est embarqué. Un asservissement visuel est ensuite appliqué sur les signatures construites sur le squelette extrait. La convergence et la stabilité de la loi de commande ont été étudiées. La robustesse et l'efficacité de la commande proposée est prouvée. Elle permet pour un robot de naviguer en toute sécurité et suivre le squelette de l'espace libre d'une manière autonome. L'approche a été validée à l'aide de plusieurs scénarios, à la fois dans des environnements de simulation et du monde réel. Les résultats montrent que la commande conçue donne des résultats satisfaisants même dans des environnements complexes, dynamiques et très encombrés.

Dans ce chapitre nous avons utilisé une méthode de suivi de proche en proche du DVG local. Cependant avec cette approche la mesure change en permanence en fonction de la topologie de l'environnement. Ceci engendre une variation importante de la mesure qui peut impacter directement la commande appliquée au robot. Dans le chapitre suivant une nouvelle stratégie de mesure/commande sera présentée. Le suivi n'est plus au plus près du squelette mais le déplacement sera plus rapide et restera sûr et autonome avec une méthode de planification adaptée dans le référentiel virtuel.

Navigation avec prédiction du diagramme de Voronoï généralisé

Résumé du chapitre

Afin de réaliser une navigation sûre, au chapitre précédent, nous avons proposé une stratégie de mesure/commande pour un suivi local du squelette. Le présent chapitre propose une extension utilisant une prédiction de l'évolution du squelette. Pour ce faire, une fois un segment du squelette choisi, nous prédisons sa future position en utilisant un cadre probabiliste. Lorsqu'une mesure devient de nouveau disponible, nous sélections alors ce qui est le segment le plus probable sur cette mesure. Cette stratégie permet de minimiser les effets des bruits de mesure. Lorsqu'aucune mesure n'est disponible, nous pouvons calculer la commande en utilisant uniquement la boucle de prédiction. Dans la mesure où le segment choisi peut être éloigné du robot, une phase de planification est réalisée. Ainsi, lorsque le segment est choisi, nous procédons à l'application complète de la loi de commande sur un horizon donné afin de vérifier la compatibilité de la trajectoire générée avec l'environnement. Si cette trajectoire rencontre des obstacles, il est possible de la déformer en modifiant les paramètres de la loi de commande ou de sélectionner un nouveau segment du squelette.

Sommaire

4.1 Introduction

Le problème du suivi du squelette a été résolu dans le chapitre précédent en utilisant un asservissement visuel basé sur une approximation linéaire du squelette dans un cadre virtuel unifié. Ce cadre permet de regrouper toutes les informations de plusieurs types de capteurs. Par conséquent, l'extraction de l'espace libre, de la signature ainsi que le calcul de la loi de commande se fait dans le repère du capteur virtuel. Cette approche se base sur une sélection du point du squelette le plus proche par rapport au robot. Il s'agit de rallier à chaque fois l'approximation linéaire du squelette sur ce point avec la bonne orientation. Cependant, avec cette méthode, la mesure change souvent en fonction de la topologie et la géométrie des frontières des obstacles ce qui engendre une variation importante et fréquente de la mesure. Une autre limitation de l'approche de suivi du squelette de proche en proche est que seule l'approximation linéaire la plus proche au robot est utilisée, nous perdons ainsi l'autre partie extraite du squelette qui représente les observations futures de l'environnement.

La solution que nous proposons dans ce chapitre est illustrée dans la figure 4.1. Il s'agit de prédire la position du squelette dans des images futures à partir d'un modèle d'évolution, puis de fusionner avec ce que le robot voit réellement. Un filtre de Kalman est utilisé pour filtrer et fusionner les données d'une approximation linéaire du squelette. Cette approche de prédiction peut limiter le bruit de mesure et peut utiliser une partie du squelette extraite pour réaliser la fusion.

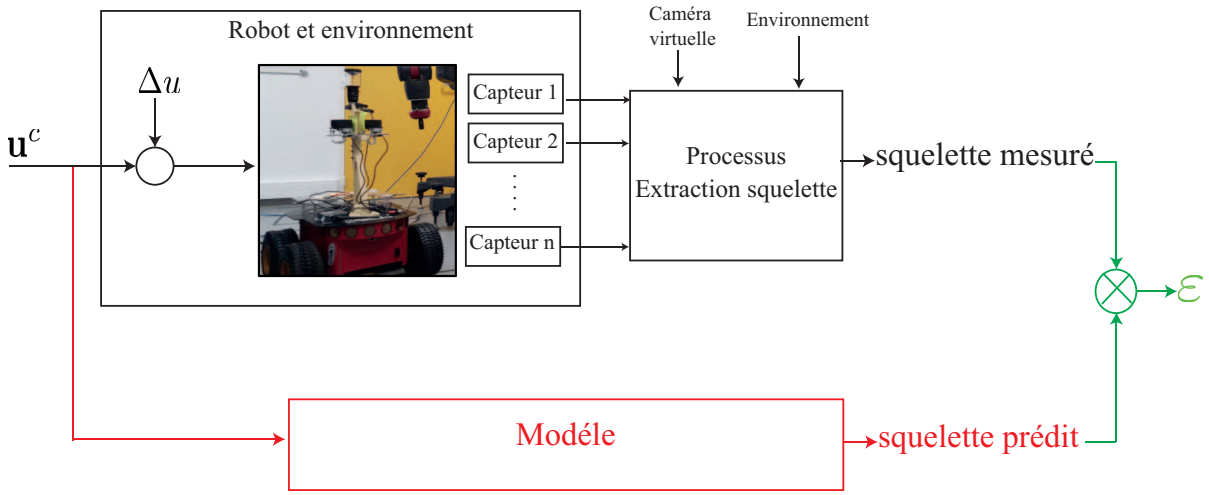


FIGURE 4.1 – Schéma bloc utilisé pour l'asservissement avec la prédiction. Le fait que la consigne ne sera pas exactement appliquée sur le robot (par exemple du fait du glissement des roues) sera modélisé par une perturbation Δu inconnue du modèle. $\mathbf{u}^c$ sont les vitesses de consignes calculées à partir de (3.17).

La première étape de cette approche est la sélection d'un point sur le squelette. Dans un second temps, on réalise le calcul de l'approximation linéaire autour de ce point (cela constitue la condition initiale du filtre). Une fois le point choisi, une étape de prédiction de la position du point dans la prochaine image à partir d'un modèle d'évolution sera réalisée. Par conséquent, lorsque la prédiction du point ne se situe pas sur le squelette, une correction est effectuée à partir de la mesure pour sélectionner le point le plus probable qui appartient au DVG. On calcule ensuite l'approximation linéaire autour de ce point et on applique la loi de commande (3.17). Les nouvelles observations constituent l'entrée

de mesure pour un filtrage de Kalman qui permet d'atténuer le bruit de perception et les erreurs de modélisation au niveau de l'application de la commande. Comme le point choisi à la première étape peut être éloignée du robot, la commande appliquée pourrait induire une trajectoire du robot non compatible avec l'environnement. Pour résoudre ce problème, une approche de planification est réalisée.

Pour résumer, ce chapitre aborde premièrement les problèmes issus de la perception et de l'action pour le suivi du squelette. Dans un second temps, nous présentons la prédiction dans l'image, en exposant des solutions existantes dans la littérature. Notre proposition d'approche pour un suivi de squelette utilisant la prédiction et une planification sera ensuite détaillée. Enfin, des résultats de simulation et expérimentaux sont présentés.

4.2 Sensibilité du DVG aux erreurs de modélisation

La loi de commande proposée au chapitre précédent (3.17) définit des vitesses de consigne pour le robot. Ces vitesses sont calculées sur une hypothèse de roulement sans glissement des roues. Dans la réalité, les roues glissent et il y a un écart entre les vitesses de consigne et les vitesses réelles du robot. De ce fait, nous pouvons modéliser ou interpréter cet écart comme une perturbation sur la commande du modèle Δu. Lorsque nous alimentons nos modèles avec les vitesses de consigne, du fait de cet écart de commande, les sorties du modèle et les sorties constatées seront différentes. Par conséquent, le squelette perçu par le robot sera différent par rapport à la réalité (figure 4.2). Le bruit de perception peut impacter aussi le fonctionnement souhaité de la loi de commande.

4.2.1 Bruit de perception

Le premier problème du suivi du squelette de proche en proche est la discontinuité de la mesure. Ceci s'explique par un changement permanent du squelette particulièrement dans des environnements de géométrie complexe ou dynamique. L'évolution rapide de la mesure peut provoquer une incohérence locale sur le squelette extrait. Par conséquent, le point le plus proche du robot choisi risque d'être perdu (cf. (3.4.1)). Le deuxième problème est lié aux bruits de mesure qui sont issus de plusieurs sources :

- défaillance de capteur,
- mesure de capteur erronée,
- erreur de segmentation de l'espace libre,
- bruit sur le calcul du squelette (branches parasites),
- discrétisation de la mesure,
- sélection du point le plus proche et calcul de l'approximation linéaire.

Une partie du deuxième problème a été résolue dans la littérature en filtrant les mesures à l'aide d'un filtre de Kalman [Bulut et al., 2011] ou d'autres techniques de prédiction [Saha et Gustafsson, 2012], [Prasov et Khalil, 2013].

4.2.2 Bruit de commande

Les difficultés de la perception ne reposent pas uniquement sur les capteurs mais également sur la chaîne d'action. Pour la modélisation d'un robot mobile, nous avons pris l'hypothèse du roulement

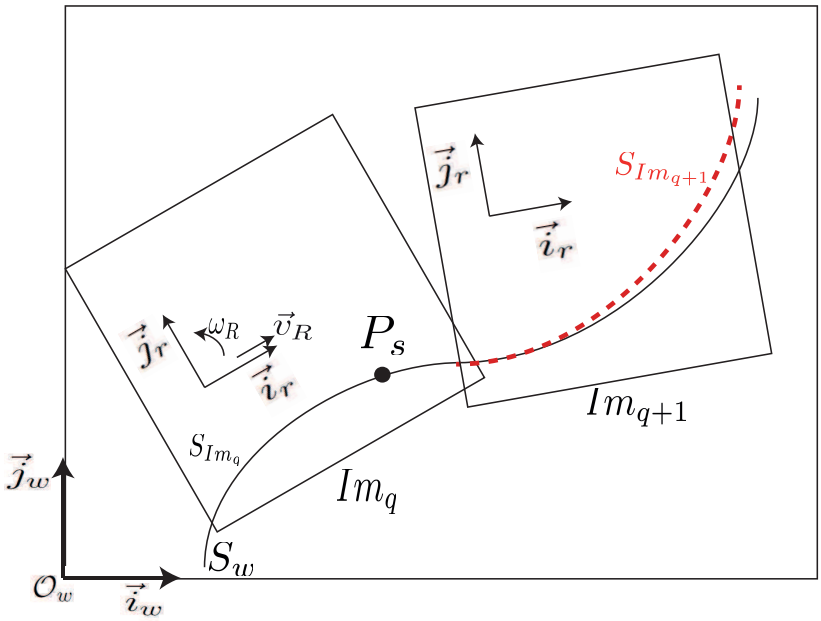


FIGURE 4.2 – Squelette réel et squelette perçu par le robot : lorsque le robot bouge dans un environnement dynamique le squelette change en permanence. Ceci engendre une différence entre ce qui était perçu à q et à q+1. Le tracé en pointillé correspond au squelette perçu par le robot, et le tracé continu correspond au squelette prédit par un modèle.

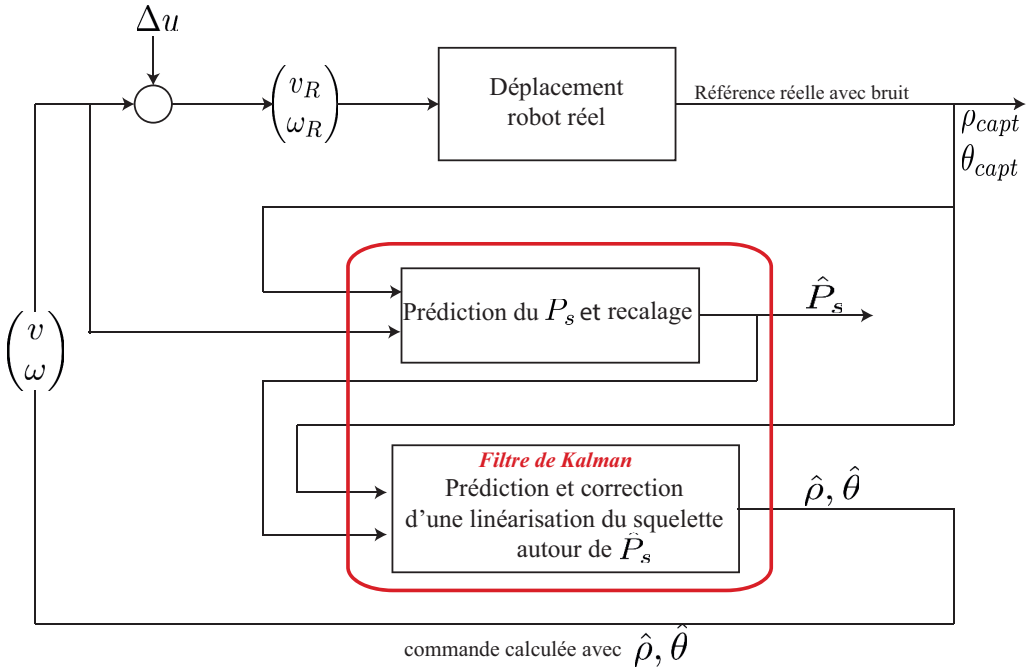


FIGURE 4.3 – Schéma bloc utilisé pour l'asservissement avec la prédiction, Δu correspond à une perturbation au niveau de la commande (par exemple le glissement des roues). P_s et $\hat{P}_s$ correspondent respectivement au point choisi sur le squelette et sa prédiction pour la mesure suivante. $\Delta(\rho, \theta)$ et $\hat{\Delta}(\hat{\rho}, \hat{\theta})$ sont les approximations linéaires du squelette réelle et estimée en un point donné. (v, ω) est la commande calculée avec l'asservissement visuel sur la prédiction. (v_R, ω_R) est la commande réelle envoyée au robot.

sans glissement. La vitesse relative de la roue au point de contact par rapport au sol est supposée nulle. Les hypothèses prises pour cette condition sont :

- Le sol est parfaitement plat
- Le contact entre la roue et le sol est ponctuel
- Les roues sont indéformables et de rayon constant.

Cependant, en pratique, le contact roue/sol n'est pas ponctuel mais surfacique, ce qui provoque de légers glissements. De même, l'hypothèse supposant que les roues sont indéformables est discutable pour des roues équipées de pneus [Bayle, 2008].

La conséquence est que les consignes de vitesse que nous envoyons aux actionneurs ne sont en pratique jamais instantanément appliquées et suivies par le robot.

Le défaut peut se trouver aussi bien au niveau de l'actionneur lui-même que de sa carte de commande. [Saied et al., 2015] propose une technique de détection de défaillance des actionneurs. Dans [Yang et Kim, 1999], le bruit dans les signaux de commande est pris en considération. Enfin, [Lhomme-Desages, 2008] proposent un calcul de glissement des roues d'un robot mobile évoluant dans un milieu naturel. Nous choisissons de modéliser le bruit de commande comme une variable aléatoire s'ajoutant aux consignes de vitesses envoyées. Ainsi, différentes sources de bruits seront modélisées.

Suite aux erreurs de perception et l'application inexacte des consignes, le squelette peut être perçu différemment par rapport à la réalité (figure 4.2). Une solution à ce problème peut être de prédire l'évolution du squelette au cours du temps. Dans le chapitre 2, nous avions conçu une stratégie de commande basée sur une combinaison modèle/observateur pour prédire et filtrer les paramètres visuels et trouver une solution lorsque les paramètres visuels sont perdus. Il s'agissait de naviguer dans un couloir, la médiane (le squelette vu dans l'image) était immobile. Dans les situations où le squelette bouge significativement, l'introduction de la prédiction de la position future du squelette peut être intéressante. Sur les principes du filtre de Kalman, nous proposons de prédire où devraient se trouver les points du squelette dans chaque image et de le comparer à un squelette réel lorsque nous le mesurons. Nous réaliserons alors une fusion entre ce qui a été prédit et ce que le robot voit réellement.

4.3 Prédiction de la future perception du squelette

Pour prédire le squelette, nous avons deux informations : un modèle d'évolution d'un point sur le squelette et un modèle d'évolution d'une approximation linéaire locale du squelette qui sera notée $(\theta_{mes}, \rho_{mes})$. Nous allons prédire, dans un premier temps, le point sur le squelette en utilisant le modèle d'évolution. Puis, dans un second temps, nous corrigerons ces variables avec ce que percevra le robot. Comme le modèle déterministe est limité, nous allons enrichir la représentation en utilisant des variables aléatoires afin de modéliser et prendre en compte les bruits et les incertitudes. Ceci correspond également au contexte utilisé dans le cadre du filtre de Kalman.

Notations

Soit $\mathbf{X}$ une variable aléatoire, et $\hat{\mathbf{X}}$ l'estimation de $\mathbf{X}$.
La variable aléatoire $\mathbf{X} \in \mathbb{R}$ suit une loi normale avec la moyenne μ et une variance σ^2, $\mathbf{X} \sim \mathcal{N}(\mu, \sigma^2)$.

Lorsque $\mathbf{X} \in \mathbb{R}^n$ avec une moyenne $\overline{\mathbf{X}}$ et la matrice de variance covariance $Cov_{\mathbf{X}}$, alors nous notons $\mathbf{X} \sim \mathcal{N}(\overline{\mathbf{X}}, Cov_{\mathbf{X}})$.

Soit $\mathbf{Y}$ le résultat d'une transformation linéaire, $\mathbf{Y} = \mathbf{C}\mathbf{X}$, alors $\mathbf{Y}$ suit la loi normale $\mathcal{N}(\mathbf{C}\,\overline{\mathbf{X}}, \mathbf{C}\,Cov_{\mathbf{X}}\,\mathbf{C}^T)$.

4.3.1 Prédiction d'un point sur le squelette

Soit P_s un point exprimé dans le repère du capteur virtuel (coordonnées métriques), on note P_{sI} sa projection dans l'image exprimée en pixels. Pour extraire le modèle d'évolution dans l'image d'un point P_{sI} sur le squelette, il faut écrire la vitesse du point P_s dans le repère du capteur virtuel puis l'exprimer dans le repère image en déduisant l'évolution de P_{sI}.

$$P_{sI} = K P_s, \text{ avec } K = \begin{pmatrix} \alpha/H & 0 & u_0 \\ 0 & \alpha/H & v_0 \\ 0 & 0 & 1 \end{pmatrix}$$

Le squelette est fixe (si l'environnement est statique) dans le référentiel global, donc l'expression de la vitesse du point P_s, $\overrightarrow{V}_{P_s/\mathcal{F}_C}$ dans le référentiel virtuel se déduit par :

$$\overrightarrow{V}_{P_s/\mathcal{F}_C} = \left[\frac{d}{dt}\overrightarrow{O_C P_s}\right]_{\mathcal{F}_C} = \left[\frac{d}{dt}\overrightarrow{O_C P_s}\right]_{\mathcal{F}_W} + \overrightarrow{\Omega}_{\mathcal{F}_W/\mathcal{F}_C} \wedge \overrightarrow{O_C P_s} \tag{4.1}$$

$$= \begin{pmatrix} -\omega(L + y_{P_s}) \\ v + \omega(W + x_{P_s}) \\ 0 \end{pmatrix}_{\mathcal{F}_C} \tag{4.2}$$

Avec W et L désignent la position du capteur virtuel relativement au robot.

Ceci permet d'écrire le modèle d'évolution du point P_s comme :

$$\begin{cases} \dot{x}_{Ps} = -\omega(L + y_{P_s}) \\ \dot{y}_{Ps} = v + \omega(W + x_{P_s}) \end{cases} \tag{4.3}$$

Pour la prédiction du point P_s appartenant au squelette, l'état est constitué des coordonnées de ce point dans le repère du capteur virtuel $\mathbf{X} = (x_{Ps}, y_{Ps})^T = (x_{1P}, x_{2P})^T$. Les vitesses linéaire et angulaire du robot représentent les variables de commande $\mathbf{U} = (v, \omega)^T = (u_1, u_2)^T$.

En notant T_e la période d'échantillonnage, l'évolution de l'état et l'équation d'observation s'écrivent :

$$\begin{cases} x_{1P}^{k+1} = x_{1P}^k + T_e f_1\left(\mathbf{X}^k, \mathbf{U}^k\right) \\ x_{2P}^{k+1} = x_{2P}^k + T_e f_2\left(\mathbf{X}^k, \mathbf{U}^k\right) \end{cases} \tag{4.4}$$

avec

$$\begin{aligned} f_1\left(\mathbf{X}^k, \mathbf{U}^k\right) &= -u_2^k\left(L + x_{2P}^k\right) \\ f_2\left(\mathbf{X}^k, \mathbf{U}^k\right) &= u_1^k + u_2^k\left(W + x_{1P}^k\right) \end{aligned} \tag{4.5}$$

La prédiction déterministe de ce point n'est pas suffisante. En effet, de nombreuses incertitudes existent. Nous allons les modéliser et enrichir le modèle par un contexte gaussien. Ainsi, nous serons

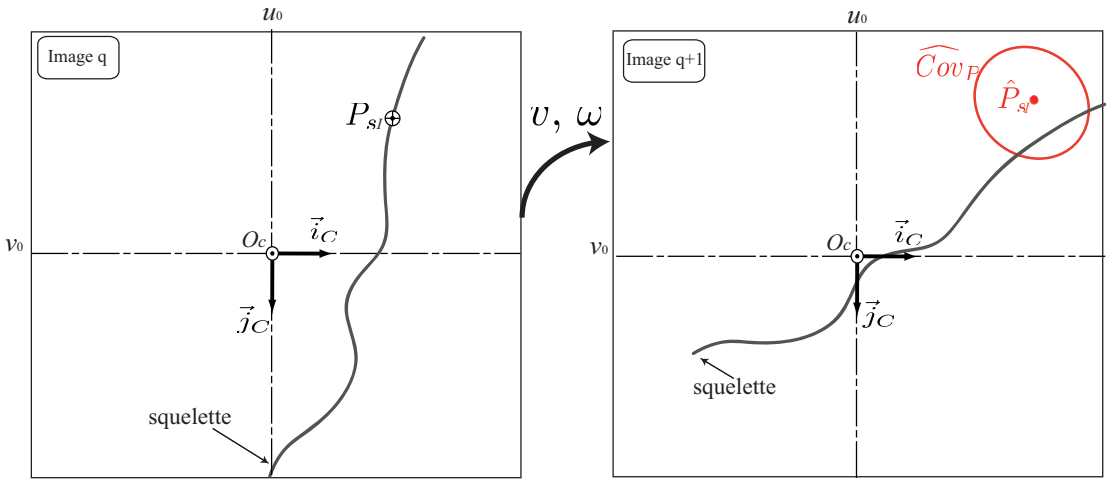


FIGURE 4.4 – Illustration de la prédiction d'un point P_{sI} dans l'image sur le squelette et son ellipse d'incertitude

capable de propager les incertitudes à travers le modèle et nous obtiendrons une estimation de la position du point suivi avec une enveloppe d'incertitude comme illustré sur la figure 4.4. Nous supposons donc que les deux variables x_{Ps} et y_{Ps} sont des variables gaussiennes aléatoires indépendantes. Le modèle discret est constitué de l'équation d'évolution et l'équation d'observation :

$$\begin{cases} \mathbf{X}_P^{k+1} = \left(x_{1P}^{k+1}, x_{2P}^{k+1}\right)^T = F_P^k\,\mathbf{X}_P^k + G_P^k\,\mathbf{U}_k + b_x^k \\ \mathbf{Y}_P^{k+1} = \mathbf{X}_P^k + b_y^k \end{cases} \tag{4.6}$$

Le bruit d'état b_x^k et le bruit de mesure b_y^k sont supposés blancs, indépendants et gaussiens à valeur moyenne nulle (centrés) et de matrices de variance-covariance $\mathbf{B_x}$ et $\mathbf{B_y}$ $\left(b_x^k \sim \mathcal{N}(0, \mathbf{B_x}),\, b_y^k \sim \mathcal{N}(0, \mathbf{B_y})\right)$.

