Navigation référencée multi-capteurs d'un robot mobile en environnement encombré

Adrien Durand-Petiteville

Université de Toulouse

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par *l'Université Toulouse III - Paul Sabatier*
Discipline ou spécialité : *Automatique et Robotique*

Présentée et soutenue par
Durand Petiteville Adrien

Le *20 janvier 2012*

Titre :
Navigation référencée multi-capteurs d'un robot mobile en environnement encombré

JURY
Président : Michel Devy - Directeur de recherche LAAS-CNRS
Rapporteur : François Chaumette - Directeur de recherche INRIA
Rapporteur : Philippe Martinet - Professeur des universités Centrale Nantes
Examinateur : David Folio - Maître de conférence ENSI Bourges
Directeur de thèse : Viviane Cadenat - Maître de conférence UPS
Directeur de thèse : Michel Courdesses - Professeur des universités UPS

Ecole doctorale : *EDSYS*
Unité de recherche : *LAAS-CNRS*
Directeurs de Thèse : *Viviane Cadenat et Michel Courdesses*

Remerciements

Tout d'abord j'exprime toute ma gratitude à Viviane Cadenat, Maître de conférences à l'université Paul Sabatier, et Michel Courdesses, Professeur des universités à l'université Paul Sabatier, pour m'avoir accueilli, encadré et formé tout au long de cette thèse. Cette première expérience de la recherche que vous m'avez permis de vivre, a comblé mes attentes et me pousse à continuer dans cette voie.

Je remercie Michel Devy, Directeur de recherche CNRS du LAAS, pour m'avoir accepté au sein du groupe Robotique Action et Perception et pour m'avoir fait l'honneur de présider le jury.

Merci à François Chaumette, Directeur de recherche INRIA à l'IRISA de Rennes, et à Philippe Martinet, Professeur des universités à l'Ecole Centrale de Nantes, d'avoir accepté de rapporter sur ma thèse. Je les remercie également pour les nombreuses remarques et suggestions qui m'ont permis d'affiner mon travail et qui me permettront de le prolonger.

Mes remerciements vont naturellement à David Folio, Maître de conférence à l'ENSI de Bourges, pour sa participation au jury et pour l'intérêt qu'il a bien voulu porter à mes travaux.

Je tiens également à remercier Seth Hutchinson, Professor of Electrical and Computer Engineering de l'Université d'Illinois à Urbana-Champaign, de m'avoir accueilli et guidé au sein de son laboratoire durant quelques mois. Je remercie également les membres de son équipe pour leur disponibilité et la richesse de nos échanges.

Je remercie les membres du pôle robotique, qu'ils soient permanents ou doctorants, pour leur aide et pour les nombreuses discussions constructives qui ont contribué à faire avancer mes travaux. De même, un grand merci à toute l'équipe pédagogique de la 61^{ime} section pour leurs conseils et leur soutien tout au long de ces quatre années d'enseignement qui furent pour moi hautement formatrices, apportant un complément précieux à ma formation par la recherche.

Merci aux personnels de l'AIP et plus particulièrement à Philippe Baillion, Ingénieur de recherche UPS, pour son aide dans la mise en œuvre des expérimentations.

Cette thèse est le résultat d'échanges, de discussions et de collaborations. Je remercie donc toutes les personnes qui ont contribué de près ou loin à ce travail.

Table des matières

TABLE DES FIGURES

Chapitre 1

Introduction

La robotique, apparue dans les années 1960, consiste à concevoir et à étudier des machines intelligentes, c'est-à-dire des machines capables de réaliser des tâches avec un certain degré d'autonomie. Les disciplines de recherche associées à cette thématique sont articulées autour de la perception (traitement du signal, reconnaissance de formes), de la décision (intelligence artificielle) et de l'action (automatique). Dans un premier temps, les chercheurs se sont intéressés à la robotique de manipulation et à ses applications notamment dans l'industrie manufacturière. A partir des années 1980, en raison de l'augmentation de la puissance de calcul embarquable et de l'amélioration des capteurs, la robotique mobile devient un thème de recherche important avec comme objectif la création de robots capables d'explorer des zones dangereuses ou inaccessibles pour l'homme.

Dans ce mémoire, nous nous intéressons à la navigation [1] d'un robot mobile dans un environnement intérieur structuré. Il s'agit alors pour le véhicule considéré d'atteindre un but préalablement connu tout en évitant les éventuels obstacles [Choset *et al.* 2005]. Pour mener à bien cette mission, la commande du robot s'appuie sur une architecture de navigation au sein de laquelle sont organisées toutes les fonctions nécessaires. Ces dernières sont listées ci-dessous :

- **Perception**

  Afin d'acquérir des données nécessaires à la navigation, un robot peut être équipé de deux types de capteurs : des capteurs proprioceptifs et des capteurs extéroceptifs. Les premiers (odomètres, gyroscopes, ...) fournissent des données relatives à l'état interne du robot tandis que les seconds (caméra, télémètres laser, télémètres ultrasons, bumpers...) fournissent des informations sur l'environnement. Les données acquises au cours de la navigation peuvent être utilisées dans quatre processus : la modélisation de l'environnement, la localisation, la décision et la commande du robot.

- **Modélisation**

  Tout processus de navigation requiert généralement la mise en place d'un modèle de l'environnement. Initialement, ce modèle est construit à partir des informations connues *a priori*. Il n'est pas toujours complet et définitif, et peut évoluer au cours du temps si cela est

1. Le problème de la navigation ne doit pas être confondu avec celui de l'exploration pour lequel aucune situation finale n'est requise.

nécessaire. Les modifications sont réalisées à partir des informations acquises au fur et à mesure de la navigation. Les données sont organisées dans des cartes dites métriques et/ou topologiques [Siegwart & Nourbakhsh 2004].

La carte métrique est une représentation continue ou discrète des espaces libres et occupés. Elle possède un repère dans lequel la situation du robot et des obstacles est connue avec une certaine précision. Lors d'une mise à jour, il est donc nécessaire de localiser les données dans ce repère avant de les ajouter à la carte.

La carte topologique est une représentation discrète de l'environnement sous forme d'un graphe. Chaque nœud représente un ensemble continu de la scène défini par une propriété caractéristique. Les ensembles sont par nature connexes et peuvent se limiter à un point unique de la scène. La propriété caractéristique, choisie par l'utilisateur, peut par exemple être la visibilité d'un amer ou l'appartenance à une même pièce. De plus, si un couple de nœuds vérifie une condition d'adjacence, ils sont alors reliés. La condition d'adjacence est choisie par l'utilisateur et peut correspondre par exemple à l'existence d'un chemin dans l'espace des configurations [2] libres permettant de relier les deux ensembles représentés par les nœuds, ou à l'existence d'une zone pour laquelle deux amers sont visibles. Une carte topologique est moins sensible qu'une carte métrique aux évolutions de la scène. En effet, seuls les événements modifiant les espaces représentés par les nœuds ou l'adjacence entre deux nœuds rendent nécessaire une mise à jour de la carte.
Il est possible d'enrichir les cartes topologiques et métriques avec des informations sensorielles, des actions ou des commandes. En d'autres termes, cela signifie qu'on associe à un nœud une ou plusieurs données sensorielles (des images par exemple) ou une commande à appliquer. Ces informations complémentaires peuvent être requises pour localiser le robot ou pour le commander. Nous verrons par la suite différentes solutions pour introduire les données supplémentaires.

- **Planification**

La phase de planification consiste à calculer, à partir du modèle de l'environnement, un itinéraire permettant au robot d'atteindre sa position d'arrivée. L'itinéraire peut consister en un chemin [3], une trajectoire [4], une suite de situations à atteindre... Il existe une grande variété de techniques de planification selon la modélisation de l'environnement considérée. Nous proposons par la suite une vue d'ensemble de ces méthodes.

Les chemins et les trajectoires peuvent être calculés lorsqu'une carte contenant une description métrique est disponible. Pour cela, l'espace géométrique est transposé dans l'espace des configurations du robot. La configuration correspondant à une paramétrisation de l'état statique du système robotique, un robot dont la géométrie peut être complexe dans l'espace de travail, est représenté par un point dans l'espace des configurations [Siegwart & Nourbakhsh 2004]. La planification consiste alors à trouver un chemin ou une trajectoire dans l'espace des configurations libres permettant d'atteindre la configuration finale [Choset *et al.* 2005]. Avec une modélisation continue de l'environnement, un chemin ou une trajectoire peut être obtenu en utilisant des graphes de visibilité ou des diagrammes de Voronoi [Okabe *et al.* 2000]. Lorsque l'environnement est représenté à l'aide d'un modèle discret, la

2. Espace des paramètres caractérisant l'état du système.
3. Un chemin est défini par une fonction continue dans l'espace des configurations.
4. Une trajectoire est définie par une fonction continue dans l'espace configuration-temps.

planification est effectuée en utilisant des méthodes issues de la théorie des graphes telles que l'algorithme A^* [Hart *et al.* 1968] ou l'algorithme de Dijkstra [Dijkstra 1971]. Que ce soit pour des cartes continues ou discrètes, la phase de planification peut s'avérer coûteuse en temps de calcul. Une solution consiste alors à utiliser des techniques de planification probabiliste telles que *probabilistic roadmap* [Kavraki *et al.* 1996] [Geraerts & Overmars 2002] ou *rapidly exploring random tree* [LaValle 1998].

Lorsque le modèle de l'environnement n'est pas complet au début de la navigation, le robot peut rencontrer des obstacles inconnus. Une première solution à ce problème consiste à ajouter ces obstacles au modèle et à planifier une nouvelle trajectoire. Dans [Lamiraux *et al.* 2004], les auteurs proposent de considérer la trajectoire comme une bande élastique qui peut être déformée si nécessaire. Cependant une replanification globale peut être requise pour une modification majeure de l'environnement.

Dans le cas où le modèle de l'environnement est une carte topologique ne contenant pas d'informations métriques, l'itinéraire planifié est généralement constitué par une suite de situations à atteindre. Celles-ci peuvent être exprimées dans un repère associé à la scène ou dans le repère capteur. L'itinéraire est calculé en utilisant également les méthodes issues de la théorie des graphes. En fonction du degré de précision avec lequel est décrit l'environnement, il est possible que l'itinéraire planifié ne tienne pas compte de tous les obstacles présents dans la scène. Cette particularité devra donc être prise en compte lors de la réalisation des mouvements.

- **Localisation**

Pour une navigation, on identifie deux types de localisations : la localisation métrique et la localisation topologique [Siegwart & Nourbakhsh 2004]. La localisation métrique consiste à calculer la situation du robot dans un repère qui peut être local ou global. La simple utilisation de données issues de capteurs proprioceptifs [Cobzas & Zhang 2001] [Wolf *et al.* 2002] ne suffit généralement pas pour résoudre ce problème car elle peut engendrer des erreurs importantes. En effet, la situation étant calculée en intégrant successivement les données acquises, une dérive peut apparaître. De plus cette intégration étant basée sur un modèle du robot, toute erreur dans la modélisation est reportée dans le calcul de la situation. Enfin, des phénomènes tels que le glissement ne peuvent être pris en compte. Afin d'améliorer la qualité de la localisation, il semble donc nécessaire de coupler, lorsque cela est possible, les données issues des capteurs proprioceptifs et extéroceptifs. L'odométrie visuelle [Nister *et al.* 2004] [Comport *et al.* 2010] [Cheng *et al.* 2006] est un exemple d'un tel couplage.

La localisation topologique [Sim & Dudek 1999] [Paletta *et al.* 2001] [Krösea *et al.* 2001] consiste à mettre en relation les données issues des capteurs avec celles associées aux nœuds du graphe modélisant l'environnement [Remazeilles 2004]. Il ne s'agit donc pas de déterminer la situation dans un repère mais de trouver la situation dans un graphe [Siegwart & Nourbakhsh 2004]. La localisation topologique est donc peu sensible aux erreurs de mesures contrairement à son alter-ego métrique. Sa précision dépend de celle avec laquelle est décrite l'environnement.

- **Action**

Afin de réaliser les actions nécessaires à la navigation, différents correcteurs doivent être

synthétisés. Ils peuvent être de type retour d'état ou retour de sortie [Levine 1996]. Dans le premier cas, il s'agir d'annuler l'erreur entre l'état courant et un état de référence correspondant à la situation finale désirée. L'état peut être défini par la situation du robot dans le repère de la scène ou par la situation du robot par rapport à une cible [Hutchinson *et al.* 1996] [Bellot 2002]. Afin de calculer la loi de commande, il est donc nécessaire de connaître l'état courant. Pour cela, nous utiliserons généralement une localisation métrique.

Dans le cas d'un retour de sortie, il s'agit d'annuler l'erreur entre la mesure courante et une mesure de référence. Cette mesure dépend de la situation relative du robot par rapport à un élément de l'environnement appelé cible ou amer. Pour la vision, les mesures correspondent à des primitives telles que le point, la droite [Chaumette 1990] ou encore les moments [Chaumette 2004]. Pour les capteurs de proximité, elles sont données par des distances pour les télémètres à ultra son [Cadenat 1999] et lasers [Thrun *et al.* 2000] [Victorino & Rives 2004]. La mesure étant directement utilisée dans la loi de commande, il n'est alors pas nécessaire de localiser le robot géométriquement. Cependant, il faut que la cible soit perceptible par le capteur tout au long de la navigation pour que la commande puisse être calculée.

- **Décision**

Afin de mener une navigation à son terme, il peut être nécessaire de prendre des décisions à différents stades du processus. Ces décisions peuvent concerner le choix du correcteur à utiliser, l'ajout d'un nouvel élément au modèle, une replanification... Les décisions peuvent donc affecter aussi bien le haut niveau, avec une replanification [Lamiraux *et al.* 2004], que le bas niveau de la navigation, en modifiant des paramètres de la loi de commande [Folio 2007]. Ces décisions sont prises par des algorithmes de supervision basant leurs choix sur les données issues des capteurs extéroceptifs.

Remarque : *Que ce soit pour la modélisation, la localisation, la commande ou la détection d'événements, il n'est pas rare que certains paramètres ne soient pas mesurables. Afin de remédier à cette situation les paramètres inconnus sont calculés à l'aide de processus d'estimation (filtre de Kalman, moindres carrés, ...) utilisant généralement les données issues des deux types de capteurs. De la même façon que pour la localisation, l'estimation est plus précise lorsque les deux types de capteurs sont exploités simultanément.*

Présentation des architectures de navigation

Les fonctions étant désormais connues, nous introduisons les différentes architectures de navigation possibles. Pour cela nous proposons d'axer notre présentation autour des deux types de correcteurs.

Architecture centrée "retour d'état"

Tout d'abord nous considérons un robot commandé par retour d'état dans un environnement libre. Les configurations initiale et finale sont définies par rapport à un repère de la scène. Pour calculer la loi de commande, l'état doit être connu à chaque instant. La capacité du robot à se localiser géométriquement est donc une condition nécessaire pour le succès de la

navigation. De plus, seule la précision de la localisation limite la distance que le robot peut parcourir puisqu'une trop grande erreur sur la valeur de l'état entrainera une loi de commande incohérente.

Nous considérons maintenant un environnement encombré d'obstacles. Dans ce cas deux solutions s'offrent à nous. La première solution consiste à commander le robot à l'aide de deux correcteurs : un premier, annulant l'erreur entre la situation courante et celle désirée, permettant d'atteindre l'objectif, un second réalisant l'évitement d'obstacle sur la base de données extéroceptives. Il est alors nécessaire de créer un module de supervision sélectionnant le correcteur à utiliser. Cette solution, qui garantit la non collision du robot avec les obstacles, ne permet pas cependant d'assurer le succès de la navigation. En effet, l'évitement d'obstacle n'est réalisé que localement, ne prenant pas en compte l'objectif final. La seconde solution consiste à suivre un chemin libre de toute collision préalablement planifié. Pour cela, il nécessaire de modéliser l'environnement à l'aide d'une carte. Si le modèle représente entièrement l'environnement, alors la navigation consiste simplement à exécuter par l'intermédiaire du correcteur de type retour d'état l'itinéraire défini. Un superviseur n'est alors plus requis. Si l'environnement n'est pas entièrement connu, il peut être nécessaire de mettre à jour son modèle lorsqu'un obstacle inconnu est rencontré. L'opération réalisée, une replanification est effectuée. Dans ce cas, un superviseur décidant de la mise à jour et de la replanification est nécessaire. Quelle que soit la solution retenue, la localisation métrique est nécessaire et elle limite la portée de la navigation.

Lorsque les déplacements sont réalisés à l'aide d'un correcteur de type retour d'état, la localisation métrique est un élément déterminant. De sa qualité dépendront les chances de succès de la navigation. De plus, dans un environnement encombré, la nécessité d'un modèle apparaît rapidement que ce soit pour converger vers la situation désirée ou pour éviter les obstacles. Les processus de localisation métrique et de modélisation sont très sensibles aux erreurs de mesure. Il est donc nécessaire de porter une attention toute particulière aux performances de ces méthodes lorsque l'on cherche à réaliser une navigation par retour d'état.

Architecture centrée "retour de sortie"

Nous considérons maintenant un environnement libre dans lequel un robot est commandé par un correcteur de type retour de sortie. La situation initiale est inconnue tandis que la situation finale par rapport à une cible est décrite par des mesures. Le robot peut converger vers la situation désirée si la cible peut être perçue à chaque instant. C'est donc la portée du capteur qui vient limiter la distance que peut parcourir le robot. Ici, aucune localisation métrique n'est requise.

Plaçons-nous maintenant dans un environnement encombré. Une des premières solutions retenues jusqu'à présent consiste à utiliser un correcteur de type retour de sortie permettant d'atteindre la situation désirée et d'éviter les obstacles. Une seconde solution propose de commander le robot à l'aide de deux correcteurs de type retour de sortie : le premier dédié à la navigation, le second à l'évitement d'obstacle. Un superviseur sélectionnant le correcteur adapté à la situation est alors nécessaire. Pour les deux solutions, des problèmes de minima locaux peuvent apparaître. De plus, la portée de la navigation est toujours limitée par la portée des capteurs. Cette dernière ne pouvant être augmentée, il devient nécessaire d'apporter des informations globales pour effectuer de longs déplacements.

En ajoutant une carte métrique, il est possible de planifier un chemin en tenant compte

des amers disponibles pour chaque situation. Plusieurs cibles, successivement utilisées comme référence pour le calcul de la commande, constituent la trajectoire planifiée. De plus lors de la planification pour un environnement fixe, les limites articulaires, la visibilité et les obstacles peuvent être pris en compte. Néanmoins, une telle planification nécessite d'être en possession de modèles fidèles de l'environnement, du robot et des capteurs. Une carte topologique peut aussi être utilisée pour apporter les informations globales. Dans ce cas, les données complémentaires associées aux nœuds du graphe correspondent généralement aux mesures de référence ou aux cibles. De même que précédemment, l'itinéraire planifié est constitué d'une série de mesures ou d'amers à atteindre. Cette approche repose sur une représentation partielle de l'environnement. Le modèle est ainsi moins sensible aux variations de l'environnement, mais ne permet pas de prendre en compte diverses contraintes, telles que la présence d'obstacles ou les limites articulaires, lors de la planification.

Présentation de notre approche

Nous proposons de réaliser une navigation en privilégiant la caméra comme capteur principal. En effet, les dispositifs de vision fournissent des informations très riches permettant de réaliser différents types de tâches [Desouza & Kak 2002] [Bonin-Font *et al.* 2008]. Afin de commander le robot, nous choisissons d'utiliser un correcteur par retour de sortie basé sur les informations visuelles. Ce choix est motivé par l'absence de localisation métrique dans le processus de commande. Nous évitons ainsi les erreurs dues à la présence de mesures bruitées. Enfin, afin d'augmenter la portée de la navigation, nous nous focalisons sur une représentation topologique de l'environnement. Celle-ci apporte suffisamment d'informations pour réaliser une navigation sans engendrer une conception trop complexe. De plus, comme nous avons pu le voir précédemment, cette représentation est moins sensible aux modifications de l'environnement que son alter-ego métrique. Enfin, ce choix requièrt une localisation topologique, qui est un processus peu sensible aux bruits dans les mesures.

Un certain nombre de travaux s'appuient sur une approche de ce type. Dans [Vassalo *et al.* 2000], un graphe dont les nœuds correspondent aux couloirs de l'environnement, est préalablement fourni par l'utilisateur. Le robot parcourt les couloirs en utilisant le point de fuite dans l'image comme information visuelle. D'autres approches proposent une modélisation de l'environnement effectuée lors d'une pré-navigation. Au cours de celle-ci, des images obtenues pour des configurations du robot relativement proches, sont mises en mémoire. Une carte topologique, aussi appelée mémoire visuelle, est ensuite réalisée en organisant les images prises [Royer *et al.* 2007]. L'itinéraire calculé est alors appelé route visuelle [Matsumoto *et al.* 1996]. Cette approche est utilisée avec une caméra omnidirectionnelle [Gaspar *et al.* 2000] [Yagi *et al.* 2005] [Goedemé *et al.* 2007] [Booij *et al.* 2007] [Courbon 2009] ou une caméra sténopée [Jones *et al.* 1997] [Blanc *et al.* 2005] [Chen & Birchfield 2006] [Krajnìk & Přeučil 2008] [Courbon 2009]. Cependant, aucune de ces approches ne tient compte des deux problèmes majeurs de la navigation visuelle : les occultations, c'est-à-dire la perte de vue de la cible, et la présence d'obstacles.

Une série de travaux [Cherubini & Chaumette 2009] [Cherubini & Chaumette 2010] [Cherubini *et al.* 2011] a abouti à une navigation référencée vision capable d'éviter des obstacles non préalablement cartographiés. De plus, des occultations partielles sont tolérées. La carte topologique est là aussi construite durant une phase de pré-navigation. L'asservissement visuel utilise une mesure de référence variant dans le temps tandis que l'évitement d'obstacles est réalisé grâce à une loi de commande basée sur les champs de potentiels. Ainsi le chemin réalisé durant la

phase de pré-navigation peut être rejoué à l'aide d'une carte topologique en garantissant la non collision.

Dans ce mémoire, nous proposons une architecture de navigation référencée vision [5] garantissant l'intégrité du robot et gérant les occultations. Dans un premier chapitre nous présentons une modélisation du système robotique ainsi qu'une loi de commande permettant de réaliser un asservissement visuel dans un environnement libre de tout obstacle. La synthèse de la commande s'appuie sur le formalisme des fonctions de tâche [Samson *et al.* 1991] qui sera brièvement rappelé. Nous nous intéressons à la gestion des occultations dans un second chapitre. Pour cela nous nous appuyons sur les travaux présentés dans [Folio 2007], que nous étendons afin de les rendre performants dans un cadre réel de navigation. Notre travail porte alors essentiellement sur l'estimation de la profondeur des indices visuels dans un cadre expérimental, c'est-à-dire bruité. Dans un dernier chapitre, nous nous intéressons à la navigation en milieu encombré. Tout d'abord nous reprenons la méthode d'évitement d'obstacle présentée dans [Cadenat 1999]. Celle-ci est réalisée par une loi de commande de type retour de sortie. Ensuite nous étudions deux approches permettant de garantir la continuité entre les correcteurs asservissement visuel et évitement. Finalement nous nous attachons à augmenter la portée de la navigation, c'est-à-dire à pouvoir atteindre un amer qui ne peut être vu depuis la position initiale du robot. Pour cela nous proposons de découper la tâche de navigation en plusieurs sous-tâches. Pour atteindre le but fixé, le robot doit donc s'asservir sur une séquence d'amers. Cette dernière est calculée à partir d'une carte topologique représentant l'environnement de navigation. Pour que cette approche soit réalisable, il est nécessaire de développer des correcteurs permettant de trouver les amers et d'assurer la transition entre chacun d'eux. Enfin, l'organisation des différentes tâches repose sur un algorithme de supervision. Tout au long du mémoire nous proposons des simulations et expérimentations permettant d'illustrer et de valider nos propos. Finalement nous concluons sur les travaux réalisés et présentons les perspectives qui peuvent être envisagées.

5. Nos contributions porteront essentiellement sur les aspects relatifs à la commande. Les aspects vision seront succinctement abordés et resteront à développer par la suite.

CHAPITRE 2

COMMANDE RÉFÉRENCÉE VISION D'UN ROBOT MOBILE

Sommaire

Dans ce chapitre, nous nous intéressons à la navigation par asservissement visuel d'un robot mobile dans un environnement libre. La commande référencée vision étant bien connue, ce chapitre est essentiellement composé de rappels. Ainsi, nous présentons dans un premier temps le type de système robotique, ainsi que les différents modèles qui y sont associés. Dans un second temps, nous introduisons les primitives utilisées pour caractériser l'environnement dans lequel doit évoluer le robot. Finalement, nous présentons une loi de commande permettant de réaliser un asservissement visuel 2D. Tous les composants nécessaires à la navigation dans un environnement libre étant connus, nous concluons par des simulations.

2.1 Le système robotique

2.1.1 Présentation du robot

FIGURE 2.1 – Le robot Pekee II (Wany Robotics)

FIGURE 2.2 – Le robot Rackham (iRobot)

Dans ce mémoire nous abordons la navigation de robots composés d'une base mobile non holonome équipée d'une caméra montée sur une platine commandable en lacet. Les robots Pekee II (Fig. 2.1) et Rackham (Fig. 2.2), ainsi que de nombreux autres systèmes correspondent à ce type de système robotique. La grande majorité des expériences présentées dans ce mémoire ont été réalisées avec le robot Pekee II. Pour cette raison, nous nous attardons plus particulièrement sur ce dernier. Ce robot a été développé par la société Wany Robotics. La base mobile, d'un diamètre et d'une hauteur de $38cm$, est équipée de deux roues motrices contrôlées indépendamment par des moteurs de 12 volts commandables en vitesse. La platine possède un débattement de $\pm 170°$ et tolère une vitesse allant jusqu'à $100°/s$. Deux ordinateurs *Mini ITX* avec un processeur *Intel Core2Duo T7100* sont embarqués. De plus, ils possèdent une mémoire vive de 2 GO et une mémoire morte de 180 GO. Finalement, un dispositif WLAN Wifi 802.11b/g leur permet de communiquer avec d'autres ordinateurs non embarqués. Le robot est en outre doté d'une caméra Axis 214 PTZ (nous reviendrons sur cette dernière dans la section 2.1.3), d'un odomètre, d'un contacteur (bumper) à 360°, de 16 télémètres infrarouges Sharp GP2D120 et d'un télémètre laser Hokuyo URG-04LX. Ce dernier possède une portée de $5.6m$, une résolution de 0.36° et une précision de $1cm$. Les diverses fonctions haut niveau permettant de contrôler le robot ont été développées en langage C++ au sein de l'AIP-PRIMECA de Toulouse.

2.1.2 Modélisation du robot

La synthèse d'une loi de commande nécessite de préalablement modéliser le système que nous désirons commander. Dans cette section nous présentons la géométrie du système, les différents repères associés au robot ainsi que les matrices de passage homogènes associées. Nous introduisons une définition des états du robot et de la caméra. Enfin, nous terminons cette section en présentant le vecteur de commande, le torseur cinématique de la caméra ainsi que la matrice jacobienne du robot. Nous verrons par la suite que les trois derniers éléments sont primordiaux dans la synthèse d'une loi de commande réalisant un asservissement visuel 2D.

2.1.2.1 Définition des repères

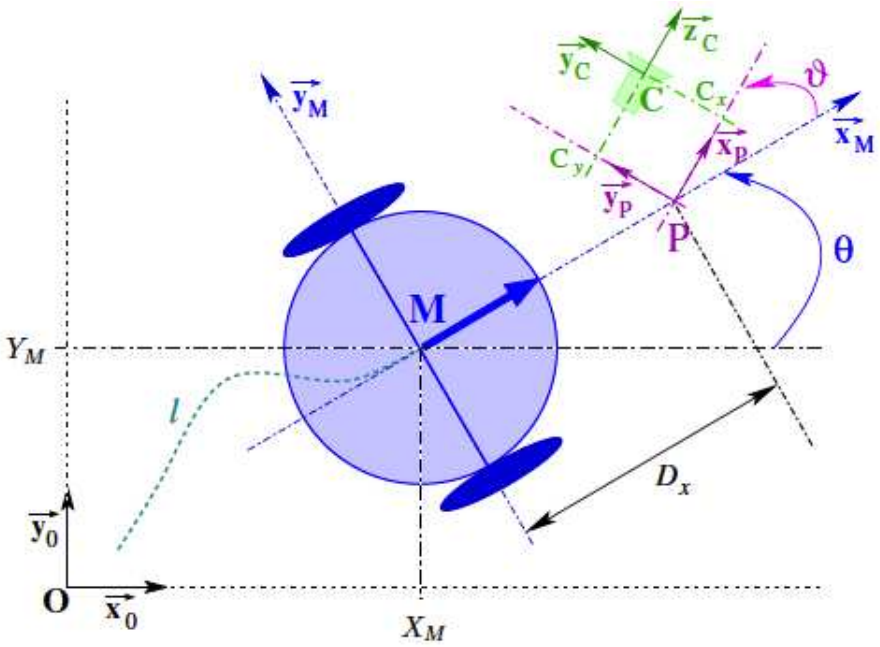


FIGURE 2.3 – Modèle du système robotique

La modélisation du robot s'appuie sur différents repères (Fig. 2.3) définis comme suit :
- $R_O(O, \vec{x}_O, \vec{y}_O, \vec{z}_O)$: le repère lié à la scène.
- $R_M(M, \vec{x}_M, \vec{y}_M, \vec{z}_M)$: le repère lié à la base mobile.
- $R_P(P, \vec{x}_P, \vec{y}_P, \vec{z}_P)$: le repère lié à la platine.
- $R_C(C, \vec{x}_C, \vec{y}_C, \vec{z}_C)$: le repère lié à la caméra.

D_x est la longueur de l'entre-axe $[MP]$, valant $10cm$ pour le robot Pekee II. Pour la base mobile, la position est représentée par les coordonnées (X_M, Y_M) du point M dans le repère R_O tandis que l'orientation est donnée par l'angle $\theta = (\vec{x}_O, \vec{x}_M)$. La position de la caméra dans le repère R_P est décrite par le vecteur $PC = (C_x\ C_y\ C_z)^T$. L'orientation de la platine est donnée par l'angle $\vartheta = (\vec{x}_M, \vec{x}_P)$.

2.1.2.2 Les matrices de passage homogènes

Dans la suite de notre étude, il sera nécessaire d'exprimer les coordonnées de points dans les différents repères associés au robot. Pour cela, nous introduisons ici les matrices de passage homogènes nécessaires.

Rappelons qu'une matrice de passage homogène du repère R_1 vers le repère R_2, est définie de la manière suivante :

$$\mathcal{H}_{R_2/R_1} = \begin{pmatrix} \mathcal{R}_{R_2/R_1} & \mathcal{T}_{R_2/R_1} \\ 0_{(1,3)} & 1 \end{pmatrix} \tag{2.1}$$

où $\mathcal{R}_{R_2/R_1}$ correspond à la matrice de rotation de R_1 vers R_2, et $\mathcal{T}_{R_2/R_1}$ correspond aux coordonnées de l'origine de R_1 exprimées dans R_2. De plus, $0_{(1,3)}$ est un vecteur nul de dimensions $(1, 3)$.

Les matrices de passage homogènes entre les différents repères sont définies comme suit :

$$\mathcal{H}_{R_O/R_M} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & X_M \\ \sin(\theta) & \cos(\theta) & 0 & Y_M \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathcal{H}_{R_P/R_C} = \begin{bmatrix} 0 & 0 & 1 & C_x \\ 0 & 1 & 0 & C_y \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
$$\mathcal{H}_{R_M/R_P} = \begin{bmatrix} \cos(\vartheta) & -\sin(\vartheta) & 0 & D_x \\ \sin(\vartheta) & \cos(\vartheta) & 0 & 0 \\ 0 & 0 & 1 & h_r \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

où h_r désigne la hauteur de la platine par rapport à la base mobile.

2.1.2.3 Définition du vecteur d'état

Nous définissons le vecteur d'état du robot à l'instant t comme étant :

$$\chi(t) = [X_M(t)\ Y_M(t)\ \theta(t)\ \vartheta(t)]^T \tag{2.3}$$

En effet ce vecteur permet de caractériser de manière unique la configuration du système robotique à chaque instant. De la même manière, nous introduisons le vecteur d'état de la caméra de la manière suivante :

$$\chi_C(t) = [X_C(t)\ Y_C(t)\ \theta_T(t)]^T \tag{2.4}$$

où $X_C(t)$ et $Y_C(t)$ sont les coordonnées à l'instant t du point C dans le repère R_0 et $\theta_T(t) = \theta(t) + \vartheta(t)$.

2.1.2.4 Définition du vecteur commande

Ce chapitre ayant pour but de présenter la loi de commande réalisant une navigation, il est nécessaire de définir le vecteur de commande. Afin de contrôler le système robotique par asservissement visuel, il est nécessaire de définir un vecteur de commande en vitesse. Dans

cette optique, nous introduisons à partir des capteurs proprioceptifs le vecteur des paramètres $q(t)$:

$$q(t) = [l(t)\ \theta(t)\ \vartheta(t)]^T \tag{2.5}$$

où $l(t)$ représente l'abscisse curviligne du point M par rapport au repère R_O. La dérivée de $q(t)$ par rapport au temps donne :

$$\dot{q}(t) = \frac{dq(t)}{dt} = [\upsilon(t)\ \omega(t)\ \varpi(t)]^T \tag{2.6}$$

où $\upsilon(t)$ et $\omega(t)$ désignent respectivement les vitesses linéaire et angulaire de la base mobile et $\varpi(t)$ correspond à la vitesse de rotation de la platine par rapport au robot. Celui-ci devant être commandé en vitesse, $\dot{q}(t)$ représente le vecteur de commande.

2.1.2.5 Le torseur cinématique de la caméra

Comme nous le verrons dans la section 2.3, la synthèse d'un correcteur réalisant un asservissement visuel nécessite de connaître le lien entre le mouvement de la caméra et le vecteur de commande. Pour cela nous utilisons le torseur cinématique de la caméra, noté $T^{R_C}_{C/R_O}$, par rapport au repère R_O et exprimé dans le repère R_C [Chaumette 1990] [Pissard-Gibollet & Rives 1995]. Le torseur est défini par

$$T^{R_C}_{C/R_O} = \left[V^{R_C\ T}_{C/R_O}\ \Omega^{R_C\ T}_{R_C/R_O}\right]^T \tag{2.7}$$

où

$$V^{R_C}_{C/R_O} = \left[V^{R_C}_{\vec{x}_C}\ V^{R_C}_{\vec{y}_C}\ V^{R_C}_{\vec{z}_C}\right]^T$$

et

$$\Omega^{R_C}_{R_C/R_O} = \left[\Omega^{R_C}_{\vec{x}_C}\ \Omega^{R_C}_{\vec{y}_C}\ \Omega^{R_C}_{\vec{z}_C}\right]^T$$

désignent respectivement les vitesses linéaires et angulaires de la caméra par rapport au repère de la scène R_O.

Les calculs [1] menant à l'expression du torseur cinématique pour notre système robotique sont présentés dans l'annexe A. Tous calculs faits, nous obtenons l'expression suivante du torseur cinématique :

$$T^{R_C}_{C/R_O} = \begin{pmatrix} 0 & 0 & 0 \\ -\sin(\vartheta) & C_x + D_x\cos(\vartheta) & C_x \\ \cos(\vartheta) & -C_x + D_x\sin(\vartheta) & -C_y \\ 0 & -1 & -1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \dot{q} \tag{2.8}$$

1. Les calculs menant à l'expression de $T^{R_C}_{C/R_O}$ sont issus de [Pissard-Gibollet & Rives 1995] et [Cadenat 1999].

Le torseur cinématique de la caméra $T_{C/R_O}^{R_C}$, que nous noterons par la suite T, décrit le mouvement de la caméra par rapport au repère de la scène. Les lignes de zéros correspondent aux mouvement non réalisables, c'est-à-dire la translation selon $\vec{x}_C$ et les rotations selon $\vec{y}_C$ et $\vec{z}_C$. Ces lignes n'apportant aucune information, nous définissons un torseur cinématique réduit T_r qui ne comporte que les degrés de liberté de la caméra réellement commandables. Nous obtenons finalement :

$$T_r = \begin{pmatrix} V_{\vec{y}_C}^{R_C} \\ V_{\vec{z}_C}^{R_C} \\ \Omega_{\vec{x}_C}^{R_C} \end{pmatrix} = \begin{pmatrix} -\sin(\vartheta) & C_x + D_x\cos(\vartheta) & C_x \\ \cos(\vartheta) & -C_x + D_x\sin(\vartheta) & -C_y \\ 0 & -1 & -1 \end{pmatrix} \dot{q} \tag{2.9}$$

2.1.2.6 Le jacobien du robot

La matrice jacobienne du robot relie le torseur cinématique de la caméra au vecteur commande du robot. A partir de l'équation 2.9, il est possible d'identifier la matrice jacobienne du robot comme étant :

$$J_r = \begin{pmatrix} -\sin(\vartheta) & C_x + D_x\cos(\vartheta) & C_x \\ \cos(\vartheta) & -C_x + D_x\sin(\vartheta) & -C_y \\ 0 & -1 & -1 \end{pmatrix} \tag{2.10}$$

Il est nécessaire de noter que $\det(J_r(\vartheta(t), t)) = -D_x$. Donc la matrice $J_r(q(t), t)$ est régulière et inversible pour $D_x \neq 0$.

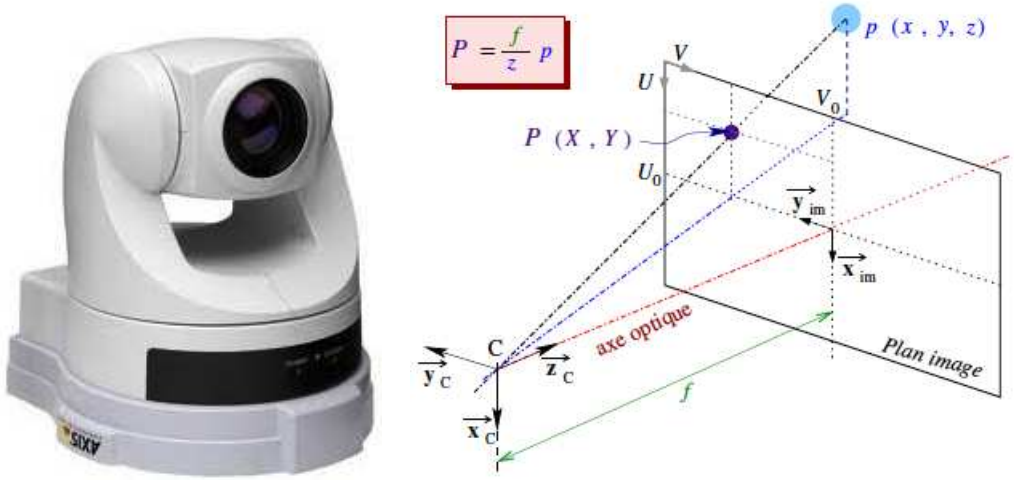
Tous les éléments en rapport avec la base mobile sont désormais décrits. Il est maintenant nécessaire de s'intéresser à la modélisation de la caméra. En effet, cette dernière tient une part importante dans la synthèse de lois de commande réalisant un asservissement visuel.

2.1.3 Modélisation de la caméra

Notre robot est équipé d'une caméra Axis 214 PTZ numérique couleur (Fig. 2.4(a)). Elle possède des capteurs CCD ExView HAD de 1/4 pouce. La valeur de la focale est comprise entre $3.54mm$ et $73.8mm$ permettant d'obtenir un zoom x18. Les images sont fournies avec un débit de 30 images par seconde. Elles sont au format JPEG avec une résolution allant de 160×120 à 704×576. Au cours des différentes expérimentations nous utiliserons une focale de $3.54mm$ et des images ayant une résolution de 704×576. Les paramètres intrinsèques de la caméra ont été obtenus lors de la phase de calibrage ou via le constructeur. Leurs valeurs sont résumées dans le tableau 2.1.

Centre optique	$U_0 = 291\ pixels$ $V_0 = 386\ pixels$	Données de calibrage
Focale	$f = 3.54\ mm$	Donnée de calibrage
Taille des pixels	$pix = 4.65\ \mu m$	Donnée constructeur

TABLE 2.1 – Valeurs des paramètres intrinsèques



(a) Caméra Axis 214 PTZ (b) Modèle sténopé de la caméra

FIGURE 2.4 – La caméra

La caméra est modélisée à l'aide du modèle sténopé (*Pinhole Camera Model*). Ce modèle est couramment utilisé pour les capteurs projectifs [Horaud & Monga 1995]. La modélisation est basée sur l'hypothèse selon laquelle tous les rayons passent par un seul point appelé centre optique. Comme nous pouvons le voir sur la figure 2.4(b), les points sont donc projetés sur le plan image selon une projection perspective en utilisant le point C comme centre optique. Ainsi, un point p de la scène ayant pour coordonnées $\underline{p}^{R_c} = [x\ y\ z]^T$ dans le repère R_C, est projeté sur le plan image en un point P de coordonnées métriques (X, Y), selon la relation suivante :

$$\begin{pmatrix} X \\ Y \end{pmatrix} = \begin{pmatrix} \frac{f}{z} & 0 & 0 \\ 0 & \frac{f}{z} & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{2.11}$$

où f représente la distance focale de la caméra (Fig. 2.4(b)). Les coordonnées de P peuvent être exprimées en pixels, $(U, V)^T$, à partir des coordonnées métriques $(X, Y)^T$ en utilisant la matrice des paramètres intrinsèques :

$$\begin{pmatrix} U \\ V \end{pmatrix} = \begin{pmatrix} \alpha_U & 0 & U_0 \\ 0 & -\alpha_V & V_0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} \tag{2.12}$$

avec $\alpha_U = \alpha_V = \frac{1}{pix}$ et pix la taille d'un pixel.

2.2 Les informations visuelles

Dans ce mémoire, nous proposons de réaliser une navigation par asservissement visuel. Pour cela, l'image obtenue par la caméra doit être préalablement traitée afin de fournir les informations visuelles utilisées dans la boucle de commande. De plus, ce type de commande nécessite

de connaître la matrice reliant les variations des informations visuelles dans l'image aux mouvements de la caméra (cf. section 2.3). Cette matrice est connue sous le nom de matrice d'interaction [Chaumette 1990].

2.2.1 Choix des primitives visuelles

Lors d'une navigation dans un environnement en milieu naturel, nous cherchons à caractériser l'environnement à partir d'informations visuelles indépendantes du zoom, de la résolution du capteur, du cadrage, de l'angle d'observation et de l'exposition. Pour cela, nous nous appuyons sur des points de Harris [Harris & Stephens 1988] ou bien des descripteurs SURF [Lowe 1999] ou SIFT [Bay *et al.* 2006]. Les informations visuelles sont alors des points. Bien que nous utilisons régulièrement des amers artificiels, notre choix pour réaliser une navigation visuelle se porte sur des primitives de type point.

2.2.2 La matrice d'interaction

Le choix du type de primitive visuelle utilisée étant fait, il est désormais nécessaire de calculer la matrice d'interaction associée. Nous rappelons ici, les calculs menant à l'obtention de son expression.

Un point p de l'espace, de coordonnées $\underline{p} = [x\ y\ z]^T$ dans R_C, est projeté sur le plan image en un point P, de coordonnées (X, Y) dans l'image, selon l'équation de projection (2.11). En différentiant l'équation (2.11), nous obtenons les variations dans l'image des coordonnées $X(t)$ et $Y(t)$ de P par rapport à la vitesse $V_{p/R_C}^{R_C}(t)$ du point p :

$$\left\{ \begin{array}{l} \dot{X}(t) = \left(\begin{array}{ccc} \frac{f}{z(t)} & 0 & \frac{-x(t)f}{z(t)^2} \end{array} \right). V_{p/R_C}^{R_C}(t) \\ \dot{Y}(t) = \left(\begin{array}{ccc} 0 & \frac{f}{z(t)} & \frac{-y(t)f}{z(t)^2} \end{array} \right). V_{p/R_C}^{R_C}(t) \end{array} \right. \tag{2.13}$$

Dans notre étude, les cibles considérées sont immobiles. Le point p appartenant à l'une des cibles, est donc fixe tandis que la caméra est animée d'une vitesse de translation $\overrightarrow{V}_{C/R_O}(t)$ et d'une vitesse de rotation $\overrightarrow{\Omega}_{R_C/R_O}(t)$. La vitesse du point p s'écrit :

$$\overrightarrow{V}_{p/R_C}(t) = -\overrightarrow{V}_{C/R_O}(t) - \overrightarrow{\Omega}_{R_C/R_O}(t) \times \overrightarrow{CP}. \tag{2.14}$$

A l'aide de (2.14), l'équation (2.13) peut alors se réécrire sous la forme :

$$\begin{array}{c} \left[\begin{array}{c} \dot{X}(t) \\ \dot{Y}(t) \end{array} \right] = L_{XY}(X(t), Y(t), z(t)) T_{C/R_0}^{R_C}(t) \\ \left[\begin{array}{c} \dot{X}(t) \\ \dot{Y}(t) \end{array} \right] = \left[\begin{array}{cccccc} -\frac{f}{z(t)} & 0 & \frac{X(t)}{z(t)} & \frac{X(t)Y(t)}{f} & -\left(f + \frac{X(t)^2}{f}\right) & Y(t) \\ 0 & -\frac{f}{z(t)} & \frac{Y(t)}{z(t)} & f + \frac{Y(t)^2}{f} & -\frac{X(t)Y(t)}{f} & -X(t) \end{array} \right] T_{C/R_0}^{R_C}(t) \end{array} \tag{2.15}$$

où $L_{XY}(X(t), Y(t), z(t))$ est la matrice d'interaction du point, reliant les variations des coordonnées d'un point dans l'image au torseur cinématique de la caméra. Notons que la matrice

d'interaction d'un point p dépend de la composante z dans le repère caméra que nous appellerons profondeur par la suite.

2.3 La commande référencée vision

2.3.1 Sélection de la loi de commande

Afin de réaliser une navigation par asservissement visuel, diverses lois de commandes s'offrent à nous. Les asservissements visuels 3D (Position Based Visual Servoing ou *PBVS*) et 2D (Image Based Visual Servoing ou *IBVS*) [Chaumette & Hutchinson 2006] constituent les approches les plus générales. Le *PBVS* contrôle la caméra à l'aide d'une boucle de commande où la situation de la caméra est explicitement prise en compte. Un processus d'estimation de la situation basé sur les données visuelles est alors mis en place. Ce dernier, assimilable à une localisation métrique, rend la méthode sensible aux bruits de mesures et aux erreurs de modélisation. Lors de l'asservissement, la caméra cherche à atteindre la situation de référence le plus directement possible, entraînant des mouvements pertinents dans l'espace de travail. Cependant l'évolution des indices visuels dans l'image n'est pas prise en compte, et il peut arriver que la caméra perde la cible de vue [Chaumette & Hutchinson 2006].

L'*IBVS* consiste à contrôler le robot en utilisant directement les informations visuelles dans la boucle de commande. Différentes primitives [2] peuvent être considérées : le point, le segment, la droite, l'ellipse [Chaumette 1990], les moments de l'image [Chaumette 2004], ... Le choix des primitives influence la trajectoire de la caméra dans l'espace de travail lors de l'asservissement. L'*IBVS* permet d'obtenir une certaine robustesse vis-à-vis des erreurs de modélisation. En effet, aucun processus d'estimation n'est présent dans la boucle. Avec cette méthode, la caméra cherche à atteindre le plus directement possible le but qui est défini dans l'image. Nous obtenons alors de très bonnes trajectoires des indices visuels dans l'image. Néanmoins, les mouvements de la caméra dans l'espace de travail nécessaires pour réaliser cette performance, peuvent apparaître compliqués et inutiles [Chaumette 1998]. De plus, les asservissements nécessitant de grandes rotations et translations selon un même axe peuvent entraîner un échec de la régulation de l'erreur.