Le bruit d'état b_x^k est l'imprécision sur la prédiction de la position du point P_s par le modèle. Il inclura le bruit au niveau de l'extraction du point P_s et l'effet des incertitudes sur les commandes appliquées. Le bruit de mesure b_y^k correspond à la dispersion donnée par la mesure.

En utilisant (4.6) et (4.4) les matrices du modèle sont :

$$F_P^k = \begin{pmatrix} \dfrac{\partial f_1}{\partial x_{1P}^k} & \dfrac{\partial f_1}{\partial x_{2P}^k} \\ \dfrac{\partial f_2}{\partial x_{1P}^k} & \dfrac{\partial f_2}{\partial x_{2P}^k} \end{pmatrix} = \begin{pmatrix} 1 & -T_e\,u_2^k \\ T_e\,u_2^k & 1 \end{pmatrix} \tag{4.7}$$

$$G_P^k = \begin{pmatrix} \dfrac{\partial f_1}{\partial u_1^k} & \dfrac{\partial f_1}{\partial u_2^k} \\ \dfrac{\partial f_2}{\partial u_1^k} & \dfrac{\partial f_2}{\partial u_2^k} \end{pmatrix} = \begin{pmatrix} 0 & -T_e(L + x_{2P}^k) \\ T_e & T_e(W + x_{1P}^k) \end{pmatrix} \tag{4.8}$$

Nous supposons que la commande $\mathbf{U}^k$ envoyée au robot est : $\mathbf{U}^k = \overline{\mathbf{U}}^k + b_{\mathbf{U}^k}$, la position du point P_s s'écrit comme : $\mathbf{X}^k = \overline{\mathbf{X}}^k + b_{\mathbf{X}^k}$. Les bruits $b_{\mathbf{U}^k}$ et $b_{\mathbf{X}^k}$ sont des bruits blancs, indépendants et gaussiens

associés au modèle discret : ($b_{\mathbf{U}^k} \sim \mathcal{N}(0,\, Cov_U)$, $b_{\mathbf{X}^k} \sim \mathcal{N}(0,\, Cov_P)$) ..

En utilisant toutes ces informations, la propagation de l'incertitude de l'état est :

$$\widehat{Cov}_P^{\,k} = F_P^k \, Cov_P \, F_P^{kT} + G_P^k \, Cov_U \, G_P^{kT} \tag{4.9}$$

La matrice de variance-covariance Cov_U est :

$$Cov_U = \begin{pmatrix} \sigma_v^2 & 0 \\ 0 & \sigma_\omega^2 \end{pmatrix} \tag{4.10}$$

A la fin de cette étape, nous avons donc la prédiction $\hat{P}_s = (x_{1P}^{k+1}, x_{2P}^{k+1})^T$ du point P_s dans le repère du capteur virtuel et sa matrice de variance-covariance $\widehat{Cov}_P^{\,k}$. Par conséquent en définissant la matrice des paramètres intrinsèques K, on peut déduire la prédiction $\hat{P}_{sI}$ dans l'image et sa matrice de variance-covariance $\widehat{Cov}_{P_{sI}}^{\,k}$ (figure 4.4).

4.3.2 Prédiction d'une approximation linéaire autour du squelette

Suite au mouvement du robot, le point $\hat{P}_{sI}$ prédit dans l'image a peu de chance de se trouver sur la nouvelle mesure du squelette. Il faut alors chercher le point P_{sI}^m du squelette qui sera le plus probable. Pour chacune des mesures, nous pouvons également associer une incertitude représentée par une matrice de variance covariance $Cov_{P_{sI}^m}$.

La loi de commande (3.17) utilise les paramètres de l'approximation linéaire du squelette, il faut reconstruire la linéarisation dans l'image $(\rho_{mes}, \theta_{mes})$ à partir des données de $\hat{P}_{sI}$, P_{sI}^m, $\widehat{Cov}_{P_{sI}}$ et $Cov_{P_{sI}^m}$.

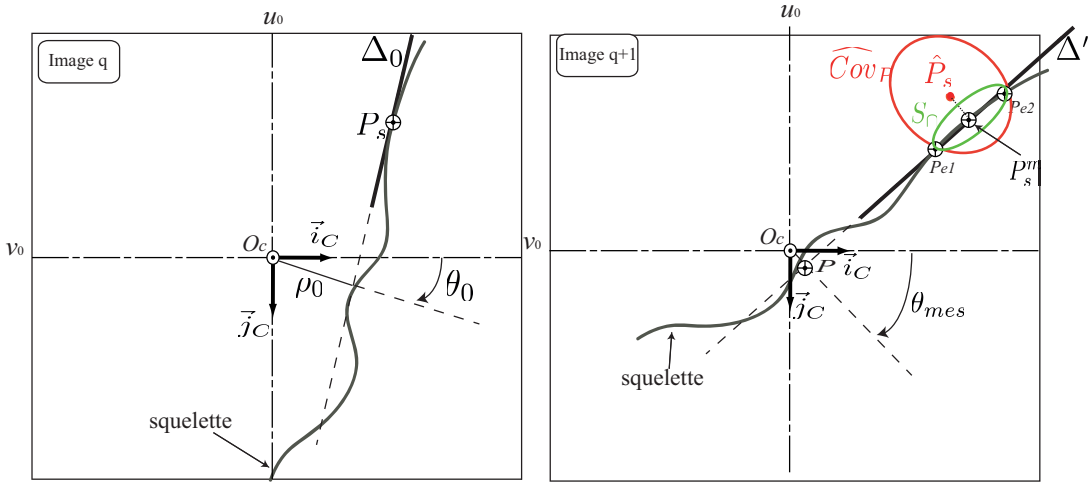
4.3.2.1 Reconstruction des paramètres de la droite cible

A partir d'une matrice de variance covariance, il est possible de définir un ellipsoïde d'incertitude. La taille de cet ellipsoïde est fonction de l'incertitude de la variable. Si nous prenons un seuil de 3σ de densité de probabilité, on peut construire un ellipsoïde d'incertitude avec un intervalle de confiance de 99%. Dans notre cas, nous traitons le cas 2D, donc l'ellipsoïde est une ellipse. Une fois l'ellipse d'incertitude déterminée, nous calculons les points d'intersection P_{e1} et P_{e2} entre l'ellipse du point $\hat{P}_{sI}$ et le squelette S dans l'image. La nouvelle mesure du point sur le squelette est le point P_{sI}^m le plus proche de $\hat{P}_{sI}$ appartenant au squelette. On peut le définir à partir de l'ensemble de points dans l'image appartenant au squelette $\mathcal{X}_{S_\cap}$. Ces points sont placés à l'intérieur de l'ellipse représentant l'ensemble des tirages induits par la matrice de variance/covariance $\widehat{Cov}_{P_{sI}}$.

$$P_{sI}^m \in \mathcal{X}_{S_\cap} \text{ tel que, pour tout les points } \mathbf{x} \text{ du squelette } \mathcal{X}_{S_\cap}, d_E(P_{sI}^m, \hat{P}_{sI}) \leq d_E(\mathbf{x}, \hat{P}_{sI}) \tag{4.11}$$

avec d_E la distance euclidienne. Si nous n'avons pas d'intersection, nous considérons qu'il n'y a pas de correspondance entre la prédiction et la mesure. Il faut alors procéder à la sélection d'un nouveau point P_s et recommencer la procédure.

Ayant calculé le point le plus probable, les paramètres de la linéarisation sont calculés à partir des pixels inclus dans l'ellipse d'incertitude. Une droite Δ' est alors obtenue par une régression linéaire.



(a) L'image à l'instant q (b) L'image à l'instant $q+1$

FIGURE 4.5 – Dans la première image qui correspond à une prise de vue à l'instant q, un point sur le squelette est choisi, l'approximation linéaire autour de ce point est calculée et les paramètres (ρ_0, θ_0) de la droite Δ_0 sont extraits. A l'instant $q+1$ une reconstruction de la mesure $(\rho_m,\ \theta_m)$ à partir de la prédiction $\hat{P}_{sI}$ du P_{sI} est réalisée sur la droite Δ'.

Le résultat est illustré sur la figure 4.5. Elle représente alors la prédiction de la droite suivie, la plus probable compte-tenu des incertitudes.

Pour extraire les paramètres servant à la commande, il faut alors projeter le centre de l'image $\mathcal{O}_C$ sur la droite extraite pour obtenir un point $P = (u_p,\ v_p)$. Ce dernier permet de calculer les paramètres $(\rho_{capt},\ \theta_{capt})$ de la droite Δ'. Les indices *capt* sont utilisés ici car la procédure décrite ci-dessus sert de capteur pour obtenir une mesure du segment du squelette suivi.

$$\begin{cases} \rho_{capt} = \sqrt{(u_p - u_0)^2 + (v_p - v_0)^2} \\ \theta_{capt} = atan2\left((v_p - v_0), (u_p - u_0)\right) \end{cases} \tag{4.12}$$

L'estimation étant terminée, pour la prédiction de la position suivante du point suivi, nous réinitialisons P_{sI}^m auquel nous affectons une ellipse $Cov_{P_{sI}^m}$. Cette matrice de variance/covariance sera définie à partir d'une ellipse passant par les deux points d'intersection P_{e1} et P_{e2} et centrée sur P_{sI}^m. Elle sélectionnera une partie du squelette telle que $S_\cap$ telle que $S_\cap = \widehat{Cov}_{P_{sI}} \cap S$ (figure 4.5).

4.3.2.2 Filtre de Kalman sur les paramètres de la droite cible

Comme nous avons une information supplémentaire de l'évolution de l'approximation linéaire (3.18), nous pouvons concevoir un filtre de Kalman sur les caractéristiques de la signature $(\rho,\ \theta)$. L'état correspond à $\mathbf{X}_{\rho\theta} = (\theta,\ \rho) = (x_1,\ x_2)$

$$\begin{cases} \dot{\mathbf{X}}_{\rho\theta} = \mathbf{\Phi}\left(\mathbf{X}_{\rho\theta}, \mathbf{U}\right) + \mathbf{b}_{\rho\theta} \\ \mathbf{Y}_{\rho\theta} = \begin{pmatrix} \theta_{capt} \\ \rho_{capt} \end{pmatrix} = \mathbf{h}\left(\theta, \rho\right) + \mathbf{b}_{Y_{\rho\theta}} \end{cases} \tag{4.13}$$

Avec $\mathbf{\Phi} = (\Phi_1,\, \Phi_2)^T$ et $\mathbf{h} = (h_1,\, h_2)^T$

$$\begin{aligned} \Phi_1\left(\mathbf{X}_{\rho\theta}, \mathbf{U}\right) &= u_2 \\ \Phi_2\left(\mathbf{X}_{\rho\theta}, \mathbf{U}\right) &= u_1\,\alpha\frac{\sin(x_1)}{H} + \left(\frac{L}{H}\cos(x_1) - \frac{W}{H}\sin(x_1)\right) u_2\,\alpha \end{aligned} \tag{4.14}$$

$\mathbf{b}_{\rho\theta} = \left(b_\rho^k\; b_\theta^k\right)^T \sim \mathcal{N}(0,\, \mathbf{B}_{\rho\theta})$ englobe les incertitudes sur les paramètres sur l'état et sur les variables de commande $\mathbf{U} = (v,\, \omega)^T$, $\mathbf{b}_{Y_{\rho\theta}} \sim \mathcal{N}(0,\, \mathbf{B}_{Y_{\rho\theta}})$ correspond à l'incertitude sur la mesure $(\rho_{capt},\, \theta_{capt})$ et α, correspond au nombre de pixels par mètre dans l'image.

Le processus de la mesure fait que les sorties des capteurs θ_{capt} et ρ_{capt} sont liées à l'état par :

$$h_2\left(\theta, \rho\right) =\mid \rho \mid \tag{4.15}$$

Le modèle (3.18) peut donner un ρ négatif. Par la suite, l'équation d'observation permet d'imposer toujours un ρ positif puisque cette variable est une distance.

$$h_1\left(\theta, \rho\right) = \begin{cases} \theta + \pi \text{ si } \rho_{capt} < 0 \\ \theta - 2\pi \text{ si } \theta_{capt} > \pi \\ \theta + 2\pi \text{ si } \theta_{capt} < -\pi \\ \theta \text{ ailleurs} \end{cases} \tag{4.16}$$

En discret, le modèle devient

$$\begin{cases} \mathbf{X}_{\rho\theta}^{k+1} = F_{\rho\theta}^k\, \mathbf{X}_{\rho\theta}^k + G_{\rho\theta}^k\, \mathbf{U}^k + \mathbf{b}_{\rho\theta}^k \\ \mathbf{Y}_{\rho\theta}^{k+1} = \begin{pmatrix} \hat{\theta}_{capt} \\ \hat{\rho}_{capt} \end{pmatrix} = \mathbf{U}_{\rho\theta}^k + \mathbf{b}_{Y_{\rho\theta}}^k \end{cases} \tag{4.17}$$

Les matrices du modèle $F_{\rho\theta}^k$ et $G_{\rho\theta}^k$ sont :

$$F_{\rho\theta}^k = \begin{pmatrix} \dfrac{\partial\Phi_1}{\partial x_1^k} & \dfrac{\partial\Phi_1}{\partial x_2^k} \\ \dfrac{\partial\Phi_2}{\partial x_1^k} & \dfrac{\partial\Phi_2}{\partial x_2^k} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \left(u_1\,\alpha\dfrac{\cos(x_1^k)}{H} + \left(-\dfrac{L}{H}\sin(x_1^k) - \dfrac{W}{H}\cos(x_1^k)\right) u_2^k\,\alpha\right) T_e & 1 \end{pmatrix} \tag{4.18}$$

$$G_{\rho\theta}^k = \begin{pmatrix} \dfrac{\partial\Phi_1}{\partial u_1^k} & \dfrac{\partial\Phi_1}{\partial u_2^k} \\ \dfrac{\partial\Phi_2}{\partial u_1^k} & \dfrac{\partial\Phi_2}{\partial u_2^k} \end{pmatrix} = \begin{pmatrix} 0 & T_e\alpha \\ T_e\alpha\dfrac{\sin(x_1^k)}{H} & \alpha\dfrac{T_e}{H}\left(L\cos(x_1^k) - W\sin(x_1^k)\right) \end{pmatrix} \tag{4.19}$$

La propagation de l'incertitude de la mesure $(\rho_{capt}, \theta_{capt})$ est

$$\widehat{Cov}_{\rho\theta} = F_{\rho\theta}^k\, Cov_{\rho\theta}\, F_{\rho\theta}^{kT} + G_{\rho\theta}^k\, Cov_U\, G_{\rho\theta}^{kT}; \tag{4.20}$$

avec

$$Cov_{\rho\theta} = \begin{pmatrix} \sigma_\theta^2 & 0 \\ 0 & \sigma_\rho^2 \end{pmatrix} \tag{4.21}$$

4.3.3 Étape de correction et mise à jour

A l'instant k, une mesure $\mathbf{Y}_{\rho\theta}^{k} = (\theta_{capt}\, \rho_{capt})^T$ par (4.12) est disponible et correspond à la linéarisation du squelette au point P_k^s. Le modèle précédent a permis d'obtenir une prédiction $\hat{\mathbf{Y}}_{\rho\theta}^{k|k-1}$ de ces paramètres. Nous allons procéder à la fusion de ces deux informations. On évalue l'erreur entre la mesure prédite par le modèle et celle obtenue par le procédé précédent

$$\varepsilon^k = \mathbf{Y}_{\rho\theta}^{k} - \hat{\mathbf{Y}}_{\rho\theta}^{k|k-1} \tag{4.22}$$

La matrice de covariance de cette erreur de mesure $\mathbf{S}^{k|k-1}$

$$\mathbf{S}^{k|k-1} = \widehat{Cov}_{\rho\theta} + \mathbf{B}_{Y_{\rho\theta}} \tag{4.23}$$

Avec la covariance de l'erreur de mesure $\mathbf{S}^{k|k-1}$, nous pouvons tester si la mesure $\mathbf{Y}_{\rho\theta}^{k}$ et le modèle sont compatibles avec le test suivant :

$$(\varepsilon^k)^T \mathbf{S}^{k|k-1}(\varepsilon_k) < \chi^2 \tag{4.24}$$

avec χ est un scalaire qui définit un seuil multidimensionnel.

Si le modèle et la mesure sont compatibles, on corrige l'estimation de l'état par l'innovation :

a) Calcul du gain de correction

$$K^k = \widehat{Cov}_{\rho\theta}(S^{k|k-1})^{-1} \tag{4.25}$$

b) Correction de la prédiction

$$\mathbf{X}_{\rho\theta}^{k|k} = \mathbf{X}_{\rho\theta}^{k|k-1} + K^k \varepsilon^k \tag{4.26}$$

c) Évaluation de la nouvelle matrice de variance covariance de l'état

$$\widehat{Cov}_{\rho\theta}^{k|k} = \widehat{Cov}_{\rho\theta}^{k|k-1} - K^k \widehat{Cov}_{\rho\theta}^{k|k-1} \tag{4.27}$$

L'algorithme de l'approche de prédiction du point sur le squelette et de la mesure est détaillé ci dessous. Il présente une vue d'ensemble de l'approche proposée du filtrage.

Algorithme : Prédiction de la position d'un point du squelette dans le repère du capteur et estimation du linéarisé du squelette en ce point

Entrées : Squelette S dans une image binaire
Sorties : Prédiction d'un point sur le squelette et filtrage de $(\theta_{capt}, \rho_{capt})$
Étape 1 - Initialisation :
Choix d'un point sur le squelette P_s
Calcul de la linéarisation autour de P_s, $\Delta_0 = (\theta_0, \rho_0)$
Définir Cov_{P0}, $Cov_{\theta\rho 0}$, Cov_U
$\mathbf{X}_P^k = P_s$
$\mathbf{X}_{\rho\theta}^k = (\theta_0, \rho_0)^T$
$Cov_P^k = Cov_{P0}$, $Cov_{\theta\rho}^k = Cov_{\theta\rho 0}$
Pour chaque instant de calcul **faire**
  Étape 2 - Prédiction de la position du point :
  $\mathbf{X}_P^{k+1} = \mathbf{X}_P^k + T_e\,\mathbf{f}$ (4.4)
  $Cov_P^{k+1} = F_P^k\, Cov_P^k\, F_P^{kT} + G_P^k\, Cov_u\, G_P^{kT}$
  Étape 3 - Sélection du point le plus probable (pour une image) :
  Si $(Cov_P^{k+1} \cap S \neq \emptyset)$ **Alors**
    Calcul $(P_{e1}, P_{e2}) = Cov_P^{k+1} \cap S$
    Recherche de P_s^m : le point le plus proche de $\mathbf{X}_P^{k+1}$ et qui appartient à $S_\cap$
    Calcul de la régression linéaire Δ' de $S_\cap$ qui passe par P_s^m
    Reconstruction de la mesure $(\theta_{capt}, \rho_{capt})$
    Algorithme : Filtrage de $(\theta_{capt}, \rho_{capt}) \rightarrow (\hat{\theta}, \hat{\rho})$
    $\mathbf{X}_P^{k+1} = P_s^m$
    $\mathbf{X}_{\rho\theta}^{k+1} = (\hat{\theta}, \hat{\rho})^T$
  Sinon
    Revenir à **Étape 1**
  Fin Si
  Étape 4 - Mise à jour :
  $\mathbf{X}_P^k = \mathbf{X}_P^{k+1}$
  $\mathbf{X}_{\rho\theta}^k = \mathbf{X}_{\rho\theta}^{k+1}$
  $Cov_P^k = Cov_P^{k+1}$
  $Cov_{\theta\rho}^k = Cov_{\theta\rho}^{k+1}$
Fin Pour

Algorithme : Filtrage de Kalman pour obtenir ce qui servira de mesure à la commande du robot. La sortie délivre les paramètre du linéarisé du squelette suivi (θ_{capt}, ρ_{capt})

Conditions initiales :
$\Delta_0 = (\theta_0, \rho_0)$: un linéarisé en un point du squelette
$Cov_{\theta\rho 0}$: une incertitude sur les paramètres

Étape de prédiction à l'instant k

1. Prédiction a priori de l'état à partir de $\hat{X}_{\rho\theta}^{k-1|k-1}$ en utilisant l'évolution de l'état (4.17). On obtient $\hat{X}_{\rho\theta}^{k|k-1}$
2. Prédiction de la matrice de covariance $\widehat{Cov}_{\rho\theta}^{k|k-1}$ en utilisant (4.20).

Étape d'innovation

3. On réalise une mesure $\mathbf{Y}_{\rho\theta}^{k}$ (4.12)
4. Calcul de l'innovation $\varepsilon^k = \mathbf{Y}_{\rho\theta}^{k} - \hat{\mathbf{Y}}_{\rho\theta}^{k|k-1}$
5. Calcul de la covariance de l'innovation $\mathbf{S}^{k|k-1} = \widehat{Cov}_{\rho\theta}^{k|k-1} + \mathbf{B}_{Y_{\rho\theta}}^{k}$
6. Test de validité de la mesure $(\varepsilon^k)^T(\mathbf{S}^{k|k-1})^{-1}\varepsilon^k \leq \chi^2$
7. Si la mesure est valide, on corrige l'état prédit.
   Étape de correction
   (a) Calcul du gain $K^k = \widehat{Cov}_{\rho\theta}^{k|k-1}(S^{k|k-1})^{-1}$
   (b) Correction de l'état par l'innovation $\hat{\mathbf{X}}_{\rho\theta}^{k|k} = \hat{\mathbf{X}}_{k|k-1} + K_k\varepsilon_k$
   (c) Correction de la matrice de variance-covariance en utilisant (4.27).

Après prédiction du squelette dans l'image nous utilisons la loi de commande (3.17) pour atteindre l'approximation linéaire la plus éloignée de la position actuelle du robot. Il est cependant nécessaire de garantir que la trajectoire générée par la commande soit libre de collision. Par conséquent il y aura une étape de test à mener avant d'envoyer les consignes de commande aux actionneurs.

4.4 Planification

L'étape de planification se définit par un test de compatibilité entre la trajectoire du robot et l'espace libre. La figure 4.7 illustre cette problématique. Pour la même cible, la trajectoire du robot peut être différente car elle dépend de l'orientation actuelle du robot et des gains choisis pour la loi de commande (3.17).

Pour l'asservissement visuel plusieurs travaux utilisent la planification de trajectoire en s'appuyant sur des informations dans l'image [Mezouar et Chaumette, 2000], [Mezouar et Chaumette, 2001], [Park et Chung, 2003]. Ces travaux étaient destinés à des robot manipulateurs.
Dans notre cas, nous allons nous appuyer sur les informations des frontières de l'espace libre.

Nous calculons la distance au bord entre la position du robot dans l'image et sa projection sur les

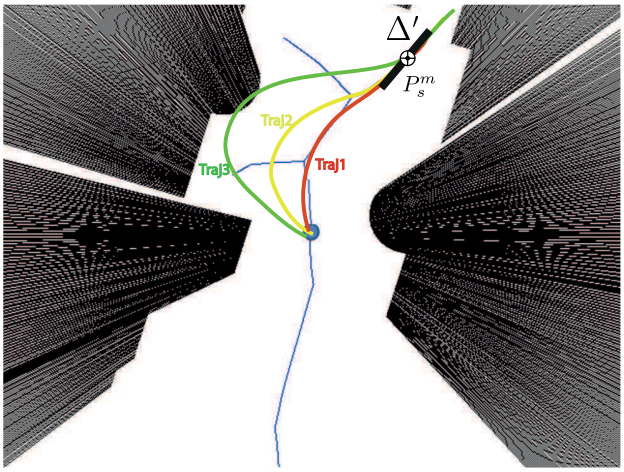


FIGURE 4.6 – L'image est fixe et le robot se déplace en simulation. Les différentes trajectoires (Traj1, Traj2 et Traj3) que le robot peut effectuer sont simulées puis transposées dans l'image et résultant d'une variation des gains dans la loi de commande (3.17)

frontières de l'espace occupé et nous comparerons cette distance par rapport à un certain seuil. Pour cela, nous estimerons la position du robot dans l'image en se basant sur le modèle cinématique du robot.

4.4.1 Génération de trajectoire dans l'image

La trajectoire du robot dans le repère monde est définie par la loi de commande et la cinématique du robot à chaque instant :

$$\begin{pmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \end{pmatrix}_{\mathcal{F}_W} = \begin{pmatrix} x_k \\ y_k \\ \theta_k \end{pmatrix}_{\mathcal{F}_W} + T_e \begin{pmatrix} v\,cos(\theta_R) \\ v\,sin(\theta_R) \\ \omega \end{pmatrix} \tag{4.28}$$

Pour déterminer la trajectoire du robot dans l'image nous fixons l'observation initiale du robot et nous déduisons la trajectoire du robot dans le repère image de l'observation fixe. L'état initial du robot est pris au centre de l'image.

$$\mathbf{X}_{R_{\mathcal{F}_I}}^{k+1} = \begin{pmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \end{pmatrix}_{\mathcal{F}_I} = \begin{pmatrix} x_k \\ y_k \\ \theta_k \end{pmatrix}_{\mathcal{F}_I} + T_e\, K\, \mathcal{R}_{\mathcal{F}_C \leftarrow \mathcal{F}_R}\, \mathcal{R}_{\mathcal{F}_R \leftarrow \mathcal{F}_W} \begin{pmatrix} v\,cos(\theta_R) \\ v\,sin(\theta_R) \\ \omega \end{pmatrix} \tag{4.29}$$

où les matrices $\mathcal{R}_{\mathcal{F}_C \leftarrow \mathcal{F}_R}$ et $\mathcal{R}_{\mathcal{F}_R \leftarrow \mathcal{F}_W}$ sont respectivement la matrice de passage entre le repère robot et le repère caméra et entre le repère monde et le repère robot.

$$\mathcal{R}_{\mathcal{F}_C \leftarrow \mathcal{F}_R} = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix} \text{ et } \mathcal{R}_{\mathcal{F}_R \leftarrow \mathcal{F}_W} = \begin{pmatrix} cos(\theta_R) & -sin(\theta_R) & 0 \\ sin(\theta_R) & cos(\theta_R) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Les variables de commande v et ω sont prises comme suit : la vitesse linéaire v est choisie constante et

la vitesse angulaire est calculée en utilisant (3.17). On procède alors à l'intégration du modèle (4.29) pour générer la trajectoire du robot dans l'image.

4.4.2 Test de compatibilité de trajectoire

Nous notons $\mathcal{X}_{free}$ l'ensemble des points dans l'image définissant l'espace libre et $\mathcal{L}_{X_{free}}$ l'ensemble des points qui constituent la limite (frontière) de l'espace libre. Pour chaque point $\mathbf{x}$ appartenant à $\mathcal{X}_{free}$, nous cherchons sa projection $\mathcal{P}(\mathbf{x})$ correspondant au point appartenant à $\mathcal{L}_{X_{free}}$ à distance minimale de $\mathbf{x}$:

$$\forall \mathbf{x}, \mathcal{P}(\mathbf{x}) = \left\{ \mathbf{y} \in \mathcal{L}_{X_{free}} \mid \forall z \in \mathcal{L}_{X_{free}},\, d_E(\mathbf{x}-\mathbf{y}) \leq d_E(\mathbf{x}-\mathbf{z}) \right\} \tag{4.30}$$

Ainsi, la distance au bord $\mathcal{D}(x)$ d'un point $\mathbf{x}$ appartenant à $\mathcal{X}_{free}$ est :

$$\mathcal{D}(x) = d_E(\mathbf{x}, \mathcal{P}(\mathbf{x})) \tag{4.31}$$

Pour notre cas la variable $\mathbf{x}$ peut être l'estimation de la position $\mathbf{X}_{R_{\mathcal{F}_I}}$ du robot dans l'image.

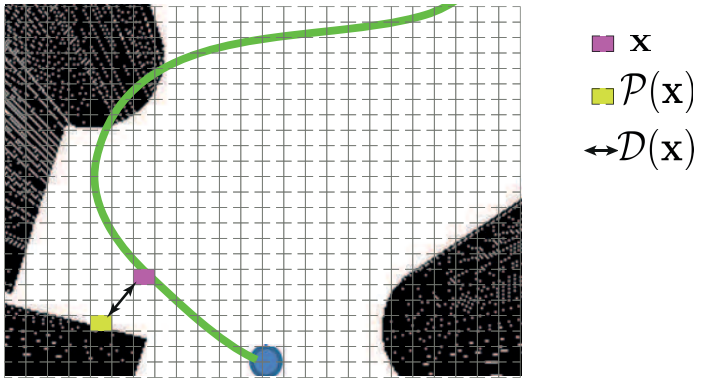


FIGURE 4.7 – Définition de la distance au bord de l'espace libre. Zoom d'une partie de l'image correspondant à la trajectoire Traj3, illustrée dans la figure 4.7

Les étapes de ce test sont :

1. Choisir l'approximation linéaire la plus loin $\Delta_\infty(\rho_\infty, \theta_\infty)$.
2. Calculer l'intégralité de la commande qui permet de rallier le segment avec (3.17) (sans envoyer aux actionneurs)
3. Calculer la position $\mathbf{X}_{R_{\mathcal{F}_I}}$ du robot dans l'image avec (4.29) puis sa distance au bord $\mathcal{D}(\mathbf{X}_{R_{\mathcal{F}_I}})$ (4.31).
4. Si $\mathcal{D}(\mathbf{X}_{R_{\mathcal{F}_I}}) > \textit{seuil}$ alors on envoie la commande aux actionneurs. Sinon, les gains de la loi de commande peuvent être modifiés ou l'approximation linéaire à rallier est changée. L'algorithme retournerait alors à l'étape 2.

4.5 Validation de l'approche

Dans cette section, nous proposons une validation en simulation de l'approche de l'asservissement sur une prédiction. Dans un premier temps, un environnement de simulation est généré avec Matlab. L'intérêt d'utiliser Matlab est de modéliser des environnements simplifiés de type couloir par morceau pour valider les performances de l'approche et du filtrage.

Dans le scénario 1 et 2 le squelette est modélisé comme un ensemble de segments. Un premier scénario permettra d'évaluer le ralliement du dernier segment du squelette (le plus loin par rapport au robot dans l'image). Cela permet de valider l'approche de la commande et de la planification avec une seule prise de vue orthographique. Le deuxième scénario permettra d'évaluer le comportement de la commande et le filtrage lors d'une succession de ralliement d'une séquence de segments. La position du capteur virtuel dans le repère robot est $(0.2,\ 0,\ 7)^T$. Dans un second temps, des environnements plus réalistes seront intégrés et la validation sera réalisée dans l'environnement Morse/Blender/Ros (scénarios 3 et 4) et différents types de bâtiments.

Scénario 1

Les résultats de ce scénario sont illustrés sur la figure 4.8. L'environnement est toujours le même. Chaque test correspond à une position initiale du robot différente (cas 1 pour les 2 images en haut (a) et (b), cas 2 pour les images de milieu (c) et (d) et cas 3 pour les deux images en bas (e) et (f)). Dans ce test, les gains de la loi de commande choisis sont $\tau_\theta = 4$ et $\tau_\rho = 4$. Ce test permet de valider l'étape de la planification de notre algorithme. Ici on simule l'application de la loi de commande qui permet de rallier le dernier segment. Entre 2 acquisitions d'images, on déduit la trajectoire du robot pour un temps simulé de 80 secondes. Pour chacune des positions simulées, la compatibilité de la trajectoire avec l'environnement est réalisée.

La première image en haut de la figure 4.8 illustre la trajectoire du robot avec sa projection dans l'image, on constate que, pour cette configuration, le robot est suffisamment loin des frontières des obstacles. Dans ce cas, la séquence de commande idéale peut être envoyée aux actionneurs. Dans un contexte idéal, la séquence idéale se déroulera comme illustré sur la figure 4.9a qui représente la convergence des 2 variables commandées ainsi que la commande appliquée au cours de l'exécution.

Par contre, pour la deuxième condition initiale (cas 2), le robot passe à la limite des frontières.

Pour la troisième position initiale du robot (cas 3), si on appliquait la commande avec les gains initialement choisis le robot entrerait en collision avec les obstacles. Deux solutions sont alors accessibles. La première consiste à choisir un nouveau segment cible. La seconde consiste en la déformation de la trajectoire en jouant sur les gains de la régulation. On choisit ici d'augmenter les gains (constantes de temps), pour que le robot atteigne plus tard le segment sans entrer en collision avec les obstacles. En sélectionnant le gain $\tau_\rho = 40$ cela permet d'avoir une annulation de l'écart latéral au segment plus lente. On conserve la valeur de τ_θ qui permet d'ajuster l'erreur d'orientation du robot par rapport au segment.

Afin d'évaluer les performances du filtrage, on ajoute un bruit à la mesure de $(\rho_{capt}, \theta_{capt})$. Au cours de l'application réelle de la commande, la mesure est alors celle de la figure 4.9b (tracé vert). Les niveaux de bruits choisis sont $\sigma_\theta = \left(\frac{10}{3}\right) deg$, $\sigma_\rho = \left(\frac{10}{3}\right) pixel$. Pour calculer la propagation de

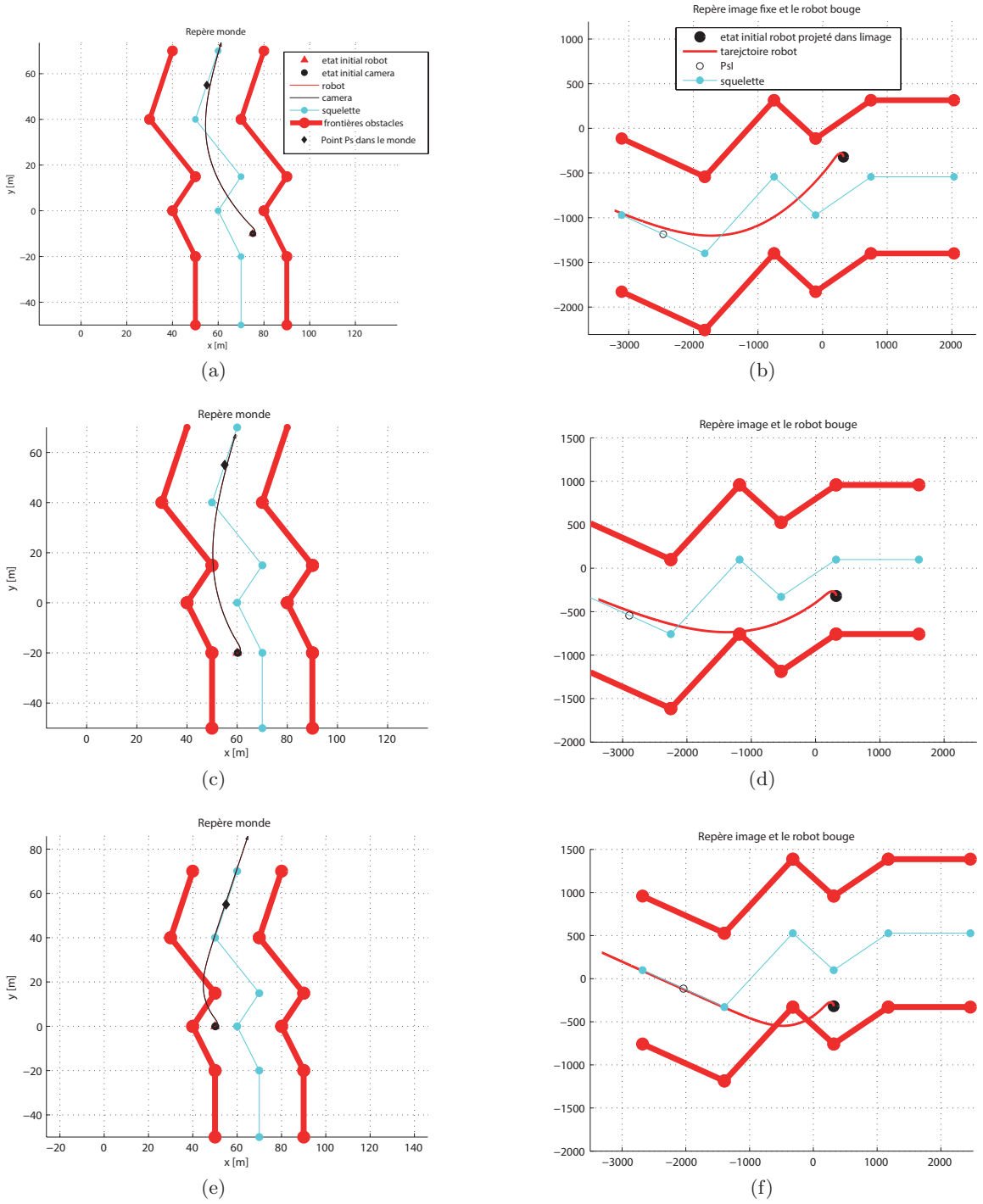
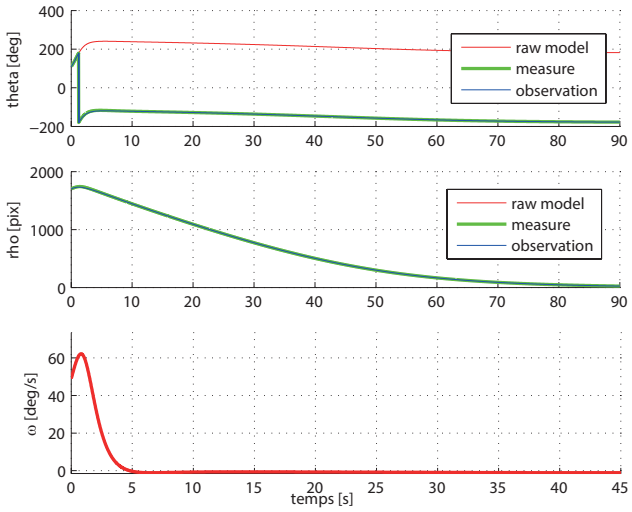
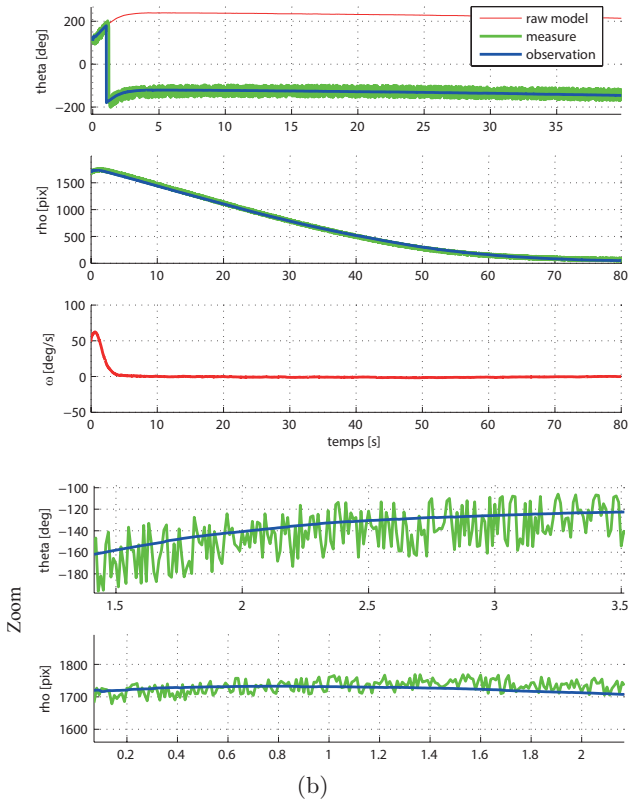


FIGURE 4.8 – Test de planification : Calcule de la trajectoire du robot par simulation sans envoyer les vitesses aux actionneurs pour le suivi du dernier segment du squelette dans le repère monde pour plusieurs positions initiales du robot (a) et dans le repère image dans (b).



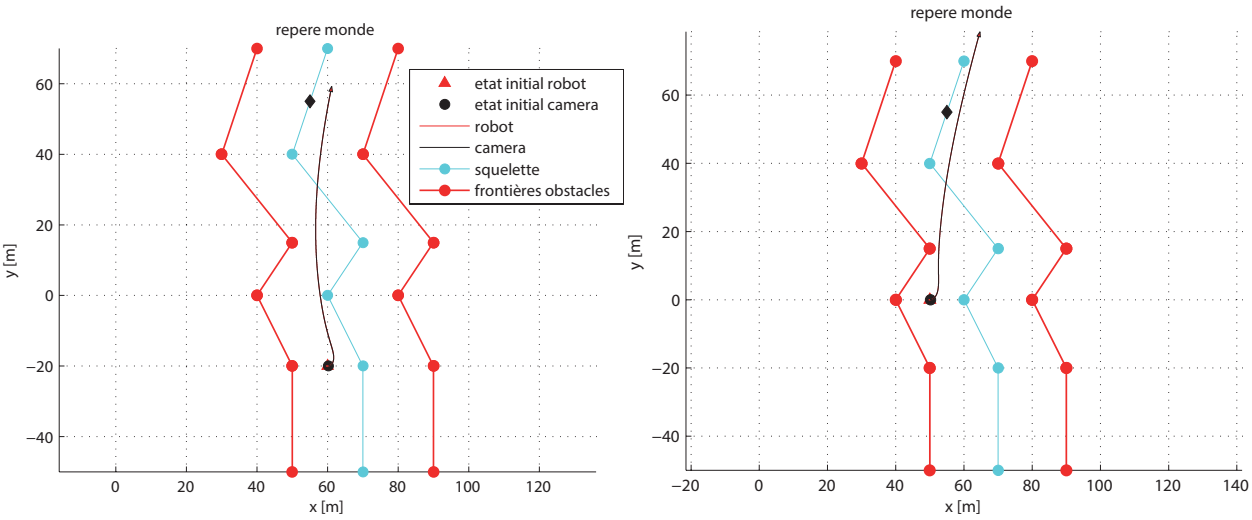
(a)



(b)

FIGURE 4.9 – Résultats pour un suivi du dernier segment du squelette : la signature visuelle (ρ_m, θ_m) du suivi d'une seule approximation linéaire et un test avec un ajout de bruit sur la mesure.

l'incertitude vis a vis de la commande appliquée les valeurs prises des erreurs sont : $\sigma_v = \left(\frac{0.2}{3}\right) m/s$ et $\sigma_\omega = \left(\frac{5}{3}\right) deg/s$



(a) (b)

FIGURE 4.10 – Test pour la planification : changement des gains de la loi de commande pour rendre la trajectoire compatible avec l'environnement pour le cas 2 et 3.

Le filtre de Kalman développé permet d'avoir des variables beaucoup moins bruitées (tracé bleu sur la figure 4.9b). Pour atteindre le segment, la variable θ doit atteindre les valeurs $\{-\pi, 0, \pi\}$. La variable θ issue du modèle (3.18) est illustrée par le tracé rouge sur la partie haute de la figure 4.9b. Malgré le bruit ajouté à la mesure, la trajectoire du robot est lisse car la commande est appliquée sur les paramètres de la droite estimée. Le robot rallie bien l'approximation linéaire du squelette choisie.