Les deux solutions n'étant pas entièrement satisfaisantes, des méthodes hybrides ont été développées telles que celle proposée par Deguchi [Deguchi 1998] ou l'asservissement visuel $2D1/2$ [Malis *et al.* 1999]. Les auteurs proposent de découpler le calcul des vitesses de translation de celui des vitesses de rotations. Dans [Deguchi 1998], les translations sont calculées sur la base d'informations 3D tandis que les rotations sont calculées à partir d'informations 2D. Dans [Malis *et al.* 1999] c'est l'idée inverse qui est développée. Dans les deux cas, les auteurs tentent de tirer avantage de l'*IBVS* et du *PBVS*. En effet, une étude présentée dans [Gans *et al.* 2003] montre que ces deux méthodes permettent de garder les indices visuels dans le champ de vue de la caméra lors de grands déplacements tout en ayant des mouvements pertinents de la caméra. Cependant, les données 3D étant estimées, elles sont sensibles aux bruits de mesures et aux erreurs de modélisation [Gans *et al.* 2003].

Une dernière approche consiste à utiliser différents jeux de primitives visuelles afin d'obtenir de bonnes trajectoires de la caméra [Chaumette 1990]. Dans [Corke & Hutchinson 2001], les auteurs proposent de calculer séparément les vitesses de translation et de rotations autour de

2. Nous recommandons vivement la lecture de [Chaumette 1990], [Chaumette 2004] et [Tahri & Chaumette 2005] qui offrent une bonne vue d'ensemble de ce problème.

l'axe optique. Celles-ci sont alors déterminées à partir de deux primitives particulières. Les autres, étant définies classiquement pour un asservissement visuel 2D, sont des points. En l'absence de processus d'estimation, cette approche est robuste aux bruits de mesures et aux erreurs de modélisation. De plus, elle permet d'obtenir des performances satisfaisantes lors de grands déplacements selon l'axe optique. Cependant, cette méthode est sensible au point de vue de la caméra lors du calcul des vitesses relatives à l'axe optique.

Dans notre étude nous nous intéressons à la commande d'une caméra embarquée sur un robot muni d'une platine. La caméra ne possédant plus tous ses degrés de liberté, les problèmes liés aux translations et rotations selon un même axe ne se posent plus. Il en est de même pour les rotations autour de l'axe optique qui ne sont plus permises. Afin de réaliser notre tâche de navigation, nous avons finalement choisi d'exploiter l'asservissement visuel 2D. Ainsi, nous bénéficions d'une loi de commande robuste aux erreurs de mesure et aux incertitudes de modélisation.

La loi de commande que nous utiliserons dans la suite de ce document est basée sur le formalisme des fonctions de tâches. Nous rappelons d'abord ce formalisme avant de présenter la synthèse de la loi de commande. Nous terminerons avec des simulations illustrant le comportement de notre système robotique lors de tâches de navigation.

2.3.2 Le formalisme des fonctions de tâche

2.3.2.1 Présentation du formalisme

Le formalisme des fonction de tâches a été introduit dans [Samson *et al.* 1991] afin de définir un cadre unificateur adapté à la plupart des schémas de commandes existants pour les robots manipulateurs holonomes. Il fut par la suite étendu au cas des robots mobiles non-holonomes dans [Pissard-Gibollet 1993]. Ce formalisme consiste à définir une tâche robotique à l'aide d'une fonction de sortie particulière : la fonction de tâche, notée $e(q(t), t)$ et prenant ses valeurs dans $\mathbb{R}^n$, où n correspond au nombre de degrés de liberté du robot. Plus particulièrement une fonction de tâche dépend d'une fonction de sortie et d'une consigne modélisant l'objectif à atteindre. La fonction de sortie peut être exprimée en termes de données proprioceptives et/ou extéroceptives selon la tâche considérée et les capteurs disponibles.

La réalisation de la tâche est effectuée à travers la régulation à zéro de la fonction de tâche sur un horizon de temps $t \in [0, T]$. La loi de commande permettant de contrôler le robot n'est alors plus exprimée dans l'espace d'état mais dans l'espace de sortie (ou encore espace capteur). Pour que le problème de commande soit correctement posé, il est nécessaire de s'assurer qu'il existe un lien régulier entre l'espace d'état et l'espace de sortie. Pour cela, la notion de $\rho-$admissibilité a été introduite dans [Samson *et al.* 1991]. Elle permet de réunir les conditions suffisantes de régularité que doit vérifier la fonction de tâche.

2.3.2.2 La notion de $\rho-$admissibilité

Soit $e(q(t), t)$ une fonction de tâche correspond à la tâche robotique à réaliser. Cette dernière est parfaitement réalisée sur un horizon de temps $t \in [0, T]$ si $e(q(t), t) = 0,\ \forall t \in [0, T]$. Une trajectoire vérifiant cette équation pour $\forall t \in [0, T]$ est dite trajectoire solution et est notée $q_r(t)$. Afin que $q_r(t)$ puisse être réalisée par le robot, elle doit être de classe $\mathcal{C}^1$ sur l'horizon $[0, T]$.

Afin de synthétiser une loi de commande dans l'espace de sortie, le lien entre l'espace d'état et l'espace de sortie doit être régulier. Pour cela, la trajectoire solution doit être unique. La résolution de l'équation $e(q(t),t) = 0$ pouvant mener à la définition de plusieurs trajectoires solutions, il est nécessaire d'introduire une condition initiale q_0, telle que $q_r(0) = q_0$ et $e(q_0, 0) = 0$, afin de faire un choix. Une trajectoire solution vérifiant cette dernière propriété est dite trajectoire idéale. Notons cependant que q_0 représente la configuration à laquelle le robot devrait se trouver à l'instant initial si la tâche était parfaitement réalisée. Elle ne désigne pas nécessairement sa position initiale.

L'introduction de la condition initiale q_0 n'est pas suffisante pour garantir l'unicité de la trajectoire solution. En effet la non régularité de la jacobienne de la tâche $\frac{\partial e}{\partial q^T}$ au voisinage de la trajectoire solution peut entrainer le fait que l'équation $e(q(t),t) = 0$ n'ait que la condition initiale comme solution, ou au contraire une infinité de solutions. Le problème de commande est alors mal posé puisque dans le premier cas, la tâche ne peut être réalisée tandis que dans le second cas, il existe une infinité de trajectoires dans l'espace d'état permettant l'annulation de $e(q(t),t)$. Il est nécessaire de ne pas confondre le dernier cas avec le problème de plusieurs trajectoires solutions distinctes. Ce problème a été résolu en introduisant la condition initiale q_0. La régularité de la matrice jacobienne de la tâche est donc essentielle dans la définition de la $\rho-$admissibilité de la fonction de tâche.

Le bon conditionnement du problème ne dépend pas uniquement de l'unicité de la trajectoire. Il est aussi nécessaire que la convergence de $e(q(t),t)$ vers 0 entraine la convergence de $q(t)$ vers $q_r(t)$. De plus, une petite variation autour de $e(q(t),t)$ ne doit pas induire une grande variation autour de $q(t)$. La propriété de $\rho-$admissibilité tient compte de ces contraintes puisqu'elle permet d'établir l'existence d'un difféomorphisme F entre l'ensemble des couples (q,t) incluant la condition initiale q_0 et l'ensemble des couples (e,t). Elle définit donc bien le lien régulier nécessaire permettant au problème d'être correctement posé. La $\rho-$admissibilité est définie formellement comme suit :

Définition 1 : *Soit $\mathcal{C}_{\rho,T}$ l'ensemble des couples (q,t) et B_ρ une boule fermée de centre 0 et de rayon ρ. Si $T < +\infty$, la propriété de $\rho-$admissibilité est équivalente aux trois assertions suivantes :*

1. *$\mathcal{C}_{\rho,T}$ est une composante connexe par arc non vide de $F^{-1}(B_\rho \times [0,T])$*
2. *$\mathcal{C}_{\rho,T}$ est une partie fermée de $\mathbb{R}^n \times \mathbb{R}$*
3. *Pour tout couple (q,t) appartenant à $\mathcal{C}_{\rho,T}$:*
   - *(a) $det(\frac{\partial e}{\partial q^T}(q,t)) \neq 0$*
   - *(b) $\|(\frac{\partial e}{\partial q^T})^{-1}\| < m_{\rho,T} < \infty$*
   - *(c) $\|\frac{\partial e}{\partial t}\| < m'_{\rho,T} < \infty$*

où $\|.\|$ représente la norme euclidienne ou la norme spectrale associée. Si $T = +\infty$, ces conditions sont seulement suffisantes.

La détermination de la composante connexe $\mathcal{C}_{\rho,T}$ définie par l'assertion (1), correspond au choix de la condition initiale $q_0 \in \mathcal{C}_{\rho,T}$. L'existence du difféomorphisme est assuré par (3a), tandis que les conditions (1) et (2) indiquent que F est défini entre deux ensembles fermés. Enfin, les constantes $m_{\rho,T}$ et $m'_{\rho,T}$ garantissent que l'inverse de la jacobienne de la tâche et la dérivée de e par rapport au temps restent bornées. Il est ainsi assuré qu'une petite variation dans l'un des espaces entraine une petite variation dans l'autre. De plus, cela signifie que même en présence d'erreurs de modélisation et de bruits de mesure, la fonction de tâche restera au voisinage de zéro et la trajectoire réelle du robot au voisinage de la trajectoire idéale.

Il est important de noter que les singularités de la matrice jacobienne de la tâche sont les principaux obstacles à l'obtention de la propriété de la $\rho-$admissibilité. La jacobienne revêt donc une importance capitale dans la définition d'une tâche $\rho-$admissible, comme le souligne le théorème suivant [Samson *et al.* 1991] :

Théorème 1 : *Soit $e(q,t);q_0$ une tâche admettant une trajectoire idéale $q_r(t)$ définie sur l'intervalle de temps $[0,T]$. Si $e(q,t)$ est de classe $\mathcal{C}^2$, si la matrice jacobienne de la tâche est inversible au voisinage de $q_r(t)$, et enfin si l'horizon de temps reste fini ($T<+\infty$), il existe un $\rho>0$ tel que la tâche considérée est $\rho-$admissible.*

L'inversibilité de la matrice jacobienne de la fonction de tâche au voisinage de la trajectoire solution est la condition essentielle permettant d'établir la $\rho-$admissibilité. Il est donc suffisant de choisir la tâche de manière à contraindre autant de degrés de liberté que d'actionneurs disponibles. L'obtention de la $\rho-$admissibilité est alors un problème de modélisation de la tâche à réaliser.

La $\rho-$admissibilité établissant un difféormorphisme entre l'espace d'état et l'espace de la tâche, elle permet de montrer qu'il est identique de synthétiser une loi de commande dans un des deux espaces.

2.3.3 Synthèse de la commande pour la navigation

Lors d'un asservissement visuel 2D, nous cherchons à réaliser une liaison virtuelle [Chaumette 1990] entre la caméra et l'environnement. Les liaisons virtuelles sont assimilables aux liaisons mécaniques classiques telles que les liaisons rigides, prismatiques ou encore rotoïdes. Par exemple, le positionnement de la caméra par rapport à une cible correspond à une liaison rigide. Le maintien de la caméra parallèle à un mur et à une certaine distance de celui-ci, correspond à une liaison appui plan. Le type de liaison à réaliser est fixé lors du choix des primitives visuelles composant le vecteur de mesure noté $s(q(t),t)$. En effet, à un type de primitive correspond une matrice d'interaction (cf. section 2.2) et son analyse, plus particulièrement du rang et des lignes et colonnes nulles, permet de déterminer le type de liaison virtuelle réalisée [Chaumette 1990]. Le rang de la matrice d'interaction nous renseigne notamment sur le nombre de degrés de libertés contraints. Dans [Chaumette 1990], l'auteur propose plusieurs jeux de primitives permettant de réaliser différentes liaisons virtuelles.

Les tâches de navigation considérées ici visent à positionner la caméra par rapport à une cible fixe. Il s'agit de réaliser une liaison virtuelle rigide. Dans cette section nous proposons donc de présenter la synthèse d'une loi de commande permettant de réaliser une telle liaison. La synthèse de la loi de commande s'appuie sur le formalisme des fonctions de tâches. Bien que ce dernier soit basé sur la régulation d'une erreur à zéro, il est régulièrement utilisé pour répondre au problème d'asservissement [Chaumette 1990]. La première étape de la synthèse consiste à définir un vecteur de mesure s permettant de réaliser une liaison virtuelle rigide. Pour une caméra possédant six degrés de liberté, quatre points $P_1(X_1,Y_1)$, $P_2(X_2,Y_2)$, $P_3(X_3,Y_3)$ et $P_4(X_4,Y_4)$ sont nécessaires pour que la tâche de positionnement puisse être correctement réalisée [Horaud 1987]. Bien que dans notre cas, la caméra ne possède que trois degrés de liberté, nous utiliserons quatre points. Le vecteur de mesure est alors défini comme suit :

$$s(q(t)) = [X_1\ Y_1\ X_2\ Y_2\ X_3\ Y_3\ X_4\ Y_4]^T \tag{2.16}$$

Le vecteur $s(q(t))$ ne dépend pas explicitement du temps, car dans notre étude les cibles sont

considérées immobiles. Nous pouvons désormais modéliser la tâche de positionnement à l'aide d'une fonction de tâche $e(q(t))$ définie comme suit :

$$e_{av}(q(t)) = C(s(q(t)) - s^*) \tag{2.17}$$

où s^* correspond au vecteur de mesures obtenu pour la configuration finale désirée de la caméra. La matrice C, dite matrice de combinaison, permet d'harmoniser la dimension de la fonction de tâche, c'est-à-dire que $\dim(e_{av}(q(t))) = n$, où n est le nombre de degrés de liberté commandables. Ainsi il est possible d'utiliser un nombre d'indices visuels plus grand que le nombre de degrés de liberté à contraindre. Nous reviendrons sur le choix de C par la suite.

Tout d'abord, il est nécessaire de s'assurer que le problème de commande est correctement posé, c'est-à-dire que la fonction de tâche est ρ-admissible. Une condition nécessaire consiste à ce que la matrice jacobienne de la fonction de tâche $J_{av} = \frac{\partial e_{av}}{\partial q}$ soit inversible. A partir de l'équation (2.17) nous obtenons l'équation de la matrice jacobienne :

$$J_{av} = \frac{\partial e_{av}}{\partial q} = C\frac{\partial s}{\partial r}\frac{\partial r}{\partial q} \tag{2.18}$$

Dans l'équation (2.18), les termes $\frac{\partial s}{\partial r}$ et $\frac{\partial r}{\partial q}$ correspondent respectivement à la matrice d'interaction de la caméra L associée aux indices visuels contenus dans le vecteur de mesure, et à la matrice jacobienne du robot J_r. Pour un vecteur de mesures s donné par l'équation (2.16), L est obtenue par la concaténation des matrices d'interactions des points données par l'équation (2.15) :

$$L = \left[L_{X_1Y_1}^T \; L_{X_2Y_2}^T \; L_{X_3Y_3}^T \; L_{X_4Y_4}^T\right]^T \tag{2.19}$$

La caméra ne possédant tous ses degrés de liberté, nous avons vu dans la section 2.1.2 qu'il était possible de considérer un torseur réduit noté T_r. Il est donc nécessaire d'adapter la matrice d'interaction à ce cas particulier. Nous obtenons alors une matrice d'interaction réduite pour quatre points définie comme suit :

$$L_r = \begin{bmatrix} L_{r_{P_1}} \\ L_{r_{P_2}} \\ L_{r_{P_3}} \\ L_{r_{P_4}} \end{bmatrix} = \begin{bmatrix} 0 & \frac{X_1}{z_1} & \frac{X_1Y_1}{f} \\ -\frac{f}{z_1} & \frac{Y_1}{z_1} & \left(f + \frac{Y_1^2}{f}\right) \\ 0 & \frac{X_2}{z_2} & \frac{X_2Y_2}{f} \\ -\frac{f}{z_2} & \frac{Y_2}{z_2} & \left(f + \frac{Y_2^2}{f}\right) \\ 0 & \frac{X_3}{z_3} & \frac{X_3Y_3}{f} \\ -\frac{f}{z_3} & \frac{Y_3}{z_3} & \left(f + \frac{Y_3^2}{f}\right) \\ 0 & \frac{X_4}{z_4} & \frac{X_4Y_4}{f} \\ -\frac{f}{z_4} & \frac{Y_4}{z_4} & \left(f + \frac{Y_4^2}{f}\right) \end{bmatrix} \tag{2.20}$$

Le rang de la matrice d'interaction L_r pour quatre points est de rang 3. Lorsque la tâche est réalisée, tous les degrés de liberté commandables de la caméra sont contraints, ce qui

correspond bien à une tâche de positionnement. L'équation (2.18) peut donc se réécrire sous la forme :

$$J_{av} = CL_rJ_r \tag{2.21}$$

Dans la mesure où la matrice jacobienne du robot J_r est une matrice régulière, la stabilité et la convergence de la fonction de tâche sont assurées par la condition suffisante suivante :

$$CL_r > 0 \tag{2.22}$$

Un choix judicieux consiste alors à choisir $C = L_r^+$ [3]. Dans certains contextes, il n'est pas possible de connaitre la valeur exacte de la matrice d'interaction. On utilise généralement une estimation de cette dernière notée $\hat{L}_r$. Un choix classique consiste à choisir $\hat{L}_r = L_r^*$ [4] et $C = L_r^{*+}$ [Chaumette 1990]. Le jacobien de la fonction de tâche peut alors s'écrire comme suit :

$$J_{av} = J_r \tag{2.23}$$

Pour notre système robotique, J_r étant une matrice carrée de dimension 3 dont le déterminant est non nul (cf. 2.1.2), la fonction de tâche e_{av} est alors ρ-admissible.

Le problème de commande étant correctement posé, nous cherchons maintenant à annuler l'erreur avec une décroissance exponentielle. Nous obtenons donc la relation suivante :

$$\dot{e}_{av} = -\lambda_{av}e_{av} \tag{2.24}$$

où λ_{av} est un scalaire strictement positif ou une matrice définie positive permettant de fixer la vitesse de décroissance. Afin d'obtenir une expression analytique de l'équation (2.24), il est nécessaire de préalablement différencier $s(q(t), t)$ par rapport à $q(t)$ et t. Nous obtenons alors la variation des informations visuelles en fonction des mouvements de la caméra :

$$\dot{s}(q(t), t) = \frac{\partial s}{\partial q}\frac{dq}{dt} + \frac{\partial s}{\partial t} = \frac{\partial s}{\partial r}\frac{\partial r}{\partial q}\frac{dq}{dt} + \frac{\partial s}{\partial t} \tag{2.25}$$

Dans l'équation (2.25), on reconnaît :

- $\frac{\partial s}{\partial r}$ qui correspond à la matrice d'interaction.
- $\frac{\partial r}{\partial q}$ qui correspond à la matrice jacobienne du robot.
- $\frac{dq}{dt}$ qui correspond à la commande en vitesse de notre système.
- $\frac{\partial s}{\partial t}$ terme qui traduit les mouvements propres de l'objet par rapport à la caméra. Notre étude portant sur des cibles fixes, il vient alors : $\frac{ds}{dt} = 0$.

Pour une matrice C constante, $\dot{e}_{av}$ peut donc se réécrire sous la forme :

$$\dot{e}_{av} = C\dot{s} = CL_rJ_r\dot{q} \tag{2.26}$$

3. $L_r^+ = (L_r^TL_r)^{-1}L_r^T$ est la pseudo-inverse à gauche de la matrice d'interaction L_r.

4. L_r^* correspond à la matrice d'interaction calculée avec les indices visuels obtenus pour la configuration finale désirée de la caméra.

A l'aide des équations (2.24) et (2.26), nous obtenons :

$$CL_r J_r \dot{q} = -\lambda_{av} e_{av} \tag{2.27}$$

Finalement, en remplaçant e_{av} par son expression donnée dans l'équation (2.17), il vient :

$$CL_r J_r \dot{q} = -\lambda_{av} C(s - s^*) \tag{2.28}$$

A partir de l'équation (2.28), il est alors possible de calculer le vecteur de commande $\dot{q}$ en fonction de l'erreur entre les indices visuels courants et ceux désirés :

$$\dot{q} = -(CL_r J_r)^{-1} \lambda_{av} C(s - s^*) \tag{2.29}$$

La loi de commande donnée par l'équation (2.29) permet de réaliser la liaison virtuelle désirée en annulant la fonction de tâche (2.17). Le vecteur de commande réalisant la navigation et calculé à l'aide de l'équation (2.29) sera noté $\dot{q}_{av}$ par la suite. La commande calculée à l'aide de l'équation (2.29) est nulle lorsque $e_{av} = 0$ ou lorsque $(s - s^*) \in \ker((CL_r J_r)^{-1} C)$.

Simulation

Nous proposons de simuler un asservissement visuel 2D permettant de positionner la caméra embarquée face à une cible. Celle-ci est composée de quatre cercles dont les centres projetés dans l'image sont utilisés pour former le vecteur de mesure. Afin de s'approcher des conditions expérimentales, nous ajoutons un bruit gaussien de deux pixels lors du calcul des indices visuels. Nous utilisons l'estimée L_r^* de la matrice d'interaction et $C = L_r^{*+}$. Finalement la période d'échantillonnage est fixée à $T_e = 100ms$ tandis que le gain de commande est $\lambda_{av} = 0.35$. L'état initial du robot est $[x = 0, y = 0, \theta = 0, \vartheta = \frac{\pi}{4}]$ et la caméra doit atteindre la situation finale $[x = 3.1, y = 2, \theta_T = 0]$, avec $\theta_T = \theta + \vartheta$.

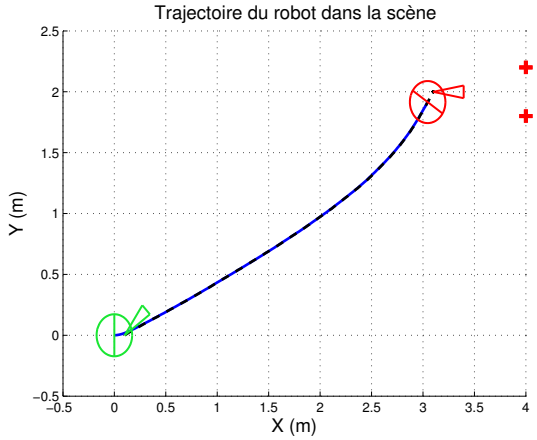
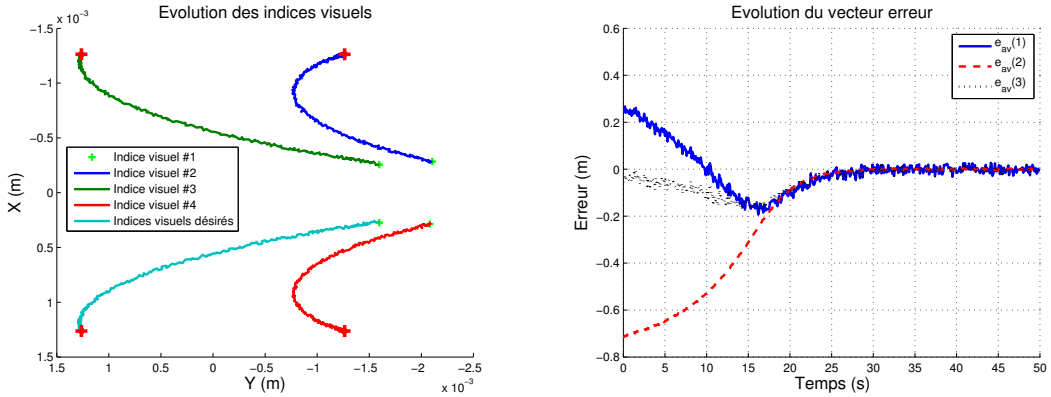


FIGURE 2.5 – Trajectoire du robot dans la scène

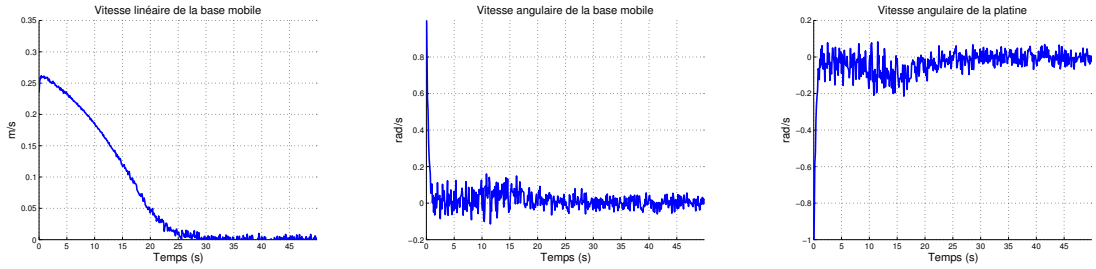
La loi de commande présentée dans l'équation (2.29) permet à la caméra d'atteindre la situation désirée (figure 2.5). Les trois composantes de la fonction de tâche convergent vers zéro (figure 2.6(b)) malgré la présence des bruits (figure 2.6(a)). De plus, la régulation à zéro de la fonction de tâche correspond à la convergence des indices visuels courants vers ceux de référence (figure 2.6(a)). Enfin la figure 2.7 présente l'évolution du vecteur de commande permettant au système robotique de positionner la caméra à la situation désirée. Nous pouvons remarquer que la

régulation à zéro de la fonction de tâche entraine une commande nulle, ce qui correspond à la liaison virtuelle rigide que nous désirions réaliser. L'asservissement visuel 2D nous permet donc de réaliser la tâche de positionnement souhaitée.



(a) Evolution des indices visuels (b) Evolution de $e_{av}(q(t))$

FIGURE 2.6 – Evolution de $s(q(t))$ et de $e_{av}(q(t))$



(a) Vitesse linéaire v (b) Vitesse angulaire ω (c) Vitesse de la platine ϖ

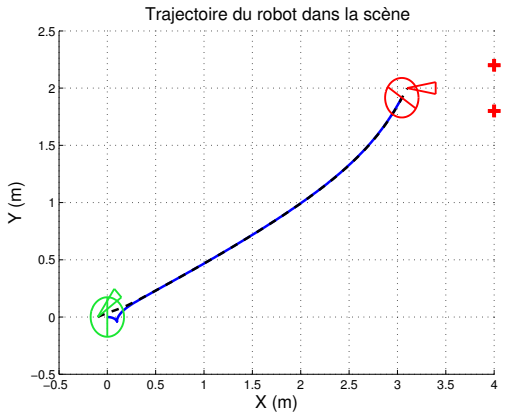
FIGURE 2.7 – Evolution du vecteur de commande $\dot{q}(t)$

Nous avons réalisé une seconde simulation similaire à la précédente mais avec une configuration initiale du robot $[x = 0, y = 0, \theta = \pi, \vartheta = \frac{-3\pi}{4}]$. De la même manière que précédemment la tâche est parfaitement réalisée. Cependant, nous pouvons remarquer sur la figure 2.8(b) [5] l'existence d'un point de rebroussement. Pour comprendre ce phénomène, il est nécessaire de rappeler que les différents asservissements visuels présentés dans la section 2.3.1 permettent de commander la caméra. En effet, dans ces méthodes on considère que celle-ci peut se mouvoir de façon holonome. Or le système ici considéré comprend une base mobile non holonome munie d'une platine. La loi de commande calcule donc un torseur cinématique pertinent pour la caméra qui est traduit en vecteur de commande pour le robot à l'aide de la matrice jacobienne du robot J_r. Dans la figure 2.8(a), nous remarquons donc la manœuvre du robot permettant de réaliser une trajectoire lisse de la caméra.

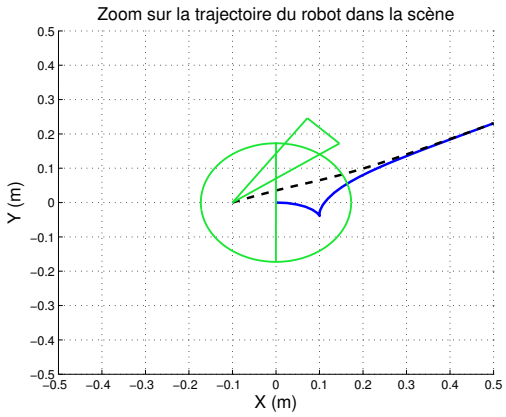
Afin de s'assurer que les points de rebroussement ne sont pas dus à la commande utilisée, nous avons rejoué la même simulation en calculant la loi de commande à partir de la méthode de Corke et Hutchinson [Corke & Hutchinson 2001]. Bien que la trajectoire globale observée dans la figure 2.8(c) soit sensiblement différente de celle obtenue précédemment, il existe toujours un point de rebroussement 2.8(d). De même, un changement d'indices visuels modifierait la

5. Le trait plein bleu correspond à la trajectoire du point M centre de la base mobile et le trait discontinu noir à la trajectoire du point C centre de la caméra.

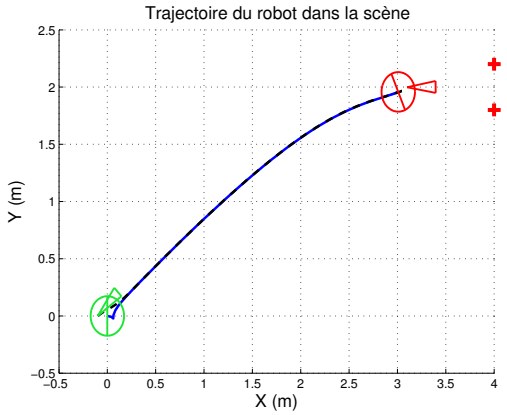
trajectoire de la caméra et la base mobile s'adapterait à celle-ci. Ce phénomène est donc dû à notre base mobile non holonome portant la caméra et apparaît pour certaines configurations initiales particulières.



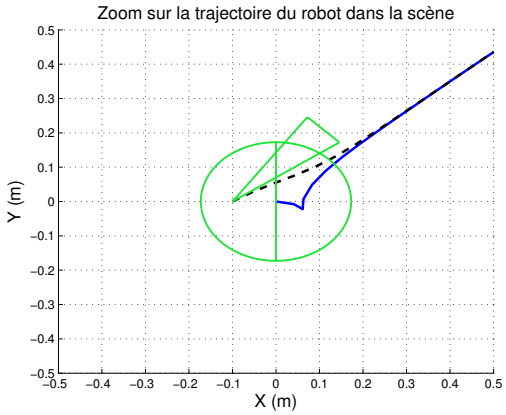
(a) Trajectoire pour un asservissement visuel 2D



(b) Zoom sur la trajectoire pour un asservissement visuel 2D



(c) Trajectoire pour un asservissement "Corke et Hutchinson"



(d) Zoom sur la trajectoire pour un asservissement "Corke et Hutchinson"

FIGURE 2.8 – Etude de la trajectoire de la base mobile avec un point de rebroussement

Bien que les points de rebroussement ne mettent pas en péril la réalisation de la tâche, nous verrons dans le chapitre 4 qu'ils présentent des inconvénients lors d'une navigation incluant de grands déplacements. Nous présenterons alors une méthode permettant de les éviter.

2.4 Conclusion

Dans ce chapitre, nous avons présenté le système robotique utilisé pour réaliser une navigation. Nous avons ensuite introduit les différents modèles nécessaires avant de nous intéresser à la synthèse d'une commande réalisant un asservissement visuel 2D. Il est désormais possible de réaliser une navigation dans un environnement libre. Cependant cette hypothèse n'est pas réaliste pour une navigation en intérieur où la présence d'obstacles est inévitable. Avec

cette nouvelle contrainte, deux problèmes pouvant entrainer l'échec de la navigation apparaissent : le risque de collision du robot avec un obstacle et la perte du signal visuel durant la navigation.

Dans un premier temps, nous nous focalisons sur la perte du signal visuel nécessaire au calcul de la loi de commande. Afin de répondre à cette problématique, nous nous appuyons sur les travaux présentés dans [Folio 2007]. L'auteur propose d'estimer l'évolution des indices visuels à partir des commandes appliquées au robot et du modèle d'évolution des indices visuels (2.15) lorsque l'image n'est plus disponible. Pour cela il est nécessaire de fournir une estimation de la "profondeur" de chaque indice visuel. Dans le prochain chapitre, après avoir rappelé les principaux résultats présentés dans [Folio 2007], nous allons nous attacher à résoudre ce problème en proposant une méthode robuste aux erreurs de mesure. Il sera alors possible de réaliser une navigation même en présence d'occultation.

CHAPITRE 3

CONTRIBUTION À LA GESTION DES OCCULTATIONS

Sommaire

Dans le chapitre précédent, nous nous sommes intéressés à la commande d'un robot par asservissement visuel 2D dans un environnement libre. Cette dernière hypothèse étant peu réaliste, il est nécessaire de considérer un environnement encombré d'obstacles possiblement occultants. Ici, nous nous intéressons à la perte de vue des indices visuels qui peut entraîner un échec de la navigation (Fig. 3.1). La perte de vue peut être due à une occultation partielle ou totale de la cible, une erreur dans le traitement d'images, une panne de la caméra ou bien encore à une utilisation de plus haute priorité de la caméra. Par la suite, toutes ces notions seront regroupées derrière le terme occultation.

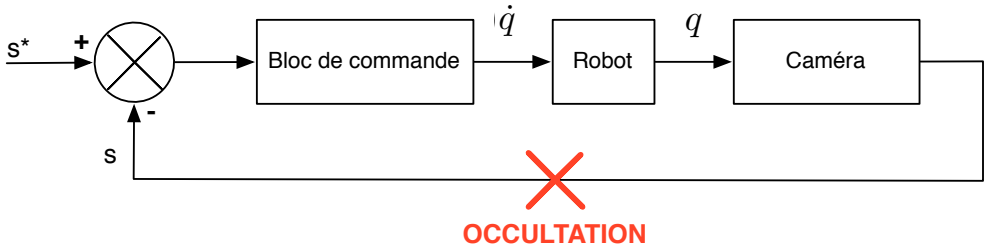


FIGURE 3.1 – Effet d'une occultation sur un asservissement visuel

Dans la littérature, une première série de travaux propose de gérer des occultation partielles [Malis *et al.* 1999] [Comport *et al.* 2004], [Garcia-Aracil *et al.* 2005]. Cependant, nous cherchons à gérer des occultations totales et les méthodes citées ne conviennent pas à notre cas. D'autres solutions basées sur le tracking [Jackson *et al.* 2004] [Lepetit & Fua 2006], le filtre de Kalman [Kalman 1960] ou le filtrage particulaire [Crisan & Doucet 2002]) [1] traitent la perte complète du signal visuel. Cependant toutes ces approches reposent sur des mesures dans l'image et ne sont pas bien adaptées pour nous.

Nous nous sommes orientés vers l'approche proposée dans [Folio 2007] et [Folio & Cadenat 2008] où les auteurs proposent de reconstruire la primitive visuelle s lorsque la caméra n'est plus en mesure de fournir d'image. Pour cela, ils proposent d'exploiter le lien existant entre s et le torseur cinématique de la caméra $T^{R_C}_{C/R_O}$ défini comme suit :

$$\dot{s} = L_s T^{R_C}_{C/R_O} \tag{3.1}$$

où L_s est la matrice d'interaction correspondant aux primitives visuelles composant s. La résolution de l'équation (3.1) permet de déterminer s. Il est alors possible d'utiliser ce dernier dans la boucle de commande comme cela est présenté dans la figure 3.2.

3.1 Gestion des occultations par reconstruction des indices visuels

Dans [Folio 2007], l'auteur propose des solutions analytiques et numériques pour différents types de primitives visuelles et de systèmes robotiques. Dans cette section, nous nous focalisons sur le cas particulier des points et du robot non holonome considéré dans ces travaux. Dans ce cas il est possible d'exhiber une solution analytique. Nous rappelons ci-après la démarche suivie permettant de l'obtenir.

1. Nous conseillons l'étude bibliographique plus approfondie proposée dans [Folio 2007].

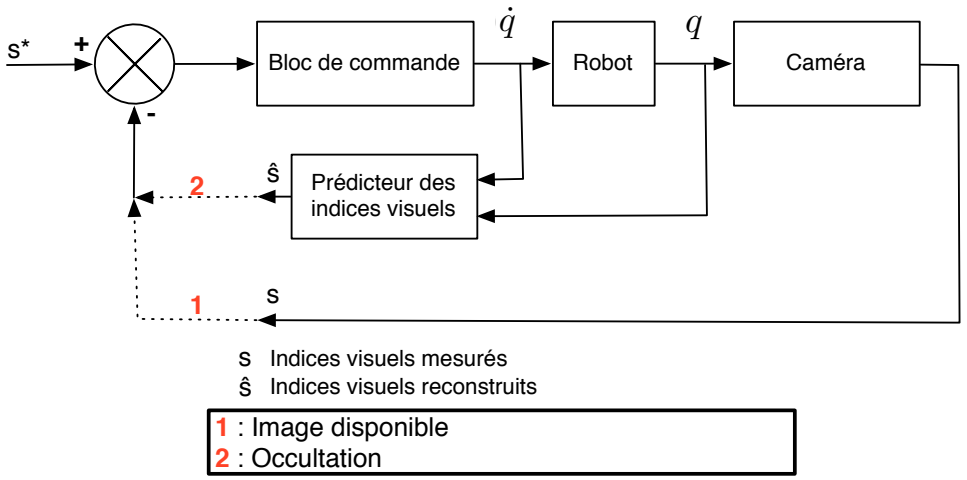


FIGURE 3.2 – Schéma bloc d'un asservissement visuel avec gestion de l'occultation par estimation des indices visuels

3.1.1 Calcul des équations d'évolution des indices visuels

Pour le système robotique considéré et un vecteur $s_P = [X_P \ Y_P]^T$ composé des coordonnées métriques d'un point P dans l'image, l'équation (3.1) se réécrit de la manière suivante [Folio 2007] :

$$\dot{s}_P(q(t),t) = L_{r_P}(X_P(t), Y_P(t), z_P(t))T_r = L_{r_P}(X_P(t), Y_P(t), z_P(t))J_r(\vartheta)\dot{q}(t) \tag{3.2}$$

où

$$J_r(\vartheta) = \begin{pmatrix} -\sin(\vartheta) & C_x + D_x\cos(\vartheta) & C_x \\ \cos(\vartheta) & -C_x + D_x\sin(\vartheta) & -C_y \\ 0 & -1 & -1 \end{pmatrix} \tag{3.3}$$

$$L_{r_P}(X_P(t), Y_P(t), z_P(t)) = \begin{pmatrix} 0 & \frac{X_P(t)}{z_P(t)} & \frac{X_P(t)Y_P(t)}{f} \\ -\frac{f}{z_P(t)} & \frac{Y_P(t)}{z_P(t)} & f + \frac{Y_P(t)^2}{f} \end{pmatrix} \tag{3.4}$$

et $(x_P \ y_P \ z_P)^T$ les coordonnées du point P dans le repère R_C. L'équation (3.2) peut se réécrire sous forme du système d'équations ci-après [Folio 2007] :

$$\begin{aligned} \dot{X}_P(t) = \frac{X_P(t)}{z_P(t)}\Big(v(t)\cos(\vartheta(t)) + \omega(t)(-C_y + D_x\sin(\vartheta(t))) - C_y\varpi(t)\Big) \\ -\frac{X_P(t)Y_P(t)}{f}(\omega(t) + \varpi(t)) \end{aligned} \tag{3.5}$$

$$\begin{aligned} \dot{Y}_P(t) = -\frac{f}{z_P(t)}\Big(-v(t)\sin(\vartheta(t)) + \omega(t)(C_x + D_x\cos(\vartheta(t))) + C_x\varpi(t)\Big) \\ +\frac{Y_P(t)}{z_P(t)}\Big(v(t)\cos(\vartheta(t)) + \omega(t)(-C_y + D_x\sin(\vartheta(t))) - C_y\varpi(t)\Big) - \Big(f + \frac{Y_P^2(t)}{f}\Big)(\omega(t) + \varpi(t)) \end{aligned} \tag{3.6}$$

Comme le montrent les équations (3.5) et (3.6), le paramètre $z_P(t)$ [2] doit être connu afin de pouvoir calculer l'évolution des indices visuels. Pour cela, il est nécessaire d'exprimer $\dot{z}_P(t)$ en

2. z_P sera par la suite régulièrement et abusivement nommé profondeur.

fonction des commandes $\dot{q}(t)$. Dans un premier temps rappelons que pour tout point p fixe et appartenant au repère scène R_O, sa vitesse $\overrightarrow{V}_{p/R_C}$ par rapport au repère mobile R_C de la caméra peut s'écrire de la manière suivante :

$$\overrightarrow{V}_{p/R_C}(t) = -\overrightarrow{V}_{C/R_O}(t) - \overrightarrow{\Omega}_{R_C/R_O}(t) \wedge \overrightarrow{CP} \tag{3.7}$$

Rappelons que $CP^{R_C} = (x_P \ y_P \ z_P)^T$ et $V^{R_C}_{p/R_C} = (\dot{x}_P \ \dot{y}_P \ \dot{z}_P)^T$. Nous avons alors :

$$\dot{z}_P(t) = -V_{\vec{z}_C}(t) - \frac{z_P(t)Y_P(t)}{f}\Omega_{\vec{x}_C}(t) + \frac{z_P(t)X_P(t)}{f}\Omega_{\vec{y}_C}(t) \tag{3.8}$$

Etant donné que pour le système robotique considéré, $\Omega_{\vec{y}_C}(t) = 0$, l'équation (3.8) devient :

$$\dot{z}_P(t) = -\Big(\upsilon(t)\cos(\vartheta(t)) + \omega(t)\big(-C_y + D_x\sin(\vartheta(t))\big) - C_y\varpi(t)\Big) + \frac{z_P(t)Y_P(t)}{f}\Big(\omega(t) + \varpi(t)\Big) \tag{3.9}$$

Le robot peut être considéré comme un système discret recevant une nouvelle commande au début de chaque période d'échantillonnage T_e. Notons qu'entre deux instants t_{k-1} et t_k, séparés par une période d'échantillonnage T_e, la commande $\dot{q}(t_{k-1})$ appliquée est constante. Sous cette condition il est possible d'obtenir les équations d'évolution d'un point P ainsi que de sa profondeur dans le repère caméra. Pour cela, il suffit d'intégrer les équations (3.5), (3.6) et (3.9) entre les instants t_{k-1} et t_k. Le lecteur intéressé est invité à se reporter à [Folio 2007] pour les détails de calculs. Nous obtenons les résultats suivants qui peuvent se scinder en quatre cas distincts [Folio 2007] :

- **Cas n° 1 : si** $\omega(k-1) \neq 0$**,** $\varpi(k-1) \neq 0$ **et** $\omega(k-1) \neq \varpi(k-1)$:

$$X_P(k) = \frac{z_p(k-1)X_P(k-1)}{z_P(k)} \tag{3.10}$$

$$Y_P(k) = \frac{f}{z_P(k)}\left(c_1\cos(A_1) - c_2\sin(A_1) + D_x\sin(\vartheta(k)) + \frac{\upsilon(k-1)}{\omega(k-1)}\cos(\vartheta(k)) - C_y\right) \tag{3.11}$$

$$z_P(k) = c_1\sin(A_1) + c_2\cos(A_1) - D_x\cos(\vartheta(k)) + \frac{\upsilon(k-1)}{\omega(k-1)}\sin(\vartheta(k)) - C_x \tag{3.12}$$

Avec :

$$A_1 = (\omega(k-1) + \varpi(k-1)) * T_e$$

$$c_1 = \frac{Y_P(k-1)z_p(k-1)}{f} - D_x\sin(\vartheta(k-1)) - \frac{\upsilon(k-1)}{\omega(k-1)}\cos(\vartheta(k-1)) + C_y$$

$$c_2 = z_p(k-1) + D_x\cos(\vartheta(k-1)) - \frac{\upsilon(k-1)}{\omega(k-1)}\sin(\vartheta(k-1)) + C_x$$

- **Cas n° 2 : si** $\omega(k-1)=0$ **et** $\varpi(k-1)\neq 0$:

$$X_P(k)=\frac{z_p(k-1)X_P(k-1)}{z_P(k)} \tag{3.13}$$

$$Y_P(k)=\frac{f}{z_P(k)}\left(c_1\cos(A_2)-c_2\sin(A_2)+\upsilon(k-1)T_e\sin(\vartheta(k))+\frac{\upsilon(k-1)}{2\varpi(k-1)}\cos(\vartheta(k))-C_y\right) \tag{3.14}$$

$$z_P(k)=c_1\sin(A_2)+c_2\cos(A_2)-\upsilon(k-1)T_e\cos(\vartheta(k))+\frac{\upsilon(k-1)}{2\varpi(k-1)}\sin(\vartheta(k))-C_x \tag{3.15}$$

Avec

$$A_2=\varpi(k-1)T_e$$

$$c_1=\frac{Y_P(k-1)z_p(k-1)}{f}-\frac{\upsilon(k-1)}{2\varpi(k-1)}\cos(\vartheta(k-1))+C_y$$

$$c_2=z_p(k-1)-\frac{\upsilon(k-1)}{2\varpi(k-1)}\sin(\vartheta(k-1))+C_x$$

- **Cas n° 3 : si** $\omega(k-1)=-\varpi(k-1)$:

$$X_P(k)=\frac{z_p(k-1)X_P(k-1)}{z_P(k)} \tag{3.16}$$

$$Y_P(k)=\frac{-f\upsilon(k-1)c_1-f\omega(k-1)D_xc_2+\varpi(k-1)z_P(k-1)Y_P(k-1)}{\varpi(k-1)z_P(k)} \tag{3.17}$$

$$z_P(k)=-\frac{\upsilon(k-1)}{\varpi(k-1)}c_2-D_xc_1+z_P(k-1) \tag{3.18}$$

avec

$$c_1=\cos(\vartheta(k))-\cos(\vartheta(k-1)) \quad c_2=\sin(\vartheta(k))-\sin(\vartheta(k-1))$$

- **Cas n° 4 : si** $\omega(k-1)=0$ **et** $\varpi(k-1)=0$:

$$X_P(k)=\frac{z_p(k-1)X_P(k-1)}{z_P(k)} \tag{3.19}$$

$$Y_P(k)=\frac{f\upsilon(k-1)\sin(\vartheta(k-1))T_e+z_P(k-1)Y_P(k-1)}{z_P(k)} \tag{3.20}$$

$$z_P(k)=-\upsilon(k-1)\cos(\vartheta(k-1))T_e+z_P(k-1) \tag{3.21}$$

Nous pouvons remarquer que les équations sont récursives dans les quatre cas. Afin de calculer les indices visuels $X_P(k)$, $Y_P(k)$ et $z_P(k)$, il est nécessaire de fournir les valeurs initiales $X_P(k-1)$, $Y_P(k-1)$ et $z_P(k-1)$.