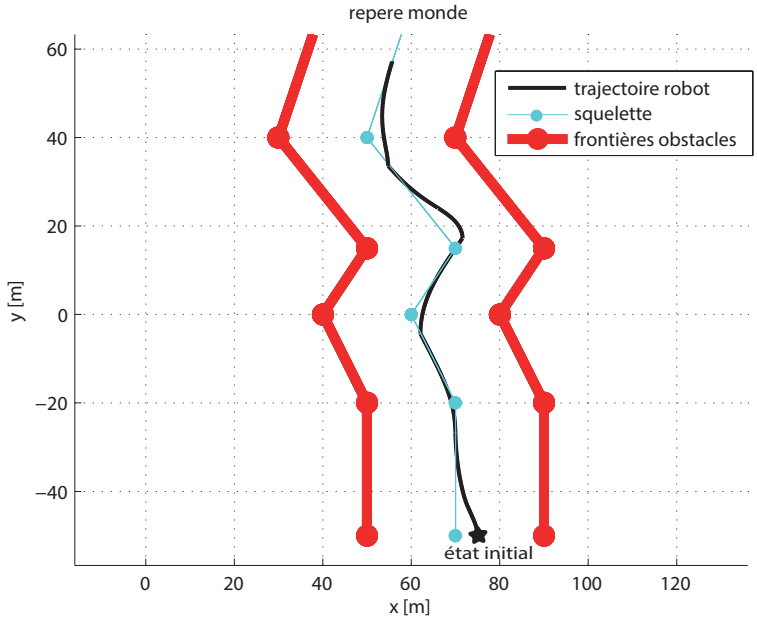
Scénario 2

Le second scénario est un suivi du squelette complet en appliquant la commande sur la prédiction. La position du robot initiale est fixée à $\overrightarrow{O_W O_R} = (75, -50)^T$ dans le repère monde.

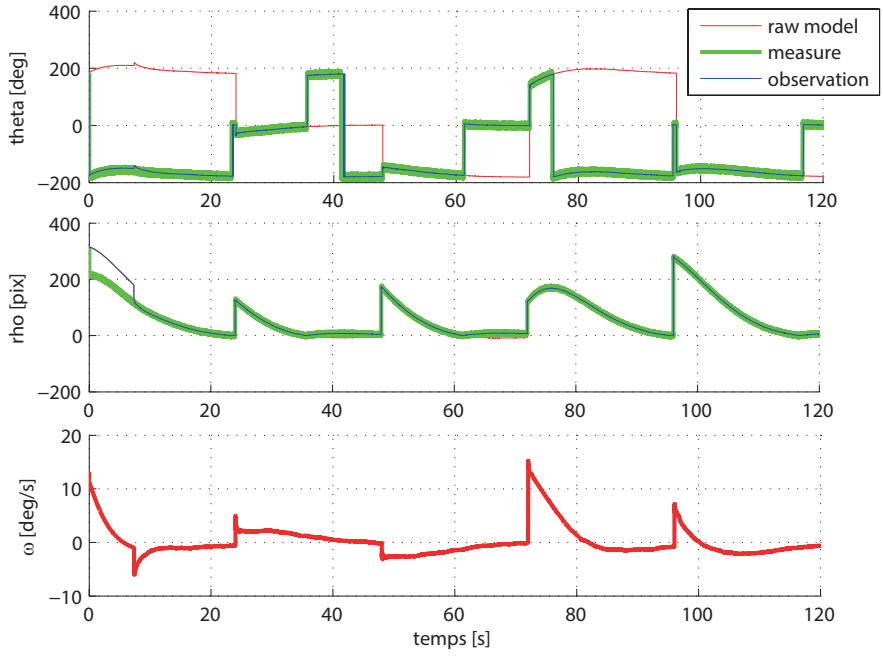
La phase de planification consiste à rallier différentes approximations linéaires du squelette. Le changement de consigne peut provoquer une modification brutale des mesures ρ et θ comme on peut le constater sur la figure 4.11b.
La stratégie consiste ici à changer de segment de squelette toutes les 23 secondes. On peut constater que les variables observées suivent bien la mesure malgré les changements de segment au cours du temps. La convergence de la signature visuelle (observée et mesurée) vers la signature désirée est obtenue pour chacun des segments sélectionnés.

Scénario 3

La figure 4.12 illustre la trajectoire du robot dans le repère monde avec un environnement modélisé sur Morse/Blender (Annexe F). Lorsque nous appliquons la commande qui permet de rallier l'approximation linéaire dans l'image la plus éloignée par rapport au robot sans l'étape de la planification, le robot peut être très proche des obstacles voire entrer en collision avec eux (figure 4.12a). L'étape de planification consiste à calculer dans l'image la distance entre la prédiction de la position du robot par rapport aux frontières de l'espace libre, et de comparer cette distance à un seuil. Ce seuil



(a)



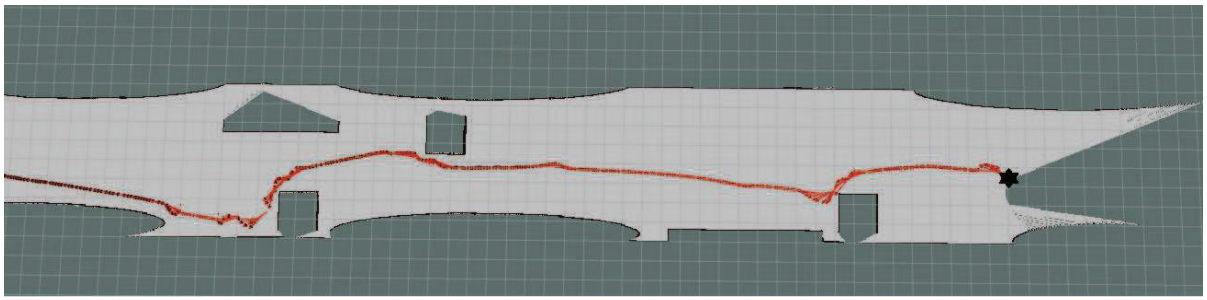
(b)

FIGURE 4.11 – Résultats pour un suivi complet du squelette : (a) La trajectoire du robot (b) Le résultat de la signature mesurée $(\rho_{capt}, \theta_{capt})$ ainsi que la signature observée $(\hat{\rho}, \hat{\theta})$ et la commande appliquée

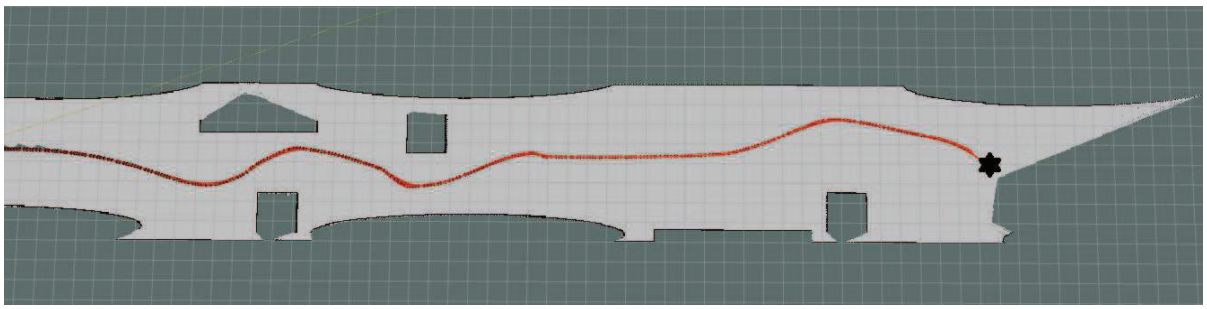
peut dépendre de la dimension du robot. Dans ce test il a été choisi à $30\,cm$ ce qui autorise le robot à frôler les obstacles. Si cette distance est inférieure à $30\,cm$, on réalise une adaptation automatique des gains de la loi de commande puisque ces derniers peuvent jouer sur la déformation de la trajectoire.

On peut constater que l'étape de la planification du robot permet de rester à bonne distance des obstacles comme l'illustre la figure 4.12b.

Le changement de segment se fait ici soit lorsque le segment de l'itération précédente est rallié soit un temps de 20 secondes est écoulé (figure 4.13). Pour l'essai illustré dans la figure 4.12b, la vitesse linéaire appliquée au robot est de $v^c = 0.5\,ms$ et la vitesse angulaire est calculée à partir de la loi de commande (3.17) (figure 4.13).



(a)



(b)

FIGURE 4.12 – Évolution du robot dans un environnement avec des obstacles. (a) sans l'étape de planification. (b) avec l'étape de planification. L'étoile correspond à la position initiale du robot.

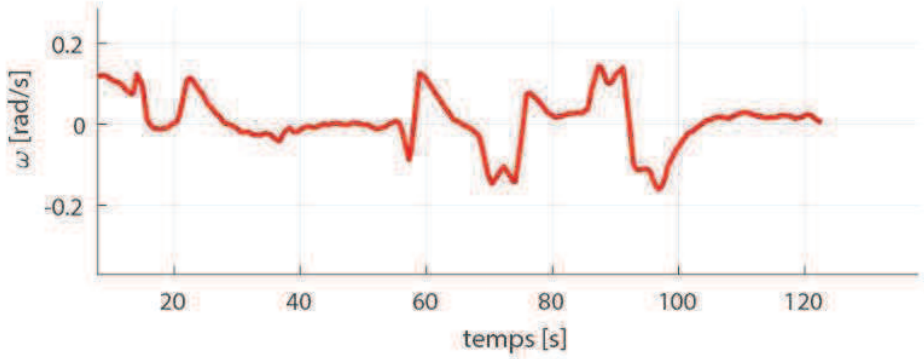


FIGURE 4.13 – La commande appliquée pour le scénario 3 pour l'essai illustrée sur la figure 4.12b

Scénario 4

Ce dernier scénario traite de la navigation dans un labyrinthe (parcours de 140 m). Dans la mesure où, ayant sélectionné une approximation linéaire du squelette éloignée et qu'est vérifiée la compatibilité de la trajectoire avec l'environnement, la trajectoire réalisée n'est plus un suivi au plus près du squelette comme réalisé au précédent chapitre (figure 3.14). La performance obtenue est un ralliement plus rapide des intersections du labyrinthe avec un déplacement restant sûr et en appliquant des commandes moins chahutées. Lorsque la trajectoire simulée passe à moins de 30 cm des obstacles, il est choisi de changer les gains de la loi de commande pour déformer la trajectoire.

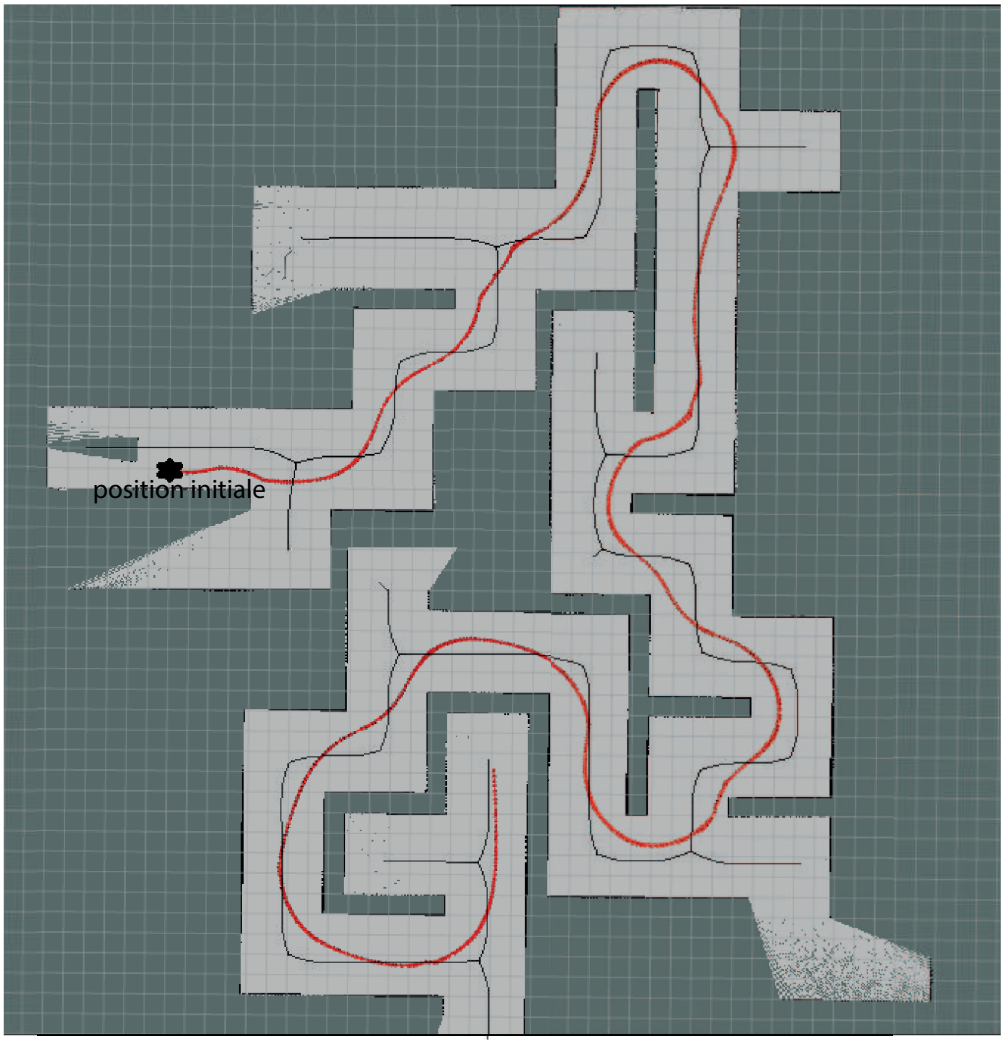


FIGURE 4.14 – Validation de l'approche de déplacement dans un labyrinthe : illustration de la trajectoire en rouge du robot avec l'outil de visualisation de ROS : RVIZ. Le test de la planification consiste à conserver une trajectoire dont la distance aux obstacles au supérieur à 30 cm (la taille d'une case de la grille est de 1 m), le DVG de l'environnement est en noir. Il est calculé en temps réel et mémorisé au cours du déplacement.

4.6 Conclusion

Nous avons proposé dans ce chapitre une nouvelle stratégie d'asservissement visuel pour le déplacement du robot en utilisant une approximation linéaire du diagramme de Voronoï. Elle se définit par un asservissement visuel basé sur la prédiction du squelette. Au lieu de faire un suivi au plus près, le robot doit rallier l'approximation linéaire la plus éloignée dans sa perception.

Dans cette méthode, nous avons pu aussi filtrer la mesure réelle et obtenir une trajectoire plus lisse du robot. Cette trajectoire reste sûre. L'approche proposée permet de limiter l'impact des bruits de mesure au niveau de la perception (un squelette qui évolue trop vite du fait de perturbations locales) et au niveau de l'actionnement (dérive du robot à cause du glissement des roues).

Nous avons pu tester cette approche dans différents environnements de simulation. Malgré l'ajout de bruit au niveau des mesures et de la commande nous avons pu filtrer les données. La performance obtenue par cette approche est un déplacement et une exploration de l'environnement plus rapide par rapport à un suivi de proche en proche, avec un mouvement lisse et sûr par rapport à l'environnement.

Une étape de planification est réalisée pour éviter la collision avec des obstacles. Cette étape se base uniquement sur la perception actuelle du robot. Avant d'envoyer les variables de commande aux actionneurs, on teste la compatibilité de la trajectoire qui devrait être générée si on appliquait la commande telle quelle avec l'environnement. Tant que la trajectoire planifiée ne présente pas de risque de collision, les consignes seront appliquées. Le cas échéant la consigne sera changée. Cela peut se faire de deux façons. La première consiste à sélectionner une nouvelle approximation linéaire sur le squelette. La seconde consiste à changer les gains de lois de commande.

Le choix des gains de la loi de commande peut influer directement sur l'allure de la trajectoire du robot. En utilisant différentes valeurs de gains τ_ρ et τ_θ, nous obtenons plusieurs trajectoires du robot (figure 4.7). En augmentant par exemple les variables $\frac{1}{\tau_\theta}$ et $\frac{1}{\tau_\rho}$, le robot atteint plus rapidement l'approximation linéaire choisie mais en contre partie, cela augmente le risque d'un dépassement incontrôlé de la commande et ceci peut nuire à la stabilité de la loi de commande. En revanche en choisissant $\frac{1}{\tau_\theta}$ et $\frac{1}{\tau_\rho}$ très petits, cela peut garantir la stabilité mais le ralliement du segment sera peu rapide et peu précis. Par conséquent une étude de gain peut être réalisée pour trouver un compromis satisfaisant entre la rapidité, la stabilité et la précision de la loi de commande.

Conclusion générale et perspectives

Sommaire

5.1 Résumé des contributions

Dans ce travail de thèse, nous avons abordé le problème de la navigation sûre et autonome d'un robot mobile dans un environnement inconnu.

Le cas du déplacement dans un environnement non encombré en utilisant un asservissement visuel adapté dans un couloir a tout d'abord été évalué. Pour généraliser la procédure à tout type d'environnement nous avons utilisé un algorithme de squelettisation dans l'image pour extraire un diagramme de Voronoï (DVG) de l'environnement local du robot. En utilisant une signature visuelle adaptée au DVG local, nous avons conçu un asservissement visuel pour un déplacement réactif autonome en temps réel du robot dans un environnement inconnu et dynamique. La signature visuelle dans l'asservissement visuel risque de ne plus pouvoir être calculée car les supports nécessaires à son extraction peuvent disparaître de l'image (murs du couloir qui disparaît ou squelette non calculé). Dans ce cas, la loi de commande est directement impactée. Elle peut ne pas être calculée ou donner un déplacement non conforme à l'environnement. Pour gérer les problèmes d'occultation ou de perte d'une signature visuelle nous n'avons pas directement utilisé un asservissement visuel issue d'une caméra réelle. Nous avons utilisé un asservissement visuel utilisant un capteur virtuel. L'image virtuelle utilisée pour l'asservissement visuel est générée en fusionnant plusieurs capteurs. Nous avons proposé un référentiel virtuel unique qui offre au robot une compréhension de son environnement. Ce référentiel regroupe toutes les informations de capteurs embarqués sur le robot. Les éléments d'extraction de la signature, la signature et la commande seront alors exprimés dans ce référentiel.

5.1.1 Navigation référencée capteurs dans un environnement type couloir

Le déplacement dans un couloir a, par exemple, été résolu dans la littérature en utilisant un asservissement visuel basé sur deux informations extraites dans l'image. La signature était constituée du point de fuite et de l'orientation de la médiane dans l'image. Nous avons conçu une nouvelle stratégie de mesure et de commande pour ce type de déplacement en utilisant le référentiel virtuel proposé. Nous avons prouvé que l'utilisation d'une seule information de la signature est suffisante pour un suivi de couloir. Une analyse détaillée des propriétés de stabilité de la loi de commande a été réalisée. Une stratégie de détection de défauts sur l'extraction de la signature a été réalisée. Si la signature ne peut pas être extraite, un modèle est généré avec de bonnes conditions initiales pour calculer la loi de commande. Dans le cas contraire, la loi de commande est calculée à partir d'une signature observée. Cette stratégie de commande est effectuée en utilisant un observateur en temps fini.

5.1.2 Navigation référencée capteur sur un DVG local

Dans le cadre virtuel construit, un DVG local peut être calculé. Ce diagramme est extrait à l'aide d'un algorithme de squelettisation dans l'image qui prend en compte la dimension du robot (*Delta Medial Axis*). Une approche de suivi du DVG avec un asservissement visuel virtuel en temps réel est réalisée pour assurer une navigation autonome et sûre des robots mobiles. La signature visuelle a été calculée par rapport à l'approximation linéaire du diagramme de Voronoï la plus proche par rapport au robot. La stabilité de la loi de commande est analysée. Cette stratégie a été implémentée et validée sur un fauteuil roulant électrique à l'Université de Kent (UK).

5.1.3 Prédiction des déplacements sur un diagramme de Voronoï

Le déplacement sur un squelette de proche en proche en utilisant une approximation linéaire présente des limites : 1) nous n'utilisons pas la totalité du DVG extrait, 2) suite aux bruits de perception et bruits de commande, le squelette perçu risque d'être différent du squelette réel. Pour résoudre ces problèmes nous avons changé la mesure. Au lieu d'asservir le robot sur l'approximation linéaire la plus proche, nous avons défini une stratégie de navigation sur n'importe quelle approximation linéaire sur le squelette perçu. Un asservissement visuel sur une prédiction d'une linéarisation du DVG est développé en utilisant un contexte probabiliste. Après avoir calculé une mesure filtrée et les vitesses de consignes, une trajectoire dans l'image peut être calculée puis une stratégie de planification dans l'image est réalisée pour éviter toute collision avec les obstacles.

5.2 Perspectives

En s'appuyant sur le travail réalisé plusieurs extensions et applications nouvelles peuvent être prévues. L'approche actuelle est conçue pour un déplacement dans des environnements inconnus statiques ou dynamiques. Le calcul du squelette dans l'image se fait en temps réel. Par conséquent le squelette peut être déformé dès qu'un obstacle dynamique apparaît et l'asservissement se fait sur le squelette déformé. Il sera intéressant aussi de prendre en compte la vitesse des obstacles dynamiques pour le déplacement autonome de notre agent mobile. Dans l'objectif de prendre en compte non seulement de la dynamique de l'agent mobile, mais également la dynamique de l'environnement qui l'entoure, [Delsart, 2010] a conçu une approche d'anticipation du mouvement d'obstacles mobiles. L'utilisation de ce type d'approche serait intéressante pour améliorer notre méthode de déplacement autonome.

Notre travail s'appuie sur un référentiel $2D$ virtuel locale. La perception, le calcul du squelette, la signature ainsi que la loi de commande se fait sur ce référentiel $2D$. Maintenant il sera intéressant d'améliorer l'approche avec une perception en $3D$ et de choisir une nouvelle signature adaptée. Cela peut être résolu par exemple par la coopération entre robots (aérien et terrestre) : le référentiel virtuel que nous avons introduit peut se construire à partir d'un autre système (un drone ou un ensemble de drones), le calcul du squelette ainsi que le calcul de la loi de commande se fait dans le nouveau référentiel virtuel dans une approche centralisée. Lorsque le référentiel virtuel est construit à partir d'un autre système qui a une meilleure vision et champs de vue comparé au robot, les branches de DVG peuvent être calculées à un certain horizon que le robot ne peut pas voir. Dans ce cas, des approches de commande prédictive peuvent être utilisées telles que celles proposées par [Morette, 2009]. Ainsi l'étape de la prédiction peut être améliorée en utilisant une portion complète du squelette pour un déplacement calculé à priori.