3.1.2 Utilisation des équations d'évolution des indices visuels pour la gestion des occultations

Afin d'utiliser les équations d'évolution des indices visuels pour calculer les prédictions $\hat{X}_P$ et $\hat{Y}_P$, il est nécessaire de les intégrer dans le schéma de commande. Comme nous pouvons le voir dans la figure 3.3, des valeurs initiales doivent être fournies à chaque itération. Deux cas se présentent : la première itération pour laquelle a lieu l'occultation et les m itérations suivantes, si l'occultation dure $m+1$ itérations. Dans le premier cas la dernière image reçue et traitée est utilisée pour fournir les valeurs initiales $X_P(I_{Occ}-1)$ et $Y_P(I_{Occ}-1)$, où I_{Occ} correspond à l'itération à laquelle se produit l'occultation. Durant les m itérations suivantes les valeurs prédites $\hat{X}_P(I_{Occ}+n_{it}-1)$ et $\hat{Y}_P(I_{Occ}+n_{it}-1)$ sont utilisées comme valeurs initiales pour le calcul de $\hat{X}_P(I_{Occ}+n_{it})$ et $\hat{Y}_P(I_{Occ}+n_{it})$, pour $n_{it} \in [1, m]$.

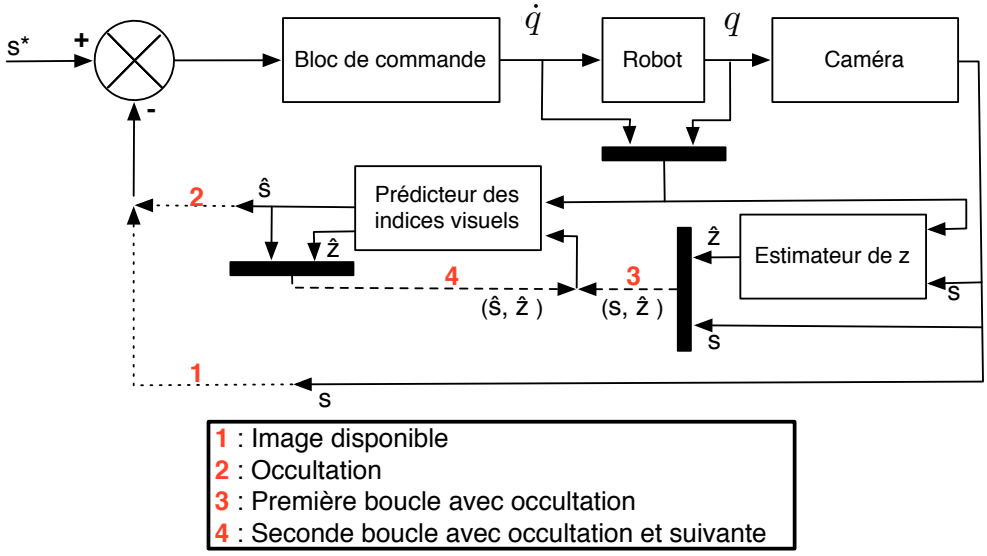


FIGURE 3.3 – Schéma de gestion des occultations

Le calcul de $\hat{X}_P(I_{Occ}+n_{it})$ et $\hat{Y}_P(I_{Occ}+n_{it})$ nécessite de plus la connaissance de $z_P(I_{Occ}+n_{it}-1)$. A nouveau ce problème se divise comme précédemment. Lors de la première itération il faut fournir une valeur de $z_P(I_{Occ}-1)$. Durant les m itérations suivantes, la valeur de $\hat{z}_P(I_{Occ}+n_{it}-1)$ fournie par les équations d'évolution des indices visuels est utilisée pour calculer $\hat{z}_P(I_{Occ}+n_{it})$. Notre contribution se situera sur la première étape mentionnée ci-dessus.

A cet instant, seule la méconnaissance de z_P lors de la première itération nous empêche de gérer les occultations à l'aide des équations d'évolution des indices visuels. Il est donc nécessaire de chercher à estimer ce paramètre lorsque les indices visuels sont disponibles. Avant de présenter les différentes techniques permettant de fournir une valeur de z_P, nous décrivons la série de contraintes que nous désirons respecter dans le cadre d'une navigation par asservissement visuel. Premièrement le temps de calcul consacré à l'obtention de z_P doit être le plus petit possible afin de ne pas augmenter la période d'échantillonnage T_e. En effet, le contrôle d'un système discret doit se faire avec la période d'échantillonnage la plus petite possible afin de ne pas compromettre la convergence du système. Deuxièmement la méthode doit converger vers une valeur proche de z_P le plus vite possible, afin de permettre la gestion d'occultation dans les meilleures conditions. En effet, la réussite de la navigation peut être remise en cause si le processus de gestion d'occultation est amorcé alors que la profondeur courante est encore mal

connue. Finalement, afin d'augmenter l'autonomie du robot, le modèle et les dimensions de la cible sont supposés inconnus.

Nous nous focalisons par la suite sur la détermination de z_P. Dans un premier temps, une approche matérielle, telle qu'un laser, peut être envisagée. Dans ce cas, deux précautions doivent être prises : s'assurer que la cible est bien dans le plan du laser et être capable d'identifier la donnée laser donnant la distance entre le robot et le point de la cible correspondant à l'indice visuel. De plus, le laser ne fournit pas directement z_P, coordonnée selon l'axe z du repère caméra, mais la distance robot/cible. Cette solution semble donc difficile à mettre en œuvre.

Afin de relâcher la contrainte concernant la présence de la cible dans le plan laser, nous nous intéressons à des méthodes utilisant les images afin d'estimer z_P. Ces méthodes peuvent se diviser en deux catégories : celles issues de la communauté vision et celles issues de la communauté automatique.

La stéréovision, solution à la fois matérielle et logicielle, permet de fournir une mesure pertinente de z_P dans le cas où le banc stéréo est correctement calibré. Bien qu'une architecture matérielle dédiée [Gibson & Marques 2008] puisse permettre une utilisation temps réel, le temps de calcul reste généralement trop élevé pour être compatible avec notre période d'échantillonnage. Les techniques dites de *Structure From Motion* [Jerian & Jain 1991], [Chaumette *et al.* 1996], [Chaumette *et al.* 1994], [Soatto & Perona 1998a, Soatto & Perona 1998b] et de *Bundle adjustment* [Triggs *et al.* 2000], nécessitent elles aussi un temps de calcul important, et sont basées sur des images obtenues pour des points de vue différents et la connaissance du mouvement entre ces images. L'estimation de la profondeur étant faite lors de la navigation, les images sont prises à des instants séparés seulement d'une période d'échantillonnage. Les images prises sont alors très semblables et n'offrent pas suffisamment d'informations pour que ces méthodes soient efficaces.

Concentrons-nous maintenant sur les méthodes issues de la communauté automatique. Dans [Bellot 2002], l'estimation de pose de la cible par rapport au robot est vue comme le dual du problème de commande. Cette méthode ne semble pas convenir car elle nécessite une connaissance *a priori* des dimensions de la cible. D'autres solutions proposent de considérer z_P comme l'un des états non mesurable du système. Une approche stochastique consiste à estimer z_P à l'aide d'un filtre de Kalman [Matthies *et al.* 1989]. Néanmoins cette méthode interdisant certains mouvements de la caméra, tels que les rotations, elle ne peut être utilisée dans le cadre de notre navigation basée sur l'asservissement visuel. Pour les approches déterministes, des observateurs sont mis en place afin de reconstruire z_P. Dans [De Luca *et al.* 2008], l'état du système est défini comme étant $[X_P, Y_P, z_P]$ et un observateur non linéaire non minimal est mis en place. De la même manière [Morbidi & Prattichizzo 2009] propose de résoudre ce problème avec un observateur non linéaire minimal. Nous présentons dans la section 3.3 ces deux méthodes de manière plus détaillée ainsi qu'une étude comparative avec notre propre technique exposée dans 3.2.3. Nous verrons à la suite de cette étude que les observateurs développés dans [De Luca *et al.* 2008] et [Morbidi & Prattichizzo 2009] sont difficilement réglables et ne proposent pas un temps de convergence suffisamment court pour répondre à nos attentes. Mais revenons tout d'abord à notre approche.

3.2 Le prédicteur/correcteur

Afin d'estimer z_P nous proposons d'utiliser les équations d'évolution des indices visuels pour mettre en place un estimateur basé sur une paire prédicteur/correcteur. Dans un premier temps, l'estimateur n'utilisera que deux images successives, puis dans un second temps, afin d'améliorer ses performances, un plus grand nombre d'images noté n sera exploité.

3.2.1 Le prédicteur/correcteur avec deux images

Dans cette section nous présentons la paire prédicteur/correcteur permettant d'estimer $z_P(k)$ en fonction des données $\hat{X}_P(k-1)$, $\hat{Y}_P(k-1)$, $\hat{X}_P(k)$ et $\hat{Y}_P(k)$ extraites des images durant les itérations $k-1$ et k. La démarche est expliquée pour le cas n° 1. Pour les autres, seuls les résultats essentiels sont donnés, le raisonnement suivi restant identique.

- **Cas n° 1 : si** $\omega(k-1) \neq 0$**,** $\varpi(k-1) \neq 0$ **et** $\omega(k-1) \neq \varpi(k-1)$:
  Dans un premier temps, l'équation (3.12) est réécrite pour obtenir une expression de $z_P(k-1)$ en fonction de $z_P(k)$:

$$z_P(k-1) = \frac{z_P(k) - \beta_1}{\alpha_1} \tag{3.22}$$

avec

$$\alpha_1 = \frac{Y_P(k-1)}{f}\sin(A_1) + \cos(A_1)$$

et

$$\begin{aligned}\beta_1 = (-D_x \sin(\vartheta(k-1)) - \tfrac{v(k-1)}{\omega(k-1)}\cos(\vartheta(k-1)) + C_y)\sin(A_1)\\ +(D_x \cos(\vartheta(k-1)) - \tfrac{v(k-1)}{\omega(k-1)}\sin(\vartheta(k-1)) + C_x)\cos(A_1)\\ -D_x\cos(\vartheta(k)) + \tfrac{v(k-1)}{\omega(k-1)}\sin(\vartheta(k)) - C_x\end{aligned}$$

L'équation (3.22) est injectée dans les équations (3.10) et (3.11), pour obtenir des expressions de $X_P(k)$ et de $Y_P(k)$ fonctions de $z_P(k)$:

$$X_P(k) = \frac{z_P(k)X_P(k-1) - \beta_1 X_P(k-1)}{\alpha_1 z_P(k)} \tag{3.23}$$

$$Y_P(k) = \frac{f}{z_P(k)}\left(\frac{Y_P(k-1)z_P(k)}{\alpha_1 f}\cos(A_1) - \frac{z_P(k)}{\alpha_1}\sin(A_1) + \lambda_1\right) \tag{3.24}$$

avec

$$\begin{aligned}\lambda_1 = \left(\tfrac{-\beta_1 Y_P(k-1)}{\alpha f} - D_x\sin(\vartheta(k-1)) - \tfrac{v(k-1)}{\omega(k-1)}\cos(\vartheta(k-1)) + C_y\right)\cos(A_1)\\ -\left(\tfrac{-\beta_1}{\alpha_1} + D_x\cos(\vartheta(k-1)) - \tfrac{v(k-1)}{\omega(k-1)}\sin(\vartheta(k-1)) + C_x\right)\sin(A_1)\\ +D_x\sin(\vartheta(k)) + \tfrac{v(k-1)}{\omega(k-1)}\cos(\vartheta(k)) - C_y\end{aligned}$$

Les expressions des prédicteurs $\hat{X}_P(k|k-1)$ et $\hat{Y}_P(k|k-1)$ dépendent des mesures $\tilde{X}_P(k-1)$ et $\tilde{Y}_P(k-1)$. Elles sont obtenues à partir des équations (3.23) et (3.24) :

$$\hat{X}_P(k|k-1) = \frac{\hat{z}_P(k|k-1)\tilde{X}_P(k-1) - \beta_1\tilde{X}_P(k-1)}{\tilde{\alpha}_1\hat{z}_P(k|k-1)} \tag{3.25}$$

$$\hat{Y}_P(k|k-1) = \frac{f}{\hat{z}_P(k|k-1)} \left(\frac{\tilde{Y}_P(k-1)\hat{z}_P(k|k-1)}{\tilde{\alpha}_1 f} \cos(A_1) - \frac{\hat{z}_P(k|k-1)}{\tilde{\alpha}_1} \sin(A_1) + \tilde{\lambda_1} \right) \tag{3.26}$$

avec

$$\tilde{\alpha}_1 = \frac{\tilde{Y}_P(k-1)}{f} \sin(A_1) + \cos(A_1)$$

et

$$\begin{aligned} \tilde{\lambda}_1 = &\left(\tfrac{-\beta_1 \tilde{Y}_P(k-1)}{\tilde{\alpha}_1 f} - D_x \sin(\vartheta(k-1)) - \tfrac{\upsilon(k-1)}{\omega(k-1)} \cos(\vartheta(k-1)) + C_y \right) \cos(A_1) \\ &- \left(\tfrac{-\beta_1}{\tilde{\alpha}_1} + D_x \cos(\vartheta(k-1)) - \tfrac{\upsilon(k-1)}{\omega(k-1)} \sin(\vartheta(k-1)) + C_x \right) \sin(A_1) \\ &+ D_x \sin(\vartheta(k)) + \tfrac{\upsilon(k-1)}{\omega(k-1)} \cos(\vartheta(k)) - C_y \end{aligned}$$

A l'aide des équations des prédicteurs, nous cherchons à obtenir une estimation de la profondeur. Dans cette optique, nous définissons le critère suivant :

$$C = (\hat{X}_P(k|k-1) - \tilde{X}_P(k))^2 + (\hat{Y}_P(k|k-1) - \tilde{Y}_P(k))^2 \tag{3.27}$$

Le critère (3.27) permet de comparer les prédicteurs basés sur les mesures à l'instant $k-1$ avec les mesures de l'instant k. La seule inconnue étant $\hat{z}_P(k|k-1)$, minimiser le critère revient à chercher une valeur de $\hat{z}_P(k|k-1)$ pour laquelle $\frac{\partial C}{\partial \hat{z}_P(k|k-1)} = 0$. La dérivée de C par rapport à $\hat{z}_P(k|k-1)$ est donnée par l'expression (3.28) :

$$\frac{\partial C}{\partial \hat{z}_P(k|k-1)} = 2(\hat{X}_P(k|k-1) - \tilde{X}_P(k)) \frac{\partial \hat{X}_P(k|k-1)}{\partial \hat{z}_P(k|k-1)} + 2(\hat{Y}_P(k|k-1) - \tilde{Y}_P(k)) \frac{\partial \hat{Y}_P(k|k-1)}{\partial \hat{z}_P(k|k-1)} \tag{3.28}$$

avec

$$\frac{\partial \hat{X}_P(k|k-1)}{\partial \hat{z}_P(k|k-1)} = \frac{\tilde{X}_P(k-1)\beta_1}{\hat{z}_P^2(k|k-1)\tilde{\alpha}_1} \qquad \frac{\partial \hat{Y}_P(k|k-1)}{\partial \hat{z}_P(k|k-1)} = \frac{-f\tilde{\lambda}_1}{\hat{z}_P^2(k|k-1)}$$

A partir des équations (3.25), (3.26) et (3.28), l'estimateur $\hat{z}_P(k|k)$ s'écrit :

$$\hat{z}_P(k|k) = \frac{\tilde{X}_P^2(k-1)\beta_1^2 + f^2\tilde{\alpha}_1^2\tilde{\lambda}_1^2}{den_z} \tag{3.29}$$

avec

$$\begin{aligned} den_z = &\tilde{X}_P(k-1)\beta_1 \left(\tilde{X}_P(k-1) - \tilde{X}_P(k)\tilde{\alpha}_1 \right) \\ &- \left(\tilde{Y}_P(k-1)\cos(A_1) - f\sin(A_1) \right) f\tilde{\alpha}_1\tilde{\lambda_1} + \tilde{Y}_P(k) f\tilde{\alpha}_1^2\tilde{\lambda}_1 \end{aligned}$$

- **Cas n° 2 : si** $\omega(k-1) = 0$ **et** $\varpi(k-1) \neq 0$:

Nous obtenons les expressions des prédicteurs $\hat{X}_P(k|k-1)$ et $\hat{Y}_P(k|k-1)$ à l'aide des équations (3.13), (3.14) et (3.15) :

$$\hat{X}_P(k|k-1) = \frac{\hat{z}_P(k|k-1)\tilde{X}_P(k-1) - \beta_2\tilde{X}_P(k-1)}{\tilde{\alpha_2}\hat{z}_P(k|k-1)} \tag{3.30}$$

$$\hat{Y}_P(k|k-1) = \frac{f}{\hat{z}_P(k|k-1)}\left(\frac{\tilde{Y}_P(k-1)\hat{z}_P(k|k-1)}{\tilde{\alpha_2}f}\cos(A_2) - \frac{\hat{z}_P(k|k-1)}{\tilde{\alpha_2}}\sin(A_2) + \tilde{\lambda_2}\right) \tag{3.31}$$

avec

$$\tilde{\alpha_2} = \frac{\tilde{Y}_P(k-1)}{f}\sin(A_2) + \cos(A_2)$$

$$\beta_2 = (-\tfrac{\upsilon(k-1)}{2\varpi(k-1)}\cos(\vartheta(k-1)) + C_y)\sin(A_2)$$
$$+(-\tfrac{\upsilon(k-1)}{2\varpi(k-1)}\sin(\vartheta(k-1)) + C_x)\cos(A_2) - \upsilon(k-1)T_e + \tfrac{\upsilon(k-1)}{2\varpi(k-1)}\sin(\vartheta(k)) - C_x$$

et

$$\tilde{\lambda_2} = \Big(\tfrac{-\beta_2\tilde{Y}_P(k-1)}{\alpha_2 f} - \tfrac{\upsilon(k-1)}{2\varpi(k-1)}\cos(\vartheta(k-1)) + C_y\Big)\cos(A_2)$$
$$-\Big(\tfrac{-\beta_2}{\alpha_2} - \tfrac{\upsilon(k-1)}{2\varpi(k-1)}\sin(\vartheta(k-1)) + C_x\Big)\sin(A_2) + \upsilon(k-1)T_e + \tfrac{\upsilon(k-1)}{2\varpi(k-1)}\cos(\vartheta(k)) - C_y$$

Les dérivées des équations (3.30) et (3.31) par rapport à $\hat{z}_P(k|k-1)$ s'écrivent comme suit :

$$\frac{\partial\hat{X}_P(k|k-1)}{\partial\hat{z}_P(k|k-1)} = \frac{\tilde{X}_P(k-1)\beta_2}{\hat{z}_P^2(k|k-1)\tilde{\alpha_2}} \tag{3.32}$$

$$\frac{\partial\hat{Y}_P(k|k-1)}{\partial\hat{z}_P(k|k-1)} = \frac{-f\tilde{\lambda_2}}{\hat{z}_P^2(k|k-1)} \tag{3.33}$$

Finalement à l'aide du critère (3.27) ainsi que des équations (3.30), (3.31), (3.32) et (3.33), $\hat{z}_P(k|k)$ s'exprime de la manière suivante :

$$\hat{z}_P(k|k) = \frac{\tilde{X}_P^2(k-1)\beta_2^2 + f^2\tilde{\alpha_2}^2\tilde{\lambda_2}^2}{den_z} \tag{3.34}$$

avec

$$den_z = \tilde{X}_P(k-1)\beta_2\Big(\tilde{X}_P(k-1) - \tilde{X}_P(k)\tilde{\alpha_2}\Big)$$
$$-\Big(\tilde{Y}_P(k-1)\cos(A_2) - f\sin(A_2)\Big)f\tilde{\alpha_2}\tilde{\lambda_2} + \tilde{Y}_P(k)f\tilde{\alpha_2}^2\tilde{\lambda_2}$$

- **Cas n° 3 : si** $\omega(k-1) = -\varpi(k-1)$:

Afin de déterminer les prédicteurs $\hat{X}_P(k|k-1)$ et $\hat{Y}_P(k|k-1)$, nous exploitons les équations (3.16), (3.17) et (3.18). Nous pouvons montrer que :

$$\hat{X}_P(k|k-1) = \frac{\hat{z}_P(k|k-1)\tilde{X}_P(k-1) - \beta_3\tilde{X}_P(k-1)}{\hat{z}_P(k|k-1)} \tag{3.35}$$

$$\hat{Y}_P(k|k-1) = \tilde{Y}_P(k-1) + \frac{\tilde{\lambda_3}}{\hat{z}_P(k|k-1)} \tag{3.36}$$

avec

$$\beta_3 = -\frac{\upsilon(k-1)}{\varpi(k-1)}(\sin(\vartheta(k)) - \sin(\vartheta(k-1)) - D_x(\cos(\vartheta(k)) - \cos(\vartheta(k-1))$$

et

$$\tilde{\lambda_3} = -f\frac{\upsilon(k-1)}{\varpi(k-1)}\Big(\cos(\vartheta(k))-\cos(\vartheta(k-1)\Big)+fD_x\Big(\sin(\vartheta(k))-\sin(\vartheta(k-1)\Big)-\tilde{Y}_P(k-1)\beta_3$$

En dérivant les équations (3.35) et (3.36) par rapport à $\hat{z}_P(k|k-1)$, il vient :

$$\frac{\partial\hat{X}_P(k|k-1)}{\partial\hat{z}_P(k|k-1)} = \frac{\tilde{X}_P(k-1)\beta_3}{\hat{z}_P^2(k|k-1)} \tag{3.37}$$

$$\frac{\partial\hat{Y}_P(k|k-1)}{\partial\hat{z}_P(k|k-1)} = \frac{-\tilde{\lambda_3}}{\hat{z}_P^2(k|k-1)} \tag{3.38}$$

Finalement, à partir du critère (3.27) et des équations (3.35), (3.36), (3.37) et (3.38), $\hat{z}_P(k|k)$ s'écrit :

$$\hat{z}_P(k|k) = \frac{\tilde{X}_P^2(k-1)\beta_3^2 + \tilde{\lambda_3}^2}{\tilde{X}_P(k-1)\beta_3\Big(\tilde{X}_P(k-1)-\tilde{X}_P(k)\Big)-\Big(\tilde{Y}_P(k-1)-\tilde{Y}_P(k)\Big)\tilde{\lambda_3}} \tag{3.39}$$

- **Cas n^o 4 : si** $\omega(k-1)=0$ **et** $\varpi(k-1)=0$:

Nous utilisons ici les équations (3.19), (3.20) et (3.21). A l'aide de ces équations, nous déduisons les expressions des prédicteurs $\hat{X}_P(k|k-1)$ et $\hat{Y}_P(k|k-1)$:

$$\hat{X}_P(k|k-1) = \frac{\hat{z}_P(k|k-1)\tilde{X}_P(k-1)-\beta_4\tilde{X}_P(k-1)}{\hat{z}_P(k|k-1)} \tag{3.40}$$

$$\hat{Y}_P(k|k-1) = \tilde{Y}_P(k-1)+\frac{\tilde{\lambda_4}}{\hat{z}_P(k|k-1)} \tag{3.41}$$

avec

$$\beta_4 = -\upsilon(k-1)\cos(\vartheta(k-1))T_e$$

et

$$\tilde{\lambda_4} = f\upsilon(k-1)\sin(\vartheta(k-1))-\tilde{Y}_P(k-1)\beta_4$$

En dérivant ces équations par rapport à $\hat{z}_P(k|k-1)$, nous avons :

$$\frac{\partial\hat{X}_P(k|k-1)}{\partial\hat{z}_P(k|k-1)} = \frac{-\tilde{X}_P(k-1)\beta_4}{\hat{z}_P^2(k|k-1)} \tag{3.42}$$

$$\frac{\partial\hat{Y}_P(k|k-1)}{\partial\hat{z}_P(k|k-1)} = \frac{-\tilde{\lambda_4}}{\hat{z}_P^2(k|k-1)} \tag{3.43}$$

Enfin, en utilisant le critère (3.27) ainsi que les équations (3.40), (3.41), (3.42) et (3.43), nous pouvons montrer que $\hat{z}_P(k|k)$ s'exprime comme suit :

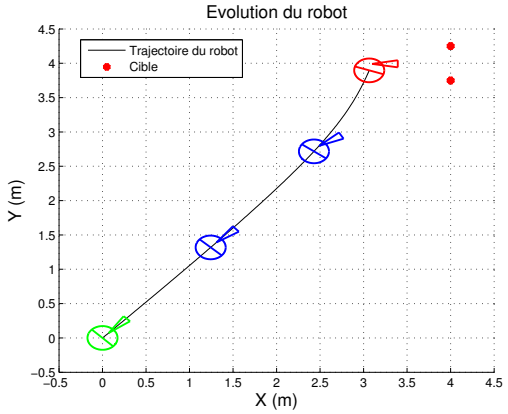
$$\hat{z}_P(k|k) = \frac{-\tilde{X}_P^2(k-1)\beta_4^2 + \tilde{\lambda_4}^2}{\tilde{X}_P(k-1)\beta_4\Big(\tilde{X}_P(k-1)-\tilde{X}_P(k)\Big)+\Big(\tilde{Y}_P(k-1)-\tilde{Y}_P(k)\Big)\tilde{\lambda_4}} \tag{3.44}$$

Nous avons ainsi obtenu des estimateurs de z_P correspondant à chacun des cas exhibés dans [Folio 2007]. Il reste maintenant à évaluer leurs efficacité.

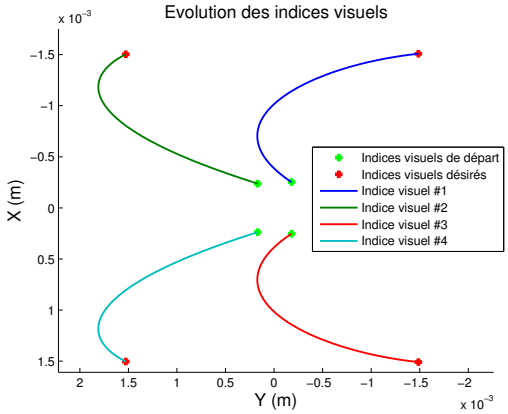
Simulation

Nous proposons de valider les différents estimateurs obtenus (cf. (3.29), (3.34), (3.39) et (3.44)) lors d'une simulation de navigation par asservissement visuel 2D. Les paramètres de simulations sont identiques à ceux précédemment présentés. La scène ainsi que l'évolution du système robotique tout au long de la navigation, sont présentées sur la figure 3.4(a).

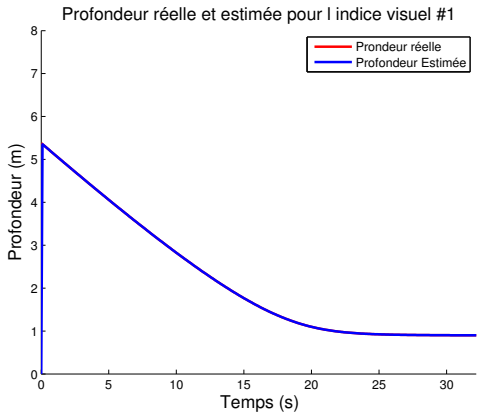
La projection des points de la cible sur le plan image ainsi que le traitement d'image sont parfaitement réalisés comme le montre l'évolution des indices visuels présentée dans la figure 3.4(b). Cela signifie qu'aucun bruit de mesure n'apparait lors de la simulation et nous place donc dans un cas parfait. De la même manière les commandes calculées sont immédiatement et parfaitement appliquées. Dans ces conditions de simulation, les équations (3.29), (3.34), (3.39) et (3.44) permettent de parfaitement reconstruire la profondeur puisque l'estimation $\hat{z}_{P_1}(k|k)$ et la valeur réelle de z_{P_1} sont parfaitement confondues sur la figure 3.4 [3].



(a) Trajectoire du robot lors d'un asservissement visuel



(b) Indices visuels



(c) Profondeur réelle et estimée

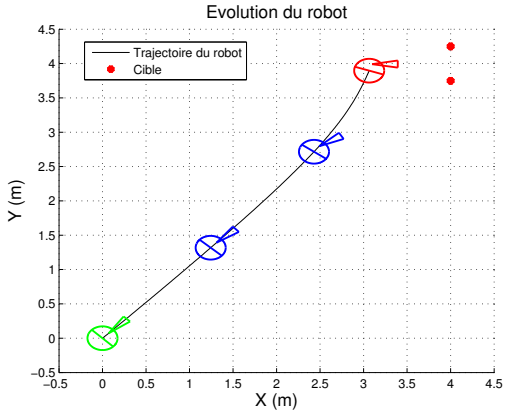
FIGURE 3.4 – Simulation avec des données non bruitées

Dans un second temps, la simulation a été rejouée en introduisant des bruits de mesure dans l'image ainsi que des perturbations au niveau de la commande afin de se rapprocher des condi-

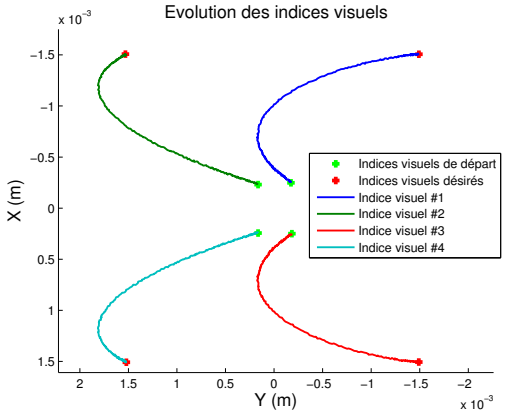
3. La première valeur de $\hat{z}_{P_1}(k|k)$ est fixée à 0 par défaut étant donné que le processus d'estimation nécessite deux images qui ne sont pas disponibles à l'instant initial.

tions expérimentales. Ainsi, une erreur de plus ou moins un pixel est aléatoirement introduite dans les coordonnées des points extraits de l'image. De même, la commande appliquée au robot correspond à la commande calculée biaisée par une erreur de plus ou moins 3%. La trajectoire (figure 3.5(a)) réalisée à l'aide des indices visuels bruités 3.5(b) est très semblable à celle présentée dans la figure 3.4(a). Cependant la figure 3.5 montre que les équations (3.29), (3.34), (3.39) et (3.44) donnent une estimation erronée de la profondeur. Ce résultat est dû à un rapport signal/bruit trop faible. En effet, les images utilisées pour l'estimation sont séparées par un seul intervalle de temps T_e et sont donc très semblables. Le bruit vient noyer l'information qui peut être extraite de la différence entre deux images successives.

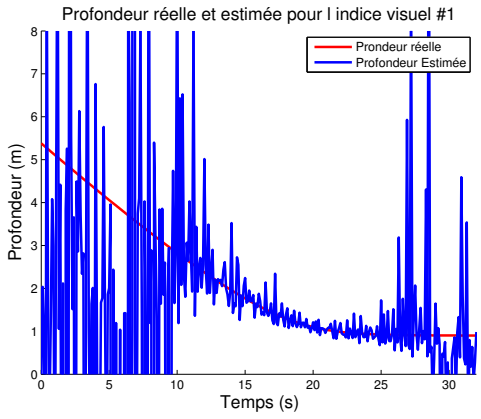
Afin d'analyser ce résultat, plaçons-nous dans une configuration simple : le robot et la caméra sont initialement face à la cible. Le système se rapproche de cette dernière de manière linéaire durant un intervalle de temps T_e. Dans le plan image les coordonnées absolues des indices visuels doivent augmenter tandis que les mesures bruitées révèlent une diminution. L'estimation de $\hat{z}_{P_1}(k|k)$ permettant de relier un déplacement linéaire positif avec une diminution des coordonnées des indices visuels est une valeur négative. Il y a donc des incohérences dans l'estimation de la profondeur lorsque les mesures sont bruitées.



(a) Trajectoire du robot lors d'un asservissement visuel



(b) Indices visuels



(c) Profondeur réelle et estimée

FIGURE 3.5 – Simulation avec des données bruitées

3.2.2 Première approche pour le prédicteur/correcteur avec n images

Afin de pouvoir estimer la profondeur en contexte bruité, il est nécessaire d'augmenter le rapport signal/bruit dans les images. Une solution consiste à utiliser des images obtenues pour des configurations du système robotique significativement différentes. Dans cette optique, nous définissons un nouveau critère :

$$C^* = \sum_{j=1}^{n} (\hat{X}_P(k|k-j) - \tilde{X}_P(k))^2 + (\hat{Y}_P(k|k-j) - \tilde{Y}_P(k))^2 \tag{3.45}$$

où n désigne le nombre d'images utilisées. Le critère (3.45) compare les mesures $\tilde{X}_P(k)$ et $\tilde{Y}_P(k)$ faites à l'itération k avec les prédicteurs $\hat{X}_P(k|k-j)$ et $\hat{Y}_P(k|k-j)$ construits à partir des mesures faites à l'itération $k-j$ ($j \in [1,n]$). Ainsi les mesures $\tilde{X}_P(k)$ et $\tilde{Y}_P(k)$ ne sont plus comparées avec un seul mais avec n prédicteurs. Il est ainsi possible d'utiliser n images $\tilde{X}_P(k-j)$ et $\tilde{Y}_P(k-j)$ non consécutives avec les mesures $\tilde{X}_P(k)$ et $\tilde{Y}_P(k)$. Les images étant maintenant très différentes, le rapport signal/bruit augmente significativement.

L'utilisation du critère (3.45) nécessite les expressions analytiques des prédicteurs $\hat{X}_P(k|k-j)$ et $\hat{Y}_P(k|k-j)$. Une première solution consiste à utiliser les équations d'évolution des indices visuels de manière récursive. Pour la coordonnée X par exemple, le processus peut être présenté de la façon suivante :

1. Ecrire $X(k-j+1) = f(X(k-j))$ où f est l'équation d'évolution de X correspondante (voir 3.1.1).
2. Ecrire $X(k-j+2) = f(X(k-j+1))$.
3. Remplacer dans l'expression précédente $X(k-j+1)$ par $f(X(k-j)$. On obtient ainsi $X(k-j+2) = f(f(X(k-j)))$.
4. Les opérations 1, 2 et 3 sont répétées jusqu'à obtenir une expression de $X(k)$ dépendant de $X(k-j)$.

Ainsi, cette approche nécessite de calculer les expressions analytiques des prédicteurs $\hat{X}_P(k|k-j)$ pour chaque valeur de j. Elle apparaît de fait rapidement fastidieuse et peu réalisable. Il faut donc chercher à réduire le nombre d'itérations permettant de relier $X(k-j)$ à $X(k)$. Pour cela, nous proposons alors de chercher la plus petite séquence de commandes permettant de relier deux images. Rappelons dans un premier temps que l'état du robot est défini comme suit :

$$\chi(k) = [X_M(k), Y_M(k), \theta(k), \vartheta(k)]^T \tag{3.46}$$

Etant donné qu'un état $\chi(k)$ (3.46) correspond à une image $s(k)$, et donc à $X(k)$ et $Y(k)$, relier $\chi(k-j)$ à $\chi(k)$ est équivalent à relier $s(k-j)$ à $s(k)$. Nous allons donc chercher la plus petite séquence de commandes permettant de relier deux états du robot. Cette séquence de commandes sera par la suite appelée "commandes équivalentes" ou "vitesses équivalentes".

Afin d'illustrer ce concept, deux exemples sont présentés dans la figure 3.6. Tout d'abord une trajectoire [4] (verte) de référence est réalisée en appliquant successivement les commandes $\dot{q}_k$, $\dot{q}_{k+1}$ et $\dot{q}_{k+2}$ permettant d'atteindre l'état $\chi(k+3)$ à partir de l'état $\chi(k)$. Dans le premier

4. Les trajectoires proposées ici n'ont qu'une valeur illustrative.

exemple, le robot peut être commandé à partir d'une seule commande équivalente $\dot{q}_{e01}$ et la trajectoire possible associée est représentée en rouge. Dans le second exemple, il faut deux commandes équivalentes $\dot{q}_{e11}$ et $\dot{q}_{e12}$ pour relier $\chi(k)$ à $\chi(k+3)$ et la trajectoire possible est représentée en jaune. Dans les deux exemples, il est supposé qu'il existe une série minimale de commandes équivalentes à la séquence $\dot{q}_k$, $\dot{q}_{k+1}$ et $\dot{q}_{k+2}$ qui garantit de relier deux états du système robotique, sans refaire la même trajectoire. Ces commandes équivalentes, si elles existent et peuvent être calculées, nous permettraient d'obtenir une expression de $X(k)$ fonction de $X(k-j)$ avec un nombre d'itérations réduit.

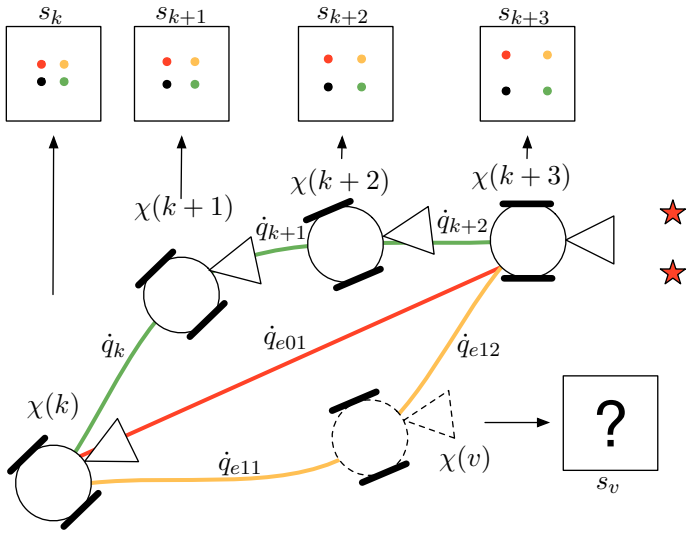


FIGURE 3.6 – Présentation des vitesses équivalentes

3.2.2.1 Etude de la commandabilité de la base mobile

Afin de déterminer le nombre de commandes nécessaires pour atteindre l'état $\chi(k)$ à partir d'un état $\chi(k-j)$, nous allons étudier la commandabilité du système non linéaire discret constitué par le robot. Pour cela, nous rappelons que le modèle cinématique associé à notre robot non-holonomme est donné par :

$$\begin{cases} \dot{X}_M(t) = \upsilon(t)\cos(\theta(t)) \\ \dot{Y}_M(t) = \upsilon(t)\sin(\theta(t)) \\ \dot{\theta}(t) = \omega(t) \\ \dot{\vartheta}(t) = \varpi(t) \end{cases} \tag{3.47}$$

Afin d'obtenir les équations d'évolution du système discret $\chi(k+1) = f(\chi(k), \dot{q}(k))$, nous intégrons le système (3.47) sur l'intervalle $[t_k, t_{k+1}]$. Les commandes appliquées au robot étant constantes durant cet intervalle, l'intégration du système est réalisable analytiquement. Il vient donc :

$$X_M(k+1) = X_M(k) + \frac{\upsilon(k)}{\omega(k)}(\sin(\theta(k)+\omega(k)*T_e) - \sin(\theta(k))) \quad (3.48)$$

$$Y_M(k+1) = Y_M(k) - \frac{\upsilon(k)}{\omega(k)}(\cos(\theta(k)+\omega(k)*T_e) - \cos(\theta(k))) \quad (3.49)$$

$$\theta(k+1) = \theta(k) + \omega(k)*T_e \quad (3.50)$$

$$\vartheta(k+1) = \vartheta(k) + \varpi(k)*T_e \quad (3.51)$$

Comme nous pouvons le voir dans l'équation (3.51), seul ϑ peut être commandé indépendament. Il n'est donc pas nécessaire d'étudier la commandabilité du système dans son ensemble, mais simplement celle de la base mobile. On considère alors un état réduit noté $\chi_r(k) = [M_x(k), M_y(k), \theta(k)]^T$. Les équations d'évolution du système discret réduit, notées $f_r(\chi_r(k), \dot{q}_r(k))$, sont données par les relations (3.48), (3.49) et (3.50).

Il reste maintenant à étudier la commandabilité de ce système discret. Pour cela nous nous appuyons sur les travaux de Djeridane [Djeridane 2004]. Ces travaux permettent de montrer qu'un système non linéaire discret défini par un état de dimension n_{etat} peut être commandé en n_{com} coups si $rang(P_{acc}) = n_{etat}$, où P_{acc} est la matrice d'accessibilité définie comme suit :

$$P_{acc} = \begin{bmatrix} \frac{\partial f_r(\chi(n_{com}-1),\dot{q}(n_{com}-1))}{\partial \dot{q}(n_{com}-1)} & & & \\ \frac{\partial f_r(\chi(n_{com}-1),\dot{q}(n_{com}-1))}{\partial \chi(n_{com}-1)} & \frac{\partial f_r(\chi(n_{com}-2),\dot{q}(n_{com}-2))}{\partial \dot{q}(n_{com}-2)} & & \\ & \cdots & & \\ \frac{\partial f_r(\chi(n_{com}-1),\dot{q}(n_{com}-1))}{\partial \chi(n_{com}-1)} & \cdots & \frac{\partial f_r(\chi(1),\dot{q}(1))}{\partial \chi(1)} & \frac{\partial f_r(\chi(0),\dot{q}(0))}{\partial \dot{q}(0)} \end{bmatrix}^T \quad (3.52)$$

Dans un premier temps, nous nous intéressons à l'étude de la commandabilité de la base mobile en un coup, c'est-à-dire avec $n_{com} = 1$ et $n_{etat} = 3$. La matrice P_1 correspondant à cette étude est la suivante :

$$P_1 = \frac{\partial f_r(\chi_r(0), \dot{q}_r(0))}{\partial \dot{q}_r(0)} \quad (3.53)$$

La matrice $\frac{\partial f_r(\chi_r(0),\dot{q}_r(0))}{\partial \dot{q}_r(0)}$ est de dimension (3,2). P_1 ne peut donc être au maximum que de rang 2, ce qui ne permet pas de remplir les conditions de commandabilité. La base mobile n'étant pas commandable en un coup, nous nous intéressons maintenant à la commandabilité en deux coups. Dans ce cas, la matrice P_2 correspondante est définie comme suit :

$$P_2 = \left[\frac{\partial f_r(\chi_r(1), \dot{q}_r(1))}{\partial \dot{q}_r(1)} \ \frac{\partial f_r(\chi_r(1), \dot{q}_r(1))}{\partial \chi_r(1)} * \frac{\partial f_r(\chi_r(0), \dot{q}_r(0))}{\partial \dot{q}_r(0)}\right] \quad (3.54)$$

où

$$\frac{\partial f_r(\chi_r(1),\dot{q}_r(1))}{\partial \dot{q}_r(1)} = \begin{bmatrix} \frac{2}{\omega(1)}\sin(\eta_1)\cos(\eta_2) & \Xi_1 \\ \frac{2}{\omega(1)}\sin(\eta_1)\sin(\eta_2) & \Xi_2 \\ 0 & T_e \end{bmatrix} \tag{3.55}$$

$$\frac{\partial f_r(\chi_r(1),\dot{q}_r(1))}{\partial \chi_r(1)} = \begin{bmatrix} 1 & 0 & -\frac{2\upsilon(1)}{\omega(1)}\sin(\eta_1)\sin(\eta_2) \\ 0 & 1 & \frac{2\upsilon(1)}{\omega(1)}\sin(\eta_1)\cos(\eta_2) \\ 0 & 0 & 1 \end{bmatrix} \tag{3.56}$$

$$\frac{\partial f_r(\chi_r(0),\dot{q}_r(0))}{\partial \dot{q}_r(0)} = \begin{bmatrix} \frac{2}{\omega(0)}\sin(\eta_3)\cos(\eta_4) & \Xi_3 \\ \frac{2}{\omega(0)}\sin(\eta_3)\sin(\eta_4) & \Xi_4 \\ 0 & T_e \end{bmatrix} \tag{3.57}$$

avec

$$\Xi_1 = -\frac{2\upsilon(1)}{\omega^2(1)}\sin(\eta_1)\cos(\eta_2) + \frac{\upsilon(1)}{\omega^2(1)}T_e\cos(\eta_1)\cos(\eta_2) - \frac{\upsilon(1)}{\omega^2(1)}T_e\sin(\eta_1)\sin(\eta_2)$$

$$\Xi_2 = -\frac{2\upsilon(1)}{\omega^2(1)}\sin(\eta_1)\sin(\eta_2) + \frac{\upsilon(1)}{\omega^2(1)}T_e\cos(\eta_1)\sin(\eta_2) + \frac{\upsilon(1)}{\omega^2(1)}T_e\sin(\eta_1)\cos(\eta_2)$$

$$\Xi_3 = -\frac{2\upsilon(0)}{\omega^2(0)}\sin(\eta_3)\cos(\eta_4) + \frac{\upsilon(0)}{\omega^2(0)}T_e\cos(\eta_3)\cos(\eta_4) - \frac{\upsilon(0)}{\omega^2(0)}T_e\sin(\eta_3)\sin(\eta_4)$$

$$\Xi_4 = -\frac{2\upsilon(0)}{\omega^2(0)}\sin(\eta_3)\sin(\eta_4) + \frac{\upsilon(0)}{\omega^2(0)}T_e\cos(\eta_3)\sin(\eta_4) + \frac{\upsilon(0)}{\omega^2(0)}T_e\sin(\eta_3)\cos(\eta_4)$$

$$\eta_1 = \frac{\omega(1)T_e}{2} \qquad \eta_2 = \frac{2\theta(1)+\omega(1)T_e}{2} \qquad \eta_3 = \frac{\omega(0)T_e}{2} \qquad \eta_4 = \frac{2\theta(0)+\omega(0)T_e}{2}$$

La matrice $\frac{\partial f_r(\chi_r(1),\dot{q}_r(1))}{\partial \dot{q}_r(1)}$ étant de dimension (3,2), la matrice P_2 est de dimension (3,4). P_2 est de rang plein (ici de rang 3) si l'un de ses mineurs est non nul. Considérons la sous-matrice SM_2 de dimension (3,3) suivante :

$$SM_2 = \begin{bmatrix} \frac{2}{\omega(1)}\sin(\eta_1)\cos(\eta_2) & \Xi_5 & \frac{2}{\omega(0)}\sin(\eta_3)\cos(\eta_4) \\ \frac{2}{\omega(1)}\sin(\eta_1)\sin(\eta_2) & \Xi_6 & \frac{2}{\omega(0)}\sin(\eta_3)\sin(\eta_4) \\ 0 & T_e & 0 \end{bmatrix} \tag{3.58}$$

avec

$$\Xi_5 = -\frac{2\upsilon(1)}{\omega^2(1)}\sin(\eta_1)\cos(\eta_2) + \frac{\upsilon(1)}{\omega^2(1)}T_e\cos(\eta_1)\cos(\eta_2) - \frac{\upsilon(1)}{\omega^2(1)}T_e\sin(\eta_1)\sin(\eta_2)$$

et

$$\Xi_6 = -\frac{2\upsilon(1)}{\omega^2(1)}\sin(\eta_1)\sin(\eta_2) + \frac{\upsilon(1)}{\omega^2(1)}T_e\cos(\eta_1)\sin(\eta_2) + \frac{\upsilon(1)}{\omega^2(1)}T_e\sin(\eta_1)\cos(\eta_2)$$

Le mineur s'écrit :

$$\begin{aligned} det(SM_2) &= \frac{2T_e}{\omega(1)}\sin(\eta_1)\sin(\eta_2)\frac{2}{\omega(0)}\sin(\eta_3)\cos(\eta_4) - \frac{2T_e}{\omega(1)}\sin(\eta_1)\cos(\eta_2)\frac{2}{\omega(0)}\sin(\eta_3)\sin(\eta_4) \\ &= \frac{4T_e}{\omega(0)\omega(1)}\sin(\eta_1)\sin(\eta_3)\left\{\sin(\eta_2)\cos(\eta_4) - \cos(\eta_2)\sin(\eta_4)\right\} \\ &= \frac{4T_e}{\omega(0)\omega(1)}\sin(\eta_1)\sin(\eta_3)\sin\left(\frac{(2\theta(1)+\omega(1)Te)-(2\theta(0)+\omega(0)Te)}{2}\right) \\ &= \frac{4T_e}{\omega(0)\omega(1)}\sin(\eta_1)\sin(\eta_3)\sin\left(\frac{(\omega(0)+\omega(1))Te}{2}\right) \end{aligned} \tag{3.59}$$

Le mineur étant non nul pour $\omega_{0,1} \neq \frac{2k\pi}{T_e}$, avec $k \in \mathbb{Z}$, SM_2 est de rang 3 et la base mobile est commandable en deux coups. Il existe donc deux commandes équivalentes permettant d'amener l'état de $\chi(k-j)$ à $\chi(k)$ en deux coups. Si $\omega_{0,1} = \frac{2k\pi}{T_e}$ cela signifie que le robot se déplace en ligne droite. Il est alors commandable en un coup et les vitesses équivalentes sont alors facilement obtenues en appliquant les résultats obtenus dans la partie 3.2.2.2.

3.2.2.2 Calcul des vitesses équivalentes

Afin de calculer les deux commandes équivalentes permettant d'atteindre l'état $\chi(k)$ à partir de l'état $\chi(k-j)$, nous utilisons les équations d'évolution du système robotique (3.48), (3.49), (3.50) et (3.51). La première commande équivalente doit permettre d'atteindre le point $(X_M(k),\, Y_M(k))$ à partir de l'état $\chi(k-j)$, mais ne garantit pas la bonne orientation de la base mobile. La seconde vitesse équivalente sera donc utilisée afin d'orienter correctement le système robotique. Le calcul des vitesses équivalentes se présente de la manière suivante :

1. **Calcul de ω_{e1}**
   Dans un premier temps, nous définissons les deux variables suivantes :

$$\Delta_X = X_M(k) - X_M(k-j) \tag{3.60}$$

$$\Delta_Y = Y_M(k) - Y_M(k-j) \tag{3.61}$$

   A partir des équations (3.48) et (3.49), il est possible d'obtenir :

$$\begin{aligned} \frac{\Delta_X}{\Delta_Y} &= \frac{\sin(\theta(k-j)+\omega_{e1}T_{ve}) - \sin(\theta(k-j)}{-\cos(\theta(k-j)+\omega_{e1}T_{ve}) + \cos(\theta(k-j)} \\ \frac{\Delta_X}{\Delta_Y} &= \frac{2\sin\left(\frac{\omega_{e1}T_{ve}}{2}\right)\cos\left(\frac{2\theta(k-j)+\omega_{e1}T_{ve}}{2}\right)}{2\sin\left(\frac{\omega_{e1}T_{ve}}{2}\right)\sin\left(\frac{2\theta(k-j)+\omega_{e1}T_{ve}}{2}\right)} \quad ; \quad \omega_{e1} \neq k\pi \end{aligned}$$

   Soit

$$\frac{\Delta_Y}{\Delta_X} = \tan\left(\frac{2\theta(k-j)+\omega_{e1}T_{ve}}{2}\right)$$

   et

$$\omega_{e1} = \frac{2}{T_{ve}}\left(\arctan(\frac{\Delta_Y}{\Delta_X}) - \theta(k-j)\right) \tag{3.62}$$

   où T_{ve} est la période d'échantillonnage pour les vitesses équivalentes.

2. **Calcul de υ_{e1}**
   Nous cherchons maintenant à obtenir une expression de υ_{e1} à partir des équations (3.48), (3.49) et de ω_{e1} [5]. Grâce à l'équation (3.48), nous obtenons :

$$\Delta_X^2 = \frac{\upsilon_{e1}^2}{\omega_{e1}^2}[\sin(\theta(k-j)+\omega_{e1}T_{ve}) - \sin(\theta(k-j))]^2 \quad (3.63)$$

$$\Delta_X^2 = \frac{\upsilon_{e1}^2}{\omega_{e1}^2}[2\sin\left(\frac{\omega_{e1}T_{ve}}{2}\right)\cos\left(\frac{2\theta(k-j)+\omega_{e1}T_{ve}}{2}\right)]^2 \quad (3.64)$$

   De la même manière, à partir de l'équation (3.49), il vient :

$$\Delta_Y^2 = \frac{\upsilon_{e1}^2}{\omega_{e1}^2}[\cos(\theta(k-j)+\omega_{e1}T_{ve}) - \cos(\theta(k-j)]^2 \quad (3.65)$$

$$\Delta_Y^2 = \frac{\upsilon_{e1}^2}{\omega_{e1}^2}[-2\sin\left(\frac{\omega_{e1}T_{ve}}{2}\right)\sin\left(\frac{2\theta(k-j)+\omega_{e1}T_{ve}}{2}\right)]^2 \quad (3.66)$$

   Nous déterminons υ_{e1} en additionnant les équations (3.64) et (3.66) :

$$\Delta_X^2 + \Delta_Y^2 = 4\frac{\upsilon_{e1}^2}{\omega_{e1}^2}\sin^2\left(\frac{\omega_{e1}T_{ve}}{2}\right) \quad (3.67)$$

$$\upsilon_{e1} = \frac{\omega_{e1}}{2\sin\left(\frac{\omega_{e1}T_{ve}}{2}\right)}\sqrt{\Delta_X^2+\Delta_Y^2} \quad (3.68)$$

3. **Calcul de ϖ_{e1}**
   La première commande équivalente est utilisée afin de simplement positionner la base mobile. Nous ne cherchons pas à orienter la platine, donc nous fixons $\varpi_{e1} = 0$.

4. **Calcul de υ_{e2}**
   Notre seconde commande équivalente permet d'orienter le système robotique. La base mobile étant correctement positionnée, nous fixons $\upsilon_{e2} = 0$.

5. **Calcul de ω_{e2}**
   Afin d'orienter correctement la base mobile, nous utilisons l'équation (3.50). Il vient :

$$\omega_{e2} = \frac{\theta(k) - (\theta(k-j)+\omega_{e1}T_{ve})}{T_{ve}} \quad (3.69)$$

6. **Calcul de ϖ_{e2}**
   Pour calculer la commande permettant d'orienter la platine, nous exploitons la relation (3.51).

$$\varpi_{e2} = \frac{\vartheta(k) - \vartheta(k-j)}{T_{ve}} \quad (3.70)$$

5. Si $\omega_{e1} = 0$, alors la base mobile se déplace en ligne droite et est commandable en un coup (voir 3.2.2.1). Alors les vitesses équivalentes sont notées $\dot{q}_e$ et $\upsilon_e = \frac{\sqrt{\Delta_X^2+\Delta_Y^2}}{T_{ve}}$, $\omega_e = 0$ et $\varpi_e = \frac{\vartheta(k)-\vartheta(k-j)}{T_{ve}}$.

Remarque : *Dans le cas d'une commandabilité en deux coups, deux commandes équivalentes $\dot{q}_{e1}$ et $\dot{q}_{e2}$ sont calculées afin de relier mathématiquement l'état $\chi(k-j)$ à l'état $\chi(k)$. Il existe alors un état virtuel noté $\chi(v)$, présenté dans la figure 3.6, qui correspond à l'état atteint virtuellement par le robot après l'application de la première commande équivalente $\dot{q}_{e1}$. D'une manière générale, l'état $\chi(v)$ ne correspond pas à un état réellement atteint par le robot lors de la navigation et l'image $s(v)$ associée à cet état ne peut être mesurée.*

Remarque : *Le calcul des vitesses équivalentes dépend de la période d'échantillonnage T_{ve}. Etant donné que ces vitesses ne sont utilisées que pour relier mathématiquement les états $\chi(k)$ et $\chi(k-j)$, la période d'échantillonnage importe peu. En effet la trajectoire associée aux commandes équivalentes n'est jamais réalisée physiquement. Il est par contre nécessaire d'utiliser T_{ve} dans les équations d'évolution du robot lorsque nous cherchons à relier les deux états $\chi(k-j)$ et $\chi(k)$ pour estimer la profondeur. Afin d'éviter toute erreur, nous choisissons une période d'échantillonnage T_{ve} égale à T_e.*

3.2.2.3 Calcul du prédicteur/correcteur avec n images

Il est maintenant possible de relier mathématiquement les états $\chi(k)$ et $\chi(k-j)$ en utilisant une séquence de commandes équivalentes comprenant au plus deux éléments [6]. Les prédicteurs requis par le critère (3.45) peuvent donc être calculés à partir des équations d'évolution des indices visuels (cf. section 3.1.1) tout en limitant la récursivité.

Chaque commande équivalente correspond à un des quatre cas d'évolution des indices visuels présentés dans la section 3.1.1. Afin de déterminer analytiquement un prédicteur, il est nécessaire de combiner les équations d'évolution des indices associés à la première commande équivalente et celles correspondant à la seconde commande. Il existe donc seize expressions analytiques des prédicteurs avec le critère (3.45), alors qu'il n'en existait que quatre pour le critère (3.27). Par exemple, dans le cas où $\dot{q}_{e1} = [1\ 1\ 0]^T$ et $\dot{q}_{e2} = [0\ 1\ -2]^T$ les prédicteurs sont calculés en utilisant deux fois les équations du cas numéro 1, tandis qu'avec $\dot{q}_{e1} = [1\ 1\ 0]^T$ et $\dot{q}_{e2} = [0\ 1\ -1]^T$ ils sont calculés avec les équations du cas numéro 1 combinées avec celles du cas numéro 3.

Dans cette partie, nous présentons la méthode de calcul amenant à l'expression de l'estimateur lorsque les deux commandes équivalentes correspondent aux équations des indices visuels du cas numéro 1. Les autres cas sont présentés dans l'annexe B. Dans un premier temps, nous rappelons les équations d'évolution des indices visuels aux instants v et k et les exprimons en fonction des commandes $\dot{q}(k-j) = \dot{q}_{e1}$ et $\dot{q}(v) = \dot{q}_{e2}$:

$$X_P(v) = \frac{z_P(k-j)X_P(k-j)}{z_P(v)} \tag{3.71}$$

$$Y_P(v) = \frac{f}{z_P(v)}\left\{\frac{Y_P(k-j)z_P(k-j)}{f}\cos(A_1(k-j)) - z_P(k-j)\sin(A_1(k-j)) + \gamma(k-j)\right\} \tag{3.72}$$

6. Dans le cas où une seule commande équivalente est nécessaire (pour un déplacement du robot en ligne droite par exemple), les prédicteurs présentés dans la section 3.2.1 sont utilisés.

$$\gamma(k-j) = \Big(-D_x \sin(\vartheta(k-j)) - \tfrac{v(k-j)}{\omega(k-j)} \cos(\vartheta(k-j)) + C_y\Big) \cos(A_1(k-j)) \\ - \Big(D_x \cos(\vartheta(k-j)) - \tfrac{v(k-j)}{\omega(k-j)} \sin(\vartheta(k-j)) + C_x\Big) \sin(A_1(k-j)) \\ + D_x \cos(\vartheta(v)) + \tfrac{v(k-j)}{\omega(k-j)} \sin(\vartheta(v)) - C_y$$

$$z_P(v) = \frac{Y_P(k-j) z_P(k-j)}{f} \sin(A_1(k-j)) + z_P(k-j) \cos(A_1(k-j)) + \beta_1(k-j) \tag{3.73}$$

$$X_P(k) = \frac{z_P(v) X_P(v)}{z_P(k)} \tag{3.74}$$

$$Y_P(k) = \frac{f}{z_P(k)} \left\{ \frac{Y_P(v) z_P(v)}{f} \cos(A_1(v)) - z_P(v) \sin(A_1(v)) + \gamma(v) \right\} \tag{3.75}$$

$$\gamma(v) = \Big(-D_x \sin(\vartheta(v)) - \tfrac{v(v)}{\omega(v)} \cos(\vartheta(v)) + C_y\Big) \cos(A_1(v)) \\ - \Big(D_x \cos(\vartheta(v)) - \tfrac{v(v)}{\omega(v)} \sin(\vartheta(v)) + C_x\Big) \sin(A_1(v)) + D_x \cos(\vartheta(k)) + \tfrac{v(v)}{\omega(v)} \sin(\vartheta(k)) - C_y$$

$$z_P(k) = \frac{Y_P(v) z_P(v)}{f} \sin(A_1(v)) + z_P(v) \cos(A_1(v)) + \beta_1(v) \tag{3.76}$$

Nous cherchons à obtenir les équations analytiques des prédicteurs $\hat{X}_P(k|k-j)$ et $\hat{Y}_P(k|k-j)$ dépendant uniquement des images mesurées $\tilde{X}_P(k-j)$ et $\tilde{Y}_P(k-j)$, ainsi que de la profondeur $\hat{z}_P(k|k-j)$. En effet, les valeurs $X_P(v)$ et $Y_P(v)$ sont inconnues car elles correspondent à une position virtuelle du robot. De plus l'expression doit être fonction seulement de $z_P(k)$, puisqu'il s'agit de la valeur à estimer. Afin d'obtenir l'expression désirée, l'équation (3.72) est réécrite de la manière suivante :

$$\frac{Y_P(v) z_P(v)}{f} = \frac{Y_P(k-j) z_P(k-j)}{f} \cos(A_1(k-j)) - z_P(k-j) \sin(A_1(k-j)) + \gamma(k-j) \tag{3.77}$$

Ensuite, nous injectons l'équation (3.77) dans l'équation (3.76) afin d'obtenir une expression de $z_P(k)$ dépendant uniquement d'images à l'instant $k-j$. Il vient :

$$z_P(k) = \Big\{ (\tfrac{Y_P(k-j)}{f} \cos(A_1(k-j)) - \sin(A_1(k-j))) \sin(A_1(v)) \\ + (\tfrac{Y_P(k-j)}{f} \sin(A_1(k-j)) + \cos(A_1(k-j))) \cos(A_1(v)) \Big\} z_P(k-j) \\ + \beta_1(k-j) \cos(A_1(v)) + \beta_1(v) + \gamma(k-j) \sin(A_1(v)) \tag{3.78}$$

Nous pouvons alors exprimer $z_P(k-j)$ comme suit :

$$z_P(k-j) = \frac{z_P(k) + \beta_1(k-j) \cos(A_1(v)) - \beta_1(v) - \gamma(k-j) \sin(A_1(v))}{\mu \cos(A_1(v)) + \nu \sin(A_1(v))} \tag{3.79}$$

avec

$$\mu = \frac{Y_P(k-j)}{f}\sin(A_1(k-j)) + \cos(A_1(k-j))$$

et

$$\nu = \frac{Y_P(k-j)}{f}\cos(A_1(k-j)) + \sin(A_1(k-j))$$

Il reste maintenant à obtenir une expression de $X_P(k)$ dépendant de $X_P(k-j)$. Pour cela, nous reprenons la relation (3.74) comme suit :

$$X_P(k) = \frac{z_P(v)X_P(v)}{z_P(k)} = \frac{z_P(k-j)X_P(k-j)}{z_P(k)} \tag{3.80}$$

En injectant l'équation (3.79) dans (3.80), il vient :

$$X_P(k) = \frac{z_P(k) + \beta_1(k-j)\cos(A_1(v)) - \beta_1(v) - \gamma(k-j)\sin(A_1(v))}{\mu\cos(A_1(v)) + \nu\sin(A_1(v))}\frac{X_P(k-j)}{z_P(k)} \tag{3.81}$$

$X_P(k)$ n'est plus fonction de $z_P(v)$. L'équation (3.81) peut être réécrite de la manière suivante :

$$X_P(k) = \frac{X_P(k-j)}{\varphi(v)} - \frac{\beta_1(k-j)\cos(A_1(v)) + \beta_1(v) + \gamma(k-j)\sin(A_1(v))}{\varphi(v)z_P(k)}X_P(k-j) \tag{3.82}$$

avec

$$\varphi(v) = \frac{Y_P(k-j)}{f}\sin(A_1(k-j) + A(v)) + \cos(A_1(k-j) + A(v))$$

De la même manière que pour $X_P(k)$, nous injectons l'équation (3.77) dans (3.75) afin d'obtenir une expression de $Y_P(k)$ qui ne dépend plus de $Y_P(v)$ mais de $Y_P(k-j)$.

$$\begin{aligned} Y_P(k) = \tfrac{f}{z_P(k)}\Big\{ \Big(\tfrac{Y_P(k-j)z_P(k-j)}{f}\cos(A_1(k-j)) - z_P(k-j)\sin(A_1(k-j)) + \gamma(k-j)\Big)\cos(A_1(v)) \\ -z_P(v)\sin(A_1(v)) + \gamma(v)\} \end{aligned} \tag{3.83}$$

Afin de remplacer $z_P(v)$ par une expression dépendant de $z_P(k-j)$, l'équation (3.73) est utilisée dans (3.83).

$$\begin{aligned} Y_P(k) = \tfrac{f}{z_P(k)}\Big\{ \Big(\tfrac{Y_P(k-j)z_P(k-j)}{f}\cos(A_1(k-j)) - z_P(k-j)\sin(A_1(k-j)) + \gamma(k-j)\Big)\cos(A_1(v)) \\ -\Big(\tfrac{Y_P(k-j)z_P(k-j)}{f}\sin(A_1(k-j)) + z_P(k-j)\cos(A_1(k-j)) + \beta_1(k-j)\Big)\sin(A_1(v)) + \gamma(v)\} \end{aligned} \tag{3.84}$$

Finalement en substituant $z_P(k-j)$ par l'équation (3.79), nous obtenons :

$$Y_P(k) = \frac{f\psi(v)}{\varphi(v)} + \frac{f}{z_P k}\varrho(k-j) \tag{3.85}$$

avec

$$\psi(v) = \frac{Y_P(k-j)}{f}\cos(A_1(k-j)+A(v)) - \sin(A_1(k-j)+A(v))$$

et

$$\begin{array}{c}\varrho(k-j) = \frac{\psi(v)}{\varphi(v)}\left(-\beta_1(k-j)\cos(A_1(v)) - \beta_1(v) - \gamma(k-j)\sin(A_1(v))\right)\\ +\gamma(k-j)\cos(A_1(v)) + \gamma(v) - \beta_1(k-j)\sin(A_1(v))\end{array}$$

Les expressions analytiques (3.82) et (3.85) dépendant uniquement de $X_P(k-j)$ $Y_P(k-j)$ et $z_P(k)$, elles peuvent être utilisées pour obtenir les prédicteurs $\hat{X}_P(k|k-j)$ et $\hat{Y}_P(k|k-j)$.

$$\hat{X}_P(k|k-j) = \frac{\tilde{X}_P(k-j)}{\tilde{\varphi}(v)} - \frac{\beta_1(k-j)\cos(A_1(v)) + \beta_1(v) + \gamma(k-j)\sin(A_1(v))}{\tilde{\varphi}(v)\hat{z}_P(k|k-j)}\tilde{X}_P(k-j) \tag{3.86}$$

$$\hat{Y}_P(k|k-j) = \frac{f\tilde{\psi}(v)}{\tilde{\varphi}(v)} + \frac{f}{\hat{z}_P(k|k-j)}\varrho(k-j) \tag{3.87}$$

avec

$$\tilde{\psi}(v) = \frac{\tilde{Y}_P(k-j)}{f}\cos(A_1(k-j)+A(v)) - \sin(A_1(k-j)+A(v))$$

et

$$\tilde{\varphi}(v) = \frac{\tilde{Y}_P(k-j)}{f}\sin(A_1(k-j)+A(v)) + \cos(A_1(k-j)+A(v))$$

Les expressions des prédicteurs étant déterminées, le critère (3.45) peut être minimisé. Pour cela nous dérivons ce critère par rapport à $z_P(k|k-j)$. Il vient :

$$\frac{\partial C^*}{\partial \hat{z}_P(k|k-j)} = \sum_{j=1}^{n}\left\{2(\hat{X}_P(k|k-j) - \tilde{X}_P(k))\frac{\partial \hat{X}_P(k|k-j)}{\partial \hat{z}_P(k|k-j)} + 2(\hat{Y}_P(k|k-j) - \tilde{Y}_P(k))\frac{\partial \hat{Y}_P(k|k-j)}{\partial \hat{z}_P(k|k-j)}\right\} \tag{3.88}$$

Afin d'obtenir une estimation de la profondeur, nous calculons les dérivées des prédicteurs par rapport à $z_P(k|k-j)$:

$$\frac{\partial \hat{X}_P(k|k-j)}{\partial \hat{X}_P(k|k-j)} = \frac{\beta_1(k-j)\cos(A_1(v)) + \beta_1(v) + \gamma(k-j)\sin(A_1(v))}{\tilde{\varphi}(v)\hat{z}_P^2(k|k-j)}\tilde{X}_P(k-j) \tag{3.89}$$

$$\frac{\partial \hat{Y}_P(k|k-j)}{\partial \hat{z}_P(k|k-j)} = -\frac{f\varrho(k-j)}{\hat{z}_P^2(k|k-j)} \tag{3.90}$$

Finalement en utilisant les équations (3.86), (3.87), (3.88), (3.89) et (3.90), l'estimateur s'écrit :

$$\hat{z}_P(k|k) = \frac{\sum_{j=1}^{n} Num_P^j}{\sum_{j=1}^{n} Den_P^j} \tag{3.91}$$

avec

$$Num_P^j = \frac{(\beta_1(k-j)\cos(A_1(v)) + \beta_1(v) + \gamma(k-j)\sin(A_1(v)))^2 \tilde{X}^2(k-j)}{\tilde{\varphi}^2(k-j)} + f^2\varrho^2(k-j)$$

et

$$Den_P^j = \frac{(\beta_1(k-j)\cos(A_1(v))+\beta_1(v)+\gamma(k-j)\sin(A_1(v)))\tilde{X}(k-j)(\tilde{X}(k-j)-\tilde{\varphi}(k-j)\tilde{X}(k))}{\tilde{\varphi}^2(k-j)} - \frac{f\tilde{\varphi}(k-j)\varrho(k-j)(f\tilde{\psi}(k-j)-\tilde{Y}(k)\tilde{\varphi}(k-j))}{\varphi^2(k-j)}$$

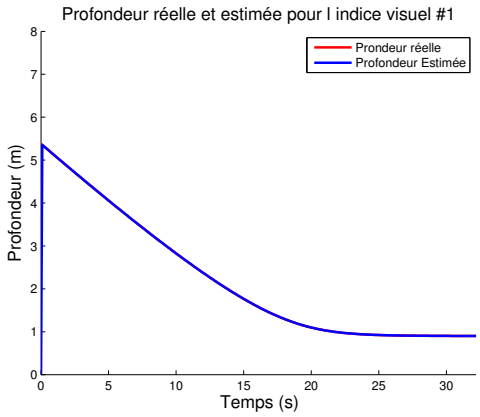
Nous avons obtenu une expression analytique de l'estimateur de $z_P(k|k)$ dépendant de l'image courante $s_P(k)$ ainsi que des n images précédentes $s_P(k-j)$, avec $j \in [1, n]$. En utilisant un plus grand nombre d'images que pour le critère (3.27), cette méthode doit permettre d'augmenter le rapport signal/bruit et donc d'estimer correctement la profondeur en contexte bruité.

Simulation

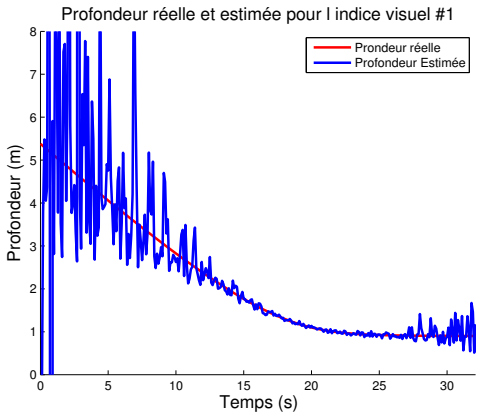
Afin de valider notre approche, nous proposons de rejouer les simulations de la section 3.2.1 en exploitant cette fois l'estimateur (3.91). Comme précédemment, la simulation est réalisée en utilisant dans un premier temps des données non bruitées puis bruitées. Etant donné que les conditions de simulation restent inchangées, la trajectoire du robot dans la scène ainsi que l'évolution des indices pour des données bruitées ou non bruitées sont similaires à celles présentées dans les figures 3.4(a), 3.4(b), 3.5(a) et 3.5(b).

La figure 3.7(a) montre l'estimation de la profondeur en contexte non bruité. Comme nous pouvons le constater sur cette figure, dans le cas où les données sont non bruitées, la profondeur est parfaitement estimée. Nous avons utilisé ici un estimateur basé sur 5 images. Naturellement, le résultat est le même quel que soit n. Les figures 3.7(b), 3.7(c), 3.7(d), 3.7(e) et 3.7(f) s'intéressent au contexte bruité. Dans ce cas, différentes valeurs de n ont été choisies : 5, 10, 20, 50 et 100 images, afin d'étudier l'influence de n sur la qualité de l'estimation. Dans chaque cas, l'estimateur parvient à converger vers la valeur réelle. Pour $n = 5$, $n = 10$ et $n = 20$, les temps de convergence sont sensiblement les mêmes. Cependant, plus le nombre d'images utilisées est grand, moins l'estimation oscille autour de la valeur réelle lors du régime permanent. De plus, pour les petites valeurs de n (figures 3.7(b), 3.7(c) et 3.7(d)), l'estimation diverge en fin de simulation. En effet à ce moment là, la caméra se déplaçant à très faibles vitesses, elle fournit des images très semblables. Le rapport signal/bruit est à nouveau trop faible pour permettre une estimation correcte de la profondeur. Dans les cas où $n = 50$ et $n = 100$, le temps de convergence est réduit et le phénomène de divergence présent dans les cas précédents n'apparait plus. En effet, n étant grand, il est possible de disposer d'un historique d'images important qui permet d'obtenir un rapport signal/bruit suffisant pour estimer correctement et rapidement la profondeur.

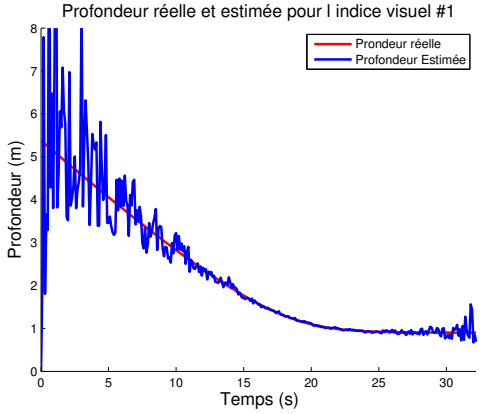
Avec un choix de n suffisament grand, il est donc possible d'estimer la profondeur en utilisant des données issues d'images ou de l'odométrie bruitées. Cependant, n ne peut pas être choisi infiniment grand pour deux raisons. Premièrement, le temps de calcul est directement lié à ce choix. Il faut donc faire un compromis entre la précision désirée et le temps de calcul qui peut être dédié à cette estimation. Secondement, afin de relier deux images, il est nécessaire d'utiliser



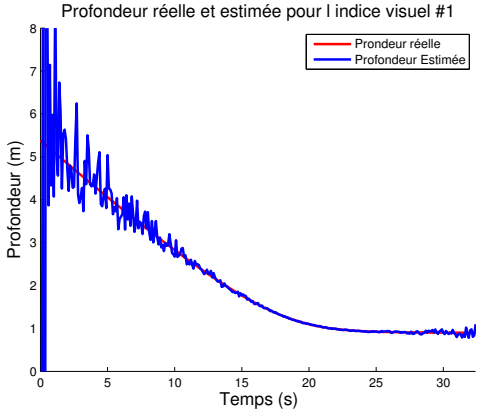
(a) Données non bruitées et $n = 5$



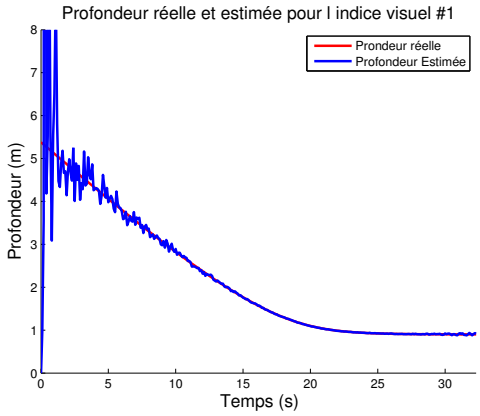
(b) Données bruitées et $n = 5$



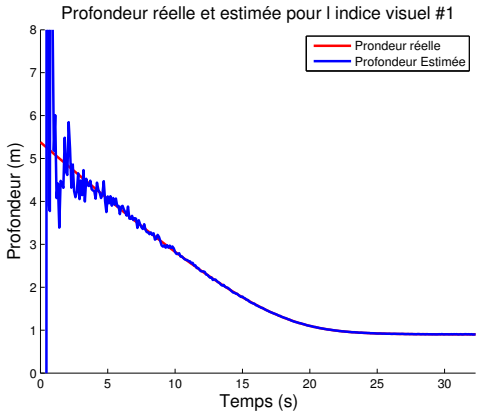
(c) Données bruitées et $n = 10$



(d) Données bruitées et $n = 20$



(e) Données bruitées et $n = 50$



(f) Données bruitées et $n = 100$

FIGURE 3.7 – Simulation pour un estimateur avec n images

les données odométriques. Plus la valeur de n augmente plus l'erreur comprise dans les données odométriques est susceptible d'être importante. Il est ainsi possible que l'augmentation de la valeur de n ne soit pas toujours un gage d'amélioration de l'estimation de la profondeur.

Expérimentation

Nous avons également testé notre méthode d'estimation sur le robot Pekee II lors d'une navigation par asservissement visuel. La cible considérée est constituée de quatre cercles rouges d'un diamètre de 16 cm. A l'aide d'un traitement d'image basé sur la couleur, les centres des cercles sont déterminés et leurs coordonnées constituent les indices visuels utilisés. Initialement, le robot est positionné à une distance de 1.9 m [7] de la cible et est légèrement décalé par rapport à cette dernière. La position souhaitée se situe à 0.75 m en face de la cible. Durant la navigation, la vitesse linéaire de la base mobile est en moyenne de 0.1 m/s, tandis que les vitesses angulaires sont limitées à quelques radians/s. Le nombre maximal d'images utilisées par le processus d'estimation est $n = 30$. Enfin, la période d'échantillonnage est de 0.2 s.

La figure 3.9 présente les profondeurs estimées pour les quatre indices visuels constituant la cible. Etant donné que le robot n'est pas équipé de capteur permettant de mesurer cette donnée, il est difficile de conclure avec certitude sur la précision de l'estimation tout au long de la navigation. Cependant la valeur finale de la profondeur estimée est de 0.74 m. Elle est donc en adéquation parfaite avec la mesure de 0.75 cm effectuée à l'issue de la mission. De plus comme la caméra évolue face à la cible pendant la seconde partie, le fait que quatre processus d'estimation indépendants convergent vers une même valeur semble démontrer l'efficacité de notre approche. Il doit être également noté que le rapport signal/bruit étant directement modifié par la focale, les vitesses et la période d'échantillonnage, les performances présentées ici ne sont qu'une illustration du bon fonctionnement du processus d'estimation.

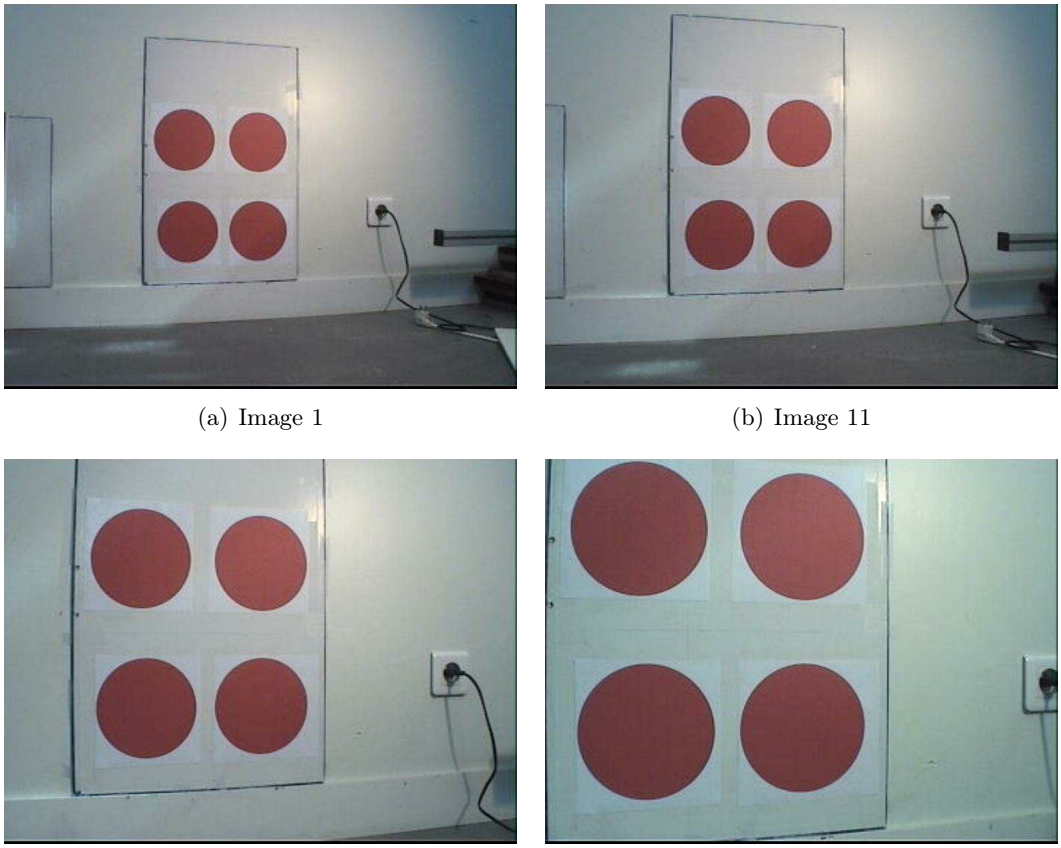
3.2.3 Seconde approche pour le prédicteur/correcteur avec n images

3.2.3.1 Etude de la commandabilité de la caméra

Dans la section 3.2.2.1, afin de déterminer la plus petite séquence de commandes reliant deux images, nous avons supposé qu'à un état du système robotique correspondait une image. Nous avons ensuite montré que notre robot était commandable en deux coups avant d'établir les expressions des deux vitesses équivalentes. A partir de ces résultats et des équations d'évolution, nous avons mis en place un estimateur de la profondeur se décomposant en 16 cas. Afin de réduire la complexité, il nous a semblé intéressant de revenir sur le problème de la commandabilité afin de voir s'il était possible de le résoudre avec une seule commande. En effet, dans un tel cas l'estimateur de la profondeur ne se décomposerait plus qu'en quatre solutions. Pour cela, nous proposons de renverser l'hypothèse qui a sous-tendu tous nos travaux jusqu'ici : une image correspond à une infinité d'états du robot, mais à un seul état de la caméra. Rappelons que ce dernier est défini à l'instant t_k comme suit :

$$\chi_C(k) = [X_C(k),\ Y_C(k),\ \theta_T(k)]^T \tag{3.92}$$

7. Des mesures précises étant difficilement réalisables, toutes les valeurs de cette partie expérimentale sont entachées d'erreurs.



(a) Image 1 (b) Image 11

(c) Image 34 (d) Image 50

FIGURE 3.8 – Images obtenues lors de la navigation

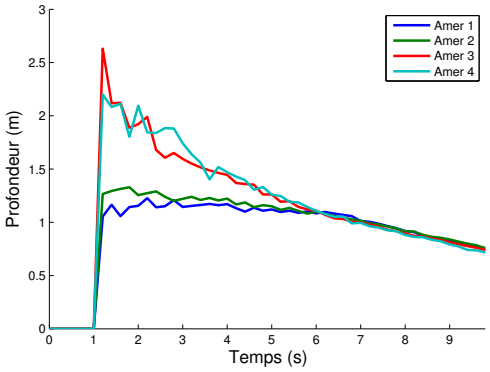


FIGURE 3.9 – Estimation de la profondeur avec le robot PekeeII

où $\theta_T(k) = \theta(k) + \vartheta(k)$. Le modèle cinématique associé à la caméra s'écrit alors :

$$\begin{cases} \dot{X}_C(t) = \upsilon(t)\cos(\theta(t)) - \omega(t)D_x\sin(\theta(t)) \\ \dot{Y}_C(t) = \upsilon(t)\sin(\theta(t)) + \omega(t)D_x\cos(\theta(t)) \\ \dot{\theta}_T(t) = \omega(t) + \varpi(t) \end{cases} \tag{3.93}$$

Comme précédemment, nous calculons la solution de ce système pour l'intervalle $[t_k, t_{k+1}]$ en supposant les commandes bloquées et $\omega \neq 0$ pendant cet intervalle. Cette solution correspond aux équations d'évolution discrète de la caméra $\chi_C(k+1) = g(\chi_C(k), \dot{q}(k))$. En posant $\eta_{C1} = \frac{\omega(0)T_e}{2}$ et $\eta_{C2} = \frac{2\theta(0)+\omega(0)T_e}{2}$, il vient :

$$X_C(k+1) = X_C(k) + \frac{2\upsilon(k)}{\omega(k)}\sin(\eta_{C1})\cos(\eta_{C2}) + 2D_x\sin(\eta_{C1})\sin(\eta_{C2}) \tag{3.94}$$

$$Y_C(k+1) = Y_C(k) - \frac{2\upsilon(k)}{\omega(k)}\sin(\eta_{C1})\sin(\eta_{C2}) + 2D_x\sin(\eta_{C1})\cos(\eta_{C2}) \tag{3.95}$$

$$\theta_T(k+1) = \theta_T(k) + (\omega(k) + \varpi(k)) * T_e \tag{3.96}$$

Afin que la caméra soit commandable en un coup, il est nécessaire que la matrice P_C soit de rang 3. Ici cette matrice s'écrit :

$$P_C = \frac{\partial g(\chi_C(0), \dot{q}(0))}{\partial \dot{q}(0)} \tag{3.97}$$

Après calcul, nous avons :

$$P_C = \begin{bmatrix} \frac{2}{\omega(0)}\sin(\eta_{C1})\cos(\eta_{C2}) & \zeta_1 & 0 \\ \frac{2}{\omega(0)}\sin(\eta_{C1})\sin(\eta_{C2}) & \zeta_2 & 0 \\ 0 & T_e & T_e \end{bmatrix} \tag{3.98}$$

avec

$$\zeta_1 = -\frac{2\upsilon(0)}{\omega^2(0)}\sin(\eta_{C1})\cos(\eta_{C2}) + \frac{\upsilon(0)}{\omega(0)}T_e\cos(\eta_{C1}+\eta_{C2}) - D_xT_e\sin(\eta_{C_1}+\eta_{C_2})$$

et

$$\zeta_2 = -\frac{2\upsilon(0)}{\omega^2(0)}\sin(\eta_{C1})\sin(\eta_{C2}) + \frac{\upsilon(0)}{\omega(0)}T_e\sin(\eta_{C_1}+\eta_{C_2}) + D_xT_e\cos(\eta_{C1}+\eta_{C2})$$

Afin d'étudier le rang de P_C, nous calculons son déterminant. Nous obtenons :

$$det(P_C) = \frac{\upsilon(0)}{\omega(0)}\sin(\eta_{C1})T_e + D_x\cos(\eta_{C1})T_e \tag{3.99}$$

Le déterminant étant non nul pour $\tan(\eta_{C1}) \neq -\frac{D_x\omega(0)}{\upsilon(0)}$, la matrice P_C est de rang plein. Il est donc possible de commander la caméra en un seul coup. Etant donné qu'il est possible de relier mathématiquement deux images par une seul commande, l'estimateur de la profondeur utilisant n images est ramené à quatre cas. Cependant, comme précédemment, il est nécessaire de déterminer les équations de la vitesse équivalente préalablement à tout autre calcul.

3.2.3.2 Calcul de la vitesse équivalente

Afin de calculer la vitesse équivalente $\dot{q}_{ec}$ permettant de relier deux images en un seul coup, nous nous appuyons sur les équations (3.94), (3.95) et (3.96). Dans un premier temps, nous cherchons à déterminer la vitesse angulaire de la base mobile ω_{ec}.

1. **Calcul de ω_{ec}**
   Tout d'abord, posons :

$$\Delta_{Xc} = X_C(k) - X_C(k-j) \qquad \Delta_{Yc} = Y_C(k) - Y_C(k-j)$$

   Les équations (3.94) et (3.95) peuvent être réécrites de la façon suivante :

$$\Delta_{Xc} + 2D_x \sin(\eta_{C1}) \sin(\eta_{C2}) = 2\frac{\upsilon_{ec}}{\omega_{ec}} \sin(\eta_{C1}) \cos(\eta_{C2}) \tag{3.100}$$

$$\Delta_{Yc} - 2D_x \sin(\eta_{C1}) \cos(\eta_{C2}) = 2\frac{\upsilon_{ec}}{\omega_{ec}} \sin(\eta_{C1}) \sin(\eta_{C2}) \tag{3.101}$$

   Nous rappelons que $\eta_{C1} = \frac{\omega(0)T_e}{2}$ et $\eta_{C2} = \frac{2\theta(0)+\omega(0)T_e}{2}$. En multipliant respectivement les équations (3.100) et (3.101) par $\sin(\eta_{C2})$ et $\cos(\eta_{C2})$ nous avons :

$$\Delta_{Xc} \sin(\eta_{C2}) + 2D_x \sin(\eta_{C1}) \sin^2(\eta_{C2}) = 2\frac{\upsilon_{ec}}{\omega_{ec}} \sin(\eta_{C1}) \cos(\eta_{C2}) \sin(\eta_{C2}) \tag{3.102}$$

$$\Delta_{Yc} \cos(\eta_{C2}) - 2D_x \sin(\eta_{C1}) \cos^2(\eta_{C2}) = 2\frac{\upsilon_{ec}}{\omega_{ec}} \sin(\eta_{C1}) \sin(\eta_{C2}) \cos(\eta_{C2}) \tag{3.103}$$

   En soustrayant ces deux équations, il vient :

$$\Delta_{Xc} \sin(\eta_{C2}) - \Delta_{Yc} \cos(\eta_{C2}) + 2D_x \sin(\eta_{C1}) = 0 \tag{3.104}$$

   Afin d'extraire ω_{ec} de l'équation (3.104), nous utilisons les relations classiques de la trigonométrie. Nous obtenons :

$$\begin{array}{c} \Delta_{Xc} \left(\sin(\theta(k-j)) \cos(\eta_{C1}) + \cos(\theta(k-j)) \sin(\eta_{C1})\right) \\ -\Delta_{Yc} \left(\cos(\theta(k-j)) \cos(\eta_{C1}) + \sin(\theta(k-j)) \sin(\eta_{C1})\right) \\ +2D_x \sin(\eta_{C1}) = 0 \end{array} \tag{3.105}$$

$$\begin{array}{c} \sin(\eta_{C1}) \left[2D_x + \Delta_{Xc} \cos(\theta(k-j)) + \Delta_{Yc} \sin(\theta(k-j))\right] \\ + \cos(\eta_{C1}) \left[\Delta_{Xc} \sin(\theta(k-j)) + \Delta_{Yc} \cos(\theta(k-j))\right] = 0 \end{array} \tag{3.106}$$

$$\omega_{ec} = \frac{2}{T_e} \arctan - \frac{\Delta_{Xc} \sin(\theta(k-j)) + \Delta_{Yc} \cos(\theta(k-j))}{2D_x + \Delta_{Xc} \cos(\theta(k-j)) + \Delta_{Yc} \sin(\theta(k-j))} \tag{3.107}$$

2. **Calcul de υ_{ec}**
   Il est désormais nécessaire de calculer la vitesse linéaire de la base mobile en fonction de

sa vitesse angulaire. Pour cela, les équations (3.94) et (3.95) sont élevées au carré puis sommées. Nous obtenons :

$$\begin{aligned}\Delta_{Xc}^2+\Delta_{Yc}^2=\tfrac{\upsilon_{ec}^2}{\omega_{ec}^2}\Big(1-2\sin(\theta(k-j)+\omega_{ec}T_e)\sin(\theta(k-j))\\ -2\cos(\theta(k-j)+\omega_{ec}T_e)\cos(\theta(k-j))+1\Big)\\ +D_x^2(1-2\cos(\theta(k-j)+\omega_{ec}T_e)\cos(\theta(k-j))-2\sin(\theta(k-j)+\omega_{ec}T_e)\sin(\theta(k-j))+1)\end{aligned} \tag{3.108}$$

$$\Delta_{Xc}^2+\Delta_{Yc}^2=4\sin^2(\eta_{C1})\left(\frac{\upsilon_{ec}^2}{\omega_{ec}^2}+D_x^2\right) \tag{3.109}$$

La vitesse linéaire équivalente pour la base mobile s'écrit donc :

$$\upsilon_{ec}=\sqrt{\omega_{ec}^2\left(\frac{\Delta_{Xc}^2+\Delta_{Yc}^2}{4\sin^2(\eta_{C1})}-D_x^2\right)} \tag{3.110}$$

3. **Calcul de ϖ_{ec}**
   A partir de l'équation (3.96), nous obtenons directement la vitesse angulaire équivalente pour la platine en fonction de ω_{ec} :

$$\varpi_{ec}=\frac{1}{T_e}(\vartheta(k)-\vartheta(k-j)+\theta(k)-\theta(k-j)+\omega_{ec}T_e) \tag{3.111}$$

3.2.3.3 Calcul du prédicteur/correcteur avec n images

Nous sommes maintenant en mesure de relier deux images à partir d'une seule commande équivalente donnée par les relations (3.107) (3.110) et (3.111). Il est donc possible d'estimer la profondeur des indices visuels en utilisant le critère (3.45) qui est fonction de n images. Ainsi l'estimation peut être réalisée à partir de données bruitées de manière similaire au travail effectué dans la section 3.2.2.3 tout en utilisant un nombre réduit d'itérations. L'expression de $z_P(k|k)$ est donnée par :

$$\hat{z}_P(k|k)=\frac{\sum_{j=1}^{n}Num_P^j}{\sum_{j=1}^{n}Den_P^j} \tag{3.112}$$

Les termes Num_P^j et Den_P^j sont calculés de la même manière que dans la partie 3.2.1. Pour cette raison, nous ne détaillerons pas les calculs permettant d'obtenir les résultats présentés ci-après. Nous rappelons que nous avons quatre cas qui dépendent de la vitesse équivalente $\dot{q}_{ec}$.