Une autre perspective du travail est de prendre en considération les contraintes mécaniques du fauteuil roulant. Il s'agit de concevoir une commande qui prend en compte toutes les contraintes de déplacement d'un fauteuil roulant. Une commande optimale peut être créée pour gérer cela.

5.3 Valorisation scientifique

Quelques travaux de ce manuscrit ont été valorisés par des publications et communications internationales :

- Hela Ben-Said, Joanny Stéphant, Ouiddad Labbani-Igbida *Sensor-based control using finite time observer of visual signatures : Application to corridor following*, International Conference on Intelligent Robots and Systems (IROS), Daejon, Corée du sud , Octobre 9-14, 2016 : 418-423
- Hela Ben-Said, Joanny Stéphant, Ouiddad Labbani-Igbida *Sensor localization for visual servoing stability, application to corridor following*, The International Federation of Automatic Control (IFAC), Toulouse, France, July 9-14, 2017 : 2259-2264
- Revue soumise : Hela Ben-Said, Romain Marie, Joanny Stéphant, Ouiddad Labbani-Igbida *Skeleton-based visual servoing in unknown environments*, IEEE/ASME Transactions on Mechatronics.

Un séjour scientifique a été réalisé au sein de l'Université de Kent, "School of Engineering and Digitals Arts".

Bibliographie

[Ali, 2011] Ali, W. G. (2011). A semi-autonomous mobile robot for education and research. *Journal of King Saud University-Engineering Sciences*, 23(2) :131–138.

[Ansar et Daniilidis, 2003] Ansar, A. et Daniilidis, K. (2003). Linear pose estimation from points or lines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5) :578–589.

[Arkin, 1998] Arkin, R. C. (1998). *Behavior-Based Robotics*, volume 1 of *1*. The MIT Press, Cambridge, Massachusetts London, England, 1 edition. 1.

[Aurenhammer et al., 2013] Aurenhammer, F., Klein, R. et Lee, D.-T. (2013). *Voronoi diagrams and Delaunay triangulations*. World Scientific Publishing Co Inc.

[Ayari et al., 2010] Ayari, E., Hadouaj, S. et Ghedira, K. (2010). A fuzzy logic method for autonomous robot navigation in dynamic and uncertain environment composed with complex traps. In *Computing in the Global Information Technology (ICCGI), 2010 Fifth International Multi-Conference on*, pages 18–23. IEEE.

[Bayle, 2008] Bayle, B. (2008). Robotique mobile. *Ecole Nationale Superieure de Physique de Strasbourg, University Louis Pasteur, France*, pages 2007–2008.

[Brandt et Algazi, 1992] Brandt, J. W. et Algazi, V. R. (1992). Continuous skeleton computation by voronoi diagram. *CVGIP : Image understanding*, 55(3) :329–338.

[Bulut et al., 2011] Bulut, Y., Vines-Cavanaugh, D. et Bernal, D. (2011). Process and measurement noise estimation for kalman filtering. In *Structural Dynamics, Volume 3*, pages 375–386. Springer.

[Buonocore et al., 2015] Buonocore, L. R., Cacace, J. et Lippiello, V. (2015). Hybrid visual servoing for aerial grasping with hierarchical task-priority control. In *Control and Automation (MED), 2015 23th Mediterranean Conference on*, pages 617–623. IEEE.

[Cadenat et al., 2000] Cadenat, V., Souères, P. et Courdesses, M. (2000). Two multi-sensor-based control strategies for driving a robot amidst obstacles. In *Decision and Control, 2000. Proceedings of the 39th IEEE Conference on*, volume 1, pages 827–832. IEEE.

[Cai et Ferrari, 2009] Cai, C. et Ferrari, S. (2009). Information-driven sensor path planning by approximate cell decomposition. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(3) :672–689.

[Carelli et Freire, 2003] Carelli, R. et Freire, E. O. (2003). Corridor navigation and wall following stable control for sonar based mobile robots. *Robotics and Autonomous Systems*, 45(34) :235–247.

[Caron et al., 2013] Caron, G., Marchand, É. et Mouaddib, E. M. (2013). Photometric visual servoing for omnidirectional cameras. *Auton. Robots*, 35(2-3) :177–193.

[Chatila et Laumond, 1985] Chatila, R. et Laumond, J.-P. (1985). Position referencing and consistent world modeling for mobile robots. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, volume 2, pages 138–145. IEEE.

[Chaumette, 1990] Chaumette, F. (1990). *La relation vision-commande : théorie et application à des tâches robotiques*. PhD thesis, Université de Rennes I.

[Chaumette, 1998] Chaumette, F. (1998). *Potential problems of stability and convergence in image-based and position-based visual servoing*, pages 66–78. Springer London, London.

[Chaumette, 2004] Chaumette, F. (2004). Image moments : a general and useful set of features for visual servoing. *IEEE Transactions on Robotics*, 20(4) :713–723.

[Chaumette et al., 2014] Chaumette, F., Rives, P. et Espiau, B. (2014). Visual servoing.

[Chen et al., 2015] Chen, L., Edwards, C. et Alwi, H. (2015). Lpv sliding mode observers for sensor fault reconstruction with erroneous scheduling parameter measurements. In *Decision and Control (CDC), 2015 IEEE 54th Annual Conference on*, pages 4080–4085. IEEE.

[Chen et al., 2012] Chen, L., Wang, S., Hu, H. et McDonald-Maier, K. (2012). Bézier curve based trajectory planning for an intelligent wheelchair to pass a doorway. In *Control (CONTROL), 2012 UKACC International Conference on*, pages 339–344. IEEE.

[Chesi et al., 2004] Chesi, G., Hashimoto, K., Prattichizzo, D. et Vicino, A. (2004). Keeping features in the field of view in eye-in-hand visual servoing : A switching approach. *IEEE Transactions on Robotics*, 20(5) :908–914.

[Choset et Burdick, 1995] Choset, H. et Burdick, J. (1995). Sensor based planning, part i : The generalized voronoi graph. In *Proceedings of the 1995 IEEE International Conference on Robotics and Automation (ICRA '95)*, volume 2, pages 1649 – 1655.

[Choset, 2005] Choset, H. M. (2005). *Principles of robot motion : theory, algorithms, and implementation*. MIT press.

[Coué et al., 2006] Coué, C., Pradalier, C., Laugier, C., Fraichard, T. et Bessière, P. (2006). Bayesian occupancy filtering for multitarget tracking : an automotive application. *The International Journal of Robotics Research*, 25(1) :19–30.

[Dan, 2016] Dan, D. (2016). Image segmentation using voronoi diagram. In *Eighth International Conference on Digital Image Processing (ICDIP 2016)*, pages 100331P–100331P. International Society for Optics and Photonics.

[De Luca et al., 2008] De Luca, A., Oriolo, G. et Robuffo Giordano, P. (2008). Feature depth observation for image-based visual servoing : Theory and experiments. *The International Journal of Robotics Research*, 27(10) :1093–1116.

[Deguchi, 1998] Deguchi, K. (1998). Optimal motion control for image-based visual servoing by decoupling translation and rotation. In *Proceedings 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications, October 13-17, 1998, Victoria, BC, Canada*, pages 705–711.

[Delage et al., 2006] Delage, E., Lee, H. et Ng, A. Y. (2006). A dynamic bayesian network model for autonomous 3d reconstruction from a single indoor image. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2418–2428. IEEE.

[Delsart, 2010] Delsart, V. (2010). *Navigation autonome en environnement dynamique : Une approche par déformation de trajectoire.* PhD thesis, Université de Grenoble.

[Deng et al., 2002] Deng, L., Janabi-Sharifi, F. et Wilson, W. J. (2002). Stability and robustness of visual servoing methods. In *Robotics and Automation (ICRA) IEEE Int. Conf. on*, volume 2, pages 1604–1609 vol.2.

[Dollár et al., 2009] Dollár, P., Wojek, C., Schiele, B. et Perona, P. (2009). Pedestrian detection : A benchmark. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 304–311. IEEE.

[Du et al., 2007] Du, Z., Qu, D., Xu, F. et Xu, D. (2007). A hybrid approach for mobile robot path planning in dynamic environments. *2007 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1058–1063.

[Durrant-Whyte et al., 2001a] Durrant-Whyte, H. et al. (2001a). Introduction to estimation and the kalman filter. *Australian Centre for Field Robotics*, 28(3) :65–94.

[Durrant-Whyte et al., 2001b] Durrant-Whyte, H. et al. (2001b). Introduction to estimation and the kalman filter. *Australian Centre for Field Robotics*, 28(3) :65–94.

[Echeverria et al., 2011] Echeverria, G., Lassabe, N., Degroote, A. et Lemaignan, S. (2011). Modular open robots simulation engine : Morse. In *Robotics and Automation (ICRA), 2011 IEEE Int. Conf. on*, pages 46–51.

[Elfes, 1989] Elfes, A. (1989). A tesselated probabilistic representation for spatial robot perception and navigation. In *Proceedings of the NASA Conference on Space Telerobotics*, volume 2, pages 341–350.

[Espiau et al., 1992] Espiau, B., Chaumette, F. et Rives, P. (1992). A new approach to visual servoing in robotics. *ieee Transactions on Robotics and Automation*, 8(3) :313–326.

[Filliat et Meyer, 2003] Filliat, D. et Meyer, J.-A. (2003). Map-based navigation in mobile robots : : I. a review of localization strategies. *Cognitive Systems Research*, 4(4) :243–282.

[Fujii, 2013] Fujii, K. (2013). Extended kalman filter. *Refernce Manual*.

[Gaddouna et al., 1994] Gaddouna, B., Maquin, D. et Ragot, J. (1994). Fault detection observers for systems with unknown inputs. In *IFAC/IMACS Symposium on Fault Detection, Supervision and Safety for Technical Processes, SAFEPROCESS'94*, volume 1, pages 69–74.

[Garrido et al., 2006] Garrido, S., Moreno, L., Abderrahim, M. et Martin, F. (2006). Path planning for mobile robot navigation using voronoi diagram and fast marching. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 2376–2381. IEEE.

[Garrido et al., 2011] Garrido, S., Moreno, L., Blanco, D. et Jurewicz, P. (2011). Path planning for mobile robot navigation using voronoi diagram and fast marching. *International Journal of Robotics and Automation (IJRA)*, 2(1) :42–64.

[Gerstmayr-Hillen et al., 2013] Gerstmayr-Hillen, L., Röben, F., Krzykawski, M., Kreft, S., Venjakob, D. et Möller, R. (2013). Dense topological maps and partial pose estimation for visual control of an autonomous cleaning robot. *Robotics and Autonomous Systems*, 61(5) :497–516.

[Ghazouani, 2012] Ghazouani, H. (2012). *Navigation visuelle de robots mobiles dans un environnement d'intérieur.* PhD thesis, Université Montpellier II-Sciences et Techniques du Languedoc.

[Godler et al., 2002] Godler, I., Honda, H. et Ohnishi, K. (2002). Design guidelines for disturbance observer's filter in discrete time. In *Advanced Motion Control, 2002. 7th International Workshop on*, pages 390–395. IEEE.

[Grewal et Andrews, 1993] Grewal, M. S. et Andrews, A. P. (1993). Kalman filtering theory and practice, prentice hall information and systems sciences series. *Thomas Kailath, Series Editor*.

[Grisetti et al., 2007] Grisetti, G., Stachniss, C. et Burgard, W. (2007). Improved techniques for grid mapping with rao-blackwellized particle filters,. *IEEE Trans. on Robotics*, 23 :34–46.

[Guechi, 2010] Guechi, E.-H. (2010). *Suivi de trajectoires d'un robot mobile non holonome : approche par modèle flou de Takagi-Sugeno et prise en compte des retards.* PhD thesis, Université de Valenciennes et du Hainaut-Cambresis.

[Hadj-Abdelkader et al., 2006] Hadj-Abdelkader, H., Mezouar, Y., Andreff, N. et Martinet, P. (2006). Decoupled homography-based visual servoing with omnidirectional cameras. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 2332–2337. IEEE.

[Hadj-Abdelkader et al., 2008] Hadj-Abdelkader, H., Mezouar, Y., Martinet, P. et Chaumette, F. (2008). Catadioptric visual servoing from 3d straight lines. *IEEE Transactions on Robotics*, 24(3) :652–665.

[Hafez et al., 2008] Hafez, A. A., Cervera, E. et Jawahar, C. (2008). Hybrid visual servoing by boosting ibvs and pbvs. In *Information and Communication Technologies : From Theory to Applications, 2008. ICTTA 2008. 3rd International Conference on*, pages 1–6. IEEE.

[Halperin, 1994] Halperin, D. (1994). Robot motion planning and the single cell problem in arrangements. *Journal of Intelligent & Robotic Systems*, 11(1) :45–65.

[Herout et al., 2013] Herout, A., Dubská, M. et Havel, J. (2013). Review of hough transform for line detection. In *Real-Time Detection of Lines and Grids*, pages 3–16. Springer.

[Huq et al., 2008] Huq, R., Mann, G. K. et Gosine, R. G. (2008). Mobile robot navigation using motor schema and fuzzy context dependent behavior modulation. *Applied soft computing*, 8(1) :422–436.

[Itoh et Matsuda, 1996] Itoh, S. et Matsuda, I. (1996). Segmentation of color still images using voronoi diagrams. In *European Signal Processing Conference, 1996. EUSIPCO 1996. 8th*, pages 1–4. IEEE.

[Iwaszko, 2012] Iwaszko, T. (2012). *Généralisation du diagramme de Voronoï et placement de formes géométriques complexes dans un nuage de points.* PhD thesis, Université de Haute Alsace-Mulhouse.

[Janabi-Sharifi, 2002] Janabi-Sharifi, F. (2002). Visual servoing : theory and applications. *Opto-Mechatronic Systems Handbook*, pages 15–1.

[Jia et al., 2012] Jia, S., Shen, H., Li, X., Cui, W. et Wang, K. (2012). Autonomous robot exploration based on hybrid environment model. In *2012 IEEE International Conference on Information and Automation*.

[Kalman et al., 1960] Kalman, R. E. et al. (1960). A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1) :35–45.

[Kanayama et al., 1990] Kanayama, Y., Kimura, Y., Miyazaki, F. et Noguchi, T. (1990). A stable tracking control method for an autonomous mobile robot. In *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, pages 384–389. IEEE.

[Keil et Sack, 1985] Keil, J. M. et Sack, J.-R. (1985). Minimum decompositions of polygonal objects. *Machine Intelligence and Pattern Recognition*, 2 :197–216.

[Khatib, 1986] Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research*, 5(1) :90–98.

[Lampaert et al., 2004] Lampaert, V., Swevers, J. et Al-Bender, F. (2004). Comparison of model and non-model based friction compensation techniques in the neighbourhood of pre-sliding friction. In *American Control Conference, 2004. Proceedings of the 2004*, volume 2, pages 1121–1126. IEEE.

[Lemaignan et al., 2012] Lemaignan, S., Echeverria, G., Karg, M., Mainprice, J., Kirsch, A. et Alami, R. (2012). Human-robot interaction in the morse simulator. In *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, pages 181–182. ACM.

[Lhomme-Desages, 2008] Lhomme-Desages, D. (2008). *Commande d'un robot mobile rapide à roues non directionnelles sur sol naturel*. PhD thesis, Université Pierre et Marie Curie-Paris VI.

[Lingelbach, 2004] Lingelbach, F. (2004). Path planning using probabilistic cell decomposition. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 1, pages 467–472. IEEE.

[Liu et al., 2002] Liu, C.-S., Peng, H. et al. (2002). Inverse-dynamics based state and disturbance observers for linear time-invariant systems. *Journal of dynamic systems measurement and control*, 124(3) :375–381.

[Luan Bui et al., 2013] Luan Bui, T., Thinh Doan, P., Sil Park, S., Kyeong Kim, H. et Sang, B. K. (2013). Agv trajectory control based on laser sensor navigation. *International Journal of Science and Engineering*, pages 16–20.

[Luenberger, 1966] Luenberger, D. (1966). Observers for multivariable systems. *IEEE Transactions on Automatic Control*, 11(2) :190–197.

[Luenberger, 1971] Luenberger, D. (1971). An introduction to observers. *IEEE Transactions on automatic control*, 16(6) :596–602.

[Luenberger, 1964] Luenberger, D. G. (1964). Observing the state of a linear system. *IEEE transactions on military electronics*, 8(2) :74–80.

[Maalouf et al., 2006] Maalouf, E., Saad, M. et Saliah, H. (2006). A higher level path tracking controller for a four-wheel differentially steered mobile robot. *Robotics and Autonomous Systems*, 54(1) :23–33.

[Mahkovic et Slivnik, 2000] Mahkovic, R. et Slivnik, T. (2000). Constructing the generalized local voronoi diagram from laser range scanner data. *IEEE Trans. Systems, Man, and Cybernetics, Part A*, 30(6) :710–719.

[Malis et al., 1998] Malis, E., Chaumette, F. et Boudet, S. (1998). Positioning a coarse-calibrated camera with respect to an unknown object by 2d 1/2 visual servoing. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 2, pages 1352–1359. IEEE.

[Malis et al., 1999] Malis, E., Chaumette, F. et Boudet, S. (1999). 2 1/2 d visual servoing. *IEEE Transactions on Robotics and Automation*, 15(2) :238–250.

[Marie, 2014] Marie, R. (2014). *Exploration autonome et construction de cartes topologiques référencées vision omnidirectionnelle*. PhD thesis, l'Université de Picardie Jules Verne.

[Marie et al., 2016] Marie, R., Labbani-Igbida, O. et Mouaddib, E. M. (2016). The delta medial axis : A fast and robust algorithm for filtered skeleton extraction. *Pattern Recognition*, 56 :26–39.

[Marinho et al., 2017] Marinho, L. B., Almeida, J. S., Souza, J. W. M., Albuquerque, V. H. C. et Rebouças Filho, P. P. (2017). A novel mobile robot localization approach based on topological maps using classification with reject option in omnidirectional images. *Expert Systems with Applications*, 72 :1–17.

[Martínez et al., 2007] Martínez, A., Martínez, J., Pérez-Rosés, H. et Quirós, R. (2007). Image processing using voronoi diagrams. In *IPCV*, pages 485–491.

[Mezouar et Chaumette, 2000] Mezouar, Y. et Chaumette, F. (2000). Path planning in image space for robust visual servoing. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 3, pages 2759–2764. IEEE.

[Mezouar et Chaumette, 2001] Mezouar, Y. et Chaumette, F. (2001). Design and tracking of desirable trajectories in the image space by integrating mechanical and visibility constraints. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 1, pages 731–736. IEEE.

[Mezouar et Chaumette, 2002] Mezouar, Y. et Chaumette, F. (2002). Path planning for robust image-based control. *IEEE Transactions on Robotics and Automation*, 18(4) :534–549.

[Minguez et Montano, 2005] Minguez, J. et Montano, L. (2005). Sensor-based robot motion generation in unknown, dynamic and troublesome scenarios. *Robotics and Autonomous Systems*, 52(4) :290–311.

[Moras et al., 2015] Moras, J., Dezert, J. et Pannetier, B. (2015). Grid occupancy estimation for environment perception based on belief functions and pcr6. In *SPIE Defense+ Security*, pages 94740P–94740P. International Society for Optics and Photonics.

[Morette, 2009] Morette, N. (2009). *Contribution à la Navigation de robots mobiles : approche par modèle direct et commande prédictive*. PhD thesis, Université d'Orléans.

[Murphy, 2000] Murphy, R. R. (2000). *Introduction to AI Robotics*, volume 1 of *1*. The MIT Press, Cambridge, Massachusetts London, England, 1 edition. 1.

[Nakhaeinia et al., 2011] Nakhaeinia, D., Tang, S., Noor, S. M. et Motlagh, O. (2011). A review of control architectures for autonomous navigation of mobile robots. *International Journal of Physical Sciences*, 6(2) :169–174.

[Narayanan et al., 2014] Narayanan, V. K., Pasteau, F., Babel, M. et Chaumette, F. (2014). Visual servoing for autonomous doorway passing in a wheelchair using a single doorpost. *IEEE/RSJ IROS Workshop on Assistance and Service Robotics in a Human.*

[Ohya et al., 1998] Ohya, I., Kosaka, A. et Kak, A. (1998). Vision-based navigation by a mobile robot with obstacle avoidance using single-camera vision and ultrasonic sensing. *IEEE Transactions on Robotics and Automation*, 14(6) :969–978.

[Organisation Mondiale de la Santé, 2011] Organisation Mondiale de la Santé, l. b. m. (2011). Rapport mondial sur le handicap. Technical report, Geneve.

[Palli et Melchiorri, 2008] Palli, G. et Melchiorri, C. (2008). Velocity and disturbance observer for non-model based load and friction compensation. In *Advanced Motion Control, 2008. AMC'08. 10th IEEE International Workshop on*, pages 194–199. IEEE.

[Palmieri et al., 2012] Palmieri, G., Palpacelli, M., Battistelli, M. et Callegari, M. (2012). A comparison between position-based and image-based dynamic visual servoings in the control of a translating parallel manipulator. *Journal of Robotics*, 2012.

[Park et Kuipers, 2011] Park, J. J. et Kuipers, B. (2011). A smooth control law for graceful motion of differential wheeled mobile robots in 2d environment. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 4896–4902. IEEE.

[Park et Chung, 2003] Park, J. S. et Chung, M. J. (2003). Path planning with uncalibrated stereo rig for image-based visual servoing under large pose discrepancy. *IEEE transactions on robotics and automation*, 19(2) :250–258.

[Park et Hashimoto, 2009] Park, S. et Hashimoto, S. (2009). Autonomous mobile robot navigation using passive rfid in indoor environment. *IEEE Transactions on Industrial Electronics*, 56(7) :2366–2373.

[Pasteau et al., 2013] Pasteau, F., Babel, M. et Sekkal, R. (2013). Corridor following wheelchair by visual servoing. *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference*.

[Pasteau et al., 2014] Pasteau, F., Krupa, A. et Babel, M. (2014). Vision-based assistance for wheelchair navigation along corridors. *IEEE Int. Conf. on Robotics and Automation*.

[Pol et Murugan, 2015] Pol, R. S. et Murugan, M. (2015). A review on indoor human aware autonomous mobile robot navigation through a dynamic environment survey of different path planning algorithm and methods. In *Industrial Instrumentation and Control (ICIC), 2015 International Conference on*, pages 1339–1344. IEEE.