• **Cas n°1 : si $\omega_{ec}\neq 0$, $\varpi_{ec}\neq 0$ et $\omega_{ec}\neq\varpi_{ec}$:**

$$Num_P^j=\tilde{X}_P^2(k-j)\beta_{e1}^2+f^2\tilde{\alpha}_{e1}^2\tilde{\lambda}_{e1}^2 \tag{3.113}$$

$$\begin{aligned}Den_P^j=\tilde{X}_P(k-j)\beta_{e1}\Big(\tilde{X}_P(k-j)-\tilde{X}_P(k)\tilde{\alpha}_{e1}\Big)\\ -\Big(\tilde{Y}_P(k-j)\cos(A_{e1})-f\sin(A_{e1})\Big)f\tilde{\alpha}_{e1}\tilde{\lambda}_{e1}+\tilde{Y}_P(k)f\tilde{\alpha}_{e1}^2\tilde{\lambda}_{e1}\end{aligned} \tag{3.114}$$

avec

$$A_{e1} = (\omega_{ec} + \varpi_{ec})T_e$$

$$\tilde{\alpha}_{e1} = \frac{\tilde{Y}_P(k-j)}{f}\sin(A_{e1}) + \cos(A_{e1})$$

$$\begin{aligned}\beta_{e1} = &\Big(- D_x \sin(\vartheta(k-j)) - \tfrac{\upsilon_{ec}}{\omega_{ec}}\cos(\vartheta(k-j)) + C_y\Big)\sin(A_{e1}) \\ +&\Big(D_x \cos(\vartheta(k-j)) - \tfrac{\upsilon_{ec}}{\omega_{ec}}\sin(\vartheta(k-j)) + C_x\Big)\cos(A_{e1}) - D_x\cos(\vartheta(k)) + \tfrac{\upsilon_{ec}}{\omega_{ec}}\sin(\vartheta(k)) - C_x\end{aligned}$$

et

$$\begin{aligned}\tilde{\lambda}_{e1} = &\Big(\tfrac{-\beta_{e1}\tilde{Y}_P(k-j)}{\tilde{\alpha}_{e1}f} - D_x\sin(\vartheta(k-j)) - \tfrac{\upsilon_{ec}}{\omega_{ec}}\cos(\vartheta(k-j)) + C_y\Big)\cos(A_{e1}) \\ &- \Big(\tfrac{-\beta_{e1}}{\tilde{\alpha}_{e1}} + D_x\cos(\vartheta(k-j)) - \tfrac{\upsilon_{ec}}{\omega_{ec}}\sin(\vartheta(k-j)) + C_x\Big)\sin(A_{e1}) \\ &+ D_x\sin(\vartheta(k)) + \tfrac{\upsilon_{ec}}{\omega_{ec}}\cos(\vartheta(k)) - C_y\end{aligned}$$

- **Cas n°2 : si $\omega_{ec} = 0$ et $\varpi_{ec} \neq 0$:**

$$Num_P^j = \tilde{X}_P^2(k-j)\beta_{e2}^2 + f^2\tilde{\alpha}_{e2}^2\tilde{\lambda}_{e2}^2 \tag{3.115}$$

$$\begin{aligned}Den_P^j = &\tilde{X}_P(k-j)\beta_{e2}\Big(\tilde{X}_P(k-j) - \tilde{X}_P(k)\tilde{\alpha}_{e2}\Big) \\ &-\Big(\tilde{Y}_P(k-j)\cos(A_{e2}) - f\sin(A_{e2})\Big)f\tilde{\alpha}_{e2}\tilde{\lambda}_{e2} + \tilde{Y}_P(k)f\tilde{\alpha}_{e2}^2\tilde{\lambda}_{e2}\end{aligned} \tag{3.116}$$

avec

$$A_{e2} = \varpi_{ec}T_e$$

$$\tilde{\alpha}_{e2} = \frac{\tilde{Y}_P(k-j)}{f}\sin(A_{e2}) + \cos(A_{e2})$$

$$\begin{aligned}\beta_{e2} = &\Big(- \tfrac{\upsilon_{ec}}{2\varpi_{ec}}\cos(\vartheta(k-1)) + C_y\Big)\sin(A_{e2}) \\ +&\Big(- \tfrac{\upsilon_{ec}}{2\varpi_{ec}}\sin(\vartheta(k-1)) + C_x\Big)\cos(A_{e2}) - \upsilon_{ec}T_e + \tfrac{\upsilon_{ec}}{2\varpi_{ec}}\sin(\vartheta(k)) - C_x\end{aligned}$$

et

$$\begin{aligned}\tilde{\lambda}_{e2} = &\Big(\tfrac{-\beta_{e2}\tilde{Y}_P(k-j)}{\alpha_{e2}f} - \tfrac{\upsilon_{ec}}{2\varpi_{ec}}\cos(\vartheta(k-1)) + C_y\Big)\cos(A_{e2}) \\ &- \Big(\tfrac{-\beta_{e2}}{\alpha_{e2}} - \tfrac{\upsilon_{ec}}{2\varpi_{ec}}\sin(\vartheta(k-1)) + C_x\Big)\sin(A_{e2}) + \upsilon_{ec}T_e + \tfrac{\upsilon_{ec}}{2\varpi_{ec}}\cos(\vartheta(k)) - C_y\end{aligned}$$

- **Cas n°3 : si $\omega_{ec} = \varpi_{ec}$:**

$$Num_P^j = \tilde{X}_P^2(k-j)\beta_{e3}^2 + \tilde{\lambda}_{e3}^2 \tag{3.117}$$

$$Den_P^j = \tilde{X}_P(k-j)\beta_{e3}\Big(\tilde{X}_P(k-j) - \tilde{X}_P(k)\Big) - \Big(\tilde{Y}_P(k-j) - \tilde{Y}_P(k)\Big)\tilde{\lambda}_{e3} \tag{3.118}$$

avec

$$\beta_{e3} = -\frac{\upsilon_{ec}}{\varpi_{ec}}\Big(\sin(\vartheta(k)) - \sin(\vartheta(k-j)\Big) - D_x\Big(\cos(\vartheta(k)) - \cos(\vartheta(k-j)\Big)$$

et

$$\tilde{\lambda}_{e3} = -f\frac{\upsilon_{ec}}{\varpi_{ec}}\Big(\cos(\vartheta(k)) - \cos(\vartheta(k-j))\Big) + fD_x\Big(\sin(\vartheta(k)) - \sin(\vartheta(k-j))\Big) - \tilde{Y}_P(k-j)\beta_3$$

- **Cas n°4 : si $\omega_{ec} = 0$ et $\varpi_{ec} = 0$:**

$$Num_P^j = -\tilde{X}_P^2(k-j)\beta_{e4}^2 + \tilde{\lambda}_{e4}^2 \tag{3.119}$$

$$Den_P^j = \tilde{X}_P(k-j)\beta_{e4}\Big(\tilde{X}_P(k-j) - \tilde{X}_P(k)\Big) + \Big(\tilde{Y}_P(k-j) - \tilde{Y}_P(k)\Big)\tilde{\lambda}_{e4} \tag{3.120}$$

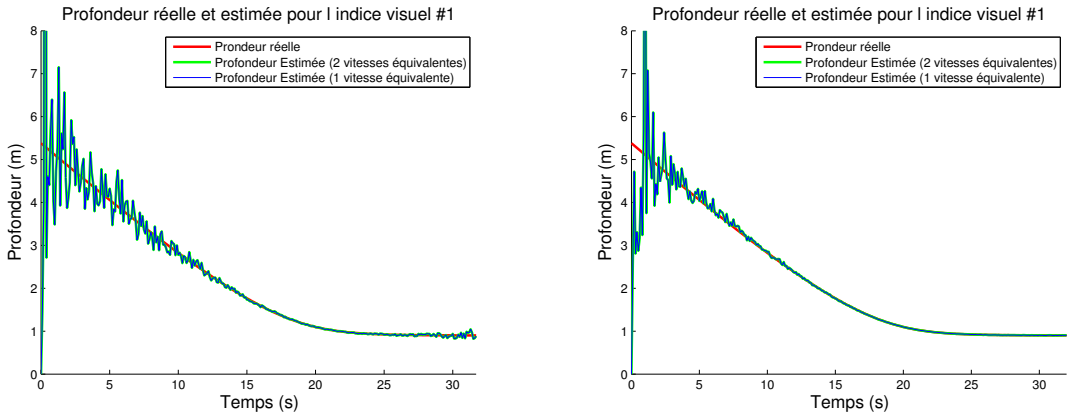
avec

$$\beta_{e4} = -\upsilon_{ec}\cos(\vartheta(k-j))T_e$$

et

$$\tilde{\lambda}_{e4} = f\upsilon_{ec}\sin(\vartheta(k-j)) - \tilde{Y}_P(k-j)\beta_{e4}$$

Simulation



(a) Données bruitées et $n = 20$ (b) Données bruitées et $n = 100$

FIGURE 3.10 – Simulation pour un estimateur avec n images pour une commandabilité en 1 coup

Nous avons simulé cette approche dans les mêmes conditions que celles présentées dans la partie 3.2.1. La trajectoire du robot dans la scène ainsi que l'évolution des indices visuels sont donc similaires à celles obtenues sur les figures 3.4(a), 3.4(b), 3.5(a) et 3.5(b). Afin de comparer les vitesses équivalentes en 1 et 2 coups, les deux processus d'estimation sont utilisés en parallèle.

Les figures 3.10(a) et3.10(b) montrent les résultats obtenus lorsque le nombre d'images utilisées pour l'estimation de la profondeur est respectivement de $n = 20$ et $n = 100$. Comme nous pouvons le voir, les processus d'estimation se comportent exactement de la même manière. Ainsi, toutes les conclusions faites sur les simulations précédentes peuvent s'appliquer à cette partie, la seule différence se situant au niveau de la mise en œuvre qui est beaucoup plus simple ici.

3.3 Comparaison avec d'autres méthodes

La méthode basée sur un prédicteur/correcteur n'étant qu'une des solutions envisageables pour l'estimation de la profondeur des indices visuels, une comparaison avec les techniques issues de l'automatique s'impose. L'étude se limite aux méthodes adaptées au cas particulier de la navigation d'un robot mobile par asservissement visuel (se référer aux critères énoncés dans la section 3.1.2). Dans un premier temps les deux approches concernées sont présentées puis une étude comparative est réalisée à l'aide de diverses simulations.

Les deux méthodes décrites ici reposent sur la représentation d'état $x = [X_P \ Y_P \ \frac{1}{z_P}]^T$. La variation de l'état par rapport au torseur de la caméra $T = [V_c^T \ \Omega_c^T]^T \in \mathbb{R}^6$ est obtenue grâce à la matrice d'interaction du point comme suit :

$$\begin{aligned} \dot{x} &= \begin{bmatrix} -fx_3 & 0 & x_1x_3 & \frac{x_1x_2}{f} & -(f+\frac{x_1^2}{f}) & x_2 \\ 0 & -fx_3 & x_2x_3 & f+\frac{x_2^2}{f} & -\frac{x_1x_2}{f} & -x_1 \\ 0 & 0 & x_3^2 & \frac{x_2x_3}{f} & -\frac{x_1x_3}{f} & 0 \end{bmatrix} T \\ y &= \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \end{aligned} \tag{3.121}$$

3.3.1 Observateur non linéaire non minimal (De Luca *et al.* 2008)

Nous proposons ici une présentation de la méthode proposée dans [De Luca *et al.* 2008] que le lecteur intéressé pourra compléter par la lecture cet article. Les auteurs proposent d'estimer la profondeur à l'aide d'un observateur non linéaire non minimal.

Le système (3.121) n'admettant pas d'observateur linéaire, une classe plus générale d'observateurs est utilisée. En définissant $\hat{x}$ comme l'estimé de l'état x partiellement connu, l'observateur non linéaire et non minimal est défini par [De Luca *et al.* 2008] :

$$\dot{\hat{x}} = \alpha_{nm}(\hat{x}, y)T + \beta_{nm}(\hat{x}, y, T) \tag{3.122}$$

Pour un tel observateur, le terme $\alpha_{nm}(\hat{x}, y)T$ peut être considéré comme la copie du système (3.121) tandis que $\beta_{nm}(\hat{x}, y, T)$ est l'élément correctif permettant de faire converger les valeurs estimées vers les valeurs réelles. Le vecteur erreur étant défini par $e_{nm} = x - \hat{x}$, le sous-vecteur $[e_{nm1} \ e_{nm2}]$ est donc directement mesurable. Finalement, l'observateur non linéaire et non minimal est donné par [De Luca *et al.* 2008] :

$$\alpha_{nm}(\hat{x}, y) = \begin{bmatrix} -f\hat{x}_3 & 0 & y_1\hat{x}_3 & \frac{y_1y_2}{f} & -(f+\frac{y_1^2}{f}) & y_2 \\ 0 & -f\hat{x}_3 & y_2\hat{x}_3 & f+\frac{y_2^2}{f} & -\frac{y_1y_2}{f} & -y_1 \\ 0 & 0 & \hat{x}_3^2 & \frac{y_2\hat{x}_3}{f} & -\frac{y_1\hat{x}_3}{f} & 0 \end{bmatrix} \tag{3.123}$$

$$\beta_{nm}(\hat{x}, y, T) = \begin{bmatrix} k_{nm1}e_{nm1} \\ k_{nm2}e_{nm2} \\ k_{nm3}((-f\upsilon_{c1} + y_1\upsilon_{c3})e_1 + (-f\upsilon_{c2} + y_2\upsilon_{c3})e_2) \end{bmatrix} \tag{3.124}$$

où k_{nm1}, k_{nm2} et k_{nm3} sont des gains scalaires strictement positifs. Bien que x_1 et x_2 soient directement mesurables, ils sont néanmoins estimés. Le calcul de $\hat{x}_1$ et $\hat{x}_2$ dépendant de $\hat{x}_3$, une estimation correcte des deux premiers termes sous-entend alors une estimation correcte du dernier. De plus, si l'estimation de $\hat{x}_3$ est erronée, alors les termes e_{nm1} et e_{nm2} sont grands et le processus correctif amplifié. Nous pouvons donc penser l'observateur proposé limite la divergence de l'estimation.

3.3.2 Observateur non linéaire minimal (Morbidi & Prattichizzo 2009)

Dans [Morbidi & Prattichizzo 2009] l'estimation de la profondeur est réalisée à l'aide d'un observateur non linéaire minimal basé sur la méthode *Immersion and Invariance* [Astolfi & Ortega 2003]. Afin d'établir un observateur non linéaire basé sur la méthode *Immersion and Invariance*, le système (3.121) est réécrit sous la forme suivante avec $y_{II} = [x_1\ x_2]^T$ et $\eta_{II} = x_3$.

$$\begin{array}{l} \dot{y}_{II} = p_1(y_{II}, t) + g(y_{II}, t)\eta_{II} \\ \dot{\eta}_{II} = p_2(y_{II}, t)\eta_{II} + \upsilon_{c2}\eta_{II}^2 \end{array} \tag{3.125}$$

avec

$$p_1(y_{II}, t) = \begin{bmatrix} \frac{x_1 x_2 \omega_{c1}}{f} - (f + \frac{x_1^2}{f})\omega_{c2} + x_2\omega_{c3} \\ (f + \frac{x_2^2}{f})\omega_{c1} f - \frac{x_1 x_2 \omega_{c2}}{f} - x_1\omega_{c3} \end{bmatrix}$$

$$p_2(y_{II}, t) = \frac{x_2\omega_{c1} - x_1\omega_{c2}}{f}$$

et

$$g(y_{II}, t) = \begin{bmatrix} -f\upsilon_{c1} + x_1\upsilon_{c3} \\ -f\upsilon_{c2} + x_2\upsilon_{c3} \end{bmatrix}$$

L'observateur suivant permet de reconstruire l'état du système (3.125) [Morbidi & Prattichizzo 2009] :

$$\hat{\eta}_{II} = \left(\frac{M_{II}}{2}(\upsilon_{c3}(x_1^2 + x_2^2) - 2f(\upsilon_{c1}x_1 + \upsilon_{c2}x_2)) - \xi_{II} \right) ||g(y_{II}, t)|| \tag{3.126}$$

où M_{II} est un scalaire positif et $\dot{\xi}_{II}$ est donné par :

$$\begin{array}{c} \dot{\xi}_{II} = M_{II} g(y_{II}, t)^T \Big(p1(y_{II}, t) + g(y_{II}, t)\hat{\eta}_{II} \Big) - \frac{M_{II}}{2} \Big(- \dot{\upsilon}_{c3}(x_1^2 + x_2^2) + 2f(\dot{\upsilon}_{c1}x_1 + \dot{\upsilon}_{c2}x_2)) \Big) \\ - \frac{p2(y_{II},t)\hat{\eta}_{II} + \upsilon_{c3}\hat{\eta}_{II}^2}{||g(y_{II},t)||} + \frac{g(y_{II},t)^T\hat{\eta}_{II}}{||g(y_{II},t)||^3} \Big(\upsilon_{c3}(p1(y_{II}, t) + g(y_{II}, t)^T\hat{\eta}_{II}) + \begin{bmatrix} -f\dot{\upsilon}_{c1} + x_1\dot{\upsilon}_{c3} \\ -f\dot{\upsilon}_{c2} + x_2\dot{\upsilon}_{c3} \end{bmatrix} \Big) \end{array}$$

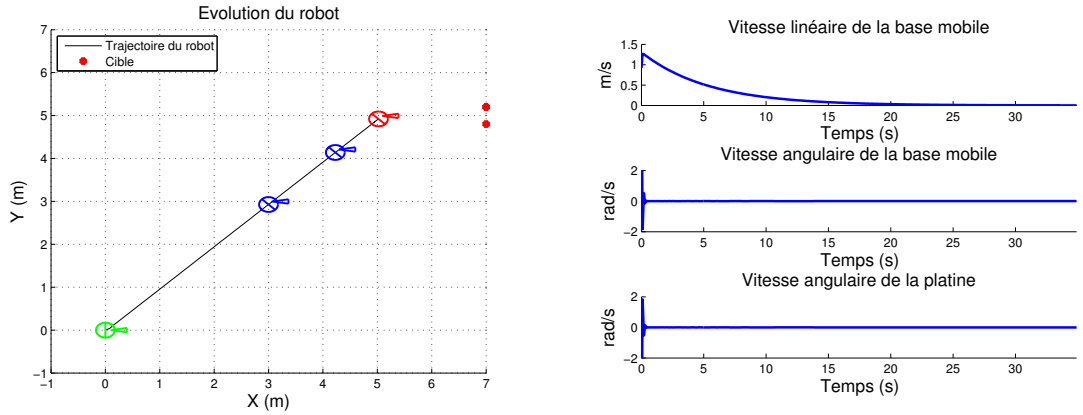
L'observateur proposé est basé sur les informations visuelles ainsi que sur la connaissance du torseur cinématique de la caméra. Contrairement à l'observateur (3.122), la profondeur estimée n'est pas rebouclée dans l'observateur. Il est donc possible que l'observateur minimal

soit davantage susceptible de diverger que son homologue non minimal. De plus, pour notre système robotique, les paramètres v_{c1}, ω_{c2} et ω_{c3} apparaissant dans l'observateur sont nuls. La quantité d'information utilisée pour l'estimation de la profondeur n'est alors pas suffisante. En effet, lors d'un asservissement visuel 2D, la rotation de la platine compense souvent la rotation de la base mobile, c'est-à-dire $\omega_{c1} = 0$. L'estimation de la profondeur est alors réalisée à partir de la simple connaissance des indices visuels, de v_{c2} et de v_{c3}. Dans ces conditions, une estimation correcte de la profondeur en utilisant l'observateur non linéaire minimal proposé, semble compromise.

3.3.3 Simulations de l'estimation par observateurs non linéaires

Dans cette section, nous réalisons une série de simulations ayant pour but d'établir une étude comparative des différents processus d'estimation préalablement présentés. Dans un premier temps les deux observateurs non linéaires sont testés lors d'un asservissement visuel dans un contexte parfait. Lors de la seconde navigation les données utilisées seront bruitées.

Pour cette première simulation, la trajectoire du robot est présentée sur la figure 3.11(a). Les gains de l'observateur non minimal sont fixés à $k_{nm1} = k_{nm2} = 1$ et $k_{nm3} = 6$. La valeur initiale de l'état de l'observateur est choisie comme suit : $\hat{x}_1(0) = x(0)$, $\hat{x}_2(0) = y(0)$ et $\hat{x}_3(0) = 0.112$ pour l'observateur non minimal et $M_{II} = 0.2$ et $\xi_{II}(0) = -1.6$ pour l'observateur minimal. Pour les deux observateurs, les termes $\dot{\xi}_{II}$ et $\dot{\hat{x}}_{nm}$ sont intégrés à l'aide de la méthode numérique Runge-Kutta d'ordre 4 [Dormand & Prince 1980]. La figure 3.12 représente l'estimation de la profondeur de l'un des quatre indices visuels à l'aide des deux observateurs. Les deux processus convergent vers la valeur réelle. Cependant, le temps de convergence est plus faible pour l'observateur non minimal.



(a) Trajectoire du robot dans la scène

(b) Vitesses du système robotique

FIGURE 3.11 – Navigation référencée image

Dans la seconde simulation les conditions de navigation étant les mêmes que précédemment, la trajectoire du robot est identique à celle présentée sur la figure 3.11(a). Nous avons introduit un bruit aléatoire de 1 pixel pour les indices visuels et de 3% pour les commandes, ainsi qu'un biais de 1% sur la valeur de la focale (voir les simulations de la section 3.2.1 pour plus de détails sur les bruits). Les différents gains ainsi que les valeurs initiales sont les mêmes que ceux utilisés lors de la première simulation.

La figure 3.13(a) présente l'estimation de la profondeur d'un indice visuel par l'observateur

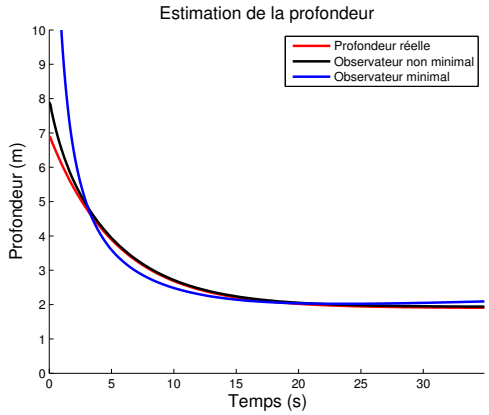
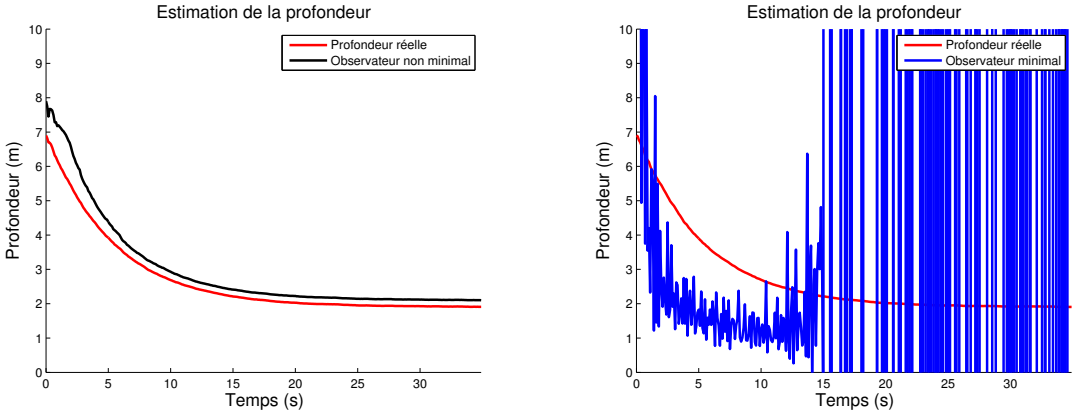


FIGURE 3.12 – Estimation de la profondeur par les observateurs non linéaires avec des donnés non bruitées

non minimal. Globalement l'estimée possède la même dynamique que la valeur réelle mais fait apparaître une erreur conservée tout au long de la navigation. Cela signifie que le terme correctif $\beta_{nm}(\hat{x}, y, T)$ n'est plus assez influent à la fin de l'estimation et que seul $\alpha_{nm}(\hat{x}, y)T$ recopie correctement la dynamique. Une solution consiste à augmenter la valeur de k_{nm3}, mais la sensibilité aux bruits est alors augmentée.

La figure 3.13(b) présente l'estimation de la profondeur par l'observateur minimal. Dans la première partie de la navigation, le processus d'estimation est très sensible aux bruits de mesure tandis que dans la seconde partie, l'estimation diverge. Dans cette seconde phase le robot se déplaçant très lentement, ce qui est caractéristique de la décroissance exponentielle, toutes les composantes du torseur de la caméra sont quasiment nulles. Le signal utilisé pour l'estimation devient donc très faible en comparaison des bruits de mesure. Cette situation explique la divergence du processus. Cet observateur ne convenant pas à l'estimation de la profondeur dans le cadre d'une navigation référencée vision, seuls l'observateur non minimal et la paire prédicteur/correcteur font l'objet d'une étude comparative.

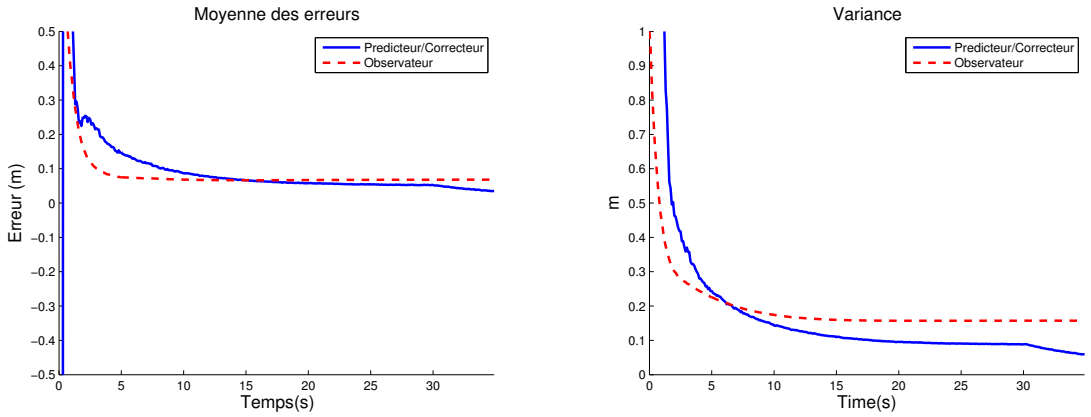


(a) Observateur non minimal (b) Observateur minimal

FIGURE 3.13 – Observateurs non linéaires avec des donnés bruitées

3.3.4 Etude comparative

Dans cette section, la profondeur est estimée lors d'une navigation visuelle effectuée dans les conditions présentées dans la partie 3.3.3. Afin de comparer les processus d'estimation, les tâches sont rejouées 100 fois [8] avec les mêmes paramètres. Nous proposons d'utiliser comme critères de comparaison la moyenne des erreurs, la variance ainsi que le temps de convergence moyen à 10% sur l'ensemble des réalisations. Pour la première simulation, les deux processus sont réglés avec un jeu de paramètres correctement adaptés à la situation. Nous choisissons donc $n = 200$ pour le prédicteur/correcteur et $k_{nm1} = k_{nm2} = 1$, $k_{nm3} = 6$, $\hat{x}_1(0) = x(0)$, $\hat{x}_2(0) = y(0)$ et $\hat{x}_3(0) = 0.112$ pour l'observateur. Les performances des deux méthodes, présentées sur les figures 3.14(a) et 3.14(b), sont comparables. Pour ces simulations, le temps moyen de convergence est de $3.8s$ pour le prédicteur/correcteur et de $8.8s$ pour l'observateur. Notre choix se porte donc sur le prédicteur/correcteur qui possède un temps de convergence moyen deux fois plus rapide que celui de l'observateur. En effet, une estimation rapide de la profondeur étant nécessaire pour une bonne gestion des occultations, le prédicteur/correcteur semble le mieux approprié à notre cas.



(a) Erreur moyenne

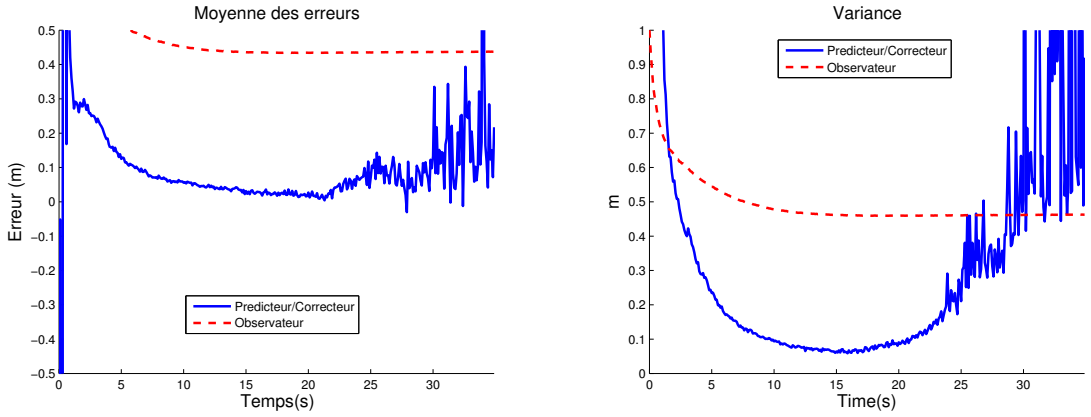
(b) Variance

FIGURE 3.14 – Performance des estimations avec réglage favorable

Lors de la seconde série de simulations, les estimateurs sont réglés défavorablement avec $n = 30$ pour le prédicteur/correcteur et $k_{nm1} = k_{nm2} = 2$, $k_{nm3} = 4$, $\hat{x}_1(0) = x(0)$, $\hat{x}_2(0) = y(0)$ et $\hat{x}_3(0) = 0.112$ pour l'observateur. Nous pouvons voir sur les figures 3.15(a) et 3.15(b) que les performances de l'observateur se sont grandement dégradées. Les réglages appliqués lors de ces simulations ne sont toutefois pas très différents de ceux proposés lors des simulations précédentes. Ceci illustre la difficulté de trouver un jeu de gains k_{nm1}, k_{nm2} et k_{nm3} permettant d'estimer correctement la profondeur. Pour le prédicteur/correcteur, la diminution du nombre d'images utilisées pour l'estimation engendre un rapport signal/bruit plus faible que celui des simulations précédentes. Ce phénomène est accentué à la fin de la navigation lorsque le robot se déplace très lentement, c'est-à-dire lorsque les images utilisées sont très semblables. La variance de la figure 3.15(b) illustre parfaitement ce problème : alors qu'elle n'est pas trop importante au milieu de la navigation, elle augmente fortement lorsque le robot ralentit 3.11(b). Cependant ce problème est facilement résolu par l'utilisation d'un nombre d'images plus important. Le prédicteur/correcteur est donc très facilement réglable : il faut choisir n le plus grand possible

8. Après différents tests, le choix de 100 réalisations s'est avéré être représentatif.

en fonction du temps de calcul attribuable à l'estimation de la profondeur [9].



(a) Erreur moyenne

(b) Variance

FIGURE 3.15 – Performance des estimations avec réglage défavorable

Le prédicteur/correcteur semble être la méthode la mieux adaptée à l'estimation de la profondeur lors de la navigation référencée vision d'un robot mobile. En effet l'utilisation d'un nombre suffisant d'images permet une estimation correcte malgré la présence de bruit dans le signal mesuré, les commandes appliquées au robot et un calibrage imparfait de la caméra. Certes l'observateur non minimal est lui aussi robuste aux bruits ; cependant il n'offre pas un temps de convergence suffisamment faible pour gérer efficacement des occultations. Néanmoins le prédicteur/correcteur est spécifique à un système composé d'une caméra sur platine montée sur une base mobile, tandis que les observateurs non linéaires s'adaptent à tous les systèmes robotiques munis d'une caméra pinhole. De plus il est envisageable que l'exploitation d'un plus grand nombre de degrés de liberté permettrait d'améliorer la qualité de leur estimation.

3.4 Applications

3.4.1 Gestion des occultations

Comme nous avons pu le voir dans la section 3.1.2, la gestion des occultations à l'aide des équations d'évolution des indices visuels nécessite de fournir une estimation de la profondeur à l'instant où le signal visuel est perdu. Il est désormais possible de fournir cette estimation $\hat{z}_P$ dans un intervalle de temps relativement court grâce au prédicteur/correcteur utilisant n images. Dans cette section nous proposons une simulation et une expérimentation montrant l'efficacité du processus de reconstruction de la profondeur dans le contexte d'une occultation.

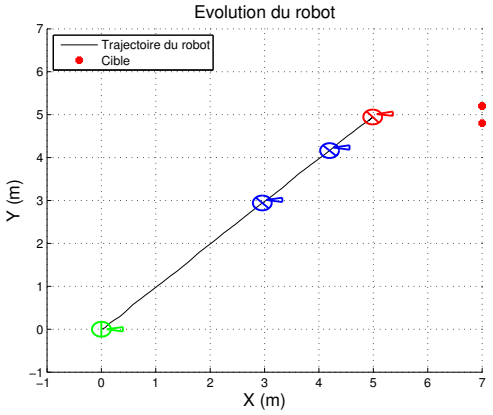
Simulation

Dans cette simulation la configuration initiale du robot est $\chi(0) = [0\ 0\ 0\ 0]^T$ et la caméra doit se situer deux mètres face à la cible à la fin de la navigation. Le nombre d'images utilisées par le prédicteur/correcteur est fixé à $n = 100$. Pour cette simulation les données utilisées aussi bien pour la commande que pour l'estimation de la profondeur sont bruitées (voir les simulations de la section 3.2.1 pour plus de détails sur les bruits).

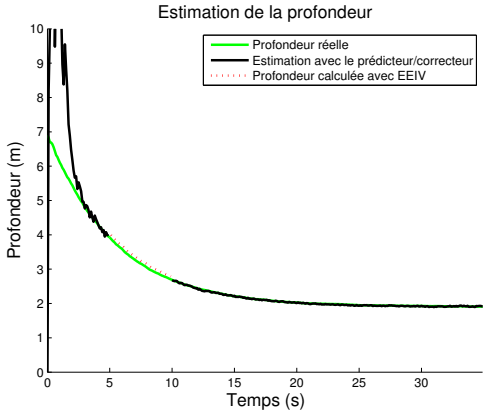
9. Le temps de calcul du prédicteur/correcteur est directement fonction du nombre n d'images utilisées.

Lors de la navigation, nous simulons une panne de caméra entre la cinquième et la dixième seconde. Le robot doit donc gérer une occultation en estimant l'évolution des indices visuels. Nous pouvons voir sur la figure 3.16(b) que la profondeur estimée par le prédicteur/correcteur est suffisamment proche de la valeur réelle au début de l'occultation pour fournir une valeur initiale de la profondeur correcte. Durant l'occultation, la profondeur (correspondant au trait en pointillé rouge) est calculée en utilisant les équations d'évolution des indices visuels. Lorsque l'occultation est terminée, le prédicteur/correcteur reprend son estimation de la profondeur en utilisant les images à sa disposition durant les n dernières itérations. Dans le cas présent, les images utilisées à la reprise de l'estimation correspondent aux 50 images comprises entre le début de la navigation et le début de l'occultation.

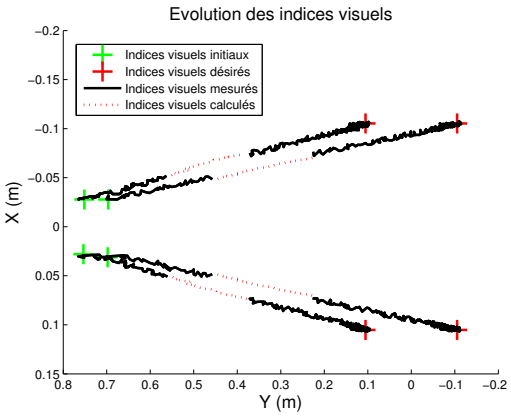
L'estimation de la profondeur rend possible le calcul de l'évolution des indices visuels durant l'occultation comme nous pouvons le voir sur la figure 3.16(c). En effet, que ce soit au début ou à la fin de l'occultation, les indices visuels calculés correspondent à ceux mesurés. De plus, comme nous pouvons le voir sur la figure 3.16(a), la trajectoire du robot durant l'occultation, comprise entre les deux robots bleus, est comparable à celle obtenue lors d'une navigation sans occultation (une telle navigation est réalisée dans la figure 3.11(a)).



(a) Trajectoire du robot dans la scène



(b) Profondeur estimée et réelle



(c) Evolution des indices visuels

FIGURE 3.16 – Navigation référencée vision avec occultation

Expérimentation

En complément des simulations, nous avons validé notre approche sur la plateforme robotique Pekee II. La cible de référence est constituée de quatre cercles rouges de diamètre 12 cm dont les centres projetés dans l'image correspondent aux indices visuels. Le robot se situe initialement à une distance de $3,5m$ et termine sa navigation à environ $1,5m$ de la cible. La période d'échantillonnage est de $200ms$ et le nombre d'images utilisées pour l'estimation est fixé à $n = 100$, ce qui correspond à un temps d'estimation d'environ $80ms$ [10] par itération.

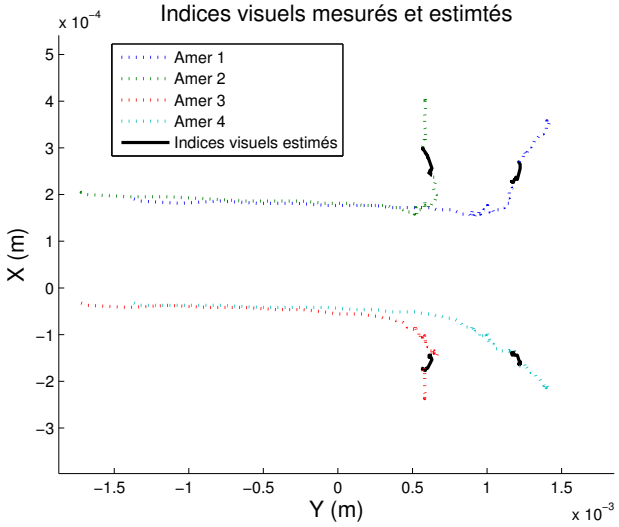


FIGURE 3.17 – Evolution des indices visuels mesurés et estimés

Au cours de la navigation, nous provoquons une occultation de la cible durant environ trois secondes. En utilisant la profondeur estimée comme valeur initiale, les indices visuels sont reconstruits à l'aide des équations d'évolution. Comme nous pouvons le voir dans la figure 3.17, les indices visuels reconstruits sont en adéquation avec ceux mesurés. Nous montrons donc ainsi qu'il est possible de gérer des occultations totales de l'amer en utilisant notre approche.

En utilisant les équations d'évolution des indices visuels et le prédicteur/correcteur pour fournir une estimation de la profondeur, il est donc possible de gérer des occultations totales en estimant les indices visuels. En utilisant cette approche, l'occultation n'a aucune influence sur la trajectoire du robot puisqu'elle est tolérée et non évitée. Bien que cette méthode ait été développée pour des indices visuels points il est tout à fait possible de l'utiliser pour d'autres types de primitives. Par exemple dans [Chaumette 1990], un segment est défini par les deux points aux extrémités. Une estimation de ces derniers permet donc de calculer l'évolution du segment. De la même manière, il est possible d'estimer l'évolution des moments discrets construits à partir de points [Chaumette 2004]. Cette méthode peut aussi être utilisée pour estimer des points caractérisant des droites, des cercles, des ellipses, des cylindres ou bien encore des sphères. Cependant les points n'étant pas les seuls paramètres définissant ces primitives [Chaumette 1990], il est nécessaire de développer une méthode permettant d'estimer les paramètres complémentaires. C'est seulement sous cette condition qu'il est envisageable d'utiliser les équations d'évolution des indices visuels pour gérer les occultations quelles que soient les primitives utilisées.

10. Le temps proposé n'a qu'une valeur indicative car aucun effort d'optimisation n'a été fait pour le réduire.

3.4.2 Calcul des indices visuels de référence s^*

Comme nous l'avons vu dans la section 2.3, un asservissement visuel 2D nécessite de connaître les indices visuels de référence s^*. Ces indices correspondent à la configuration qui doit être atteinte par la caméra à la fin de la navigation. Dans [Chaumette 2002], l'auteur identifie deux méthodes pour obtenir s^* : par apprentissage ou en fixant *a priori* la configuration de la caméra relativement à la cible. La méthode par apprentissage consiste à placer la caméra à la position finale désirée pour prendre une image de la cible. Ainsi, à l'aide du traitement d'images, il est possible d'obtenir s^*. Cette approche nous contraint donc à placer la caméra en position finale, puis en position initiale afin de démarrer l'asservissement. Bien que ne nécessitant aucune connaissance *a priori*, cette méthode apparaît peu adaptée au cadre de la navigation d'un robot mobile. Cependant elle semble bien appropriée à des bras manipulateurs devant répéter la saisie d'objets d'un même type. En effet, une seule phase d'apprentissage est nécessaire pour une série de prise et de dépose d'objet. La seconde solution pour obtenir s^* consiste à fixer la configuration finale de la caméra par rapport à la cible, puis à projeter les points caractéristiques de la cible dans l'image. Contrairement à l'approche précédente, aucun déplacement de la caméra n'est requis, mais un modèle de la cible est nécessaire. En première analyse, cette seconde solution semble plus appropriée à la navigation car elle ne nécessite pas d'amener le robot en position désirée. Cependant elle limite l'autonomie du système car elle requiert un modèle de la cible constitué des points caractéristiques et de leur position relative. Afin de relâcher cette dernière contrainte, nous proposons d'utiliser la connaissance de la profondeur des indices visuels pour calculer les indices visuels de référence s^*.

Pour cela, nous allons chercher à estimer les coordonnées de la cible dans le repère initial du robot noté R_O. Nous serons alors en possession d'un modèle de la cible. Ce dernier sera ensuite utilisé pour calculer s^* pour une configuration finale donnée du robot dans R_O à partir des équations de projection perspective. A l'aide des différentes matrices de passage homogènes nécessaires (cf. section 2.1.2), nous pouvons écrire :

$$\underline{P} = H_{I/C} * H_{P/C}^{-1} * H_{M/P}^{-1} * H_{O/M}^{-1} * \underline{p} \tag{3.127}$$

avec

$$H_{I/C} = \begin{bmatrix} \frac{f}{z_P} & 0 & 0 & 0 \\ 0 & \frac{f}{z_P} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.128}$$

où $\underline{p} = [x_P^O \ y_P^O \ z_P^O \ 1]^T$ désigne les coordonnées homogènes d'un point caractéristique de la cible dans le repère R_O et $\underline{P} = [X_P \ Y_P \ z_P \ 1]^T$ les coordonnées homogènes de l'indice visuel correspondant. L'équation (3.127) ne peut être utilisée que lorsque z_P, qui est nécessaire au calcul de $H_{I/C}$, est connue. Cependant, les coordonnées de p ne sont parfaitement calculées que si les mesures ne sont pas bruitées. Cette hypothèse n'étant pas réalisable en contexte expérimental, il est nécessaire de multiplier les informations pour améliorer l'estimation de p. Nous proposons de considérer plusieurs images offrant un point de vue différent d'une même cible. Ensuite, nous utiliserons les moindres carrés pour déterminer p. Notons n_{s^*} le nombre

d'images exploitées. Notre problème d'estimation peut être réécrit comme suit :

$$IM_p = \Theta_p * p + R_p \tag{3.129}$$

avec

$$IM_p = \begin{bmatrix} P(k) \\ \vdots \\ P(k - n_{s^*} + 1) \end{bmatrix} \qquad \Theta_p = \begin{bmatrix} \Theta(k) \\ \vdots \\ \Theta(k - n_{s^*} + 1) \end{bmatrix} \qquad R_p = \begin{bmatrix} R(k) \\ \vdots \\ R(k - n_{s^*} + 1) \end{bmatrix}$$

$$\Theta(k-w) = \begin{bmatrix} 0 & 0 & \frac{-f}{z_P} \\ [-\cos(\theta)\sin(\vartheta) - \sin(\theta)\cos(\vartheta)]\frac{f}{z_P} & [-\sin(\theta)\sin(\vartheta) + \cos(\theta)\cos(\vartheta)]\frac{f}{z_P} & 0 \\ \cos(\theta)\cos(\vartheta) - \sin(\theta)\sin(\vartheta) & \sin(\theta)\cos(\vartheta) + \cos(\theta)\sin(\vartheta) & 0 \end{bmatrix}$$

et

$$R(k-w) = \begin{bmatrix} \frac{h_r f}{z_P} \\ \{-[-M_x\cos(\theta) - M_y\sin(\theta)]\sin(\vartheta) + [M_x\sin(\theta) - M_y\cos(\theta)]\cos(\vartheta) + D_x\sin(\vartheta) - C_y\}\frac{f}{z_P} \\ [-M_x\cos(\theta) - M_y\sin(\theta)]\cos(\vartheta) + [M_x\sin(\theta) - M_y\cos(\theta)]\sin(\vartheta) - D_x\cos(\vartheta) - C_x \end{bmatrix}$$

où $w \in [0, ..., n_{s^*} - 1]$ et les termes z_P, M_x, M_y, θ et ϑ désignent respectivement $z_P(k-w)$, $M_x(k-w)$, $M_y(k-w)$, $\theta(k-w)$ et $\vartheta(k-w)$.

La solution de l'équation (3.129) au sens des moindres carrés est donnée par :

$$p = (\Theta_p^T\Theta_p)^{-1}\Theta_p^T(IM_p - R_p) \tag{3.130}$$

En utilisant l'estimation des coordonnées p donnée par l'équation (3.130) il est possible de calculer les indices visuels de référence. Pour cela, nous utilisons l'équation (3.127) où z_P, M_x, M_y, θ et ϑ prennent les valeurs désirées pour la configuration finale désirée dans le repère R_O. Il n'est donc plus nécessaire de déplacer préalablement le robot ou d'avoir un modèle *a priori* de la cible. Dans la section 3.4.4, cette approche a été validée expérimentalement.

Remarque : *Le robot n'étant pas équipé de capteur pouvant mesurer la profondeur des indices visuels, il est difficile de détecter la convergence du processus d'estimation de z. Une solution consiste à utiliser l'estimation des indices visuels de référence. Alors que la valeur réelle de la profondeur est différente à chaque itération, les indices visuels de référence, qui correspondent à une configuration fixe du robot, sont identiques à chaque instant. s^* étant calculé à partir de z, la convergence de l'estimation des premiers correspond à la convergence de l'estimation de la seconde.*

3.4.3 Calcul de la matrice d'interaction

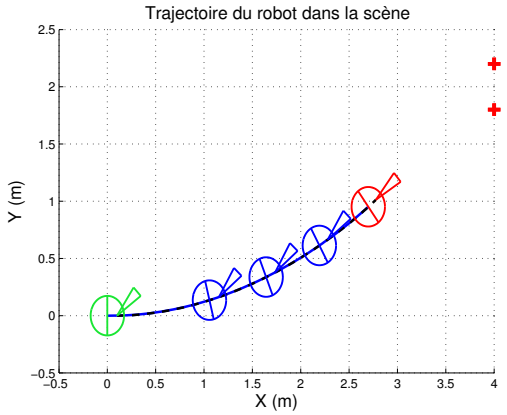
Comme nous l'avons vu dans la section 2.2.2, la matrice d'interaction L_{Point} peut être calculée pour la situation courante ou pour la situation finale désirée. La profondeur étant estimée, il est désormais possible de calculer la valeur exacte de L_{Point} (aux bruits et incertitudes près).

Cependant, l'auteur de [Chaumette 1998] a montré qu'utiliser z_P^* au lieu de z_P permet d'éviter certains problèmes de minima locaux.