[Prasov et Khalil, 2013] Prasov, A. A. et Khalil, H. K. (2013). A nonlinear high-gain observer for systems with measurement noise in a feedback control framework. *IEEE Transactions on Automatic Control*, 58(3) :569–580.

[Rekleitis et al., 2004] Rekleitis, I., Lee-Shue, V., New, A. P. et Choset, H. (2004). Limited communication, multi-robot team based coverage. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 4, pages 3462–3468. IEEE.

[Rizzi et Koditschek, 1996] Rizzi, A. A. et Koditschek, D. E. (1996). An active visual estimator for dexterous manipulation. *IEEE Transactions on Robotics and Automation*, 12(5) :697–713.

[Roset et al., 2006] Roset, B., Lazar, M., Nijmeijer, H. et Heemels, W. (2006). Stabilizing output feedback nonlinear model predictive control : An extended observer approach. In *17th Symposium on Mathematical Theory for Networks and Systems. Kyoto, Japan.*

[Saha et Gustafsson, 2012] Saha, S. et Gustafsson, F. (2012). Particle filtering with dependent noise processes. *IEEE Transactions on Signal Processing*, 60(9) :4497–4508.

[Saied et al., 2015] Saied, M., Fantoni, I., Francis, C., Lussier, B. et Shraim, H. (2015). Détection et diagnostic d'une défaillance d'actionneur dans un octorotor. In *QUALITA'2015.*

[Samson et al., 1991] Samson, C., Espiau, B. et Borgne, M. L. (1991). *Robot control : the task function approach.* Oxford University Press.

[Santos et al., 2013] Santos, J. M., Portugal, D. et Rocha, R. P. (2013). An evaluation of 2d slam techniques available in robot operating system. In *Safety, Security, and Rescue Robotics (SSRR), 2013 IEEE International Symposium on*, pages 1–6. IEEE.

[Sassano et al., 2010] Sassano, M., Carnevale, D. et Astolfi, A. (2010). Observer design for range and orientation identification. *Automatica*, 46(8) :1369–1375.

[Sbnchez et Martinez, 2000] Sbnchez, A. et Martinez, J. M. (2000). Robot-arm pick and place behavior programming system using visual perception. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 4, pages 507–510. IEEE.

[Schmuck et al., 2016] Schmuck, P., Scherer, S. A. et Zell, A. (2016). Hybrid metric-topological 3d occupancy grid maps for large-scale mapping. *IFAC-PapersOnLine*, 49(15) :230–235.

[Sequeira et Ribeiro, 2004] Sequeira, J. et Ribeiro, M. I. (2004). Behavior-based control for semi-autonomous robots. *IFAC Proceedings Volumes*, 37(8) :18–23.

[Seraji et Howard, 2002] Seraji, H. et Howard, A. (2002). Behavior-based robot navigation on challenging terrain : A fuzzy logic approach. *IEEE Transactions on Robotics and Automation*, 18(3) :308–321.

[Shahriari et al., 2013] Shahriari, N., Fantasia, S., Flacco, F. et Oriolo, G. (2013). Robotic visual servoing of moving targets. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 77–82. IEEE.

[Shen et al., 2010] Shen, X., Huang, D. et Qin, X. (2010). A first step towards hybrid visual servoing control based on image moments. *Artificial Intelligence and Computational Intelligence*, pages 301–310.

[Shirai et Inoue, 1973] Shirai, Y. et Inoue, H. (1973). Guiding a robot by visual feedback in assembling tasks. *Pattern recognition*, 5(2) :99IN3107–106108.

[Siegwart et al., 2011] Siegwart, R., Nourbakhsh, I. R. et Scaramuzza, D. (2011). *Introduction to autonomous mobile robots*. MIT press.

[Skrabánek et al., 2016] Skrabánek, P., Vodicka, P. et Yildirim-Yayilgan, S. (2016). Control system of a semi-autonomous mobile robot. *IFAC-PapersOnLine*, 49(25) :460–469.

[Slotine et Li, 1991] Slotine, J.-J. E. et Li, W. (1991). *Applied Nonlinear Control*. Prentice Hall.

[Stéphant, 2004] Stéphant, J. (2004). *Contribution à l'étude et à la validation expérimentale d'observateurs appliqués à la dynamique du véhicule*. PhD thesis, Compiègne.

[Swingler et Ferrari, 2010] Swingler, A. et Ferrari, S. (2010). A cell decomposition approach to cooperative path planning and collision avoidance via disjunctive programming. In *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 6329–6336. IEEE.

[Tang et al., 2012] Tang, S., Khaksar, W., Ismail, N. et Ariffin, M. (2012). A review on robot motion planning approaches. *Pertanika Journal of Science and Technology*, 20(1) :15–29.

[Tarantino et al., 2000] Tarantino, R., Szigeti, F. et Colina-Morles, E. (2000). Generalized luenberger observer-based fault-detection filter design : an industrial application. *Control Engineering Practice*, 8(6) :665–671.

[Thrun, 2003] Thrun, S. (2003). Learning occupancy grid maps with forward sensor models. *Autonomous robots*, 15(2) :111–127.

[Vassallo et al., 2000] Vassallo, R. F., Schneebeli, H. J. et Santos-Victor, J. (2000). Visual servoing and appearance for navigation. *Robotics and Autonomous Systems*, 31(34) :87–97.

[Victorino et al., 2004a] Victorino, A. C., Rives, P. et Borrelly, J.-J. (2004a). Safe navigation for indoor mobile robots. part i : A sensor-based navigation framework. *The Int. Journal of Robotics Research*, 22(12) :1005–1118.

[Victorino et al., 2004b] Victorino, A. C., Rives, P. et Borrelly, J.-J. (2004b). Safe navigation for indoor mobile robots. part ii : Exploration, self localization and map building. *The Int. Journal of Robotics Research*, 22(12) :1019–1041.

[Vincent et Folorunso, 2009] Vincent, O. et Folorunso, O. (2009). A descriptive algorithm for sobel image edge detection. In *Proceedings of Informing Science & IT Education Conference (InSITE)*, volume 40, pages 97–107.

[Wang, 2013] Wang, B. (2013). *Observateurs d'état pour le diagnostic de comportement dynamique de véhicules automobiles en environnement réel de conduite*. PhD thesis, Compiègne.

[Weiss, 1984] Weiss, L. (1984). Dynamic visual servo control of robots : an adaptative image-based approach. these de doctorat.

[Weiss et al., 1987] Weiss, L., Sanderson, A. et Neuman, C. (1987). Dynamic sensor-based control of robots with visual feedback. *IEEE Journal on Robotics and Automation*, 3(5) :404–417.

[Welch et Bishop, 1997] Welch, G. et Bishop, G. (1997). Scaat : Incremental tracking with incomplete information. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 333–344. ACM Press/Addison-Wesley Publishing Co.

[Wenzel et al., 2006] Wenzel, T. A., Burnham, K., Blundell, M. et Williams, R. (2006). Dual extended kalman filter for vehicle state and parameter estimation. *Vehicle System Dynamics*, 44(2) :153–171.

[Wilmarth et al., 1999] Wilmarth, S. A., Amato, N. M. et Stiller, P. F. (1999). MAPRM : A probabilistic roadmap planner with sampling on the medial axis of the free space. In *1999 IEEE International Conference on Robotics and Automation, Marriott Hotel, Renaissance Center, Detroit, Michigan, May 10-15, 1999, Proceedings*, pages 1024–1031.

[Wung Choi et Elkaim, 2008] Wung Choi, J. et Elkaim, G. H. (2008). Bézier curve for trajectory guidance. In *World Congress on Engineering and Computer Science*.

[Xiong et al., 2015] Xiong, S., Wang, W., Liu, X., Chen, Z. et Wang, S. (2015). A novel extended state observer. *ISA transactions*, 58 :309–317.

[Yang et Kim, 1999] Yang, J.-M. et Kim, J.-H. (1999). Sliding mode control for trajectory tracking of nonholonomic wheeled mobile robots. *IEEE Transactions on robotics and automation*, 15(3) :578–587.

[Yang et Tsai, 1999] Yang, Z.-F. et Tsai, W.-H. (1999). Viewing corridors as right parallelepipeds for vision based vehicule localization. *IEEE Trans. on Industrial Electronics*, 46(3) :653–661.

[Ye et Wang, 2000] Ye, C. et Wang, D. (2000). A novel behavior fusion method for the navigation of mobile robots. In *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, volume 5, pages 3526–3531. IEEE.

[Yu et al., 2014] Yu, C., Cherfaoui, V. et Bonnifait, P. (2014). An evidential sensor model for velodyne scan grids. In *Control Automation Robotics & Vision (ICARCV), 2014 13th International Conference on*, pages 583–588. IEEE.

[Yuksel et al., 2011] Yuksel, C., Schaefer, S. et Keyser, J. (2011). Parameterization and applications of catmull–rom curves. *Computer-Aided Design*, 43(7) :747–755.

[Zeitz, 1987] Zeitz, M. (1987). The extended luenberger observer for nonlinear systems. *Systems & Control Letters*, 9(2) :149–156.

[Zhang et al., 2009] Zhang, Y., Zhao, Z., Lu, T., Yuan, L., Xu, W. et Zhu, J. (2009). A comparative study of luenberger observer, sliding mode observer and extended kalman filter for sensorless vector control of induction motor drives. In *Energy Conversion Congress and Exposition, 2009. ECCE 2009. IEEE*, pages 2466–2473. IEEE.

[Zhao et Guo, 2017] Zhao, Z.-L. et Guo, B.-Z. (2017). A novel extended state observer for output tracking of mimo systems with mismatched uncertainty. *IEEE Transactions on Automatic Control*.

Annexes

Matrice d'interaction d'un point

Nous présentons ici les briques nécessaires pour calculer la matrice d'interaction d'un point dans l'image qui va nous être utile pour le chapitre 2 et le chapitre 3.

A.1 Mouvement d'un point par rapport au capteur

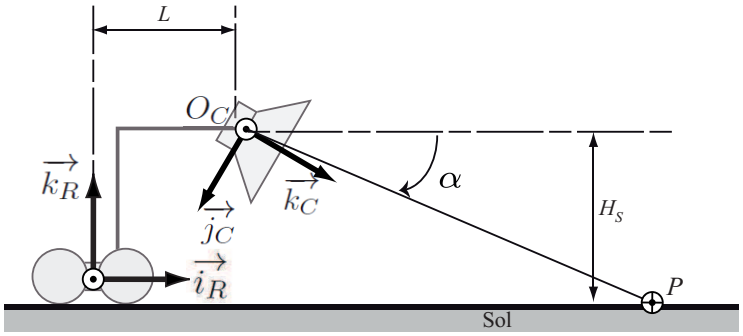


FIGURE A.1 – Variables utilisées pour le calcul.

Nous cherchons à identifier les mouvements d'un point fixe P 3D appartenant au sol vu par le capteur virtuel lorsque le robot se déplace.

$$\left[\frac{d}{dt}\overrightarrow{O_CP}\right]_{\mathcal{B}_C} = \begin{pmatrix}\dot{X}_P\\ \dot{Y}_P\\ \dot{Z}_P\end{pmatrix}_{\mathcal{B}_C} \tag{A.1}$$

Il est également possible d'écrire cela relativement au monde et aux mouvements du robot.

$$\begin{aligned}\left[\frac{d}{dt}\overrightarrow{O_CP}\right]_{\mathcal{B}_C} &= \left[\frac{d}{dt}\left(\overrightarrow{O_CO_W}+\overrightarrow{O_WP}\right)\right]_{\mathcal{B}_C}\\ &= \left[\frac{d}{dt}\overrightarrow{O_CO_W}\right]_{\mathcal{B}_W} + \overrightarrow{\Omega}_{\mathcal{B}_W/\mathcal{B}_C}\wedge\overrightarrow{O_CP} + \left[\frac{d}{dt}\overrightarrow{O_WP}\right]_{\mathcal{B}_W}\\ &= -\left[\frac{d}{dt}\overrightarrow{O_WO_C}\right]_{\mathcal{B}_W} - \overrightarrow{\Omega}_{\mathcal{B}_C/\mathcal{B}_W}\wedge\overrightarrow{O_CP} + \vec{0}\\ &= -\overrightarrow{V}_{O_C\in\Sigma_C/\mathcal{R}_W} - \overrightarrow{\Omega}_{\mathcal{B}_C/\mathcal{B}_W}\wedge\overrightarrow{O_CP}\end{aligned} \tag{A.2}$$

On écrit (A.2) en explicitant les coordonnées

$$\left[\frac{d}{dt}\overrightarrow{O_C P}\right]_{\mathcal{B}_C} = -\begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix}_{\mathcal{B}_C} - \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix}_{\mathcal{B}_C} \wedge \begin{pmatrix} X_P \\ Y_P \\ Z_P \end{pmatrix}_{\mathcal{B}_C} \tag{A.3}$$

En identifiant (A.1) et (A.3), on écrit la formule générale

$$\begin{pmatrix} \dot{X}_P \\ \dot{Y}_P \\ \dot{Z}_P \end{pmatrix}_{\mathcal{B}_C} = \begin{pmatrix} -v_x - \omega_y Z_P + \omega_z Y_P \\ -v_y - \omega_z X_P + \omega_x Z_P \\ -v_z - \omega_x Y_P + \omega_y X_P \end{pmatrix}_{\mathcal{B}_C} \tag{A.4}$$

La vitesse linéaire de la caméra dans le repère monde est

$$\begin{aligned} \overrightarrow{V}_{O_C \in S_C/\mathcal{F}_W} &= \left[\frac{d}{dt}\overrightarrow{O_W O_C}\right]_{\mathcal{B}_W} = \left[\frac{d}{dt}\overrightarrow{O_W O_R}\right]_{\mathcal{B}_W} + \left[\frac{d}{dt}\overrightarrow{O_R O_C}\right]_{\mathcal{B}_R} + \overrightarrow{\Omega}_{\mathcal{B}_R/\mathcal{B}_W} \wedge \overrightarrow{O_R O_C} \\ &= \begin{pmatrix} v - W\dot{\theta}_R \\ L\dot{\theta}_R \\ 0 \end{pmatrix}_{\mathcal{B}_R} = \begin{pmatrix} -L\dot{\theta}_R \\ \left(v - W\dot{\theta}_R\right) sin(\alpha) \\ \left(v - W\dot{\theta}_R\right) cos(\alpha) \end{pmatrix}_{\mathcal{B}_C} \end{aligned} \tag{A.5}$$

Dans le cas d'une caméra perspective frontale, en remplaçant par les expressions de (A.5)

$$\begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix}_{\mathcal{B}_C} = \begin{pmatrix} -L.\dot{\theta}_R \\ 0 \\ v - W.\dot{\theta}_R \end{pmatrix}_{\mathcal{B}_C}$$

$$\begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix}_{\mathcal{B}_C} = \begin{pmatrix} 0 \\ -\dot{\theta}_R \\ 0 \end{pmatrix}_{\mathcal{B}_C}$$

Nous avons le vecteur $\mathbf{U}$ composé de la vitesse linéaire et la vitesse de rotation du robot. $\mathbf{U} = \begin{pmatrix} v & \omega \end{pmatrix}^T$.

Les mouvements s'écrivent alors en fonction des commandes

$$\begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix}_{\mathcal{B}_C} = \begin{pmatrix} -L.\omega \\ 0 \\ v - W.\omega \end{pmatrix}_{\mathcal{B}_C} \tag{A.6}$$

$$\begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix}_{\mathcal{B}_C} = \begin{pmatrix} 0 \\ -\omega \\ 0 \end{pmatrix}_{\mathcal{B}_C} \tag{A.7}$$

On obtient l'expression des variations des coordonnées du point P fixe dans le monde, exprimées

dans la base de la caméra, en fonction des commandes appliquées du robot

$$\begin{pmatrix} \dot{X}_P \\ \dot{Y}_P \\ \dot{Z}_P \end{pmatrix}_{\mathcal{B}_C} = \begin{pmatrix} \omega(L+Z_P) \\ 0 \\ -v+\omega(W-X_P) \end{pmatrix}_{\mathcal{B}_C} \tag{A.8}$$

A.2 Mouvement d'un point du monde dans l'image

A.2.1 Mouvement d'un point quelconque

Nous prenons un modèle de caméra perspective. Dans ce cas, nous rappelons que les coordonnées métriques dans l'image telles que

$$x_P = \frac{X_P}{Z_P} \tag{A.9}$$

$$y_P = \frac{Y_P}{Z_P} \tag{A.10}$$

Les variations des coordonnées de P dans l'image lorsque le robot bouge s'écrivent alors

$$\begin{aligned} \frac{d}{dt}(x_P) &= \frac{d}{dt}\left(\frac{X_P}{Z_P}\right) \\ &= \frac{1}{Z_P}\frac{d}{dt}(X_P) - \frac{X_P}{Z_P^2}\frac{d}{dt}(Z_P) \\ &= \frac{1}{Z_P}\dot{X}_P - \frac{X_P}{Z_P^2}\dot{Z}_P \end{aligned}$$

$$\begin{aligned} \frac{d}{dt}(y_P) &= \frac{d}{dt}\left(\frac{Y_P}{Z_P}\right) \\ &= \frac{1}{Z_P}\frac{d}{dt}(Y_P) - \frac{Y_P}{Z_P^2}\frac{d}{dt}(Z_P) \\ &= \frac{1}{Z_P}\dot{Y}_P - \frac{Y_P}{Z_P^2}\dot{Z}_P \end{aligned}$$

En utilisant l'expression générale des mouvements de P vus de la caméra (A.4)

$$\begin{aligned} \dot{x}_P &= \frac{1}{Z_P}(-v_x - \omega_y Z_P + \omega_z Y_P) - \frac{X_P}{Z_P^2}(-v_z - \omega_x Y_P + \omega_y X_P) \\ &= \begin{pmatrix} -\frac{1}{Z_P} & 0 & \frac{X_P}{Z_P^2} \end{pmatrix}\begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} + \begin{pmatrix} \frac{X_P Y_P}{Z_P^2} & -\left(\frac{X_P^2}{Z_P^2}+1\right) & \frac{Y_P}{Z_P} \end{pmatrix}\begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} \end{aligned}$$

En réutilisant le modèle de caméra (A.9) et (A.10), on obtient alors la relation liant les mouvements du point dans l'image en fonction des anciennes coordonnées, des mouvements du robot et de la distance

de la caméra au point (Z_P).

$$\dot{x}_P = \begin{pmatrix} -\frac{1}{Z_P} & 0 & \frac{x_P}{Z_P} \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} + \begin{pmatrix} x_P.y_P & -(x_P^2+1) & y_P \end{pmatrix} \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} \tag{A.11}$$

$$\begin{aligned} \dot{y}_P &= \frac{1}{Z_P}(-v_y - \omega_z X_P + \omega_x Z_P) - \frac{Y_P}{Z_P^2}(-v_z - \omega_x Y_P + \omega_y X_P) \\ &= \begin{pmatrix} 0 & -\frac{1}{Z_P} & \frac{Y_P}{Z_P^2} \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} + \begin{pmatrix} \left(1+\frac{Y_P^2}{Z_P^2}\right) & -\frac{X_P Y_P}{Z_P^2} & -\frac{X_P}{Z_P} \end{pmatrix} \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} \end{aligned}$$

Soit en réutilisant le modèle de caméra (A.9) et (A.10)

$$\dot{y}_P = \begin{pmatrix} 0 & -\frac{1}{Z_P} & \frac{y_P}{Z_P} \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} + \begin{pmatrix} (1+y_P^2) & -x_P.y_P & -x_P \end{pmatrix} \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} \tag{A.12}$$

A.3 Mouvement du point de fuite

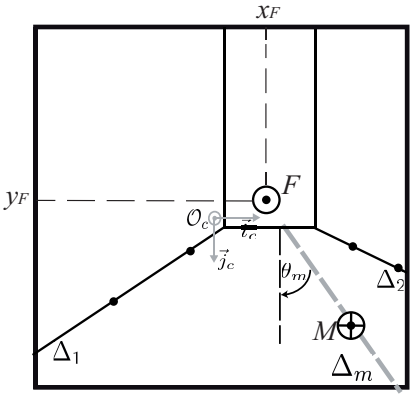


FIGURE A.2 – Illustration du point de fuite dans une image perspective.

Pour une caméra perspective, les droites d'intersection des murs et du sol convergent vers un point que nous noterons F. Ce point n'existe pas, il est le résultat de la déformation induite par la caméra. Les coordonnées de ce point le repère caméra plan image sont (x_F, y_F). Ce point est également rejeté à l'infini dans le repère de la caméra $Z_F = \infty$. Lorsque le robot se déplace, le point de convergence des murs se déplace dans l'image. Ses variations peuvent s'écrire grâce aux relations (A.11) et (A.12) en utilisant les coordonnées d'un point F.

$$\dot{x}_F = \begin{pmatrix} -\frac{1}{Z_F} & 0 & \frac{x_F}{Z_F} \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} + \begin{pmatrix} x_F.y_F & -(x_F^2+1) & y_F \end{pmatrix} \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} \tag{A.13}$$

$$\dot{y}_F = \begin{pmatrix} 0 & -\frac{1}{Z_F} & \frac{y_F}{Z_F} \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} + \begin{pmatrix} (1+y_F^2) & -x_F.y_F & -x_F \end{pmatrix} \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} \tag{A.14}$$

En utilisant le fait que $Z_F = \infty$, hypothèse clef de la méthode,

$$\dot{x}_F = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} + \begin{pmatrix} x_F.y_F & -(x_F^2+1) & y_F \end{pmatrix} \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} \tag{A.15}$$

$$\dot{y}_F = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} + \begin{pmatrix} (1+y_F^2) & -x_F.y_F & -x_F \end{pmatrix} \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} \tag{A.16}$$

En injectant la relation des mouvements du robot à partir des commandes (A.7), on obtient les mouvements du point de convergence des murs en fonction de leurs positions et de la commande en rotation du robot.

$$\dot{x}_F = \omega(x_F^2+1) \tag{A.17}$$

$$\dot{y}_F = \omega x_F y_F \tag{A.18}$$

On peut noter ici que la vitesse longitudinale du robot commandée par v n'a pas d'influence sur ces deux variables. Seule la rotation du robot permet de déplacer la position de ce point dans l'image. La convergence de x_F vers 0 va impliquer la convergence de y_F également vers 0.

Modèle de la caméra et d'une droite au sol

Cette partie nous permet de calculer la matrice d'interaction de l'évolution d'une droite dans une image. Soit M_W un point du monde appartenant à la ligne médiane du couloir. Les coordonnées de ce point dans le repère caméra s'écrivent :

$$\overrightarrow{O_C M_W} = \begin{pmatrix} X_M \\ Y_M \\ Z_M \end{pmatrix}_{\mathcal{B}_C}$$

Il est possible de décrire cette droite dans le repère de la caméra comme étant l'intersection de 2 plans.

$$\begin{cases} a_1 X_M + b_1 Y_M + c_1 Z_M + d_1 &= 0 \\ a_2 X_M + b_2 Y_M + c_2 Z_M + d_2 &= 0 \end{cases} \tag{B.1}$$

En supposant que le sol est parallèle au plan $\left(\overrightarrow{j_C}, \overrightarrow{k_C}\right)$ de la caméra, on peut écrire

$$\begin{cases} Y_M &= H_s \\ a_2 X_M + b_2 Y_M + c_2 Z_M + d_2 &= 0 \end{cases} \tag{B.2}$$

où H_s représente la hauteur de la caméra par rapport au sol.

En appliquant le modèle de la caméra perspective, on obtient les coordonnées du point M, projection de M_W dans le plan image, exprimées dans la base de la caméra par

$$\overrightarrow{O_C M} = \begin{pmatrix} x_M \\ y_M \\ f \end{pmatrix}_{\mathcal{B}_C}$$

où

$$x_M = \frac{X_M}{Z_M} \tag{B.3}$$

$$y_M = \frac{Y_M}{Z_M} \tag{B.4}$$

avec f la distance focale de la caméra. On obtient alors une représentation telle que celle de la B.1.

La ligne médiane du couloir dans le plan image est portée par la droite notée Δ_M de la figure B.1.

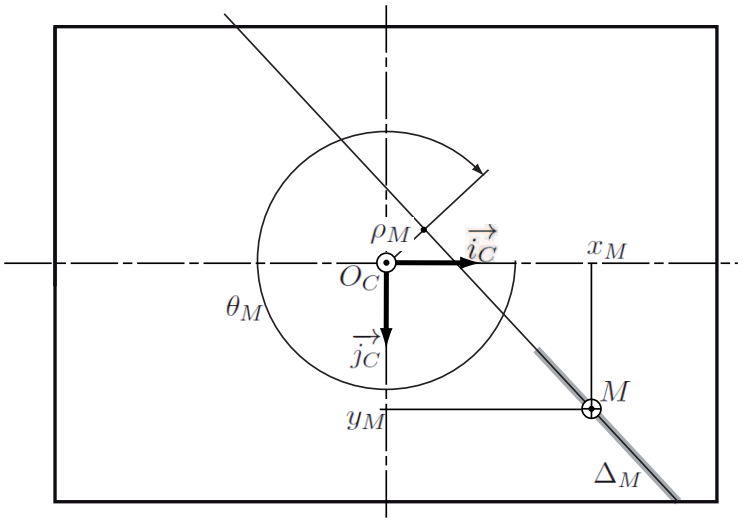


FIGURE B.1 – Modélisation dans le repère caméra, au niveau du plan image. Le point de vue est placé au centre de la caméra et regarde vers l'avant de la scène.

Cette droite peut être paramétrée par les variables ρ_M et θ_M telles que θ_M soit l'angle entre le vecteur $\overrightarrow{i_C}$ et la normale à cette droite passant par la projection de O_C sur le plan image. ρ_M représente la distance de Δ_M à la projection de O_C sur le plan image.

Dans ce cas, le point M se décrit comme

$$x_M \cos(\theta_M) + y_M \sin(\theta_M) - \rho_M = 0 \tag{B.5}$$

De part la représentation utilisée, lorsque la droite Δ_M est verticale dans l'image, cela indique que l'axe de la caméra est aligné sur la ligne médiane du couloir.

Afin de construire l'asservissement visuel, il faudra donc décrire comment cette droite évolue en fonction des mouvements du torseur cinématique de la caméra par rapport au repère monde et au centre de la caméra $\left\{\Upsilon_{\Sigma_C/\mathcal{R}_W}\right\}$ (2.17). Nous devons donc trouver une relation décrivant $\dot{\theta}_M = f_\theta\left(\left\{\Upsilon_{\Sigma_C/\mathcal{R}_W}\right\}\right)$ et $\dot{\rho}_M = f_\rho\left(\left\{\Upsilon_{\Sigma_C/\mathcal{R}_W}\right\}\right)$.

En dérivant l'équation de la droite (B.5) par rapport au temps, on obtient

$$\dot{\rho}_M + \dot{\theta}_M \left(x_M \sin(\theta_M) - y_M \cos(\theta_M)\right) = \dot{x}_M \cos(\theta_M) + \dot{y}_M \sin(\theta_M) \tag{B.6}$$

Le terme de gauche peut se reformuler pour faire disparaître une des deux variables, en utilisant l'équation (B.5)

$$x_M = \frac{\rho_M}{\cos(\theta_M)} - y_M \tan(\theta_M) \tag{B.7}$$

Dans ce cas,

$$\begin{aligned} & \dot{\rho}_M + \dot{\theta}_M \left(x_M \sin(\theta_M) - y_M \cos(\theta_M)\right) \\ = \;& \dot{\rho}_M + \dot{\theta}_M \left(\left(\frac{\rho_M}{\cos(\theta_M)} - y_M \tan(\theta_M)\right) \sin(\theta_M) - y_M \cos(\theta_M)\right) \\ = \;& \dot{\rho}_M + \dot{\theta}_M \rho_M \tan(\theta_M) - y_M \frac{\dot{\theta}_M}{\cos(\theta_M)} \end{aligned} \tag{B.8}$$

Le terme de droite de l'équation (B.6) peut être développé en utilisant l'expression des déplacements de la projection du point M_W sur le plan image (équations (A.11) et (A.12) en considérant le point M_W)

$$\dot{x}_M = \begin{pmatrix} -\frac{1}{Z_M} & 0 & \frac{x_M}{Z_M} \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} + \begin{pmatrix} x_M.y_M & -(x_M^2+1) & y_M \end{pmatrix} \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} \tag{B.9}$$

$$\dot{y}_M = \begin{pmatrix} 0 & -\frac{1}{Z_M} & \frac{y_M}{Z_M} \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} + \begin{pmatrix} (1+y_M^2) & -x_M.y_M & -x_M \end{pmatrix} \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} \tag{B.10}$$

En utilisant la formulation du torseur cinématique, on obtient alors l'expression suivante pour le terme de droite de l'équation (B.6)

$$\begin{aligned} & \dot{x}_M \cos(\theta_M) + \dot{y}_M \sin(\theta_M) \\ = & \begin{pmatrix} -\frac{1}{Z_M} & 0 & \frac{x_M}{Z_M} & x_M.y_M & -(x_M^2+1) & y_M \end{pmatrix} \cos(\theta_M)\mathbf{T} \\ & + \begin{pmatrix} 0 & -\frac{1}{Z_M} & \frac{y_M}{Z_M} & (1+y_M^2) & -x_M.y_M & -x_M \end{pmatrix} \sin(\theta_M)\mathbf{T} \end{aligned} \tag{B.11}$$

A partir de l'équation d'un plan quelconque dans le repère caméra (B.1), on peut écrire la relation suivante

$$\frac{1}{Z_M} = -\left(a_1 x_M + b_1 y_M + c_1\right) \frac{1}{d_1} \tag{B.12}$$

En réutilisant la formulation (B.7) pour x_M, on a

$$\begin{aligned} \frac{1}{Z_M} &= -\left(a_1 \left(\frac{\rho_M}{\cos(\theta_M)} - y_M \tan(\theta_M) \right) + b_1 y_M + c_1 \right) \frac{1}{d_1} \\ &= -\left(\left(c_1 + a_1 \left(\frac{\rho_M}{\cos(\theta_M)} \right) \right) + y_M \left(b_1 - a_1 \tan(\theta_M) \right) \right) \frac{1}{d_1} \end{aligned} \tag{B.13}$$

En notant

$$\lambda_2 = -\frac{c_1 + a_1 \left(\frac{\rho_M}{\cos(\theta_M)} \right)}{d_1} \tag{B.14}$$

et

$$\lambda_1 = \frac{b_1 - a_1 \tan(\theta_M)}{d_1} \tag{B.15}$$

Alors (B.12) devient

$$\frac{1}{Z_M} = -\left(\lambda_2 + \lambda_1 y_M\right) \tag{B.16}$$

Dans notre cas, utilisant l'équation du plan du sol dans le repère caméra, (B.2), on obtient une expression de λ_1.

$$\lambda_1 = -\frac{1}{H_s} \tag{B.17}$$

En injectant l'expression précédente dans l'équation (B.11)

$$\dot{x}_M \cos(\theta_M) + \dot{y}_M \sin(\theta_M) = y_M K_1 \mathbf{T} + K_2 \mathbf{T} \tag{B.18}$$

avec

$$K_1 = \begin{pmatrix} \lambda_1 \cos(\theta_M) & \lambda_1 \sin(\theta_M) & -\lambda_1 \rho_M & \rho_M & \rho_M \tan(\theta_M) & \dfrac{1}{\cos(\theta_M)} \end{pmatrix} \tag{B.19}$$

et

$$K_2 = \begin{pmatrix} \lambda_2 \cos(\theta_M) & \lambda_2 \sin(\theta_M) & -\lambda_2 \rho_M & \sin(\theta_M) & \left(-\cos(\theta_M) - \dfrac{\rho_M^2}{\cos(\theta_M)} \right) & -\rho_M \tan(\theta_M) \end{pmatrix} \tag{B.20}$$

$$K_1 = \begin{pmatrix} -\frac{\cos(\theta_m)}{H_s} & -\frac{\sin(\theta_m)}{H_s} & \frac{\rho_m}{H_s} & \rho_m & \rho_m \tan(\theta_m) & \dfrac{1}{\cos(\theta_m)} \end{pmatrix} \tag{B.21}$$

et

$$K_2 = \begin{pmatrix} 0 & 0 & 0 & \sin(\theta_M) & \left(-\cos(\theta_m) - \dfrac{\rho_m^2}{\cos(\theta_m)} \right) & -\rho_M \tan(\theta_m) \end{pmatrix} \tag{B.22}$$

Identifiant maintenant les deux termes de l'équation (B.6)

$$-y_M \frac{\dot{\theta}_M}{\cos(\theta_M)} + \dot{\rho}_M + \dot{\theta}_M \rho_M \tan(\theta_M) = y_M K_1 \mathbf{T} + K_2 \mathbf{T} \tag{B.23}$$

Par identification, on obtient alors

$$\dot{\theta}_M = -K_1 \mathbf{T} \cos(\theta_M) \tag{B.24}$$

puis

$$\dot{\rho}_M = K_2 \mathbf{T} + \rho_M \sin(\theta_M) K_1 \mathbf{T} \tag{B.25}$$

En injectant la relation des mouvements du robot à partir des commandes, (A.7), on obtient finalement les mouvements de ces deux variables en fonction de leur position et des commandes en vitesse appliquées au robot.

Méthodes d'observation

Nous rappelons dans cette section le principe des méthodes d'estimation utilisés. Pour construire un observateur il faut un modèle décrivant l'évolution de l'état $\mathbf{x} \in \mathbb{R}^n$ par rapport au temps. Pour modéliser chaque système il nous faut la commande $\mathbf{u} \in \mathbb{R}^q$ et les mesures $\mathbf{y} \in \mathbb{R}^m$ qui représente la sortie du système.

C.1 Observateurs linéaires

Un système linéaire invariant dans le temps est décrit par le modèle suivant :

$$\left\{\begin{array}{rcc} \dot{\mathbf{x}}(t) & = & \mathbf{A}\,\mathbf{x}(t) + \mathbf{B}\,\mathbf{u}(t) \\ \mathbf{y}(t) & = & \mathbf{C}\,\mathbf{x}(t) + \mathbf{D}\,\mathbf{u}(t) \\ \mathbf{x}(t=t_0) & = & \mathbf{x}_0 \end{array}\right. \tag{C.1}$$

Avec les matrices réelles $A \in \mathbb{R}^{n*n}$, $B \in \mathbb{R}^{n*p}$, $C \in \mathbb{R}^{m*n}$, $D \in \mathbb{R}^{m*q}$. Généralement dans la réalité les systèmes linéaires n'existent pas tous les systèmes réels sont décrit par une non linéarité. Pour rendre les calculs et la modélisation plus simple, un système non linéaire peut être linéarisé autour d'un point d'équilibre. Cette linéarisation décrit correctement le système non linéaire au voisinage de ce point d'équilibre.
Pour construire un observateur d'état il faut faire une étude d'observabilité pour savoir s'il est possible d'estimer l'état.
Le système C.1 est observable si l'observation de ses entrées et de ses sorties pendant un temps fini $t_k \in [t_i\, t_f]$ permet de déterminer l'état initial $x(t_i)$ et donc connaitre $x(t)$ par intégration de l'équation de l'évolution de l'état à tout instant t_k. Il faut trouver que la matrice d'observabilité $\mathcal{O}_n = \left(\mathbf{C} \quad \mathbf{CA} \quad \mathbf{CA^2} \quad ... \quad \mathbf{CA^{n-1}}\right)^T$ est de rang n. Le problème de l'observation des systèmes linéaires a été résolu par l'estimation de Luenberger [Luenberger, 1964], [Luenberger, 1966], [Luenberger, 1971]. Cette estimation est définit par le calcul d'un gain $\mathbf{K}$ (C.2). Ce gain représente une pondération de l'état estimé $\hat{\mathbf{x}}$ par l'erreur entre la mesure réelle $\mathbf{y}$ et la mesure estimée $\hat{\mathbf{y}}$.

$$\left\{\begin{array}{rcc} \dot{\mathbf{x}} & = & \mathbf{A}\,\mathbf{x} + \mathbf{B}\,\mathbf{u} \\ \mathbf{y} & = & \mathbf{C}\,\mathbf{x} \\ \dot{\hat{\mathbf{x}}}(t) & = & \mathbf{A}\,\hat{\mathbf{x}} + \mathbf{B}\,\mathbf{u} + \mathbf{K}\,(\mathbf{y} - \hat{\mathbf{y}}) \\ \hat{\mathbf{y}} & = & \mathbf{C}\,\hat{\mathbf{x}} \end{array}\right. \tag{C.2}$$

L'objectif d'un observateur est de faire converger l'état estimé vers la vraie valeur de l'état. Ceci

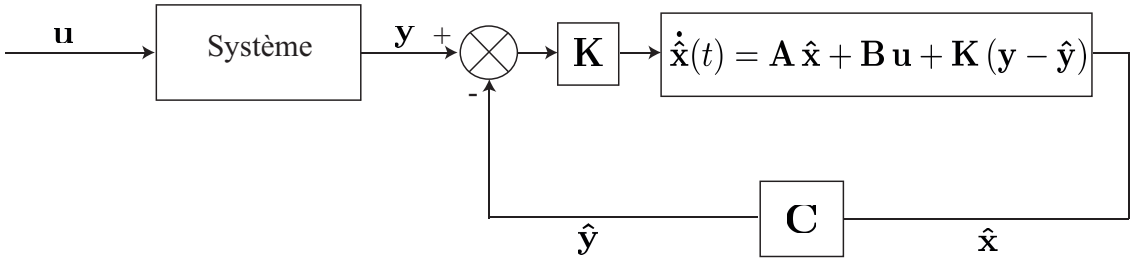


FIGURE C.1 – Schéma d'un observateur linéaire)

peut se traduire de la manière suivante :

$$\lim_{t \to +\infty} (\hat{\mathbf{x}} - \mathbf{x})\,(t) = 0 \tag{C.3}$$

En définissant l'erreur d'estimation d'état par $\epsilon_{\mathbf{x}}$. Cette dernière peut être calculée à partir de (C.2)

$$\dot{\varepsilon_{\mathbf{x}}} = (\mathbf{A} - \mathbf{KC})\,\varepsilon_{\mathbf{x}} \tag{C.4}$$

Pour créer un observateur linéaire il faut trouver une matrice $\mathbf{K}$ qui assure la convergence définie dans (C.3). La matrice $\mathbf{K}$ peut être calculée si le paire $(\mathbf{A}, \mathbf{C})$ qui décrit le système (C.2) est observable telle que toutes les valeurs propres de la matrice $(\mathbf{A} - \mathbf{KC})$ ont une partie réelle strictement négative : $\Re[\lambda_i] < 0$ pour toute valeur propre λ_i de $(\mathbf{A} - \mathbf{KC})$.

C.2 Filtre de Kalman

Le filtre de Kalman est une approche qui sert à estimer des variables d'un système évoluant dans le temps à partir de mesures bruitées. Ce filtre a été initialement conçu pour les systèmes discrets, puis il a été étendu aux systèmes continus [Durrant-Whyte et al., 2001b]. L'avantage de ce filtre est sa capacité de prédiction des variables et de correction par rapport aux erreurs, non seulement des capteurs, mais aussi du modèle d'évolution.
Le but est d'estimer un vecteur aléatoire $\mathbf{Z}$.

Pour le filtre de Kalman nous utiliserons des variables suivantes :
- x_k l'état réel à l'instant k
- $\hat{x}_{k|k}$ l'estimation de x_k en considérant les mesures jusqu'à l'instant k.
- $\mathbf{P}_{k|k} = \mathbf{E}\left[\left(\hat{x}_{k|k} - x_k\right)\left(\hat{x}_{k|k} - x_k\right)^T\right]$ la covariance a posteriori sur l'estimation du vecteur x_k.
- $\hat{x}_{k|k-1}$ la prédiction de x_k en considérant les mesures jusqu'à l'instant $k-1$.
- $\mathbf{P}_{k|k-1} = \mathbf{E}\left[\left(\hat{x}_{k|k-1} - x_{k-1}\right)\left(\hat{x}_{k|k-1} - x_{k-1}\right)^T\right]$ la covariance a priori sur l'estimation du vecteur x_{k-1}.

Le modèle complet auquel le filtrage de Kalman s'applique est :

$$\begin{cases} \mathbf{x}_k &= \mathbf{F}_k\,\mathbf{x}_{k-1} + \mathbf{G}_k\,\mathbf{u}_k + v_k \\ \mathbf{y}_k &= \mathbf{C}_k\,\mathbf{x}_k + w_k \end{cases} \tag{C.5}$$

avec $\mathbf{x}_k \in \mathbb{R}^n$, $\mathbf{y}_k \in \mathbb{R}^m$, $\mathbf{u}_k \in \mathbb{R}^p$.
Le bruit d'état $v_k \in \mathbb{R}^n$ et le bruit de mesure $w_k \in \mathbb{R}^m$ sont des bruits gaussiens, blancs, et indépendants de covariances $\mathbf{Q}_k \in \mathbb{R}^{n\times n}$ (resp.$\mathbf{R}_k \in \mathbb{R}^{m\times m}$) connues.
Avec ce filtre nous cherchons à chaque itération :

- $\hat{x}$ la meilleure estimation de l'état $\mathbf{x}_k$
- $\mathbf{P} = \mathbb{E}\left[\tilde{x}\tilde{x}^T\right]$ la covariance de l'erreur d'estimation $\tilde{x} = x - \hat{x}$
- $\mathbf{S} = \mathbb{E}\left[\varepsilon\varepsilon^T\right]$ l'erreur sur la mesure.

L'algorithme d'un filtrage de Kalman est réalisé en plusieurs étapes. La première consiste à faire l'initialisation de l'état estimé $\hat{x}_{0|-1} = x_0$, la matrice de variance-covariance $\mathbf{P} = \mathbf{P}_0$, les matrices de bruits $\mathbf{Q}$ et $\mathbf{R}$
Une fois qu'une mesure $\mathbf{y}_k$ arrive à partir des capteurs, le calcul de l'erreur de mesure se fait (calcul de l'innovation).

$$\varepsilon_k = y_k - \hat{y}_{k|k-1} \tag{C.6}$$

Puis il s'agit du calcul de la covariance de cette erreur de mesure $\mathbf{S}_{k|k-1}$

$$\mathbf{S}_{k|k-1} = C_k.\hat{\mathbf{P}}_{k|k-1}.C_k^T + R_k \tag{C.7}$$

Avec la covariance de l'erreur de mesure $\mathbf{S}_{k|k-1}$ nous pouvons tester si la mesure z_k est bien conforme au modèle ou inversement. Soit $\mathcal{D}$ le domaine d'acceptation tel que si $\varepsilon_k \in \mathcal{D}$ on conclue la compatibilité de modèle/mesure. Ce domaine est défini par l'ellipsoïde de centre $\hat{y}_{k|k-1}$ et dont les axes principaux sont les vecteurs propres de la matrice $\mathbf{S}_{k|k-1}$.

$$\varepsilon_k^T \mathbf{S}_{k|k-1}^{-1} \varepsilon_k \leq \mathcal{X}^2(P_r) \tag{C.8}$$

avec $\mathcal{X}(P_r)$ est un seuil multidimensionnel sous la forme d'un scalaire, il s'agit de la probabilité de réalisation P_r de y_k. Nous utilisons l'inégalité (C.8) comme un test de validité de l'innovation. Si l'innovation n'assure pas la condition (C.8) c'est-à-dire la mesure réalisée et la prédiction par le modèle ne sont pas compatibles. Sinon l'innovation est valide et on peut corriger l'estimation de l'état :
a) calcul du gain de correction

$$K_k = \mathbf{P}_{k|k-1}.C_k^T.\mathbf{S}_{k|k-1}^{-1} \tag{C.9}$$

b) correction de la prédiction

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k.\varepsilon_k \tag{C.10}$$

c) évaluation de la nouvelle matrice de variance covariance de l'erreur d'estimation

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - K_k.C_k.\mathbf{P}_{k|k-1} \tag{C.11}$$

Puis il y a l'étape de la prédiction il faut prédire l'état, sa covariance et la mesure pour la prochaine itération

$$\left\{\begin{array}{rcl} \hat{\mathbf{x}}_{k+1|k} & = & \mathbf{F}_k\,\hat{\mathbf{x}}_{k|k} + \mathbf{G}_k\,\mathbf{u}_k \\ \mathbf{P}_{k+1|k} & = & \mathbf{F}_k.\mathbf{P}_{k|k}.\mathbf{F}_k^T + \mathbf{Q}_k \\ \mathbf{y}_{k+1|k} & = & \mathbf{C}_k\,\mathbf{x}_{k+1|k} \end{array}\right. \tag{C.12}$$

En conclusion, pour appliquer un filtre de Kalman, il faut avant tout obtenir un modèle linéaire. Son avantage est de délivrer en plus de l'estimation de l'état, un ellipsoïde de confiance de cette estimation. Les coûts de calcul sont principalement dus au calcul du gain de Kalman (C.9), puisqu'il s'agit d'inversion de matrice de taille du vecteurs de mesures y_k.
Dans le cas ou les bruits sur le modèle ou sur la mesure sont non-blancs, le filtre de Kalman produit une estimation biaisée. Il existe des solutions pour ce problème si la fonction de corrélation des bruits est connue [Durrant-Whyte et al., 2001a].
Dans le cas où le modèle est non linéaire, nous ne pouvons plus appliquer les relations précédentes. Il s'agit de la limitation principale du filtre de Kalman. Des approches existent pour traiter ces problèmes, c'est le cas notamment du filtre de Kalman étendu (Extended Kalman Filter : EKF).