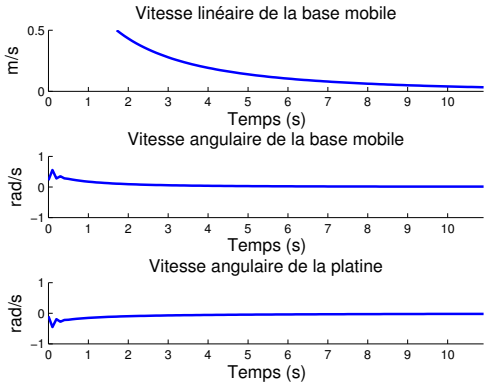
Simulation

Afin d'illustrer le problème relatif aux minima locaux soulevé dans [Chaumette 1998], nous proposons deux simulations d'asservissement visuel. Dans la première simulation la matrice d'interaction est calculée avec les valeurs réelles tandis que dans la seconde elle est déterminée à partir des valeurs de référence. Pour les deux simulations, la position initiale du robot est $\chi(0) = [0\ 0\ 0\ \frac{\pi}{4}]^T$ et la position finale désirée de la caméra est $\chi_C = [3.07\ 2.92\ -\frac{\pi}{4}]^T$. La période d'échantillonnage est de $T_e = 100ms$ tandis que les gains de commande sont respectivement fixés à $\lambda_{av} = 0.15$ et $\lambda_{av} = 0.35$ pour la seconde. Enfin, toutes les données concernant les images, l'odométrie et le calibrage de la caméra sont parfaitement connues. Nous nous plaçons donc en contexte non bruité.

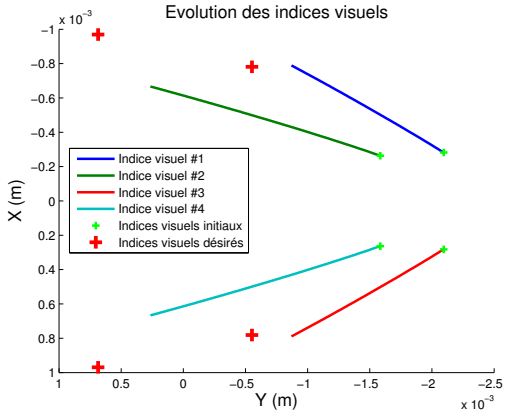
Sur la figure 3.18(a), nous pouvons voir que la caméra n'atteint pas la configuration désirée. En effet, les vitesses du robot sont nulles (figure 3.18(b)) alors que l'erreur dans l'image n'est toujours pas régulée à zéro (figures 3.18(c) et 3.18(d)). Nous sommes donc dans le cas où un minimum local a été atteint.



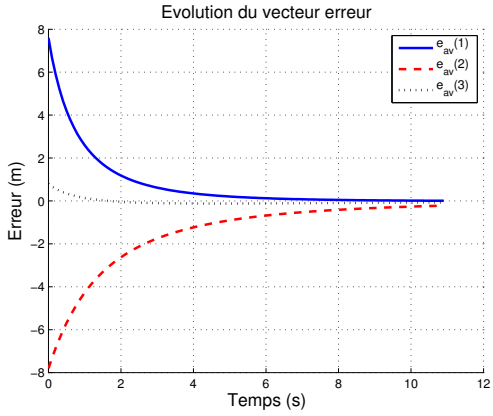
(a) Trajectoire du robot dans la scène



(b) Vitesses du système robotique



(c) Evolution des indices visuels

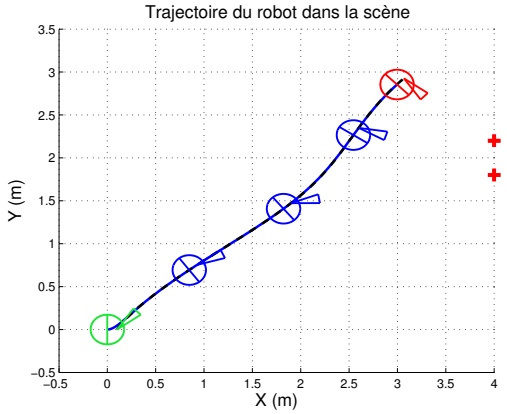


(d) Evolution du vecteur erreur

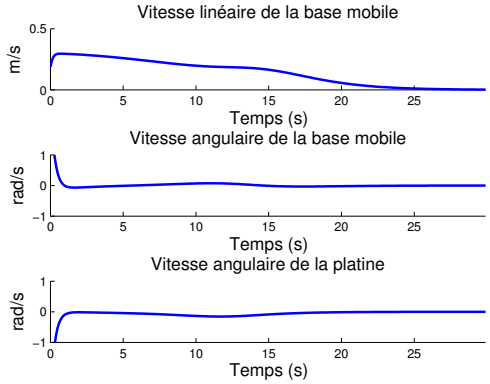
FIGURE 3.18 – Asservissement visuel avec $L(X_P(t), Y_P(t), z_P(t))$

Contrairement à la première simulation, dans la figure 3.19(a) la caméra atteint la configuration

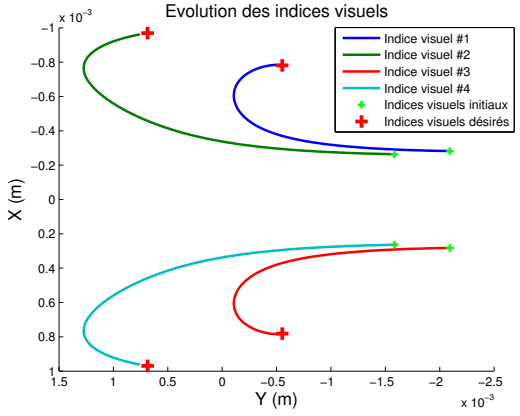
désirée à la fin de la navigation. L'erreur dans l'image est annulée (figures 3.19(c) et 3.19(d)) et aucun minimum local n'est atteint durant l'asservissement visuel (figure 3.19(b)). Cela ne signifie pas qu'il n'existe pas de minimum local, mais que celui-ci est évité lorsque la matrice d'interaction est calculée avec les valeurs de référence.



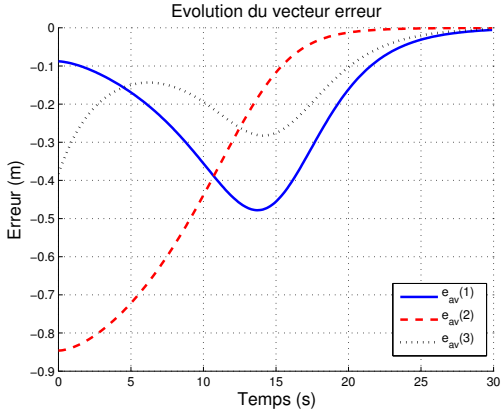
(a) Trajectoire du robot dans la scène



(b) Vitesses du système robotique



(c) Evolution des indices visuels



(d) Evolution du vecteur erreur

FIGURE 3.19 – Asservissement visuel avec $L(X_P^*, Y_P^*, z_P^*)$

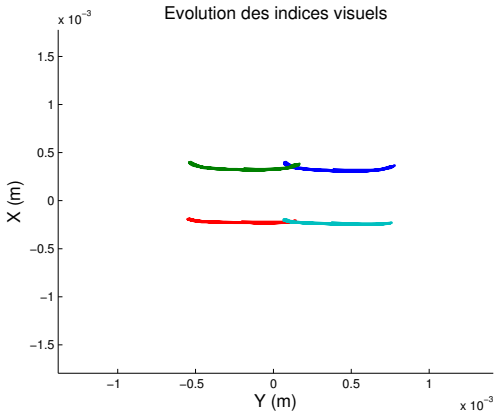
3.4.4 Phase d'initialisation

Dans les sections précédentes nous avons constaté la nécessité de connaître la profondeur pour réaliser un asservissement visuel, notamment pour gérer les occultations. Cependant le processus d'estimation de la profondeur nécessite un certain temps pour converger vers la valeur réelle. Durant ce laps de temps, l'apparition d'une occultation peut faire échouer la navigation, étant donné qu'aucune valeur correcte de la profondeur n'est disponible. Une solution pour pallier ce problème consiste en une phase d'initialisation durant laquelle le robot estime la profondeur en ne réalisant que des rotations sur lui-même. Le choix des rotations est motivé par deux raisons : premièrement le processus d'estimation nécessite des images prises depuis des points de vue différents, même s'ils sont très proches, et secondement, en ne réalisant que des rotations le robot limite tout risque de collision avec un obstacle.

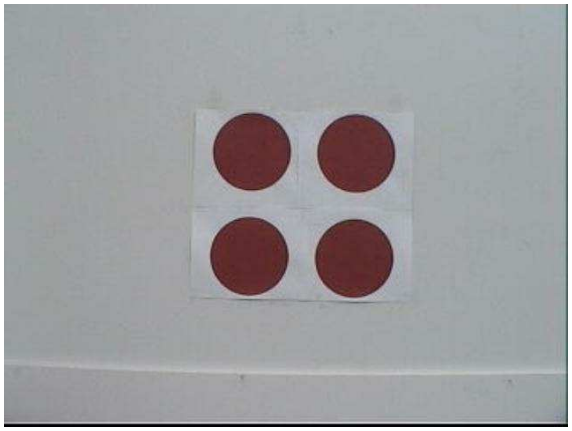
Nous proposons donc une phase d'initialisation préalable à la navigation durant laquelle le robot estime la profondeur des indices visuels à l'aide des différentes images. Une fois la profondeur estimée, il est possible de calculer les indices visuels de référence s^* et de fournir immédiatement une valeur initiale au processus de gestion des occultations. Les chances de succès de la navigation sont ainsi significativement augmentées.

Expérimentation

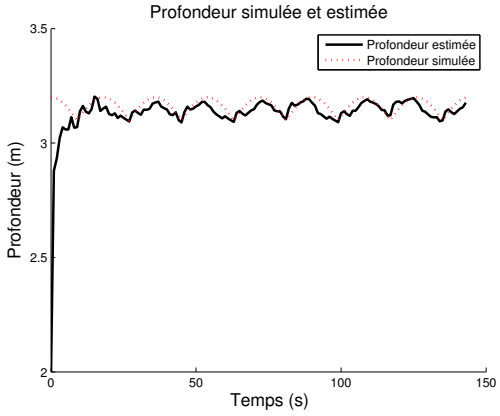
Nous avons réalisé une phase d'initialisation sur le robot Pekee II. Pour cela, nous avons considéré une cible composée de quatre cercles rouges de $12cm$ de diamètre. Le robot est initialement positionné à $3,1m$ en face de la cible. La caméra fournit des images de 480×640 pixels avec une focale estimée à $3,54$ mm (cf. 3.20(b)). Enfin, le prédicteur/correcteur utilisé exploite $n = 100$ images.



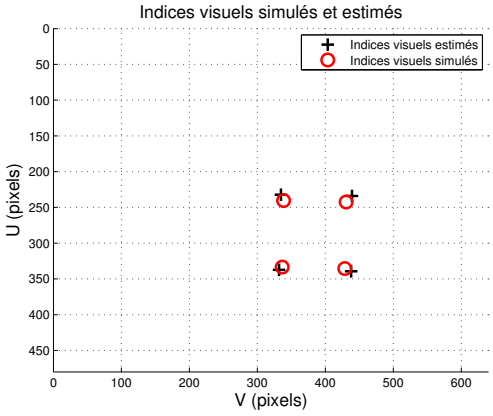
(a) Evolution des indices visuels



(b) Exemple de prise de vue



(c) Profondeur estimée et simulée



(d) Indices visuels de références estimés et simulés

FIGURE 3.20 – Exemple de phase d'initialisation

Afin d'obtenir les images nécessaires à la phase d'initialisation, le robot et la platine tournent sur eux-mêmes de manière à ce que la caméra effectue une rotation de -90° à 90°. Le déplacements des indices visuels correspondants dans l'image est totalement cohérent avec ce mouvement (cf. 3.20(a)). A partir des $n = 100$ dernières images (cf. 3.20(b)), le prédicteur/correcteur calcule la profondeur (cf. 3.20(c)). Sur cette base, en appliquant la relation (3.130), il est possible de déterminer les indices visuels de référence (cf. 3.20(d)) correspondant à la situation désirée pour la caméra. Les figures 3.20(c) et 3.20(d) présentent une bonne cohérence entre

les données simulées et les résultats expérimentaux, ce qui montre la validité de l'approche proposée.

Nous avons ainsi proposé une méthode permettant de calculer les indices visuels de référence sans apprentissage ni connaissance *a priori* d'un modèle de la cible. Nous avons de plus montré qu'il peut être intéressant d'introduire une phase d'initialisation permettant d'obtenir une valeur correcte de la profondeur. De cette manière, il est possible de gérer efficacement une occultation survenant dès le début de la mission et d'accroître l'autonomie du robot.

3.5 Conclusion

Dans ce chapitre nous nous sommes attachés à développer un processus d'estimation de la profondeur des indices visuels. Celui-ci est basé sur une paire prédicteur/ correcteur exploitant un nombre d'images paramétrable par l'utilisateur. Il est ainsi possible d'obtenir une estimée précise malgré la présence de bruit sur les mesures visuelles et sur l'odométrie. Cette estimée est essentiellement utilisée comme valeur initiale dans l'algorithme de reconstruction des indices visuels proposé dans [Folio 2007]. De plus, comme nous l'avons vu, elle peut être exploitée pour d'autres applications : calcul automatique des indices visuels de référence, détection de la convergence, ... Les résultats obtenus en simulation et expérimentation montrent que le couplage des méthodes d'estimation et de reconstruction permet de gérer efficacement les occultations.

Nous avons donc étendu les travaux proposés dans [Folio 2007] de manière à obtenir une méthode complète permettant de traiter la perte du signal visuel. Cependant les occultations ne sont pas les seules contraintes à prendre en compte lors de la réalisation d'une tâche de navigation référencée vision dans un environnement encombré. Il faut de plus être capable de gérer les problèmes de collision et d'effectuer de longs déplacements. C'est à ces deux problèmes que nous nous intéressons dans le chapitre suivant.

CHAPITRE 4

NAVIGATION PAR ASSERVISSEMENT VISUEL EN ENVIRONNEMENT ENCOMBRÉ

Sommaire

Dans ce chapitre, notre objectif est de réaliser une navigation référencée vision nécessitant de grands déplacements dans un environnement peu connu. Pour cela, il est nécessaire d'une part de traiter les occultations et les collisions et d'autre part d'accroître la portée de la navigation. Disposant de techniques permettant de tolérer les pertes du signal visuel, nous nous attachons ici à résoudre les deux autres problèmes mentionnés.

Dans un premier temps, nous considérons le problème de l'évitement d'obstacles pendant la réalisation d'une tâche référencée vision. Ce problème a été au coeur de travaux antérieurs réalisés au LAAS [Cadenat 1999] [Souères & Cadenat 2003] [Folio 2007] que nous rappellerons brièvement et sur lesquels nous nous appuierons fortement pour proposer notre solution. Ces méthodes consistent à réaliser un suivi de chemin sur la seule base d'informations fournies par le télémètre laser. De la même manière que pour l'asservissement en position, nous utiliserons donc une méthode locale. Dans un second temps, nous nous focaliserons sur l'augmentation de la portée de la navigation qui constitue ici notre contribution essentielle. En effet, les lois de commande utilisées pour la navigation ne requièrent pas de connaissances *a priori* sur l'environnement, et sont calculées sur la base de données acquises au cours de la navigation. Il n'est donc pas possible de réaliser de grands déplacements dans un environnement encombré avec la seule seule utilisation de ces outils. Nous proposons donc d'ajouter une carte de l'environnement qui, couplée aux méthodes précédemment présentées, offrira une portée globale à la navigation. De plus, nous verrons que l'ajout de cette carte nécessite de définir une stratégie de navigation au long cours. Pour cela, nous introduirons un algorithme de supervision garantissant le bon déroulement de tâche globale. Nous présentons donc d'abord les travaux plus anciens concernant l'évitement d'obstacles et les stratégies de commande, avant de montrer comment utiliser ces outils pour réaliser de grands déplacements par asservissement visuel 2D.

4.1 Outils pour la navigation

Dans cette section nous nous intéressons aux outils nécessaires à la navigation au long cours. Comme nous l'avons vu plus haut, il est nécessaire de donner au robot la capacité d'éviter les obstacles imprévus. Pour cela, nous proposons d'utiliser la stratégie de commande développée dans [Cadenat 1999]. Celle-ci consiste à basculer entre l'asservissement visuel et un correcteur assurant la non collision préalablement synthétisé. Afin que cette stratégie puisse être mise en place, il est nécessaire de savoir sélectionner le correcteur adéquat au bon moment et de garantir la continuité de la loi de commande lors de la transition entre les correcteurs. Nous rappelons d'abord les travaux concernant l'évitement d'obstacle puis nous présentons les méthodes permettant d'enchaîner les différentes lois envisagées.

4.1.1 Evitement d'obstacle

La méthode proposée dans [Cadenat 1999] s'appuie sur le formalisme du suivi de chemin [Samson 1992]. L'idée consiste à suivre un chemin de contournement, noté ξ_0, situé à une distance d_0 de l'obstacle (cf. figure 4.1). Pour cela, seule la base mobile doit être commandée. Cependant, il est nécessaire de contrôler la caméra afin de garder les indices visuels dans le champ de vue. Ainsi, l'évitement d'obstacle terminé, il sera possible de reprendre l'asservissement visuel. Il est donc nécessaire de découpler les mouvements de la base mobile de celle de la caméra. Nous

présentons dans une première partie le correcteur associé à l'évitement d'obstacle, puis dans une seconde celui permettant de garder en vue l'amer d'intérêt.

4.1.1.1 Commande de la base mobile

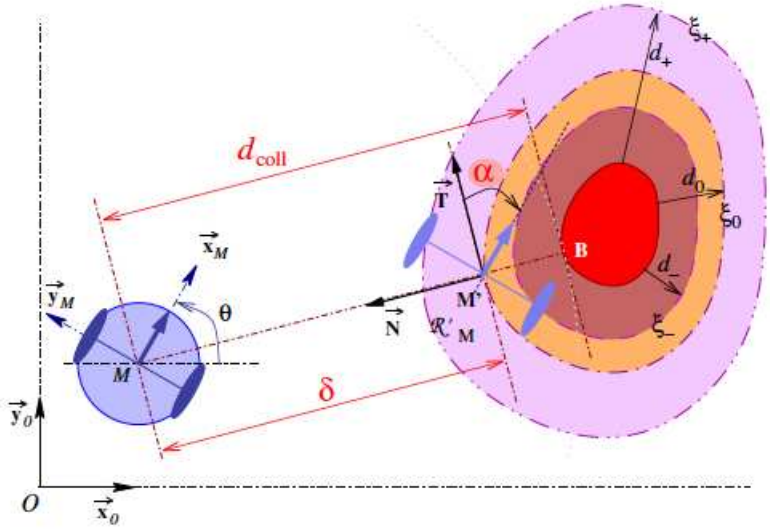


FIGURE 4.1 – Evitement d'obstacle par suivi de chemin

Dans cette section, nous nous intéressons donc au calcul des vitesses v_{coll} et ω_{coll} permettant de réaliser un évitement d'obstacle. Considérons la figure 4.1. En projetant orthogonalement le robot sur l'enveloppe ξ_0, nous définissons un repère de Frenet R'_M d'origine M'. Soient d_{coll} la distance entre le point M et l'obstacle, et α la différence, exprimée dans le repère R'_M, entre l'orientation de la platine et l'orientation de la tangente au point M'. Le robot étant équipé d'un télémètre laser, ces deux paramètres peuvent être calculés en ligne durant la navigation. Nous pouvons désormais définir l'erreur de positionnement latéral par $\delta = (d_{coll} - d_0)$ et l'erreur d'orientation par α. La dynamique du couple (δ, α) par rapport au repère R'_M s'exprime comme suit [Canudas de Wit *et al.* 1993] :

$$\begin{cases} \dot{\delta} = v_{coll} \sin(\alpha) \\ \\ \dot{\alpha} = \omega_{coll} - \frac{\Lambda v_{coll} \cos\alpha}{1+\sigma\Lambda\delta} \end{cases} \tag{4.1}$$

où $\Lambda = \frac{1}{R}$ représente la courbure de l'obstacle au point M' et R désigne son rayon de courbure. σ est défini selon le sens de contournement, de la manière suivante :

- $\sigma = +1$ si le robot contourne l'obstacle dans le sens trigonométrique.
- $\sigma = -1$ si le robot contourne l'obstacle dans le sens horaire.
- $\sigma = 0$ si le contour de l'obstacle est localement une ligne droite.

Dans le formalisme du suivi de chemin [Samson 1992], l'objectif est d'assurer la convergence géométrique du centre M du véhicule vers le chemin à suivre ξ_0 indépendamment de la vitesse linéaire du robot. Le but est donc de déterminer une loi de commande ω_{coll} permettant d'amener le couple (δ, α) à zéro, étant donné une vitesse linéaire non nulle définie a priori. Pour cela, nous nous appuyons sur les techniques de régime glissant et considérons la variable

de glissement γ suivante :

$$\gamma = \delta + \kappa\alpha \ , \qquad \forall \kappa \in \mathbb{R}^* \tag{4.2}$$

où κ est un gain scalaire non nul qui permet de déterminer la vitesse relative de convergence de δ et α. On peut montrer que la convergence de γ à zéro entraîne celle du couple (δ, α), répondant ainsi au problème du suivi de chemin [Souères *et al.* 1998] [Cadenat 1999]. Cependant, la régulation à zéro de γ ne permettant de fixer que ω_{coll}, nous imposons de plus au robot de suivre l'enveloppe de contournement ξ_0 avec une vitesse constante $v_0 \neq 0$ prédéfinie. Pour cela, nous chercherons à réguler à zéro l'erreur $(l - v_0 t)$, où l représente l'abscisse curviligne du centre M du robot dans le repère R_M défini à l'instant auquel débute l'évitement, ce qui conduit à définir la fonction de tâche e_{coll} suivante [Souères & Cadenat 2003] :

$$e_{coll} = \begin{pmatrix} l - v_0 t \\ \delta + \kappa\alpha \end{pmatrix} \tag{4.3}$$

En dérivant e_{coll} par rapport au temps, il vient :

$$\dot{e}_{coll} = \begin{pmatrix} v_{coll} - v_0 \\ \dot{\delta} + \kappa\dot{\alpha} \end{pmatrix} \tag{4.4}$$

A l'aide de la relation (4.1), $\dot{e}_{coll}$ peut se réécrire comme suit :

$$\dot{e}_{coll} = J_{coll} \begin{pmatrix} v_{coll} \\ \omega_{coll} \end{pmatrix} + A = \begin{pmatrix} 1 & 0 \\ \sin(\alpha) - \frac{\kappa\Lambda\cos\alpha}{1+\sigma\Lambda\delta} & \kappa \end{pmatrix} \begin{pmatrix} v_{coll} \\ \omega_{coll} \end{pmatrix} + \begin{pmatrix} -v_0 \\ 0 \end{pmatrix} \tag{4.5}$$

Notons que $\det(J_{coll}) = \kappa$ et donc que la tâche est ρ-admissible pour $\kappa \neq 0$. Nous désirons fixer une décroissance exponentielle telle que :

$$\dot{e}_{coll} = -\lambda_{coll} e_{coll} \tag{4.6}$$

A partir des équations (4.5) et (4.6), nous obtenons la commande d'évitement d'obstacle suivante [Souères & Cadenat 2003] :

$$\dot{q}_{base} = \begin{pmatrix} v_{coll} \\ \omega_{coll} \end{pmatrix} = J_{coll}^{-1}(-\lambda_{coll} e_{coll} - A) \tag{4.7}$$

4.1.1.2 Commande de la platine

Dans la section précédente, nous nous sommes intéressés au calcul des vitesses v_{coll} et ω_{coll} permettant de contourner un obstacle. Cependant, cette méthode ne tient pas compte de la présence de la platine portant la caméra. Nous rappelons que le correcteur réalisant la navigation utilise des informations visuelles relatives à un amer. Il est donc primordial de garder l'amer dans le champ de vue de la caméra lors de l'évitement d'obstacle. Ainsi, les informations visuelles requises par la navigation restent disponibles durant toute la phase de contournement. Dans cette section, nous nous intéressons donc au calcul de la vitesse ϖ_{coll} permettant de garder l'amer en vue lors d'un évitement d'obstacle.

Notre caméra ne réalisant des déplacements que dans le plan horizontal, nous proposons de ne considérer que les coordonnées Y des points dans l'image. De plus, cherchant à contrôler un seul degré de liberté, il est nécessaire que la fonction de tâche soit un scalaire pour garantir la ρ-admissibilité. Nous définissons donc la fonction de tâche e_{pl} comme suit [Cadenat 1999] :

$$e_{pl} = Y_0 \tag{4.8}$$

où Y_0 correspond à la coordonnée selon l'axe Y de l'image du centre de gravité des points caractérisant l'amer. Ainsi la régulation à zéro de e_{pl} permet de centrer l'amer dans l'image, garantissant ainsi sa présence dans le champ de vue de la caméra. En dérivant e_{pl} par rapport au temps, nous obtenons :

$$\dot{e}_{pl} = \begin{pmatrix} -\frac{f}{z_0} & \frac{Y_0}{z_0} & f + \frac{Y_0^2}{f} \end{pmatrix} T_r = L_{Y_0} T_r \tag{4.9}$$

avec z_0 la profondeur du centre de gravité. Il faut noter que la matrice d'interaction L_{Y_0} n'est qu'une approximation. En effet, la matrice d'interaction associée au centre de gravité se déduit des matrices d'interactions associées aux moments de l'image [Chaumette 2004].

Afin d'imposer une décroissance exponentielle à e_{pl}, nous posons :

$$\dot{e}_{pl} = -\lambda_{pl} e_{pl} \tag{4.10}$$

où λ_{pl} est un scalaire positif. Il reste désormais à déterminer la commande de la platine satisfaisant cette dynamique. Pour cela, nous partitionnons la matrice jacobienne J_r de manière à séparer les termes relatifs à la base de ceux de la platine. Nous obtenons alors :

$$T_r = \begin{pmatrix} J_{base} & J_{pl} \end{pmatrix} \begin{pmatrix} \dot{q}_{base} \\ \dot{q}_{pl} \end{pmatrix} \tag{4.11}$$

où

- J_{base} désigne la matrice constituée des deux premières colonnes de la matrice jacobienne J_r, correspondant aux mouvements de la base mobile.
- J_{pl} est donnée par la troisième colonne de la matrice jacobienne J_r, correspondant au mouvement de la platine.
- $\dot{q}_{base} = (\upsilon_{coll}\ \omega_{coll})^T$ représente les vitesses linéaire et angulaire de la base lors de la phase de contournement.
- $\dot{q}_{pl}$ désigne la commande de la platine lors de la phase d'évitement, c'est-à-dire la vitesse de rotation ϖ_{coll}.

En considérant les équations (4.9), (4.10) et (4.11), nous pouvons établir :

$$L_{Y_0} J_{base} \dot{q}_{base} + L_{Y_0} J_{pl} \dot{q}_{pl} = -\lambda_{pl} e_{pl} \tag{4.12}$$

Nous en déduisons l'expression de la commande de la platine permettant de centrer la cible dans l'image lors de la phase d'évitement [Cadenat 1999] :

$$\dot{q}_{pl} = -\frac{1}{L_{Y_0} J_{pl}} \left(\lambda_{pl} e_{pl} + L_{Y_0} J_{base} \dot{q}_{base} \right) \tag{4.13}$$

Ainsi, la commande de la platine intègre l'erreur de suivi de l'amer visuel, tout en compensant les déplacements de la base mobile en phase d'évitement. De cette manière, la conservation de l'amer visuel dans l'image est garantie pendant la phase de contournement. Le vecteur de commande $\dot{q}_{eo}$ permettant de contourner un obstacle tout en gardant l'amer d'intérêt dans le champ de vue de la caméra est défini comme suit [Cadenat 1999] :

$$\dot{q}_{eo} = \begin{pmatrix} \dot{q}_{base} \\ \dot{q}_{pl} \end{pmatrix} \tag{4.14}$$

Remarque : *De la même manière que pour l'asservissement visuel 2D, lors d'un évitement d'obstacle avec occultation, la caméra est commandée à l'aide d'indices visuels calculés à partir des équations d'évolution.*

4.1.2 Transition entre les correcteurs

Nous sommes maintenant en possession de deux correcteurs permettant de réaliser une navigation en environnement encombré : le premier, $\dot{q}_{av}$, utilisé pour un asservissement visuel 2D positionnant le robot à la situation désirée, le second $\dot{q}_{eo}$, pour contourner les obstacles. Ces deux lois de commande sont indépendantes et ne peuvent être utilisées simultanément. Afin de sélectionner le correcteur adapté à la situation, il est nécessaire de définir les événements-clés intervenant dans le déroulement de la tâche (risques de collision, d'occultation, etc). De plus, il n'est pas possible de commuter sans chercher à assurer la continuité des vitesses. Afin de répondre à ces attentes, deux méthodes de transition ont été développées : la combinaison convexe [Cadenat 1999] et le séquencement dynamique [Souères & Cadenat 2003]. Dans cette section, nous proposons dans un premier temps une présentation des deux méthodes, puis nous réalisons une étude comparative entre celles-ci.

4.1.2.1 Enchaînement par combinaison convexe

La combinaison convexe consiste à concevoir un correcteur global $\dot{q}_{global}$ à l'aide de n_c correcteurs élémentaires et indépendants $\dot{q}_i$, avec $i \in [0, ..., n_c - 1]$. Pour cela, $\dot{q}_{global}$ est calculé comme suit :

$$\dot{q}_{global} = \sum_{i=0}^{n_c-1} \lambda_i \dot{q}_i \tag{4.15}$$

où $0 < \lambda_i < 1$ et $\sum \lambda_i = 1$. λ_i sont des variables permettant d'activer les n_c correcteurs. Ces variables étant généralement calculées de manière à assurer la continuité de la commande lors du basculement, celle-ci est garantie par construction.

Dans notre cas, nous cherchons à définir un correcteur global $\dot{q}_{nav}$ réalisant la fusion des correcteurs élémentaires $\dot{q}_{av}$ et $\dot{q}_{eo}$. A partir de l'équation (4.15), nous définissons $\dot{q}_{nav}$ comme suit :

$$\dot{q}_{nav} = (1 - \mu_{coll})\dot{q}_{av} + \mu_{coll}\dot{q}_{eo} \tag{4.16}$$

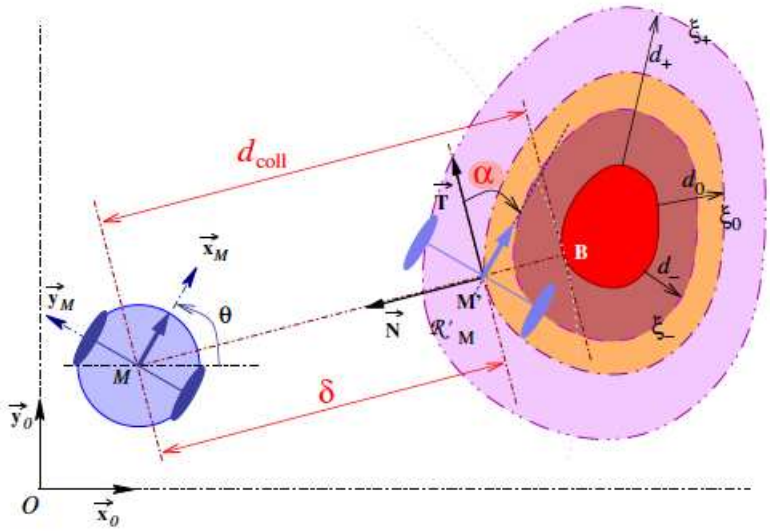


FIGURE 4.2 – Enchaînement par combinaison convexe pour la navigation

où μ_{coll} est une fonction de la distance à l'obstacle qui permet d'activer le correcteur adéquat tout en lissant le basculement. Cette fonction dépend de la stratégie de commande souhaitée. Un choix possible est proposé ci-après (cf. figure 4.2) [Folio 2007] :

$$\begin{cases} \mu_{coll} = 0 & si \quad d_{coll} > d_0 \quad ET \quad EVITE_{coll} = 0 \quad ET \quad SORTIR_{coll} = 0 \\ \mu_{coll} = \frac{d_{coll} - d_0}{d_- - d_0} & si \quad d_{coll} \in [d_-, d_0] \quad ET \quad EVITE_{coll} = 0 \\ \mu_{coll} = \frac{d_{coll} - d_+}{d_S - d_+} & si \quad d_{coll} \in [d_S, d_+] \quad ET \quad SORTIR_{coll} = 1 \\ \mu_{coll} = 1 & sinon \end{cases} \tag{4.17}$$

où

- d_- est la distance entre l'enveloppe ξ_- et l'obstacle. L'enveloppe ξ_- définit une zone à l'intérieur de laquelle seul le correcteur $\dot{q}_{eo}$ est actif.
- d_+ est la distance entre l'enveloppe ξ_+ et l'obstacle. L'enveloppe ξ_+ définit une zone à l'extérieur de laquelle seul le correcteur $\dot{q}_{av}$ est actif.
- $EVITE_{coll}$ est un indicateur positionné à 1 lorsque le robot franchit l'enveloppe de sécurité ξ_-. Il permet de maintenir μ_{coll} à 1, et par conséquent la commande de contournement $\dot{q}_{eo}$, tant qu'il y a un risque de collision.
- $SORTIR_{coll}$ est un indicateur de sortie qui passe à 1 lorsque la condition de sortie est satisfaite. Cette condition doit être vérifiée lorsque la tâche référencée vision et la tâche d'évitement d'obstacle deviennent compatibles. Dans [Cadenat 1999], l'auteur propose de définir cet événement dès lors que les directions de la base mobile et de la platine deviennent parallèle à l'obstacle, se traduisant par l'annulation des angles ϑ et α.
- d_S est la distance du robot à l'obstacle relevée par le télémètre laser lorsque la condition de sortie intervient. Elle permet de garantir la continuité de l'évolution de μ_{coll} au moment où le robot est autorisé à quitter l'enveloppe de contournement de l'obstacle.

Notons que lors des phases de transition, la commande appliquée au robot est une combinaison des commandes calculées par les deux correcteurs élémentaires. Il est possible que les vitesses obtenues soient incompatibles entrainant alors l'existence de minima locaux. De plus, la durée

de la transition est fixée à l'aide des enveloppes ξ_- et ξ_+, c'est-à-dire de manière géométrique. Il peut s'avérer difficile de régler ces valeurs *a priori* pour chacun des obstacles présents dans la scène.

4.1.2.2 Enchaînement par séquencement dynamique

Le séquencement dynamique a été initialement développé dans [Souères & Cadenat 2003]. Il est basé sur le formalisme des fonctions de tâches et permet de garantir la continuité de la commande à l'instant t_s de commutation entre les différents correcteurs. Soient deux fonctions de tâches e_i et e_{i+1} à enchainer. Sous l'hypothèse qu'elles sont ρ-admissibles, les correcteurs permettant de les réguler à zéro sont donnés par :

$$\left[\begin{array}{c} \dot{q}_i = J_i^{-1}\dot{e}_i^* \\ \dot{q}_{i+1} = J_{i+1}^{-1}\dot{e}_{i+1}^* \end{array} \right. \tag{4.18}$$

où $\dot{e}_i^*$ et $\dot{e}_{i+1}^*$ sont les dynamiques des fonctions de tâches e_i et e_{i+1}. Le séquencement dynamique cherche à garantir l'égalité des commandes $\dot{q}_i$ et $\dot{q}_{i+1}$ à l'instant de commutation t_s. En d'autres termes :

$$\dot{q}(t) = \left[\begin{array}{cc} \dot{q}_i(t) & \forall t \leq t_s \\ \dot{q}_{i+1}(t) & \forall t \geq t_s \end{array} \right. \tag{4.19}$$

Pour cela, une solution consiste à choisir [Mansard & Chaumette 2007] pour $\dot{e}_{i+1}^*$:

$$\dot{e}_{i+1}^*(t) = -\lambda_{i+1}e_{i+1}(t) + \rho(t) \tag{4.20}$$

où $\rho(t)$ doit être défini de telle manière que $\dot{e}_{i+1}^*(t) = -\lambda_{i+1}e_{i+1}(t) = \dot{e}_{i+1}(t), \forall t >> t_s$. Dans [Mansard & Chaumette 2007], les auteurs proposent de définir $\rho(t)$ comme étant :

$$\rho(t) = \big(J_{i+1}J_i^{-1}\dot{e}_i(t_s) + \lambda_{i+1}e_{i+1}(t_s)\big)\, e^{-\tau(t-t_s)} \tag{4.21}$$

où $\tau > 0$ est un scalaire fixant le temps de décroissance. Pour $t >> t_s, \rho(t) = 0$ et $\dot{e}_{i+1}^*(t) = \dot{e}_{i+1}(t)$. Ainsi à partir des équations (4.18), (4.19), (4.20) et (4.21), nous obtenons à l'instant $t = t_s$:

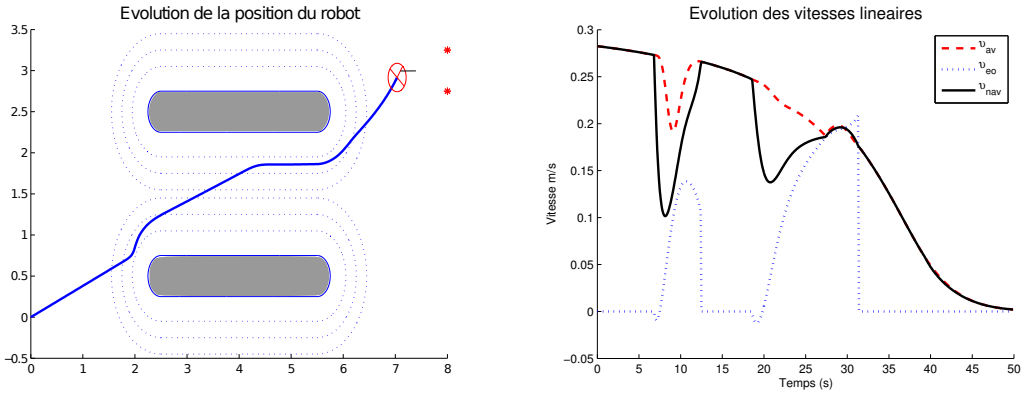
$$\begin{array}{rcl} \dot{q}_{i+1}(t_s) & = & J_{i+1}^{-1}(t_s)\big(-\lambda_{i+1}e_{i+1}(t_s) + \rho(t_s)\big) \\ & = & J_{i+1}^{-1}(t_s)\big(-\lambda_{i+1}e_{i+1}(t_s) + J_{i+1}(t_s)J_i(t_s)^{-1}\dot{e}_i(t_s) + \lambda_{i+1}e_{i+1}(t_s)\big) \\ & = & J_i(t_s)^{-1}\dot{e}_i(t_s) \\ & = & \dot{q}_i(t_s) \end{array} \tag{4.22}$$

Le séquencement dynamique permet donc de garantir la continuité de la loi de commande lorsque le problème est modélisé sous la forme de fonctions de tâches à séquencer. La transition se faisant dans le temps et non dans l'espace, l'approche ne nécessite que de fixer le temps de décroissance τ. De plus, la condition déclenchant la commutation est extérieure à la méthode. Dans notre cas, il s'agit du franchissement de l'enveloppe ξ_0. Au premier abord,

le séquencement dynamique semble donc être plus facilement réglable que la combinaison convexe.

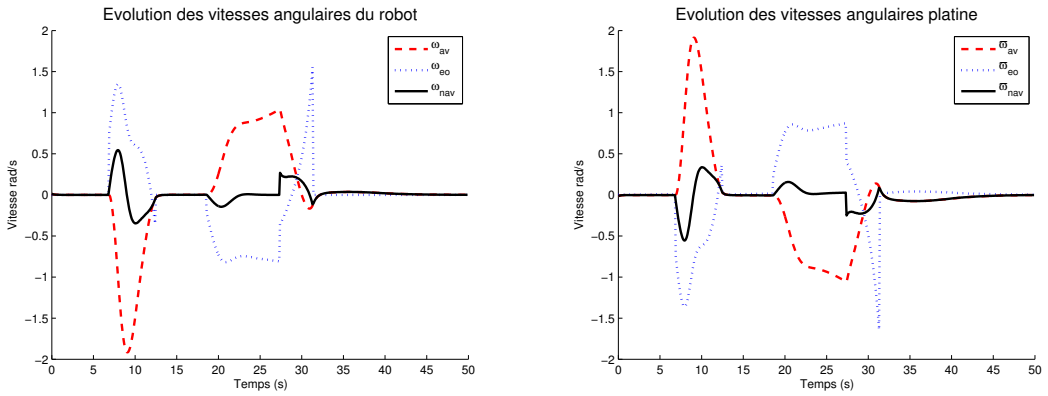
4.1.2.3 Etude comparative

Dans cette section, nous proposons de simuler des navigations par asservissement visuel dans un environnement encombré d'obstacles non occultants. Pour cela, nous utilisons deux lois de commande, l'une basée sur l'asservissement visuel et l'autre sur le contournement d'obstacles. Les transitions sont réalisées soit par combinaison convexe, soit par séquencement dynamique. Pour les deux types de simulations, la navigation est effectuée par rapport à une cible caractérisée par quatre points et avec un gain $\lambda_{av} = 0.35$. Le robot a pour état initial $\chi = [0\ 0\ 0\ \frac{\pi}{4}]^T$ tandis que la caméra doit atteindre la situation $\chi_C = [7.1\ 3\ 0]^T$. La loi de commande réalisant le contournement d'obstacle est calculée avec $\lambda_{coll} = 0.7$ et $\kappa = 0.1$. Enfin, la période d'échantillonnage est fixée à 100 ms.



(a) Trajectoire du robot (b) Evolution des vitesses linéaires

FIGURE 4.3 – Navigation avec combinaison convexe



(a) Evolution des vitesses angulaires de la base (b) Evolution des vitesses angulaires de la platine

FIGURE 4.4 – Navigation avec combinaison convexe

Pour la première simulation, la navigation est réalisée à l'aide de la combinaison convexe. Nous fixons $d_- = 0.3m$, $d_0 = 0.5m$ et $d_+ = 0.7m$. Comme nous pouvons le voir dans la figure 4.3(a), le robot atteint le but prédéfini tout en évitant les obstacles. De plus l'évolution des

vitesses de navigation $\dot{q}_{nav}$, présentée dans les figures 4.3(b), 4.4(a) et 4.4(b), ne présente pas de discontinuité. Cela est dû à la régularité du paramètre μ_{coll}, dont l'évolution est présentée dans la figure 4.5. L'enchaînement par combinaison convexe permet donc de sélectionner le correcteur adéquat, tout en garantissant la continuité de la loi de commande.

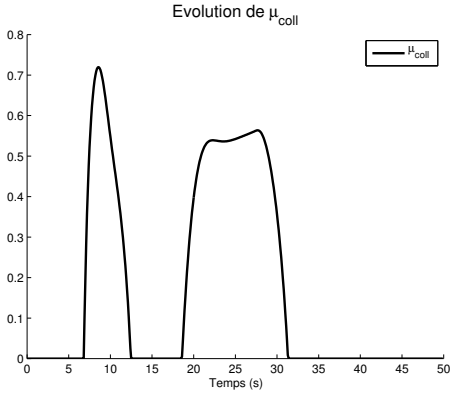
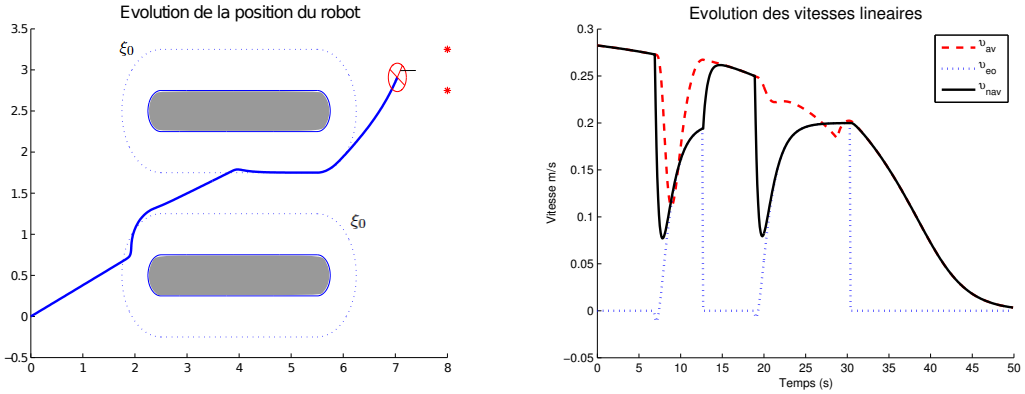


FIGURE 4.5 – Evolution de μ_{coll} lors d'une navigation avec combinaison convexe



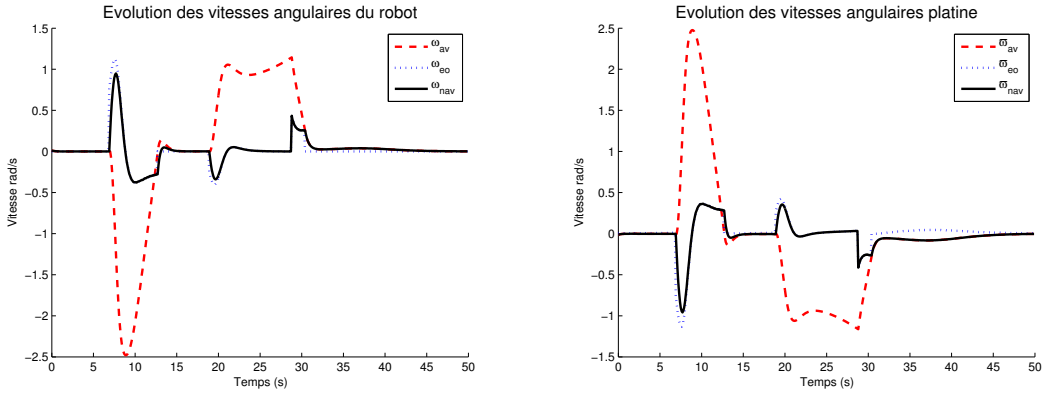
(a) Trajectoire du robot

(b) Evolution des vitesses linéaires

FIGURE 4.6 – Navigation avec séquencement dynamique

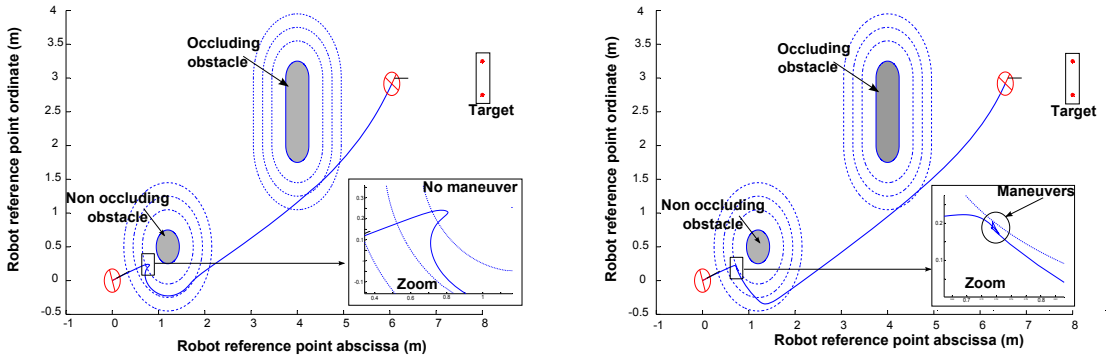
Pour la seconde simulation, la navigation est réalisée à l'aide d'un enchaînement par séquencement dynamique avec $\tau = 2s$. Comme précédemment, le robot atteint son but tout en évitant les obstacles (cf. figure 4.6(a)). La trajectoire est par ailleurs globalement identique. De plus, il n'existe pas de discontinuité dans l'évolution des commandes appliquées au robot (cf. figures 4.6(b), 4.7(a) et 4.7(b)).