C.3 Filtre de Kalman étendu

Dans ce paragraphe nous abordons le problème de filtrage au cas où la dynamique du système (état et observation) n'est pas linéaire. Considérons la dynamique non linéaire suivante :

$$\begin{cases} \mathbf{x}_k &= f(x_k, u_k) + v_k \\ \mathbf{y}_k &= h(x_k) + w_k \end{cases} \tag{C.13}$$

Les fonctions f et h doivent être dérivables.
Le filtre de Kalman étendu donne une approximation de l'estimation optimale. Les non-linéarités de la dynamique de système sont approximées par une version linéarisée du modèle du système non linéaire autour de la dernière estimation de l'état. Cette linéarisation doit être valide le domaine d'incertitude associé à l'estimation de l'état.
Les étapes pour construire ce filtre sont :

1. Mesure de la mesure y_k
2. Calcul de l'innovation $\varepsilon = y_k - \hat{y}_k$
3. Calcul de la covariance de l'erreur de la mesure $S = CPC^T + R$
4. Test de validité de la mesure (C.8)
5. Si la mesure est valide il faut faire la correction de l'estimation de l'état
   (a) Calcul du gain $K = PC^T S^{-1}$
   (b) Correction par l'innovation $\hat{x} = \hat{x} + K\varepsilon$
   (c) Calcul du linéarisé du modèle d'évolution $C = [H_x]_{x=\hat{x}}$
   (d) Adaptation de la matrice de variance covariance de l'erreur d'estimation $P = P - KCP$
6. Prédiction de l'état pour l'itération suivante $\hat{x}_{k+1|k} = f(\hat{x}, u)$
7. Calcul du linéarisé du modèle d'évolution $F = [F_x]_{x=\hat{x}_{k+1|k}}$
8. Calcul de la covariance de l'erreur d'estimation $P = FPF^T + Q$
9. Prédiction de la mesure $\hat{y}_{k+1|k} = h(\hat{x})$

Explication du choix de l'observateur de la signature

Cette partie nous intéresse pour la construction de l'observateur dans le chapitre 2.
L'état est constitué de la signature

$$\mathbf{X} = \begin{pmatrix} x_f & y_f & \theta_m \end{pmatrix}^T = \begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix}^T$$

L'état est mesuré grâce aux principes explicités à la section B où x_F, y_F et θ_m sont extraits dans l'image après une extraction des droites supports des intersection murs/sol.

$$\mathbf{Y} = \begin{pmatrix} x_f & y_f & \theta_M \end{pmatrix}^T = \begin{pmatrix} y_1 & y_2 & y_3 \end{pmatrix}^T$$

La commande est :

$$\mathbf{U} = \begin{pmatrix} v & \omega \end{pmatrix}^T = \begin{pmatrix} u_1 & u_2 \end{pmatrix}^T$$

La dynamique du système et l'équation d'observation s'écrit

$$\Sigma \begin{cases} \dot{\mathbf{X}} &= f(\mathbf{X}, \mathbf{U}) \\ \mathbf{Y} &= \mathbf{X} \end{cases} \tag{D.1}$$

où

$$f(\mathbf{X}, \mathbf{U}) = \begin{pmatrix} f_1(\mathbf{X}, \mathbf{U}) \\ f_2(\mathbf{X}, \mathbf{U}) \\ f_3(\mathbf{X}, \mathbf{U}) \end{pmatrix}$$

explicitées par (A.17), (A.18) et (B.24) (dans laquelle ρ_M est remplacé par l'expression (2.31))

$$f_1(\mathbf{X}, \mathbf{U}) = -u_2 \left(1 + x_1^2\right) \tag{D.2}$$

$$f_2(\mathbf{X}, \mathbf{U}) = u_2 x_1 x_2 \tag{D.3}$$

$$\begin{aligned} f_3(\mathbf{X}, \mathbf{U}) &= \cos^2(x_3) \frac{1}{H} \left(-L u_2 - u_1 x_1 + x_1 u_2 W\right) + \sin^2(x_3) x_2 u_2 \\ &\quad + \cos(x_3) sin(x_3) \left(-\frac{u_1 x_2}{H} + u_2 \left(x_2 \frac{W}{H} + x_1\right)\right) \end{aligned} \tag{D.4}$$

Nous proposons l'observateur

$$\Sigma \begin{cases} \dot{\hat{\mathbf{X}}} & = f(\hat{\mathbf{X}}, \mathbf{U}) + l(\tilde{\mathbf{Y}}, \mathbf{U}, \hat{\mathbf{X}}) \\ \hat{\mathbf{Y}} & = \hat{\mathbf{X}} \end{cases} \tag{D.5}$$

avec l'erreur d'observation

$$\tilde{\mathbf{Y}} = \mathbf{Y} - \hat{\mathbf{Y}}$$

et

$$l(\tilde{\mathbf{Y}}, \mathbf{U}, \hat{\mathbf{X}}) = \begin{pmatrix} l_1(\tilde{\mathbf{Y}}, \mathbf{U}, \hat{\mathbf{X}}) \\ l_2(\tilde{\mathbf{Y}}, \mathbf{U}, \hat{\mathbf{X}}) \\ l_3(\tilde{\mathbf{Y}}, \mathbf{U}, \hat{\mathbf{X}}) \end{pmatrix}$$

On souhaite que $\hat{\mathbf{X}} \to \mathbf{X}$.

En proposant $l_1(\tilde{\mathbf{Y}}, \mathbf{U}, \hat{\mathbf{X}})$ tel que définit en (D.6), on garantit que $\hat{x}_1 \to x_1$.

$$l_1\left(\tilde{\mathbf{Y}}, \mathbf{U}, \hat{\mathbf{X}}\right) = u_2.\tilde{y}_1^2 + 2.u_2.\tilde{y}_1.\hat{x}_1 + K_1.sign(\tilde{y}_1) \tag{D.6}$$

On propose le gain $l_2(\tilde{\mathbf{Y}}, \mathbf{U}, \hat{\mathbf{X}})$ tel que

$$l_2(\tilde{\mathbf{Y}}, \mathbf{U}, \hat{\mathbf{X}}) = u_2\left((\tilde{y}_1 + \hat{x}_1)(\tilde{y}_2 + \hat{x}_2) - \hat{x}_1\hat{x}_2\right) + K_2.sign(\tilde{y}_2) \tag{D.7}$$

avec K_2 un scalaire positif. La dynamique de l'erreur s'écrit :

$$\begin{aligned} \dot{\tilde{x}}_2 &= \dot{x}_2 - \dot{\hat{x}}_2 \\ &= u_2 x_1 x_2 - u_2 \hat{x}_1 \hat{x}_2 - l_2(\tilde{\mathbf{Y}}, \mathbf{U}, \hat{\mathbf{X}}) \\ &= u_2(x_1 x_2 - \hat{x}_1 \hat{x}_2) - u_2\left((\tilde{y}_1 + \hat{x}_1)(\tilde{y}_2 + \hat{x}_2) - \hat{x}_1\hat{x}_2\right) - K_2.sign(\tilde{y}_2) \\ &= -K_2.sign(\tilde{y}_2) \end{aligned} \tag{D.8}$$

Soit la fonction de Lyapunov $V_2 = \frac{1}{2}\tilde{x}_1^2 + \frac{1}{2}\tilde{x}_2^2$. Sa dérivée par rapport au temps s'écrit

$$\dot{V}_2 = -K_1|\tilde{y}_1| - K_2|\tilde{y}_2| \tag{D.9}$$

ce qui garantit la convergence de l'erreur d'observation des deux premières variables.

On propose le gain $l_3(\tilde{\mathbf{Y}}, \mathbf{U}, \hat{\mathbf{X}})$ tel que

$$l_3(\tilde{\mathbf{Y}}, \mathbf{U}, \hat{\mathbf{X}}) = \tilde{y}_1\left(u_2(\frac{W}{H} + 1) - \frac{u_1}{H}\right) + \tilde{y}_2\left(u_2(1 + \frac{W}{H}) - \frac{u_1}{H}\right) + K_3.sign(\tilde{y}_3) \tag{D.10}$$

Soit la fonction de Lyapunov $V_3 = \frac{1}{2}\tilde{x}_3^2$. Sa dérivée par rapport au temps s'écrit

$$\begin{aligned}
\dot{V}_3 &= \tilde{x}_3\dot{\tilde{x}}_3 \\
&= \tilde{x}_3\left(f_3(\mathbf{X},\mathbf{U}) - f_3(\hat{\mathbf{X}},\mathbf{U}) - l_3(\tilde{\mathbf{Y}},\mathbf{U},\hat{\mathbf{X}})\right) \\
&= \tilde{x}_3\left(\varepsilon_3 - l_3(\tilde{\mathbf{Y}},\mathbf{U},\hat{\mathbf{X}})\right)
\end{aligned} \tag{D.11}$$

$$\begin{aligned}
\varepsilon_3 &= f_3(\mathbf{X},\mathbf{U}) - f_3(\hat{\mathbf{X}},\mathbf{U}) \\
&= \cos^2(x_3)\frac{1}{H}\left(-Lu_2 - u_1x_1 + x_1u_2W\right) + \sin^2(x_3)x_2u_2 + \cos(x_3)sin(x_3)\left(-\frac{u_1x_2}{H} + u_2\left(x_2\frac{W}{H} + x_1\right)\right) \\
&\quad - \cos^2(\hat{x}_3)\frac{1}{H}\left(-Lu_2 - u_1\hat{x}_1 + \hat{x}_1u_2W\right) - \sin^2(\hat{x}_3)\hat{x}_2u_2 - \cos(\hat{x}_3)sin(\hat{x}_3)\left(-\frac{u_1\hat{x}_2}{H} + u_2\left(\hat{x}_2\frac{W}{H} + \hat{x}_1\right)\right) \\
&< \frac{1}{H}\left(-Lu_2 - u_1x_1 + x_1u_2W\right) + x_2u_2 - \frac{u_1x_2}{H} + u_2\left(x_2\frac{W}{H} + x_1\right) \\
&\quad - \frac{1}{H}\left(-Lu_2 - u_1\hat{x}_1 + \hat{x}_1u_2W\right) - \hat{x}_2u_2 + \frac{u_1\hat{x}_2}{H} - u_2\left(\hat{x}_2\frac{W}{H} + \hat{x}_1\right) \\
&< \frac{1}{H}\left(-u_1(x_1 - \hat{x}_1) + (x_1 - \hat{x}_1)u_2W\right) + (x_2 - \hat{x}_2)u_2 - \frac{u_1(x_2 - \hat{x}_2)}{H} + u_2\left((x_2 - \hat{x}_2)\frac{W}{H} + (x_1 - \hat{x}_1)\right) \\
&< \frac{\tilde{x}_1}{H}\left(-u_1 + u_2W\right) + \tilde{x}_2u_2 - \frac{u_1\tilde{x}_2}{H} + u_2\left(\tilde{x}_2\frac{W}{H} + \tilde{x}_1\right) \\
&< \tilde{x}_1\left(-\frac{u_1}{H} + u_2 + u_2\frac{W}{H}\right) + \tilde{x}_2\left(u_2 - \frac{u_1}{H} + u_2\frac{W}{H}\right) \\
&< \tilde{x}_1\left(u_2(\frac{W}{H} + 1) - \frac{u_1}{H}\right) + \tilde{x}_2\left(u_2(1 + \frac{W}{H}) - \frac{u_1}{H}\right)
\end{aligned}$$

En injectant cette information dans (D.11), on obtient

$$\dot{V}_3 < \tilde{x}_3\left(\tilde{x}_1\left(u_2(\frac{W}{H} + 1) - \frac{u_1}{H}\right) + \tilde{x}_2\left(u_2(1 + \frac{W}{H}) - \frac{u_1}{H}\right) - l_3(\tilde{\mathbf{Y}},\mathbf{U},\hat{\mathbf{X}})\right) \tag{D.12}$$

En remplaçant $l_3(\tilde{\mathbf{Y}},\mathbf{U},\hat{\mathbf{X}})$ par la valeur proposée, on arrive à

$$\dot{V}_3 < -K_3.|\tilde{y}_3| \tag{D.13}$$

qui prouve la convergence de l'erreur d'observation.

Extraction du squelette dans la représentation virtuelle

Cette partie nous intéresse pour le chapitre 3. L'extraction du squelette en utilisant l'algorithme du DMA est calculé directement dans l'image, et ne prend pas en considération le contexte lié à sa formation. Cela revient à déterminer dans l'image une nouvelle métrique $\widetilde{d}$ conforme à la distance euclidienne dans le plan du sol [Marie, 2014].

Pour cela il faut chercher la relation entre la distance euclidienne de deux points 3D et la distance euclidienne dans le plan d'image normalisé extraite à partir du capteur virtuel.
Soit $P_1 = (X_1, Y_1, Z_1)^T$, $P_2 = (X_2, Y_2, Z_2)^T$ deux points du sol exprimés dans la base de la caméra, et $p_1 = (x_1, y_1)^T$, $p_2 = (x_2, y_2)^T$ leurs projections dans le plan d'image normalisé.
La métrique adaptée $\widetilde{d}$ doit vérifier :

$$\widetilde{d}(p_1, p_2) \propto d(P_1, P_2) \tag{E.1}$$

Rappelons que $x_i = X_i/Z_i$, $y_i = Y_i/Z_i$, et nous avons l'équation du plan du sol dans le repère du capteur qui s'écrit $Z_i = H$, pour $i = 1, 2$, nous pouvons écrire que :

$$\widetilde{d}(p_1, p_2) = H\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{E.2}$$

Cette métrique adaptée est très importante pour l'extraction du squelette (section.3.3). L'expression de (E.2) demande un temps de calcul très coûteux par rapport à une simple distance euclidienne. Les tableaux de "Lookup Tables" sont utilisés pour assurer le calcul dans une application temps réel.

En se basant sur la nouvelle métrique, les projections adaptées $\widetilde{\mathcal{P}_{\mathbf{x}}}$ ainsi que les distances $\widetilde{\mathcal{D}_{\mathbf{x}}}$ sont définies pour chaque $\mathbf{x} \in \mathcal{X}$:

$$\widetilde{\mathcal{P}_{\mathbf{x}}} = argmin_{\mathbf{y} \in \mathcal{X}^c} \widetilde{d}(\mathbf{x}, \mathbf{y}) \qquad \widetilde{\mathcal{D}_{\mathbf{x}}} = \widetilde{d}(\mathbf{x}, \widetilde{\mathcal{P}_{\mathbf{x}}}) \tag{E.3}$$

Par extension, la valeur delta maximale adaptée est donnée par :

$$\widetilde{\delta_M}(\mathbf{x}) = \sup\{\widetilde{r}_{\mathbf{x},\mathbf{y}} \mid \mathbf{y} \in N_{\mathbf{x}},\ \widetilde{d}(\mathbf{m}, \widetilde{\mathcal{P}_{\mathbf{x}}}) \geq \widetilde{d}(\mathbf{m}, \widetilde{\mathcal{P}_{\mathbf{y}}})\} \tag{E.4}$$

avec $\widetilde{r}_{\mathbf{x},\mathbf{y}} = sup\{\widetilde{\mathcal{D}_{\mathbf{z}}} \mid \mathbf{z} \in [\widetilde{\mathcal{P}_{\mathbf{x}}}, \widetilde{\mathcal{P}_{\mathbf{y}}}\}$ for a direct neighbor $\mathbf{y} \in N_{\mathbf{x}}$.

L'axe delta médian adapté est le défini par [Marie et al., 2016] : Soit $\mathcal{X} \cup \mathcal{X}^c$ un masque binaire,

et $\delta \in \mathbb{R}_+^*$. L'axe delta médian adapté $\widetilde{MA}_\delta(\mathcal{X})$ de $\mathcal{X}$ est l'ensemble de points $\mathbf{x} \in \mathcal{X}$ avec une valeur de delta maximale adaptée d'au moins δ :

$$\widetilde{MA}_\delta(\mathcal{X}) = \{\mathbf{x} \in \mathcal{X} \mid \widetilde{\delta_M}(\mathbf{x}) \geq \delta\} \tag{E.5}$$

Environnement de simulation

Morse (The Modular OpenRobots Simulation Engine), le moteur de simulation modulaire OpenRobots est un outil open-source développé pour la recherche en robotique. C'est un simulateur indépendant, où des robots virtuels peuvent interagir avec un environnement 3D, en utilisant des capteurs et des actionneurs qui se comportent de la même manière que leurs homologues dans le monde réel [Lemaignan et al., 2012]. L'intégration de Morse avec le logiciel Blender permet une simulation semi-réaliste d'environnements complexes. Cette intégration permet de modéliser l'environnement et le robot [Echeverria et al., 2011] en utilisant un middleware.
Les middlewares sont une couche intermédiaire entre différents systèmes logiciels qui leur permet de communiquer et de partager des données. Différents middlewares existent en robotique, tels que YARP, ROS, ou Pocolibs. Dans notre cas nous avons utilisé le middleware ROS.

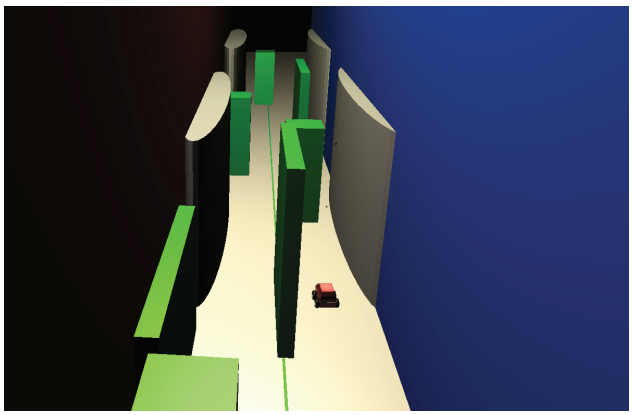


FIGURE F.1 – Le robot utilisé et un exemple d'environnement construit avec Morse/Blender.

Les différents composants robotiques qui peuvent être définis dans MORSE sont :

- Les capteurs : récupèrent les données du monde simulé, émulant la fonctionnalité des capteurs réels.
- Les actionneurs : Produisent des actions sur les robots ou composants associés. En particulier, les actionneurs déplacent les robots en fonction d'un paramètre donné : les coordonnées d'une cible ou les vitesses linéaires/angulaires.
- Les robots : Les plates-formes où les capteurs et les actionneurs sont montés. Ils définissent également les propriétés physiques (taille, poids, friction, mobilité, collision) du robot virtuel. Les bras robotiques sont également inclus dans cette catégorie. Ceux-ci sont composés de plusieurs segments qui peuvent être articulés et qui ont des actionneurs spéciaux.

L'environnement 3D sera définit par le logiciel Blender.

Navigation autonome et commande référencée capteurs de robots d'assistance à la personne

L'autonomie d'un agent mobile se définit par son capacité à naviguer dans un environnement sans intervention humaine. Cette tâche s'avère très demandée pour les robots d'assistance à la personne. C'est pour cela que notre contribution s'est portée en particulier sur l'augmentation de l'autonomie d'un fauteuil roulant pour les personnes à mobilité réduite. L'objectif de ce travail est de concevoir des lois de commande qui permettent la navigation autonome en temps réel dans un environnement inconnu. Une approche de navigation autonome et sûre a été conçue pour se déplacer dans un environnement peu encombré dont la structure peut être assimilée à un couloir (lignes au sol, murs, délimitation herbes, routes...). La problématique a été résolue en utilisant le formalisme de l'asservissement visuel. Pour assurer une navigation sûre et lisse, même lorsque les caractéristiques visuelles ne peuvent pas être extraites, nous avons conçu un observateur d'état pour les estimer dans le but de maintenir la commande fonctionnelle du robot. La première contribution de cette thèse a été étendue en traitant tout type d'environnement encombré statique ou dynamique. Cela a été réalisé en utilisant le diagramme de Voronoï Généralisé (DVG). Dans ce travail, une approche d'asservissement visuel basée sur le squelette extrait en temps réel était proposée pour une navigation autonome et sûre des robots mobiles. La commande est basée sur une approximation du DVG local en utilisant le Delta Medial axis, un algorithme de squelettisation rapide et robuste. Cette approche peut faire face aux bruits de mesure au niveau de la perception et au niveau de la commande à cause du glissement des roues. Nous avons alors conçu une approche d'asservissement visuel sur une prédiction d'une linéarisation du DVG.

Mots-clés : robots mobiles, navigation autonome, asservissement visuel, diagramme de Voronoï.

Autonomous navigation and sensor based control of personal assistance robots

The autonomy of a mobile agent is defined by its ability to navigate in an environment without human intervention. This task is very required for personal assistance robots. That's why our contribution has been particularly **focused** on increasing the autonomy of a wheelchair for reduced mobility peaple. The objective of this work is to design control laws that for a autonomous navigation in real time in an unknown environment.
First we designed an autonomous and safe navigation approach in environment whose structure can be assimilated to a corridor (lines on the ground, walls, delimitation of grasses, roads ...). We have solved this problem by using the formalism of visual servoing. To ensure safe and smooth navigation, even when the visual characteristics can not be extracted, we have designed a finite-time state observer to estimate those parameters in order to maintain the robot's control efficient. We have extended the first contribution of this work with dealing with any type of static or dynamic environment. This was done using the Generalized Voronoi diagram (GVD). In this work, a real time skeleton based visual servoing approach is proposed for a safe autonomous navigation of mobile robots. The control is based on an approximation of the local GVD using the Delta Medial Axis, a fast and robust skeletonization algorithm. This approach can cope with measurement noises at the perception and control with the wheel slip. This is why we have designed a visual servoing approach on a prediction of a GVD linearization.

Keywords : mobile robot, autonomous navigation, visual servoing, Voronoï diagram.