Après un certain nombre de tests, nous sommes arrivés à la conclusion que les méthodes de transitions sont équivalentes lorsque l'on utilise un correcteur pour l'asservissement visuel et un autre pour le contournement d'obstacles. Cependant, dans [Cadenat *et al.* 2012], nous avons pu mettre en évidence les limites de l'enchaînement par combinaison convexe. Dans cet article, la commande globale du robot est calculée à partir de trois correcteurs : le premier est associé à l'asservissement visuel, le second à l'évitement d'obstacle et le dernier à l'évitement d'occultations. Lorsque l'enchaînement est réalisé par séquencement dynamique (cf. figure 4.8(a)), la navigation est parfaitement réalisée. Dans le cas où l'on utilise la combinaison convexe (cf. figure 4.8(b)), la commande calculée provoque des déplacements inappropriés à la navigation.



(a) Evolution des vitesses angulaires de la base (b) Evolution des vitesses angulaires de la platine

FIGURE 4.7 – Navigation avec séquencement dynamique



(a) Navigation avec séquencement dynamique (b) Navigation avec combinaison convexe

FIGURE 4.8 – Navigation avec trois correcteurs

Avec le séquencement dynamique, la loi de commande est calculée à partir d'un seul correcteur. De plus, à chaque basculement, il existe un régime transitoire permettant de garantir la continuité de la loi de commande. Ainsi la navigation est réalisée si chaque correcteur est correctement défini. Lorsque l'on utilise l'enchaînement par combinaison convexe, la loi de commande appliquée au robot est donnée par une combinaison de plusieurs correcteurs possiblement antagonistes, ce qui peut occasionner des problèmes de minima locaux ou des manoeuvres. C'est donc pour cette raison que notre choix se portera sur l'enchaînement par séquencement dynamique pour la suite de nos travaux, dans la mesure où toutes nos sous-tâches peuvent s'exprimer sous la forme d'une fonction de tâche ρ-admissible.

4.1.3 Simulation de navigation en environnement encombré

Dans cette section, nous proposons de simuler une navigation dans un environnement encombré. Le robot doit atteindre une cible composée de quatre cercles en naviguant dans un environnement encombré d'obstacles non-occultant et occultant, respectivement représentés en gris et noir dans la figure 4.9. Pour cela, nous disposons des correcteurs réalisant l'asservissement visuel 2D et le contournement d'obstacle. Les transitions entre ces derniers sont réalisées par séquencement dynamique. Enfin, nous utilisons la paire prédicteur/correcteur pour estimer la profondeur des indices visuels lorsque ceux-ci sont mesurables. Cette estimation est utilisée comme valeur initiale par l'algorithme de calcul d'évolution des indices visuels lorsqu'une occultation se produit. Durant toute la simulation, les données fournies par les capteurs sont bruitées comme précédemment afin de s'approcher des conditions expérimentales.

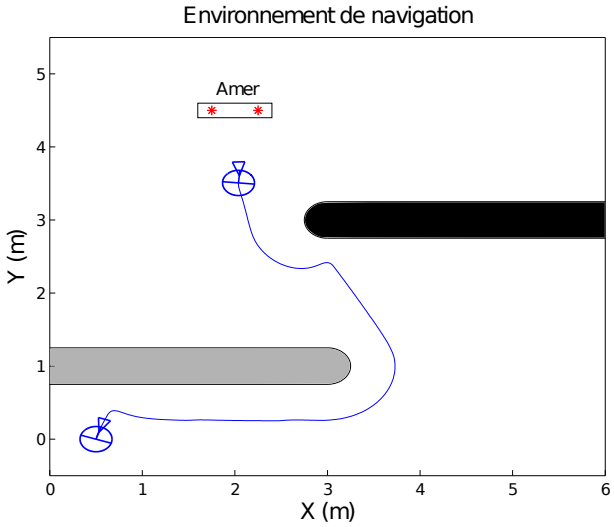


FIGURE 4.9 – Exemple de navigation dans un environnement encombré

Malgré la présence d'obstacles, le robot accomplit la tâche de navigation comme nous pouvons le voir dans la figure 4.9. Les correcteurs réalisant l'asservissement 2D et l'évitement d'obstacle sont successivement utilisés. Comme nous le montre la figure 4.11(b), l'enchaînement par séquencement dynamique permet de garantir la continuité de la loi de commande appliquée au robot.

Enfin, au cours de la navigation la cible est occultée par un des obstacles. Cependant, grâce au processus d'estimation de la profondeur (cf. figure 4.10) et à l'algorithme de calcul de l'évolution des indices visuels, l'occultation est parfaitement gérée. En effet, nous pouvons voir

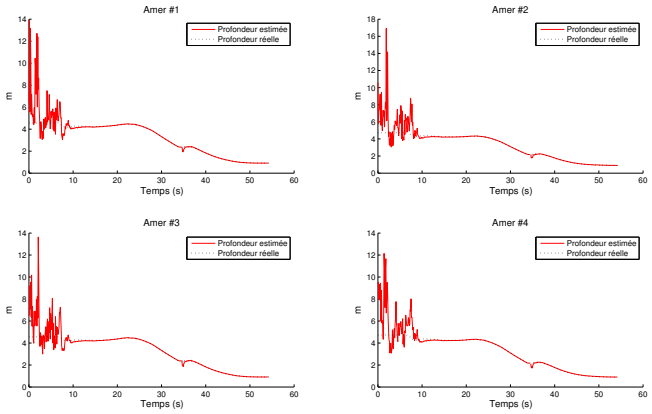
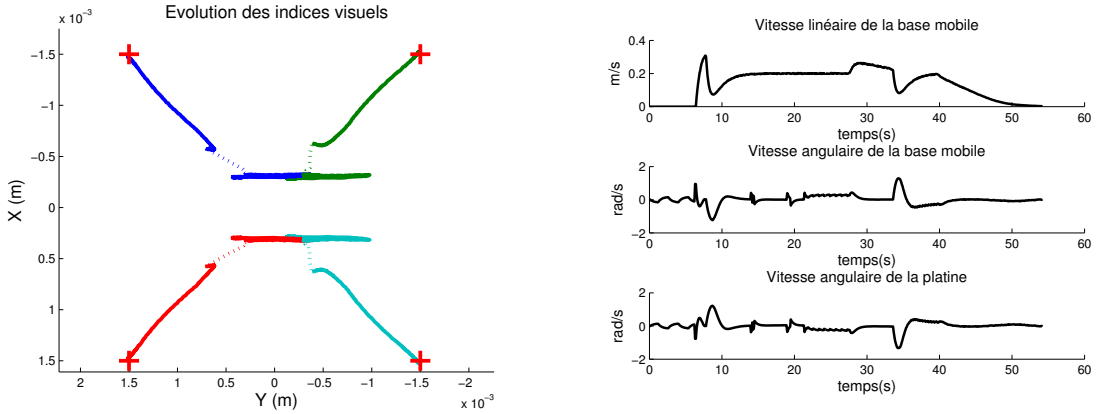


FIGURE 4.10 – Estimation de la profondeur des indices visuels

dans la figure 4.11(a) que les indices visuels estimés (en pointillés) sont en adéquation avec ceux mesurés (traits pleins). A l'aide des différentes méthodes présentées jusqu'à présent, il est donc possible de réaliser une navigation en environnement encombré lorsque l'amer de référence est perceptible dès le début.



(a) Evolution des indices visuels

(b) Evolution des commandes

FIGURE 4.11 – Navigation dans un environnement encombré

4.2 La navigation au long cours

A l'aide des correcteurs réalisant l'asservissement visuel 2D (cf. partie 2.3.3) et le contournement d'obstacles (cf. partie 4.1.1), ainsi que des équations d'évolution des indices visuels (cf. partie 3.2.2.3), il est possible de réaliser une navigation référencée multi-capteurs dans un environnement encombré d'obstacles occultants. Cependant l'amer utilisé pour obtenir le signal de mesure doit être visible par la caméra dès le début de la navigation pour que cette dernière puisse démarrer. A ce stade, la navigation est donc limitée par la portée de la caméra. Pour relâcher cette contrainte, nous proposons d'utiliser un modèle de l'environnement. Nous avons vu dans le premier chapitre que deux solutions s'offraient à nous : la carte métrique et la carte topologique. Nous rappelons qu'une carte métrique est sensible aux erreurs de mesures, ainsi qu'aux variations de l'environnement. De plus, les informations métriques que l'on en

extrait ne sont pas en adéquation avec nos correcteurs basés sur un retour de sortie. La carte topologique, plus robuste vis-à-vis des deux points mentionnés plus haut, permet quant à elle de représenter la scène à l'aide de données de notre choix. Nous proposons donc d'utiliser une carte topologique comme modèle de l'environnement pour augmenter la portée de la navigation et réaliser de longs déplacements par asservissement visuel. Nous parlerons alors de navigation au long cours.

Nous rappelons qu'une carte topologique est une représentation discrète de l'environnement sous la forme d'un graphe [Thrun 2002]. De plus, elle est définie par une propriété caractéristique définissant les ensembles représentés par les nœuds et une condition d'adjacence. Enfin, il est possible d'associer à chaque nœud des informations nécessaires à la commande ou à la localisation. Plusieurs types de cartes topologiques ont été proposés afin de réaliser une navigation visuelle. Dans [Matsumoto *et al.* 1996], les auteurs proposent d'utiliser une carte topologique pour se localiser, mais aussi pour fournir les commandes permettant de réaliser la navigation. Pour cela, une phase de pré-navigation durant laquelle des images sont enregistrées [1] est réalisée. Pour chaque nouvelle image acquise, un nœud est créé. Il est ajouté au graphe en le reliant au nœud précédent et l'image lui est attribuée. Enfin, une commande est associée à chaque nouveau nœud. Le processus de construction de la carte permet ainsi de créer une base de données contenant des images auxquelles correspondent des commandes. Durant la navigation, le robot se localise en comparant l'image courante avec la base de données. La localisation est alors un test de similitude entre images. Une fois le robot localisé, la commande associée à l'image est appliquée. Dans [Booij *et al.* 2007], les auteurs proposent de ne pas utiliser toutes les images acquises lors de la pré-navigation. Pour cela ils extraient des descripteurs SIFT des images issues de la base de données. Ils sélectionnent alors des images clés en cherchant à maximiser la différence entre les situations de prise de vue de deux images clés, tout en ayant un nombre minimal de descripteurs SIFT [Bay *et al.* 2006] communs. Un nœud auquel sont associés des descripteurs SIFT, est créé pour chaque image clé sélectionnée. Lors d'une navigation, il est ainsi possible de réaliser une localisation à l'aide d'une étude de similitude entre les descripteurs issus de l'image courante et ceux issus de la base de données. Finalement, le robot est commandé sur la base d'informations 3D obtenues à l'aide d'une estimation de la matrice d'homographie entre l'image courante et l'images clé. Une approche similaire est proposée pour différents types de robots, caméras et descripteurs dans [Courbon 2009]. Enfin, dans [Cherubini *et al.* 2011] les auteurs proposent d'extraire des points de Harris des images acquises au cours de la pré-navigation. De même que précédemment, des images clés sont sélectionnées et un nœud est créé pour chacune d'entre elles. Les images clés sont utilisées pour localiser le robot dans le graphe, mais aussi pour calculer la commande permettant de réaliser la navigation.

Les trois dernières approches présentées proposent donc de construire une carte représentant un chemin appris lors d'une phase de pré-navigation. Le robot doit ensuite rejouer ce chemin avec plus ou moins de fidélité à l'aide des informations visuelles enregistrées et courantes. Le processus de sélection des images clés garantit au robot de percevoir l'image suivante lorsqu'il a atteint un but intermédiaire. De plus le robot est assuré de pouvoir atteindre chaque image clé depuis la précédente puisque cela a été le cas lors de la pré-navigation. Cependant, le fait que les situations pour lesquelles les images clé ont été obtenues soient relativement proches ne permet pas au robot de dévier du chemin appris. Il semble alors difficile d'éviter des obstacles qui n'étaient pas présents lors de l'apprentissage. Seuls les auteurs de [Cherubini *et al.* 2011] proposent de synthétiser un correcteur permettant de remédier à cette situation.

1. La phase de pré-navigation est commune à toutes les approches que nous présentons dans ce paragraphe.

Néanmoins, les occultations totales ne sont tolérées pour aucune des méthodes présentées dans cette section.

Dans ce mémoire, nous proposons de décomposer la tâche de navigation au long cours en plusieurs tâches de sous-navigation [2]. Pour atteindre la cible finale qui n'est pas visible pour la situation initiale, le robot devra donc atteindre successivement une série de cibles constituant des sous-buts. La carte topologique correspondant à nos besoins est donc constituée de nœuds représentant les amers. Deux nœuds doivent être reliés si les amers associés sont mutuellement visibles et atteignables depuis leur voisinage respectif. A l'aide de la carte topologique, il est alors possible de calculer un chemin correspondant à la séquence de cibles devant être atteintes par le robot. Enfin, il est nécessaire de définir une stratégie de navigation au long cours. En effet, tout au long de la navigation, le robot doit détecter les amers présents dans la scène, se localiser, sélectionner la cible par rapport à laquelle il doit s'asservir, éviter les obstacles, gérer les occultations, ... Nous proposons de définir un algorithme de supervision qui détermine les actions à réaliser en fonction des données perçues par les différents capteurs. Dans cette section, nous présentons donc dans un premier temps les outils relatifs à la carte topologique puis nous nous attardons sur la stratégie de navigation au long cours et à l'algorithme de supervision qui en découle. Enfin, nous terminons notre présentation par des simulations présentant nos premiers résultats concernant la navigation au long cours.

4.2.1 La carte topologique

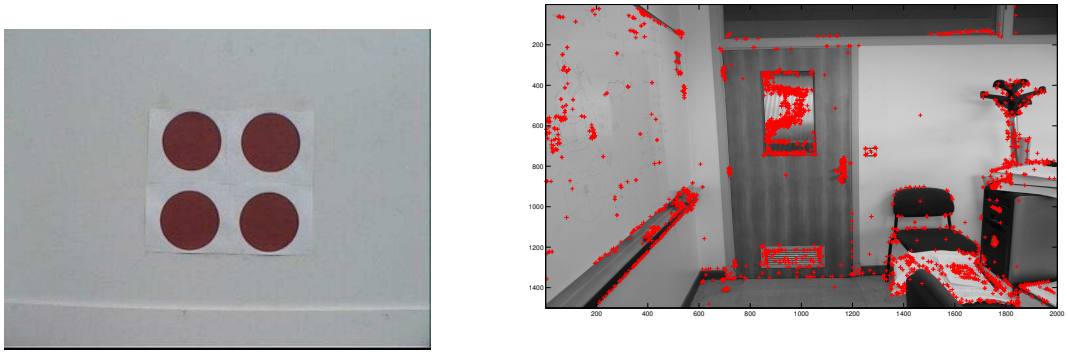
4.2.1.1 Définition de notre carte topologique

Afin de concevoir la carte topologique adaptée à notre approche, il est nécessaire d'identifier dans un premier temps les amers utilisés pour caractériser l'environnement. Dans notre étude, nous proposons d'utiliser des cibles artificielles, chacune composée d'un nombre différent de cercles rouges (cf. figure 4.12(b)) et disposées en des lieux particuliers de la scène. Par la suite, nous prévoyons d'utiliser des amers naturels, qui seront caractérisées par des points de Harris [Harris & Stephens 1988], des descripteurs SURF [Lowe 1999] ou SIFT [Bay *et al.* 2006] (cf. figure 4.12(b)). Une étude des performances de ces méthodes pour différents types de caméras est réalisée dans [Courbon 2009]. Les cibles étant identifiées, nous devons maintenant concevoir la carte topologique de l'environnement.

Nous nous intéressons à la définition de la carte topologique adaptée à notre approche. Pour chaque amer il existe une zone de la scène à l'intérieur de laquelle ce dernier est visible. Nous proposons alors de définir les nœuds de notre graphe comme étant les zones de visibilité des amers. Notre propriété caractéristique définissant les ensembles représentés par les nœuds est donc la visibilité d'un amer. Afin d'illustrer nos propos, nous proposons de considérer l'environnement, présenté dans la figure 4.13, à l'intérieur duquel quatre cibles numérotées de 1 à 4 ont été placées. Pour chacune d'entre elles, les zones de visibilité sont présentées dans les figures 4.14(a), 4.14(b), 4.14(c) et 4.14(d). Le graphe associé à la scène (cf. figure 4.16) contient donc quatre nœuds correspondant chacun à une zone de visibilité.

Nous proposons de définir la condition d'adjacence comme étant la présence d'une zone commune de visibilité pour deux cibles. Cela signifie que les deux cibles peuvent être perçues par la caméra si le robot est à l'intérieur de cette zone. Les zones de visibilité des cibles 1 et 4 sont

2. Une sous-navigation correspond à un déplacement du robot vers une seule cible, utilisant l'asservissement visuel 2D et l'évitement d'obstacles.



(a) Exemple d'amer artificiel

(b) Exemple d'amer naturel

FIGURE 4.12 – Exemples d'amer

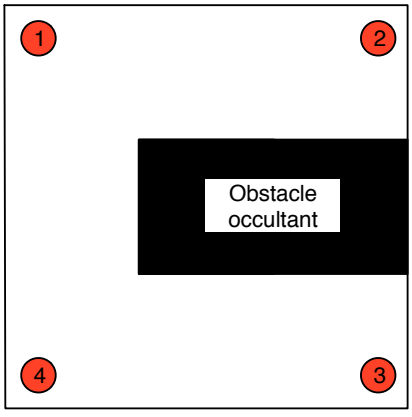
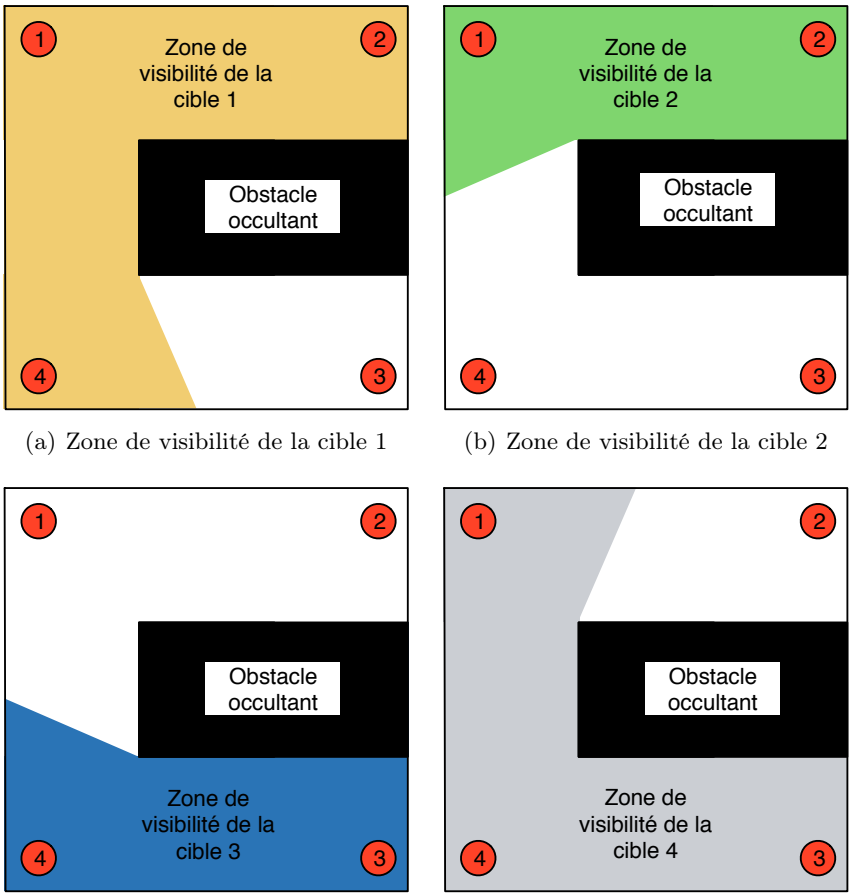


FIGURE 4.13 – Exemple d'environnement de navigation



(a) Zone de visibilité de la cible 1

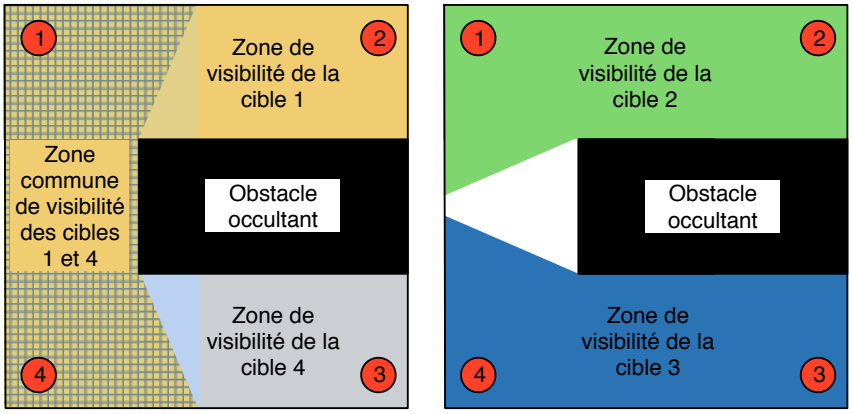
(b) Zone de visibilité de la cible 2

(c) Zone de visibilité de la cible 3

(d) Zone de visibilité de la cible 4

FIGURE 4.14 – Zones de visibilité des différentes cibles

regroupées dans la figure 4.15(a). Il existe une zone de visibilité commune pour ces deux cibles représenté par le quadrillage. Il est donc possible de relier les nœuds du graphe correspondant aux zones de visibilité des amers 1 et 4 (cf. figure 4.16). Dans la figure 4.15(b), nous avons regroupé les zones de visibilité des cibles 2 et 3. Nous pouvons voir qu'il n'y a pas de recouvrement et donc nous ne relions pas les nœuds 2 et 3 du graphe (cf. figure 4.16). L'opération est répétée pour chaque couple de cibles et finalement nous obtenons le graphe présenté dans la figure 4.16.



(a) Zone commune de visibilité des cibles 1 et 4

(b) Zone commune de visibilité des cibles 2 et 3

FIGURE 4.15 – Exemple de zone commune de visibilité

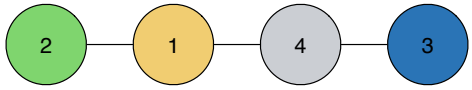


FIGURE 4.16 – Graphe représentant l'environnement de navigation

4.2.1.2 Données associées à la carte topologique

Les règles définissant notre carte topologique sont désormais posées. Cependant la carte dans son état actuel n'est pas suffisante pour permettre de réaliser une navigation visuelle au long cours. En effet, il est nécessaire d'y ajouter les informations requises par les processus de navigation et de localisation.

Afin de permettre au robot de se localiser dans le graphe, nous associons des données de localisation à chaque nœud. Dans le cas où nous utilisons des amers artificiels, nous donnons le nombre de cercles composant la cible. Pour des amers naturels, les données de localisation correspondent aux descripteurs obtenus à partir d'images prises depuis différents points de vue de l'amer.

Pour naviguer vers chaque cible, nous ajoutons à chaque nœud des données de navigation. Elles correspondent à l'image de référence utilisée par la loi de commande. Les situations auxquelles sont acquises les images doivent appartenir aux zones de visibilité commune. Ainsi, lorsque le robot a accompli la tâche de sous-navigation par rapport à une cible, nous garantissons que la caméra peut effectivement percevoir le prochain amer d'intérêt. Pour cela, il est possible que le robot doive pivoter sur lui-même afin de le faire entrer dans son champ de vue. De plus, il est envisageable que la cible suivante puisse être vue bien avant la fin de la sous-navigation

par rapport à l'amer courant. Il peut alors être intéressant de définir une tâche de recherche de cible durant la navigation.

4.2.1.3 Construction de la carte topologique

Nous considérons une scène où n_a amers T_i ont été identifiés, avec $i \in [1, ..., n_a]$. Notre carte topologique est construite lors d'une phase d'apprentissage ou pré-navigation. Cependant, le robot n'est pas télé-commandé par l'utilisateur pour effectuer un chemin comme par exemple dans [Matsumoto *et al.* 1996], mais pour être positionné successivement aux situations $S_i^* = [X_{M_i}^* \; Y_{M_i}^*]^T$ appartenant au voisinage de chaque amer T_i. En effet, à chaque amer correspond un nœud N_i dans le graphe et nous cherchons alors à identifier les connexions C_{ij} dans ce dernier, avec $j \in [1, ..., n_a]$. Afin de réaliser cette opération, nous ne cherchons pas à déterminer les zones de visibilité pour chaque amer, mais à prouver l'existence d'une zone de visibilité commune entre deux cibles T_i et T_j. Le robot étant positionné en S_i^*, il effectue alors une rotation sur lui-même pour détecter les cibles visibles depuis cette position. Pour chaque cible T_j perçue, une connexion C_{ij} est créée entre les nœuds N_i et N_j. Enfin, une image de référence par rapport à la cible T_i est enregistrée et associée au nœud N_i. Elle sera utilisée durant la navigation au long cours pour réaliser la sous-navigation par rapport à la cible T_i.

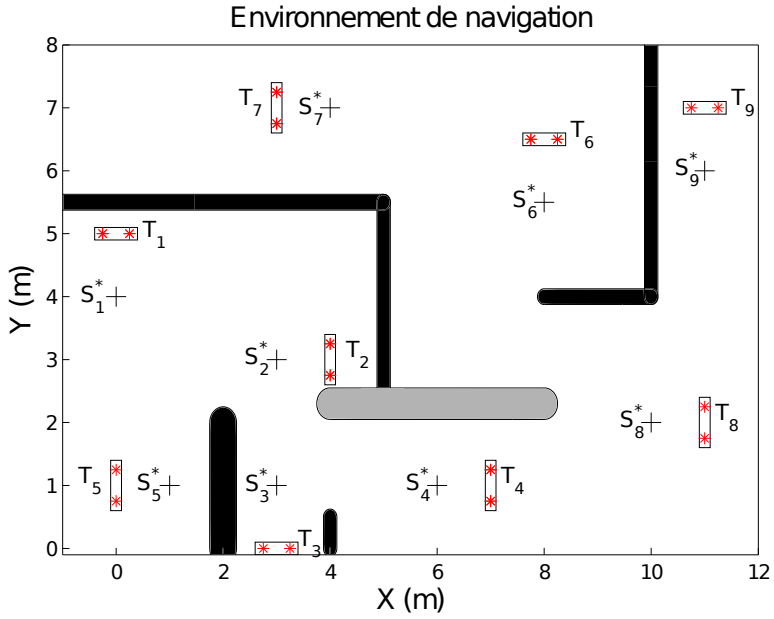


FIGURE 4.17 – Environnement de navigation

Afin de valider notre approche, nous avons réalisé un ensemble de simulations pour l'environnement présenté sur la figure 4.17. Il est composé d'obstacles occultants qui sont représentés en noir, d'un obstacle gris non-occultant et de $n_a = 9$ amers. La carte topologique modélisant l'environnement est donc un graphe constitué de 9 nœuds. Afin de déterminer l'existence de liens entre les nœuds, le robot est successivement placé aux situations S_i^* relatives aux amers T_i, avec $i \in [1, ..., n_a]$. Pour chaque position S_i^*, le robot détecte les amers visibles lors d'une rotation sur lui-même. Nous obtenons alors le graphe présenté dans la figure 4.18. Notons que le choix de la situation S_i^* a une influence sur le graphe. En effet, pour la situation S_2^*, l'amer T_8 n'est pas détecté alors qu'il existe une zone de visibilité commune entre les cibles T_2 et T_8. Un choix différent pour S_2^* aurait permis de lier les nœuds N_2 et N_8 et donc de compléter la

modélisation de la scène. Enfin, les images de références s_i^* utilisées dans les lois de commande sont obtenues pour $\chi_i^* = [X_{M_i}^* \; Y_{M_i}^* \; \theta_i^* \; 0]^T$, où θ_i^* est choisi de manière à ce que le robot soit face à la cible considérée.

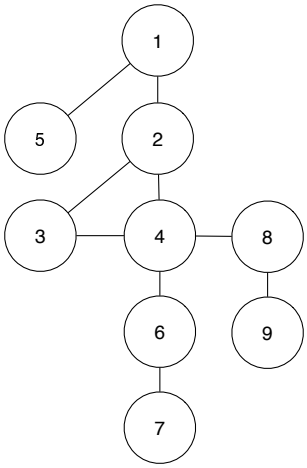


FIGURE 4.18 – Carte topologique de l'environnement

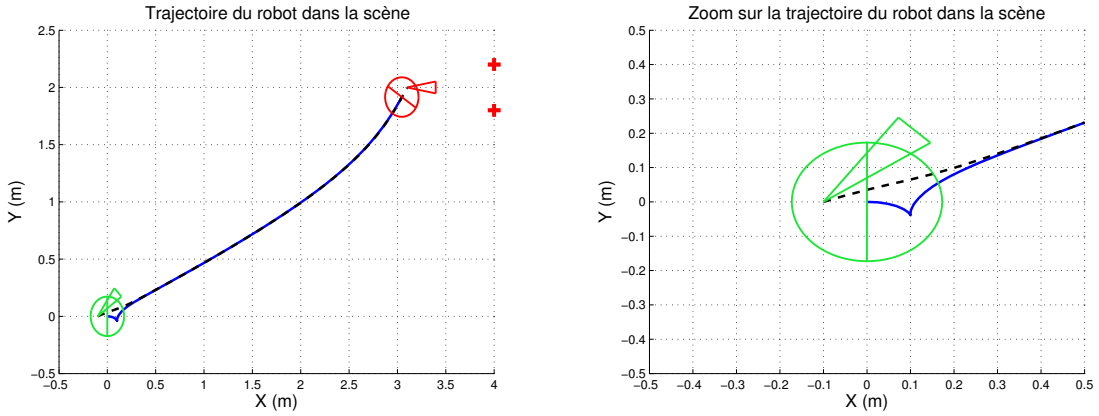
Nous sommes désormais en possession d'une carte topologique à partir de laquelle il est possible de calculer un chemin constitué d'une série d'amers lorsque les amers initial et final sont définis. Le robot peut donc réaliser une navigation pour laquelle la cible d'intérêt n'est pas visible pour la situation initiale.

4.2.2 Lois de commande complémentaires

Avant de pouvoir réaliser une navigation au long cours dans les meilleures conditions, deux situations spécifiques doivent être analysées. Premièrement, il est nécessaire de garantir le bon comportement du robot lors d'une transition entre deux sous-navigations par rapport à des amers distincts. En effet, comme nous avons pu le voir dans la section 2.3.3, il se peut que le robot doive faire une manœuvre afin de réaliser l'asservissement visuel désiré. Nous devons donc nous attacher à définir une stratégie permettant d'éviter ce mouvement indésirable. Nous développerons une loi de commande spécifique que nous appellerons "réorientation". Deuxièmement, nous avons vu dans la section 4.2.1.1, qu'il pouvait être intéressant de rechercher le prochain amer pendant la sous-navigation avec l'amer courant. En effet, l'approche proposée jusqu'à présent garantit la visibilité de la cible lorsque le robot atteint une situation de référence. Cependant il est possible que la prochaine cible soit perceptible bien avant d'atteindre cette situation. Il apparait donc nécessaire de développer une méthode de recherche d'amer durant la navigation.

4.2.2.1 Loi de commande de réorientation

Au début d'une sous-navigation par asservissement visuel 2D, le robot peut être contraint de faire une manœuvre (cf. figure 4.19). Un point de rebroussement apparaît, ce qui peut occasionner inutilement des risques de collision. En effet, le robot se déplace alors en marche arrière et ne peut plus détecter la présence d'obstacles du fait de l'angle de débattement du laser réduit à 270°. Il est donc nécessaire dans notre cas de chercher à éviter ces manœuvres.



(a) Trajectoire pour un asservissement visuel 2D (b) Zoom sur la trajectoire pour un asservissement visuel 2D

FIGURE 4.19 – Trajectoire avec un point de rebroussement

La présence d'un point de rebroussement est dépendante de la situation initiale du robot par rapport à la cible. Ainsi, il n'y a pas de manœuvre lorsque celui-ci est initialement orienté en direction de cette dernière. Nous proposons donc de synthétiser un correcteur permettant au robot d'être correctement situé au début de l'asservissement visuel 2D. Il doit donc permettre d'orienter la base mobile en direction de la cible. Pour cela, nous proposons de découpler la commande de la base de celle de la platine (cf. section 4.1.1.2) de manière à atteindre deux objectifs. Premièrement, nous désirons garder l'amer d'intérêt dans le champ de vue de la caméra durant la phase de réorientation. Deuxièmement, la caméra étant dirigée vers la cible, orienter la base mobile en direction de cette dernière est équivalent à réguler à zéro l'angle ϑ. Pour synthétiser le correcteur permettant de réorienter la base mobile, nous utilisons le formalisme des fonctions de tâche. Il est ainsi possible de réaliser les basculements entre les différents correcteurs par séquencement dynamique. Nous définissons donc la fonction de tâche suivante :

$$e_{ro} = \begin{pmatrix} l_{ro} - v_{ro}t \\ \theta_{ro} - \omega_{ro}t \end{pmatrix} \tag{4.23}$$

où l_{ro} et θ_{ro} sont l'abscisse curviligne et l'orientation de la base mobile exprimés dans le repère du robot R_M défini à l'instant où démarre la réorientation. De plus, v_{ro} et ω_{ro} sont respectivement les vitesses linéaire et angulaire du robot que nous désirons atteindre. v_{ro} doit être positif afin d'éviter tout point de rebroussement, tandis que le signe de ω_{ro} est identique à celui de ϑ. Ainsi, la réalisation de la tâche e_{ro} permet de réguler ϑ à zéro avec des vitesses de la base mobile données.

Afin de calculer la loi de commande permettant de réguler à zéro la fonction de tâche (4.23), nous proposons d'imposer une décroissance exponentielle comme suit :

$$\dot{e}_{ro} = -\lambda_{ro} e_{ro} \tag{4.24}$$

La dérivée par rapport au temps de (4.23) est donnée par :

$$\dot{e}_{ro} = J_{ro}\dot{q}_{base} + A_{ro} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \dot{q}_{base} + \begin{pmatrix} -\upsilon_{ro} \\ -\omega_{ro} \end{pmatrix} \tag{4.25}$$

En utilisant les équations (4.23), (4.24) et (4.25), nous obtenons :

$$\dot{q}_{base} = J_{ro}^{-1}(-\lambda_{ro} e_{ro} - A_{ro}) \tag{4.26}$$

A l'aide de la commande (4.26), nous imposons au robot d'avancer tout en tournant dans une direction imposée. Cependant pour que la caméra garde dans son champ de vue l'amer d'intérêt, il est nécessaire de commander la platine. Pour cela, nous nous appuyons sur la loi de commande (4.14) utilisée pour commander la platine lors d'un évitement d'obstacle.

Il n'est pas nécessaire d'orienter la base mobile exactement en direction de la cible pour éviter les points de rebroussement. Nous proposons donc d'utiliser la loi de commande (4.26) jusqu'à que l'angle ϑ soit proche de zéro. Ainsi, nous pouvons garantir que les transitions entre chaque sous-navigation conduisent à des trajectoires lisses et qu'aucun danger de collision non détectable ne sera introduit par des manœuvres indésirables.

4.2.2.2 Recherche d'une cible durant une navigation

Lors d'une navigation au long cours, le robot doit enchaîner des sous-navigations par rapport à des amers T_i, avec $i \in [1, ..., n_a]$. De plus, il doit trouver l'amer T_{i+1} à partir d'un certain voisinage de T_i. La méthode selon laquelle nous avons construit la carte topologique, nous garantit que T_{i+1} est visible pour la situation S_i^*. Cependant, T_{i+1} peut éventuellement être aperçue bien avant que cette situation soit atteinte. Nous proposons de développer une stratégie permettant au robot de chercher l'amer T_{i+1} tout en réalisant la sous-navigation courante. Ainsi, le robot peut basculer vers la tâche suivante sans avoir complètement réalisé la précédente.

Dans le chapitre 3, nous avons vu qu'il était possible de naviguer par rapport à un amer même lorsque celui-ci est occulté. Nous proposons de nous appuyer sur ce résultat pour réaliser la recherche de cible durant une sous-navigation. En effet, nous prévoyons de déclencher des "occultations volontaires" de l'amer durant lesquelles le robot navigue à l'aide d'indices visuels estimés. La caméra n'étant plus utilisée pour la navigation, elle balaie la scène à la recherche de la prochaine cible. Pour cela, la platine parcourt la zone comprise entre ses deux butées. Si la cible est trouvée alors la sous-navigation par rapport à T_{i+1} est lancée. Sinon la sous-navigation par rapport à T_i est prolongée, en utilisant à nouveau des indices visuels mesurés. L'opération de recherche de l'amer suivant est régulièrement relancée jusqu'à le trouver ou terminer la mission courante.

Remarque : *Lors d'une phase de recherche de la prochaine cible, des indices visuels estimés sont utilisés. Il est donc nécessaire d'être en possession d'une estimation correcte de la profondeur avant de lancer ou relancer le processus de recherche. Nous proposons d'utiliser l'estimation des indices visuels de référence pour détecter la convergence de l'estimation de la profondeur (cf. 3.4.2). Tant que les indices visuels de référence n'ont pas convergé vers ceux mesurés lors de phase de pré-navigation, cela signifie que la profondeur n'est pas correctement estimée, et donc qu'il n'est pas judicieux de créer une "occultation volontaire". Nous proposons*

d'utiliser ce critère pour définir les instants auxquels nous pouvons déclencher une phase de recherche de cible sans mettre en péril la navigation.

4.2.3 Stratégie de navigation au long cours

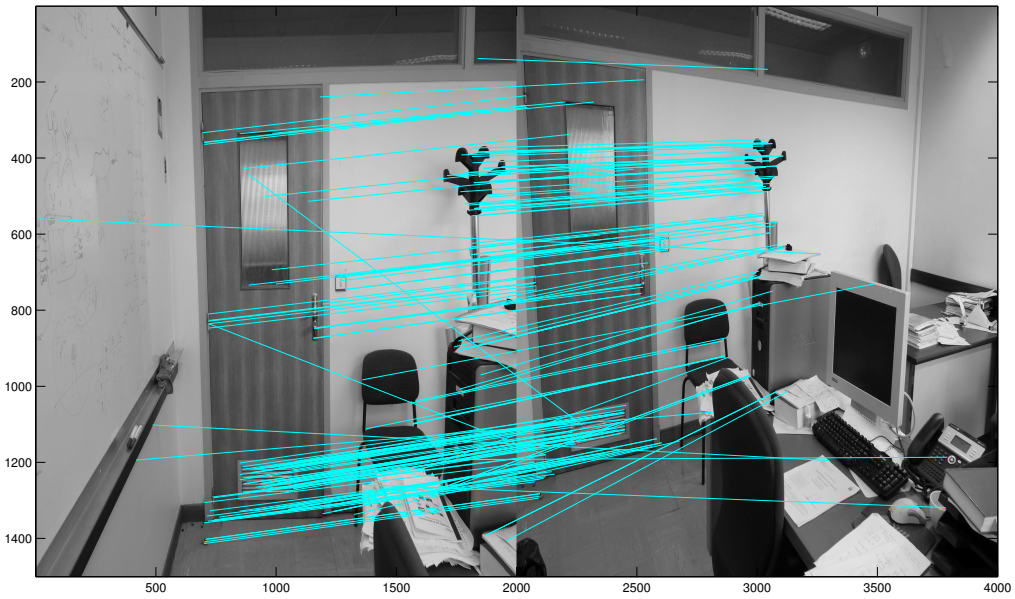
Dans ce manuscrit, nous avons présenté différents outils permettant de réaliser une navigation au long cours dans un environnement encombré d'obstacles possiblement occultants. Nous proposons dans cette section de reprendre les six phases nécessaires à la navigation qui ont été présentées dans le premier chapitre, tout en précisant les choix qui ont été les nôtres.

- **Perception**
  Afin de réaliser une navigation au long cours, nous proposons d'utiliser une caméra sténopée comme capteur principal. De plus, les données fournies par un télémètre laser sont utilisées lors des phases d'évitement d'obstacles.

- **Modélisation**
  La scène dans laquelle se déroule la navigation est modélisée à l'aide d'une carte topologique. Chaque nœud représente la zone de visibilité d'un amer et un couple de nœuds est relié s'il existe une zone de visibilité commune pour les deux amers. De plus, des données sensorielles sont associées à chacun des nœuds. Ces dernières sont exploitées par les processus de localisation et de commande.

- **Localisation**
  La localisation consiste à identifier la cible qui se situe dans le champ de vue de la caméra. Ainsi, il est possible de connaître la situation du robot à l'intérieur du graphe. La localisation est réalisée à l'aide des données associées à chacun des nœuds de la carte topologique. Dans le cas de cibles artificielles, il s'agit du nombre de cercles composant la cible. Il suffit alors simplement de compter les cercles pour savoir de quelle cible il s'agit. Dans le cas d'amers naturels, les informations sensorielles correspondent aux descripteurs issus d'images obtenues pour différents points de vue. La localisation consiste alors à réaliser un test de similitude entre les images. Pour cela, les descripteurs de l'image courante et ceux de la base de données sont appariés. L'image issue de la base de données ayant le plus grand nombre de points communs avec l'image courante est sélectionnée. Le nœud correspondant est alors considéré comme celui où se situe le robot. Nous présentons un exemple de localisation pour des amers naturels dans les figures 4.20(a) et 4.20(b). L'image courante se situe à gauche tandis que les images issues de la base de données sont à droite. Dans notre cas le test de similitude est concluant pour l'image de référence de la figure 4.20(a).

- **Planification**
  La scène contenant n_a amers $[T_1, ..., T_{n_a}]$ est modélisée par une carte topologique de n_a nœuds. Il est alors possible de calculer un chemin T_P constitué par une séquence de n_P amers $[T_{P1}, ..., T_{Pn_P}]$ à atteindre. La première étape consiste à identifier les cibles dans le champ de vue de la caméra pour la situation initiale $[X_M(0)\ Y_M(0)]$ lors d'une première phase de localisation. Dans un second temps, la cible finale étant donnée par l'utilisateur, nous pouvons calculer le chemin à réaliser à l'aide de méthodes telles que l'algorithme de Dijkstra [Dijkstra 1971] qui permet de trouver le plus court chemin dans un graphe.
  Nous considérons l'environnement présenté dans la figure 4.17 contenant $n_a = 9$ amers et la carte topologique associée présentée dans la figure 4.18. Le robot est initialement situé en $[X_M(0) = 1\ Y_M(0) = 0.5]$ et doit atteindre la situation S_7^* relative à l'amer T_7. Après



(a) Test de similitude validé

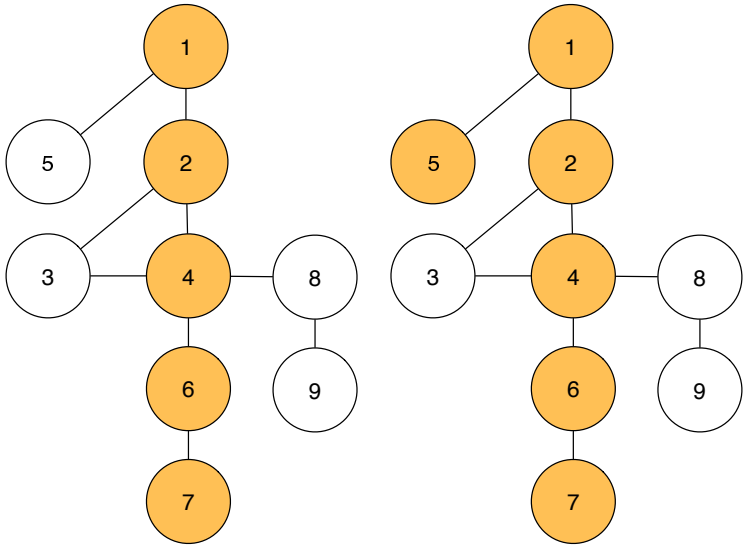
(b) Test de similitude non validé

FIGURE 4.20 – Exemple de localisation avec des amers naturels

la phase de localisation, il apparait que les cibles T_1 et T_5 sont les points d'entrée possibles dans le graphe. En utilisant l'algorithme de Dijkstra, il vient alors les deux chemins présentés dans les figures 4.21(a) et 4.21(b) :

$$T_P^1 = [T_{P1}, T_{P2}, T_{P3}, T_{P4}, T_{P_5}] = [T_1, T_2, T_4, T_6, T_7]$$

$$T_P^5 = [T_{P1}, T_{P2}, T_{P3}, T_{P4}, T_{P_5}, T_{P_6}] = [T_5, T_1, T_2, T_4, T_6, T_7]$$



(a) Chemin calculé avec le nœud 1 comme entrée (b) Chemin calculé avec le nœud 5 comme entrée

FIGURE 4.21 – Exemple de chemin dans le graphe

Nous n'avons introduit aucune information relative à la distance dans la carte topologique. Notre choix se porte donc sur le chemin contenant le moins d'éléments, c'est-à-dire le chemin T_P^1 contenant $n_P = 5$ amers. Cependant, notre choix n'est pas forcément toujours le plus pertinent. En effet, il peut arriver qu'un chemin contenant un plus grand nombre de nœuds soit le plus court dans la scène.

- **Action**
  Tout au long de ce manuscrit, nous avons identifié trois phases d'action nécessaires à la réalisation d'une navigation au long cours. Premièrement, nous avons défini un correcteur permettant au robot de réaliser des navigations sur une courte distance, appelées sous-navigations. Ce correcteur cherche à annuler une erreur entre l'image courante donnée par la caméra et une image de référence obtenue lors de la phase de pré-navigation. Il s'agit donc d'un correcteur par retour de sortie. Deuxièmement, nous avons présenté un correcteur permettant d'éviter des obstacles. Celui-ci essaie de stabiliser le robot sur un chemin défini à partir des données fournies par le télémètre laser. Troisièmement, nous avons synthétisé un correcteur permettant d'éviter les points de rebroussement lors de la transition entre deux sous-navigations. Ce correcteur utilise l'image fournie par la caméra pour orienter cette dernière en direction de la cible d'intérêt, tandis que la base mobile est contrôlée à l'aide de données issues des capteurs proprioceptifs. La transition entre chacun de ces correcteurs est assurée par un séquencement dynamique qui garantit la continuité de la loi de commande.

- **Décision**
  Le processus de décision doit permettre au robot d'activer ou de désactiver les outils que nous avons présentés, afin de garantir le succès de la navigation au long cours. Nous proposons d'utiliser un algorithme de supervision pour réaliser le processus de décision. L'algorithme est construit selon une stratégie de navigation que nous allons développer par la suite.

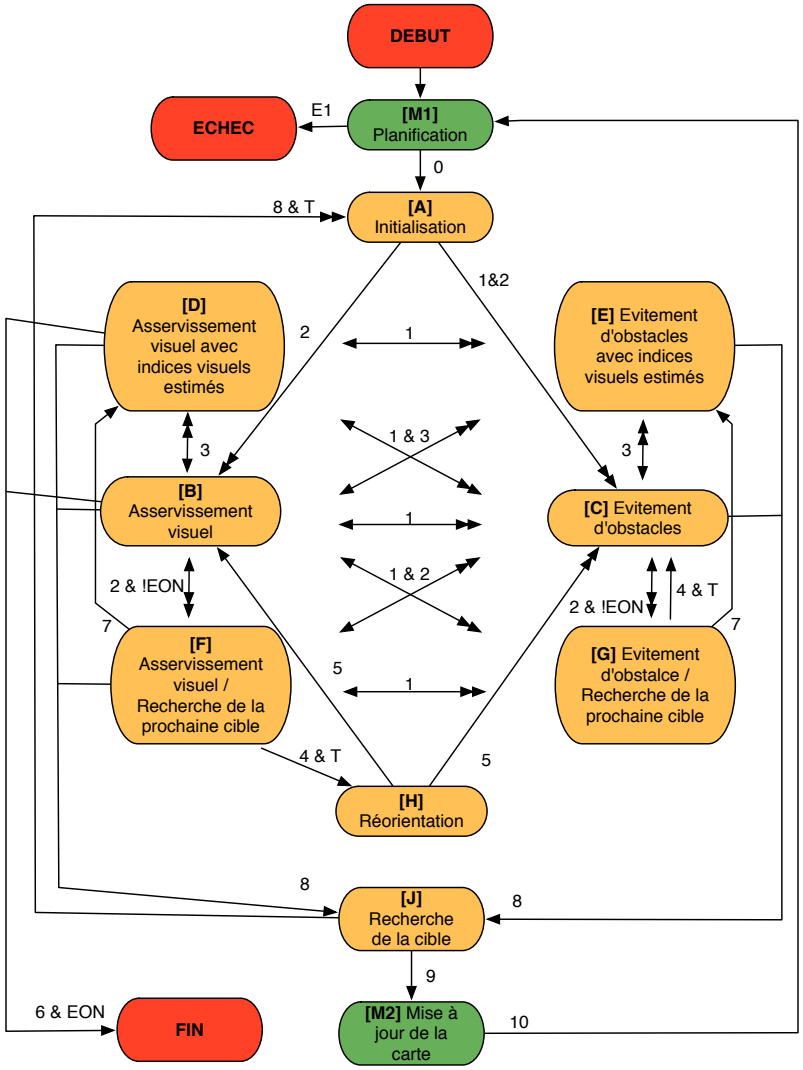
  La scène où se déroule la navigation a été modélisée lors d'une phase de pré-navigation. Le robot doit alors identifier les amers qu'il peut voir depuis sa position $[X_M(0)\ Y_M(0)]$. Pour cela il réalise une rotation de 360^o sur lui-même. Les points d'entrées dans le graphe et l'amer final étant connus, un ou plusieurs chemins sont calculés à l'aide de l'algorithme de Dijkstra. Si plusieurs chemins sont disponibles, alors celui contenant le moins d'éléments est sélectionné. La séquence d'amers T_P à atteindre successivement est désormais déterminée.

  Le robot exécute alors la phase d'initialisation qui consiste à réaliser de petites rotations permettant d'estimer la profondeur des indices visuels de la cible T_{P1}. Cette phase se termine lorsque la reconstruction de la profondeur a convergé, c'est-à-dire lorsque les indices visuels de référence estimés sont égaux à ceux mesurés lors de la pré-navigation. Nous utiliserons ce critère tout au long de la navigation pour détecter la convergence du processus d'estimation. Il est alors possible de gérer une occultation dès le début de la mission.

  La sous-navigation par rapport à l'amer T_{P1} est alors démarrée. Si le robot s'approche trop près d'un obstacle, le correcteur permettant de contourner l'obstacle est utilisé. De plus, si une occultation est détectée, alors le robot navigue à l'aide d'indices visuels estimés. Durant la sous-navigation le robot cherche régulièrement la prochaine cible T_{P2}. Si elle n'est pas trouvée, le robot continue la sous-navigation courante et redémarre le processus d'estimation de la profondeur. Celui-ci ayant convergé, la recherche de T_{P2} est relancée. Cette boucle est répétée jusqu'à que la prochaine cible soit repérée ou que la sous-navigation soit terminée. Dans ce dernier cas, le robot effectue une rotation sur lui-même afin d'identifier la prochaine cible. Si elle n'est pas trouvée, alors le graphe est mis à jour et un nouveau chemin est calculé. Dans le cas où aucun chemin ne permet d'atteindre la cible finale, nous estimons que la navigation a échoué.

  Nous considérons maintenant que la cible T_{P2} a été trouvée. La phase de réorientation est donc lancée. Celle-ci est stoppée lorsque le robot est face à la cible T_{P2}. Les phases de sous-navigation, d'évitement d'obstacle et de recherche sont répétées selon les mêmes critères que précédemment tant que le robot n'a pas atteint la cible finale ou que la navigation a échoué.

  Nous présentons dans la figure 4.22 l'algorithme de supervision permettant de réaliser la stratégie de navigation que nous venons de développer. Les cases rouges correspondent aux états de la navigation, les oranges aux processus de navigation réalisés à l'aide de correcteurs, et les vertes aux processus effectués lorsque le robot est à l'arrêt. De plus, les différentes conditions permettant de passer d'un processus à l'autre sont résumées dans un tableau. Nous obtenons ainsi une vue d'ensemble de la stratégie proposée pour réaliser une navigation au long cours dans un environnement encombré.



0 : → Chemin calculé
1 : → Evitement terminé →→ Obstacle détecté
2 : →→ Convergence de l'estimation des indices visuels de références
→ Prochaine cible non trouvée
3 : → Fin de l'occultation →→ Occultation détectée
4 : → Convergence de l'estimation des prochains indices visuels de références
5 : → Fin de la réorientation
→→ Fin de la réorientation et évitement d'obstacle
6 : → Fin de l'asservissement visuel
7 : → Prochaine cible non trouvée et détection d'occultation
8 : → Sous-naviagtion terminée et prochaine cible non trouvée
→→ Prochaine cible trouvée
7 : → Prochaine cible non trouvée et détection d'occultation
9 : → Prochaine cible non trouvée
10 : → Mise à jour de la carte terminée
E1 : → Aucun chemin ne mène à la cible finale
T : → Mise à jour de la cible courante
EON : → La cible courante est la dernière cible du chemin
!EON : → La cible courante n'est pas la dernière cible du chemin

FIGURE 4.22 – Algorithme de supervision de la navigation au long cours

4.2.4 Simulations de navigation au long cours

Dans cette section, nous proposons d'illustrer en simulation la stratégie de navigation présentée dans ce chapitre. Pour cela, nous considérons l'environnement schématisé sur la figure 4.17, où les obstacles occultants sont représentés en noir tandis que les non-occultants sont gris. Chacune des cibles est constituée d'un nombre différent de cercles rouges. Le robot se localise donc en comptant le nombre de cercles. Cependant, la loi de commande est calculée à partir de quatre indices visuels quel que soit l'amer considéré. Comme pour les précédentes simulations, du bruit est ajouté aux données fournies par les capteurs afin de s'approcher des conditions d'expérimentation.

4.2.4.1 Exemple n°1

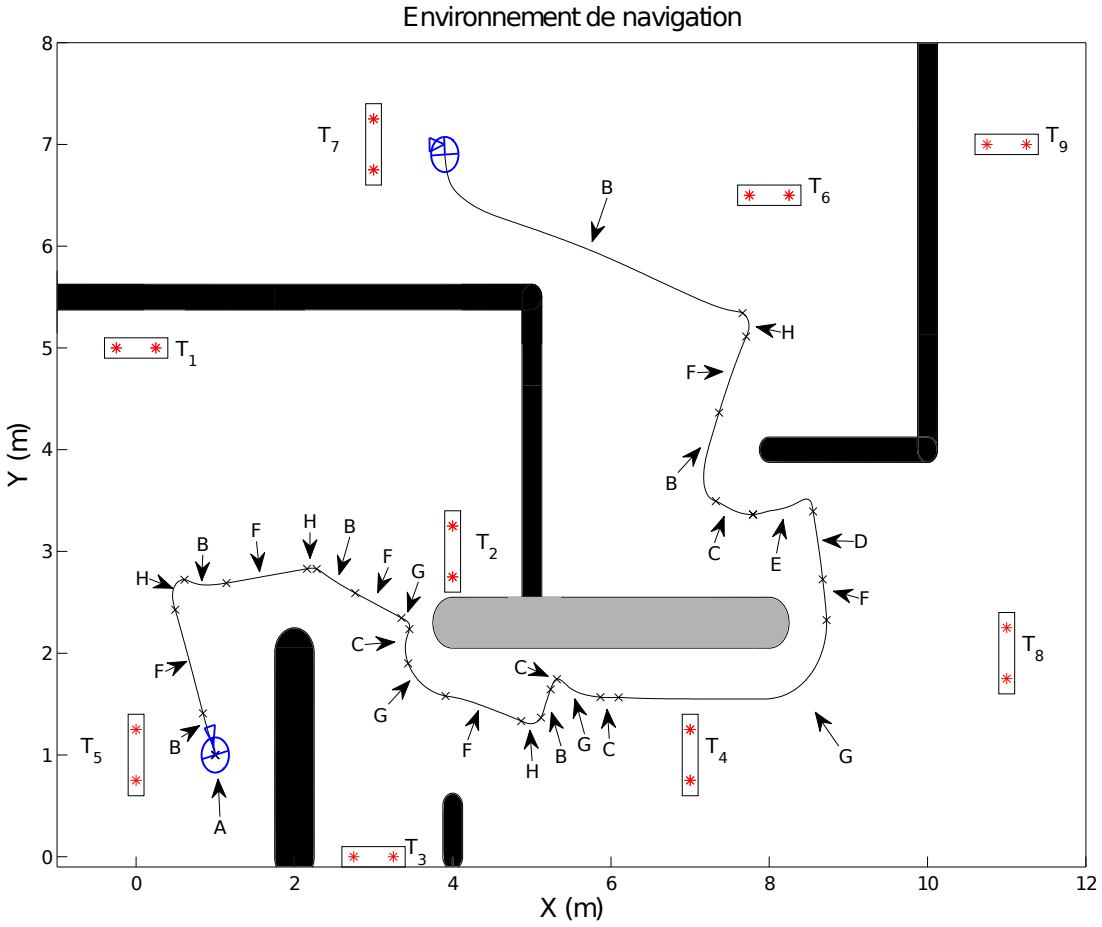


FIGURE 4.23 – Trajectoire du robot dans la scène

Pour cette première simulation l'environnement est modélisé par la carte topologique 4.18 obtenue lors de la pré-navigation pour les situations présentées dans la figure 4.17. La base mobile est initialement positionnée en $[X_M = 1\ Y_M = 1]^T$ et la caméra doit atteindre l'amer T_7. s_7^*, a été choisi de manière à ce que la situation atteinte S_7^* soit définie par $[X_C = 4.1\ Y_C = 7\ \theta_T = \pi]^T$. Après une première phase de localisation les amers T_1 et T_5 sont identifiés comme les points

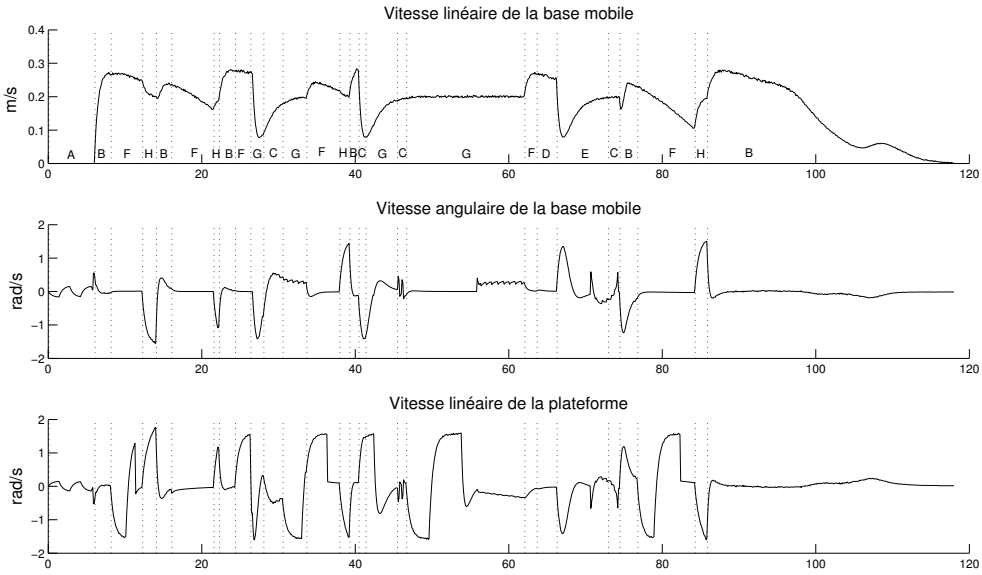


FIGURE 4.24 – Evolution des vitesses du système robotique

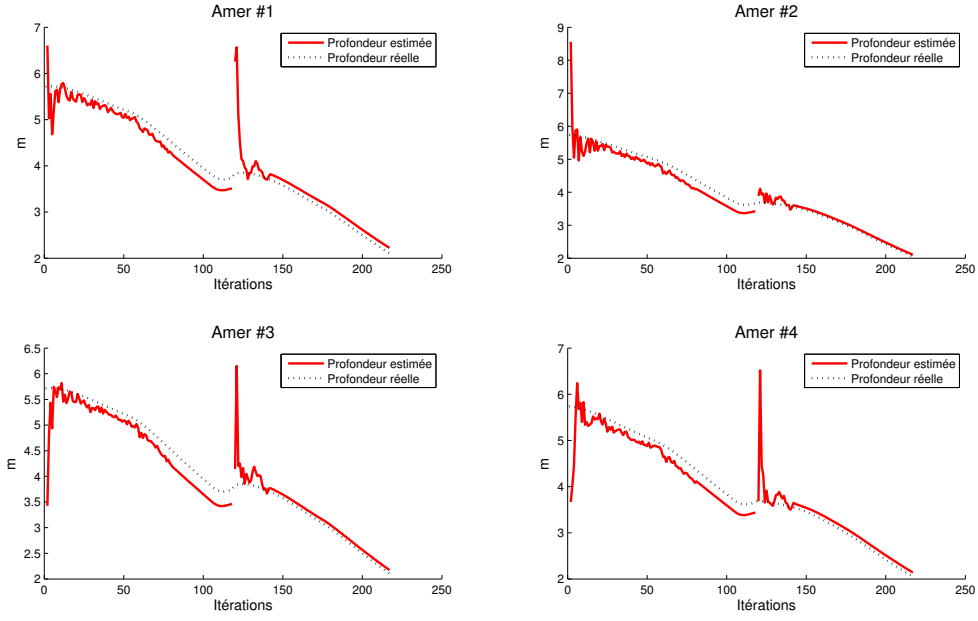


FIGURE 4.25 – Profondeur estimée des indices visuels de la cible T_4

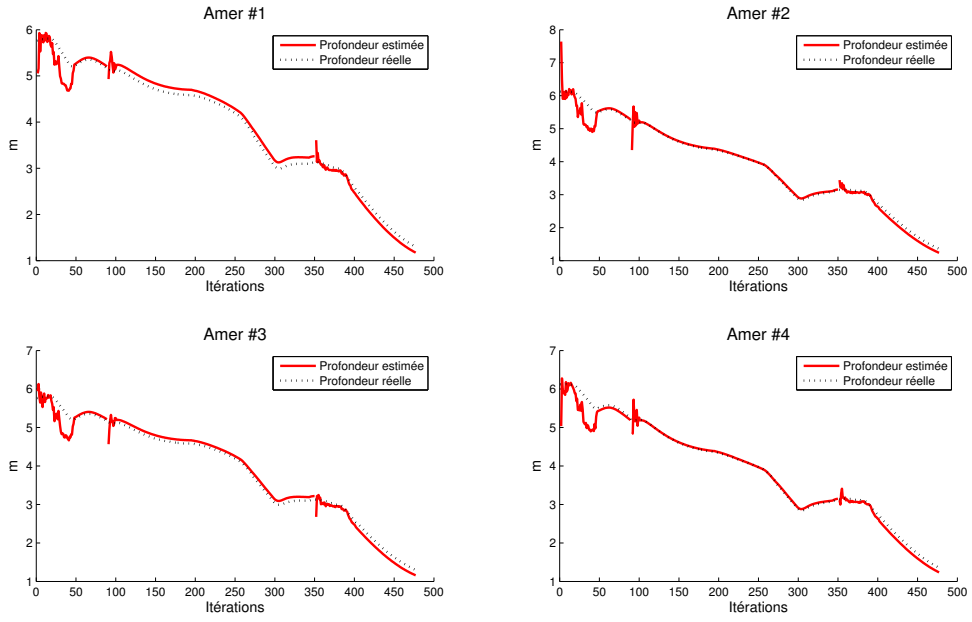
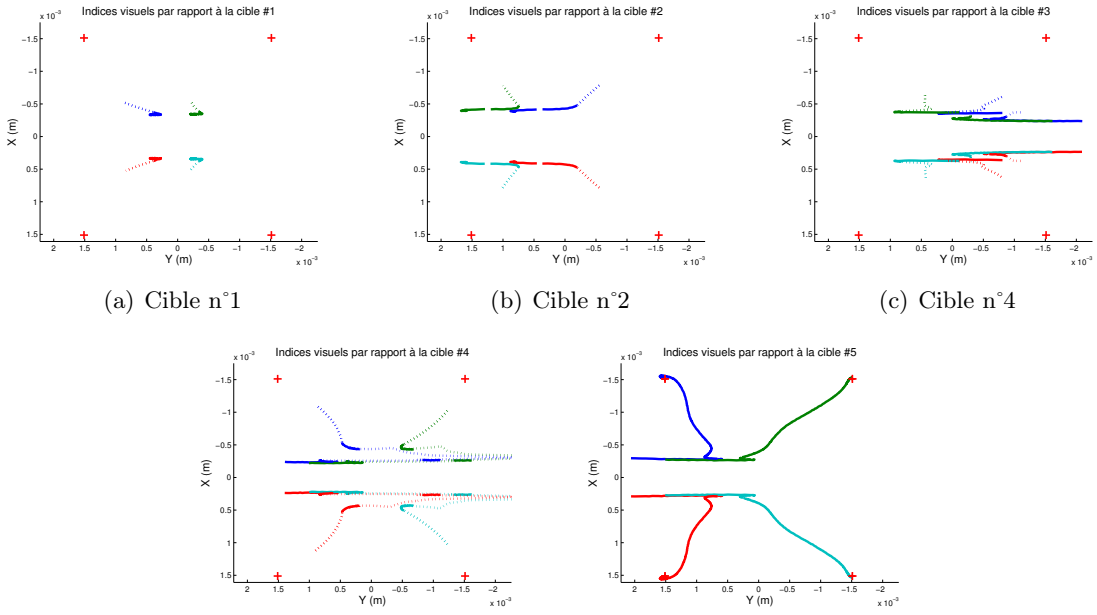


FIGURE 4.26 – Profondeur estimée des indices visuels de la cible T_6



(a) Cible n°1 (b) Cible n°2 (c) Cible n°4

(d) Cible n°6 (e) Cible n°7

FIGURE 4.27 – Evolution des indices visuels pour les différentes cibles

d'entrée dans le graphe. Deux chemins sont alors planifiés (cf. figures 4.21(a) et 4.21(b)) et celui contenant le moins de nœuds est conservé. Pour réussir sa mission, le robot doit donc atteindre successivement les cibles $T_P = [T_{P1}, T_{P2}, T_{P3}, T_{P4}, T_{P_5}] = [T_1, T_2, T_4, T_6, T_7]$.

Comme, nous pouvons le voir dans la figure 4.23, la navigation est accomplie bien que des obstacles soient présents dans la scène et que l'amer T_7 ne puisse être perçu dès le début de la tâche. Pour mener à bien sa mission, le robot a exploité les différents correcteurs à sa disposition ainsi que les processus d'estimation évoqué dans le chapitre précédent. La figure 4.24 montre que les transitions entre correcteurs sont parfaitement gérées et que la loi de commande appliquée au robot ne comporte pas de discontinuité. Les figures 4.25 et 4.26 représentent l'évolution des profondeurs estimées pour les indices visuels composant les amers T_4 et T_6. L'estimation est réalisée par deux processus : le prédicteur/correcteur lorsque des images sont disponibles, l'équation d'évolution de la profondeur lors des occultations. La qualité du premier dépend directement du nombre d'images utilisées, et c'est pourquoi nous remarquons une convergence vers la valeur réelle au fur et à mesure des acquisitions (entre les itérations 1 et 75 pour la figure 4.25). Le second processus nécessite une valeur initiale. L'erreur introduite par cette dernière est conservée tout au long de l'estimation (entre les itérations 75 et 125 pour la figure 4.25). Enfin, notons que lorsque l'estimation est de nouveau réalisée avec la paire prédicteur/correcteur (itération 125 pour la figure 4.25), il existe une discontinuité pour la valeur de la profondeur. Cela est dû au fait qu'il est nécessaire d'être en possession d'au moins deux images pour obtenir une estimation de la profondeur.

Finalement, nous nous intéressons à l'évolution des indices visuels composant chaque cible. Premièrement, nous pouvons remarquer que les indices visuels mesurés et estimés s'enchaînent parfaitement. La paire prédicteur/correcteur fournit donc une estimation de la profondeur suffisamment correcte pour permettre de gérer les occultations à l'aide des équations d'évolution des indices visuels. Deuxièmement, notons que l'image de référence n'est jamais atteinte sauf pour le dernier amer. En effet, le robot n'a jamais eu à atteindre les situations intermédiaires finales pour percevoir la prochaine cible contenue dans le chemin.

Avec cette simulation, nous avons donc montré qu'il était possible de réaliser une navigation au long cours en suivant la stratégie que nous avons développée tout au long de ce chapitre.

4.2.4.2 Exemple n°2

Pour cette seconde simulation, nous considérons l'environnement présenté dans la figure 4.28 où seuls des obstacles occultants sont présents. Celui-ci étant un sous-ensemble de la scène initiale, nous utilisons dans un premier temps la carte topologique 4.29(a) obtenue lors de la phase de pré-navigation précédente. Notons que celle-ci a été réalisée avec un obstacle non-occultant qui est désormais considéré comme occultant. A partir du graphe 4.29(a), nous planifions le chemin à réaliser 4.29(b).

Comme nous pouvons le voir dans la figure 4.28, la caméra ne peut percevoir l'amer T_4 lorsqu'il navigue par rapport à T_2. Le robot ayant terminé la sous-navigation par rapport à T_2, la carte topologique de l'environnement est mise à jour 4.29(c) : la connexion entre T_2 et T_4 est supprimée. A partir de cette nouvelle situation un chemin est planifié 4.29(d). Celui-ci existant, la navigation est relancée et le robot atteint l'objectif initial, c'est-à-dire l'amer T_4.

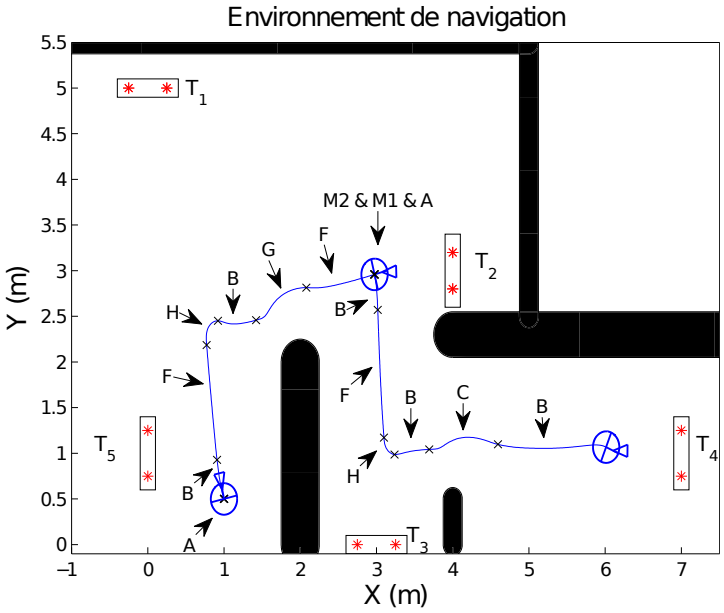
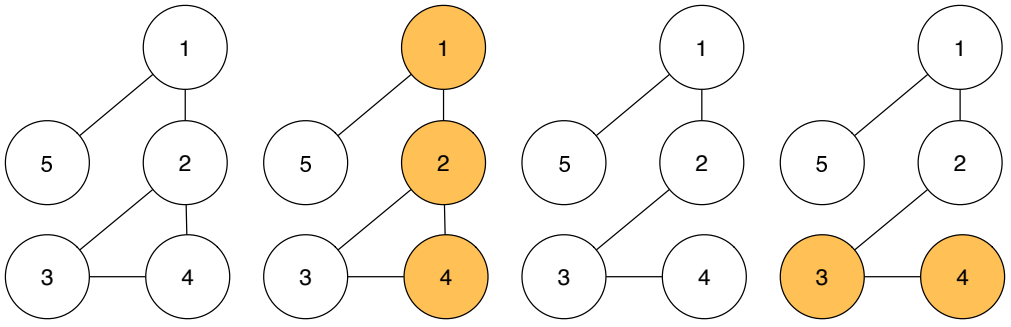


FIGURE 4.28 – Exemple de navigation avec replanification



(a) Graphe initial (b) Chemin planifié (c) Graphe modifié (d) Chemin replanifié

FIGURE 4.29 – Evolution de la carte topologique lors d'une re-planification

4.2.4.3 Exemple n°3

A partir de la carte topologique 4.18, il n'aurait pas été possible d'atteindre l'amer T_7 si tous les obstacles avaient été occultants. En effet, la navigation se serait arrêtée à la cible T_8. Une fois cette dernière sous-navigation terminée, le robot n'aurait pas été capable de détecter la présence de T_6. Nous aurions alors considéré que la navigation avait échoué.

Pour que le robot puisse atteindre l'amer T_7 alors que tous les obstacles sont occultants, il est nécessaire de fournir une carte topologique plus complète que la précédente. En effet, lors de la construction du graphe 4.18, de nombreuses zones de visibilité commune n'ont pas été détectées. Nous proposons désormais de réaliser la pré-navigation en considérant les situations de référence de la figure 4.30. Le graphe correspondant et le chemin planifié sont présentés sur les schémas 4.32(a) et 4.32(b). Il doit être noté que la zone de visibilité commune entre les amers T_6 et T_8 est détectée grâce à un choix différent de la situation de référence. A partir de cette modélisation, le robot peut accomplir sa mission (cf. figure 4.30) malgré la présence d'obstacles occultants.

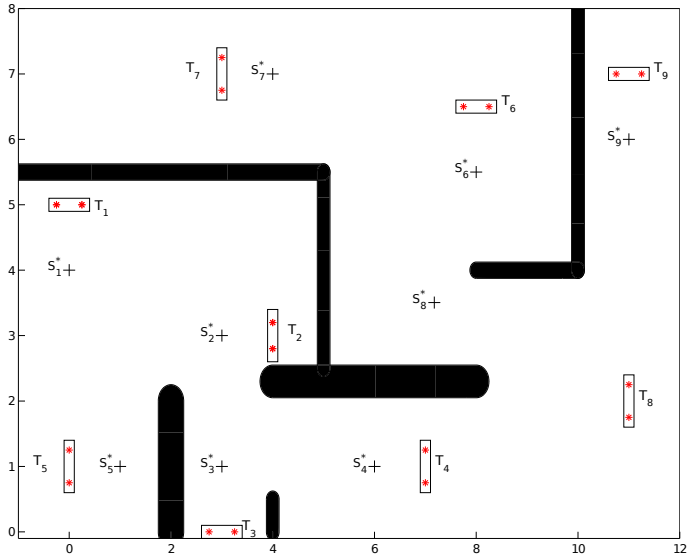


FIGURE 4.30 – Environnement de navigation

A l'aide de ces différentes simulations, nous avons montré qu'il est possible de réaliser une navigation au long cours dans un environnement encombré d'obstacles occultants à l'aide de la stratégie que nous avons proposée. Cette approche repose sur une représentation partielle de l'environnement sous forme de graphe couplé à un algorithme de supervision. Cependant, le succès de la navigation dépend fortement de la modélisation de la scène qui est accessible au robot. L'utilisateur doit donc s'attacher à fournir un graphe pour lequel un maximum de zones de visibilité communes ont été détectées.

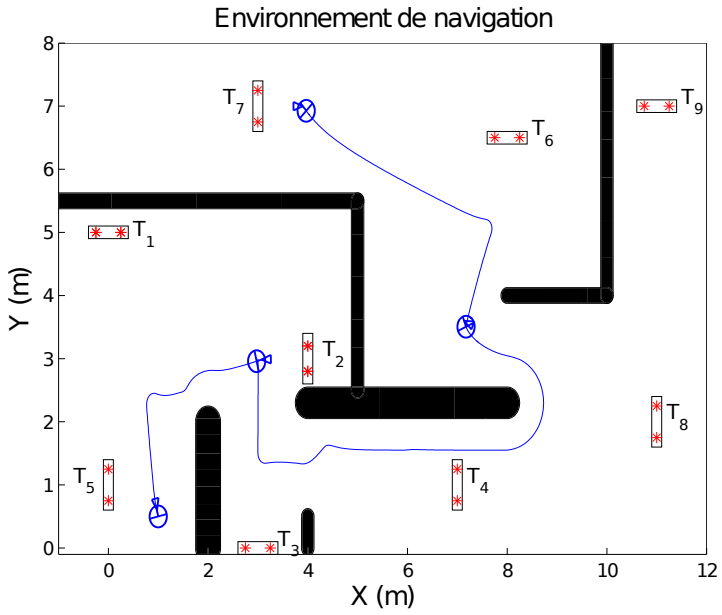
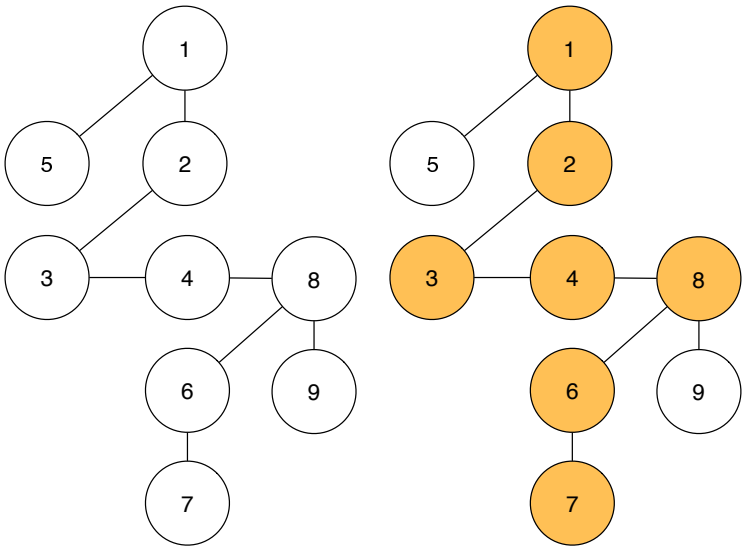


FIGURE 4.31 – Evolution du robot dans la scène



(a) Carte topologique (b) chemin planifié

FIGURE 4.32 – Graphe pour la troisième simulation

4.3 Conclusion

Dans ce chapitre, nous nous sommes intéressés à la navigation au long cours d'un robot mobile dans un environnement encombré d'obstacles possiblement occultants. Pour cela nous nous sommes focalisés sur deux sous-problèmes : les risques de collision occasionnés par les obstacles et la réalisation de longs déplacements. Il y a donc un volet local et un autre global. Concernant le premier aspect, nous avons repris la stratégie de commande proposée dans [Cadenat 1999]. Celle-ci consiste à réaliser la navigation à l'aide de deux correcteurs : l'un dédié à l'asservissement visuel et l'autre à l'évitement d'obstacles que nous avons rappelé brièvement. De plus, nous avons présenté deux techniques d'enchaînement permettant de garantir la continuité de la loi de commande appliquée au robot. Sur la base de ces résultats, nous nous sommes ensuite intéressés au volet global, qui constitue le cœur de ce chapitre. Nous avons ici cherché à étendre la portée de la navigation qui était jusque là limitée par les capacités de perception de la caméra. Pour cela nous avons proposé de coupler une carte topologique à un algorithme de supervision permettant d'étendre les stratégies de commande sus-mentionnées. Les résultats obtenus montrent la pertinence et l'efficacité de cette approche qui bénéficie de trois avantages principaux : une représentation allégée de l'environnement, la tolérance des occultations et la garantie de non-collision. Il devient maintenant possible pour le robot de réaliser de grands déplacements par asservissement visuel dans des environnements peu connus.

CHAPITRE 5

CONCLUSION

Dans ce manuscrit, nous nous sommes intéressés à la navigation d'un robot mobile dans un environnement encombré d'obstacles possiblement occultants. Le système robotique étant muni d'une caméra et d'un télémètre laser, nous avons exploité les techniques d'asservissement visuel 2D pour guider le robot vers son objectif. Celui-ci est défini par une mesure dans l'image relative à un amer d'intérêt et correspond à la situation devant être atteinte par la caméra. Cependant, ces techniques ne permettent de réaliser la tâche considérée que si le robot évolue dans un environnement libre. En effet, la présence d'obstacles peut conduire à des collisions, des occultations ou à la non visibilité de l'amer d'intérêt dès le début de la tâche. Nous nous sommes focalisés sur les deux derniers problèmes mentionnés.

Afin de répondre à la perte du signal visuel due à une occultation, nous avons choisi de nous appuyer sur les solutions développées dans [Folio 2007]. Les techniques proposées dans ces travaux consistent à reconstruire les indices visuels lorsqu'ils ne sont plus disponibles. Elles sont basées sur l'intégration d'un système dynamique particulier et nécessitent une valeur initiale fiable de la profondeur à l'instant où se produit l'occultation. Notre première contribution a donc consisté en le développement d'une paire prédicteur/correcteur capable d'estimer cette donnée de manière rapide et précise. Nous avons d'abord montré qu'un estimateur exploitant seulement deux images s'avérait inefficace dans un contexte réaliste. Nous avons alors amélioré cette approche en utilisant un plus grand nombre d'images dans le but de limiter la sensibilité du processus aux bruits de mesures. Pour cela nous avons d'abord analysé la commandabilité de la base mobile seule puis celle de la caméra. Nous avons ainsi pu exhiber une séquence de commandes équivalentes permettant de relier mathématiquement deux images quelconques. Les résultats précédents peuvent alors être ré-utilisés pour construire un estimateur exploitant plusieurs images et fournissant une valeur pertinente de la profondeur en contexte réaliste. Sur cette base, nous avons ensuite développé une méthode permettant de déterminer automatiquement les indices visuels de référence sans aucune connaissance *a priori*. Ce calcul peut être réalisé au cours de la navigation ou lors d'une phase d'initialisation. Grâce à ces travaux, nous sommes maintenant en mesure de traiter correctement le problème de la perte du signal visuel, quelle qu'en soit son origine (occultation, panne de la caméra ou du traitement d'images, utilisation temporaire de la caméra pour une autre tâche, etc.). Le robot est donc devenu capable d'évoluer sur la base d'indices visuels virtuels pendant un laps de temps donné.

Dans le prolongement de ces travaux, nous nous sommes focalisés sur des missions pour lesquelles l'amer d'intérêt n'est pas visible initialement. Cela se produit généralement lorsque l'on cherche à réaliser de longs déplacements. Il s'agit alors d'une navigation au long cours. Pour

répondre à ce problème, la stratégie de navigation élaborée consiste à définir une séquence d'amers permettant d'atteindre la cible finale tout en gérant les problèmes liés à la présence d'obstacles. Pour cela nous avons couplé une carte topologique à un algorithme de supervision. La première fournit les informations globales nécessaires pour effectuer de longs déplacements. Elle est définie par un graphe dont les nœuds correspondent aux amers discriminants de l'environnement. Deux nœuds étant connectés si l'on détecte une zone de visibilité commune, il est possible de déterminer la séquence d'amers permettant d'atteindre le but à l'aide d'une commande référencée vision. La tâche de navigation globale peut alors être vue comme une suite de sous-navigations locales. L'algorithme de supervision quant à lui regroupe un ensemble de correcteurs locaux permettant de réaliser respectivement l'asservissement visuel, l'évitement d'obstacle, la recherche de la cible suivante et la réorientation. Les deux premiers sont issus de travaux antérieurs tandis que les seconds ont été synthétisés spécifiquement pour notre problème. La sélection du correcteur adéquat est réalisée en ligne à l'aide des informations extéroceptives fournies par les capteurs embarqués. De plus, lors des transitions, la continuité de la loi de commande est garantie par le biais d'une méthode dédiée. L'ensemble des techniques développées pour l'estimation de la profondeur et la navigation au long cours a été validé en simulation et/ou expérimentation, démontrant la pertinence des approches proposées.

Ces travaux ont permis d'ouvrir un certain nombre de perspectives intéressantes. Tout d'abord, nous nous focalisons sur les aspects vision. Nos méthodes s'appuyant sur des cibles artificielles, l'extension logique consiste à exploiter des amers naturels. Pour cela, nous pensons que les points de Harris et les descripteurs SIFT/SURF sont des outils pertinents. En effet, ceux-ci permettent de caractériser un amer naturel par des descripteurs invariants à plusieurs paramètres tels que l'exposition ou bien l'angle de vue. Il apparaît alors possible de réutiliser les algorithmes de reconstruction et de commande dans un environnement non instrumenté. De plus, à l'aide de ces descripteurs il peut être intéressant de caractériser de manière générique des éléments récurrents dans les scènes de navigation (portes, couloirs, fenêtres, etc.), ce qui permettra de réduire la phase d'apprentissage significativement. En effet, celle-ci consistera simplement à transformer le plan des locaux en un graphe topologique. On pourra alors s'affranchir de l'étape de pré-navigation. A plus long terme, nous envisageons aussi d'élargir la gamme des tâches réalisables aux missions d'exploration. Il s'agirait ici de construire la carte topologique de l'environnement de manière automatique sans que le robot cherche à atteindre une situation prédéfinie.

Nous nous focalisons maintenant sur des aspects liés à l'estimation et la commande. Comme nous l'avons vu, la méthode de reconstruction proposée est spécifique à notre système robotique et à des indices visuels de type point. Si le second aspect n'est pas fondamentalement gênant, le premier apparaît par contre clairement limitatif. C'est pourquoi nos premiers efforts dans ce domaine consisteront à lever cette contrainte. Pour cela nous nous appuierons sur les travaux présentés dans [Folio 2007]. Ceux-ci proposent une expression analytique de l'évolution des points pour un torseur cinématique de la caméra constant, valable pour tout robot porteur. Nous chercherons à calculer un torseur constant équivalent en suivant un principe similaire à celui développé dans ce manuscrit. Nous obtiendrons ainsi une méthode de reconstruction des indices visuels plus générique. D'un point de vue davantage orienté vers la commande trois aspects nous semblent particulièrement intéressants. En premier lieu, nous pensons que l'utilisation du zoom peut apporter des améliorations significatives pour la réalisation des tâches de navigation au long cours. En effet, la distance pouvant être parcourue étant limitée par la portée de la caméra, il nous semble intéressant d'essayer de l'accroître en exploitant

judicieusement cet outil. Ensuite, nous souhaitons nous intéresser à la commande prédictive. En effet, disposant d'un outil d'estimation des indices visuels et de leur profondeur performant, il apparaît intéressant de le coupler avec ce type d'approche. Il sera ainsi possible de prendre en compte de nouvelles contraintes telles que la saturation des actionneurs. Enfin, comme nous avons pu le voir dans les simulations de navigation au long cours, le correcteur garantissant la non-collision ne permet pas d'anticiper les obstacles. Cela conduit à des trajectoires peu naturelles du point de vue de l'utilisateur. Nous proposons donc de développer une nouvelle stratégie de contournement répondant à ce problème. Elle sera déterminée sur la base des informations visuelles et proximétriques afin de définir un mouvement du robot mieux adapté. Enfin, nous pensons aussi qu'une partie importante des travaux futurs devra être consacré à la prise en compte de la dynamique de l'environnement. En effet, d'un point de vue sécurité, il sera nécessaire de considérer des obstacles mobiles, en particulier le cas de piétons susceptibles de se trouver dans un voisinage proche du robot. Ce dernier point, qui nécessitera de repenser les stratégies d'évitement et les techniques d'estimation mises en place jusqu'ici, sera essentiel pour permettre au véhicule de naviguer de manière sûre et autonome dans des environnements où le robot côtoie l'homme.

ANNEXE A

CALCUL DU TORSEUR CINÉMATIQUE DE LA CAMÉRA

Dans cette annexe, nous présentons les calculs permettant d'établir l'expression du torseur cinématique de la caméra pour le système robotique modélisé dans la figure A.1. Celui-ci est déterminé par rapport au repère de la scène R_O et exprimé dans le repère caméra R_C. Dans un premier temps, nous chercherons à déterminer son expression dans le repère de la base mobile R_M puis la projetterons dans R_C.

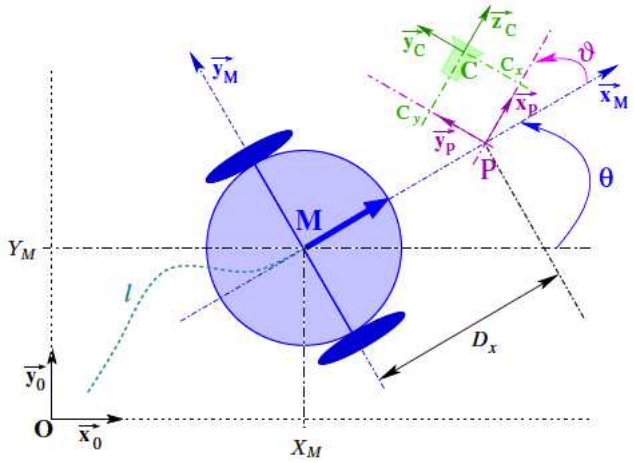


FIGURE A.1 – Modèle du système robotique

Tout d'abord, nous rappelons les expressions de $\mathcal{R}_{R_M/R_P}$ et $\mathcal{R}_{R_P/R_C}$, qui sont respectivement les matrices de passage entre R_P et R_M, et R_C et R_P :

$$\mathcal{R}_{R_M/R_P} = \begin{pmatrix} \cos(\vartheta) & -\sin(\vartheta) & 0 \\ \sin(\vartheta) & \cos(\vartheta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad \mathcal{R}_{R_P/R_C} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{pmatrix} \tag{A.1}$$

De plus, le torseur T_{C/R_O} s'écrit comme suit :

$$T_{C/R_O} = \begin{pmatrix} \overrightarrow{V}_{C/R_O} \\ \overrightarrow{\Omega}_{R_C/R_O} \end{pmatrix} = \begin{pmatrix} \overrightarrow{V}_{C/R_M} + \overrightarrow{V}_{M/R_O} + \overrightarrow{\Omega}_{R_M/R_O} \wedge \overrightarrow{MC} \\ \overrightarrow{\Omega}_{R_C/R_M} + \overrightarrow{\Omega}_{R_M/R_O} \end{pmatrix} \tag{A.2}$$

Le calcul de T_{C/R_O} nécessite de déterminer $\overrightarrow{MC}$, $T_{C/R_M} = \left[\overrightarrow{V}_{C/R_M} \quad \overrightarrow{\Omega}_{R_C/R_M}\right]$ et $T_{M/R_O} = \left[\overrightarrow{V}_{M/R_O} \ \overrightarrow{\Omega}_{R_M/R_O}\right]$. Nous allons donc décomposer le calcul de T_{C/R_O} exprimé dans R_C de la manière suivante :

- Calcul de T_{M/R_O} exprimé dans R_M.
- Calcul de T_{C/R_M} donnée dans R_M.
- Calcul de $\overrightarrow{MC}$ exprimé dans R_M.
- Détermination de T_{C/R_O} donné dans R_M à partir des trois résultats précédents et de l'équation (A.2).
- Détermination de T_{C/R_O} exprimé dans R_C à partir de T_{C/R_O} donné dans R_M et de la matrice de passage entre les repères R_M et R_C.

- **Calcul de $T^{R_M}_{M/R_O}$**

  A partir de la géométrie du robot présentée sur la figure A.1, $T^{R_M}_{M/R_O}$ est donné par :

$$T^{R_M}_{M/R_O} = \begin{pmatrix} V^{R_M}_{M/R_O} \\ \Omega^{R_M}_{R_M/R_O} \end{pmatrix} = \begin{pmatrix} \upsilon \\ 0 \\ 0 \\ 0 \\ 0 \\ \omega \end{pmatrix} \tag{A.3}$$

- **Calcul de $T^{R_M}_{C/R_M}$**

  Le torseur T_{C/R_M} est défini de la manière suivante :

$$T_{C/R_M} = \begin{pmatrix} \overrightarrow{V}_{C/R_M} \\ \overrightarrow{\Omega}_{R_C/R_M} \end{pmatrix} = \begin{pmatrix} \overrightarrow{V}_{C/R_P} + \overrightarrow{V}_{P/R_M} + \overrightarrow{\Omega}_{R_P/R_M} \wedge \overrightarrow{PC} \\ \overrightarrow{\Omega}_{R_C/R_P} + \overrightarrow{\Omega}_{R_P/R_M} \end{pmatrix} \tag{A.4}$$

  Les points P et C étant respectivement liés aux repères R_M et R_P, les vitesses $\overrightarrow{V}_{P/R_M}$ et $\overrightarrow{V}_{C/R_P}$ sont nulles. De plus, R_C étant fixe par rapport à R_P, $\overrightarrow{\Omega}_{R_C/R_P} = \overrightarrow{0}$. L'expression (A.4) du torseur T_{C/R_M} se réduit donc à :

$$T_{C/R_M} = \begin{pmatrix} \overrightarrow{\Omega}_{R_P/R_M} \wedge \overrightarrow{PC} \\ \overrightarrow{\Omega}_{R_P/R_M} \end{pmatrix} \tag{A.5}$$

  Le calcul de T_{C/R_M} dans R_M nécessite donc la détermination de $\overrightarrow{\Omega}_{R_P/R_M}$ et de $\overrightarrow{PC}$ dans ce repère. A l'aide de la géométrie du robot, PC^{R_P} est directement donné par le vecteur $[C_x \ C_y \ C_z]^T$. Pour obtenir PC^{R_M} à partir de PC^{R_P}, nous utilisons la matrice de passage R_{R_M/R_P} comme suit :

$$PC^{R_M} = \mathcal{R}_{R_M/R_P} PC^{R_P} \tag{A.6}$$

  Nous obtenons alors :

$$PC^{R_M} = \begin{pmatrix} C_x \cos(\vartheta) - C_y \sin(\vartheta) \\ C_x \sin(\vartheta) + C_y \cos(\vartheta) \\ C_z \end{pmatrix} \tag{A.7}$$

Pour déterminer complètement $T_{C/R_M}^{R_M}$, il nous reste à calculer la vitesse de rotation de R_P par rapport à R_M. Compte tenu de la géométrie du robot représenté sur la figure A.1, $\Omega_{R_P/R_M}^{R_M}$ est donné par le vecteur $[0\ 0\ \varpi]^T$. A l'aide de cette dernière équation et des relations (A.5) et (A.7), il est possible de déduire l'expression du torseur cinématique de la caméra dans R_M :

$$T_{C/R_M}^{R_M} = \begin{pmatrix} V_{C/R_M}^{R_M} \\ \Omega_{R_C/R_M}^{R_M} \end{pmatrix} = \begin{pmatrix} -C_x\sin(\vartheta) - C_y\cos(\vartheta) \\ C_x\cos(\vartheta) - C_y\sin(\vartheta) \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \varpi \tag{A.8}$$

- **Calcul de $\overrightarrow{MC}$ dans R_M**

Pour obtenir l'expression de $\overrightarrow{MC}$, nous décomposons ce dernier à l'aide de la relation de Chasles de la manière suivante :

$$\overrightarrow{MC} = \overrightarrow{MP} + \overrightarrow{PC} \tag{A.9}$$

A l'aide de la géométrie du robot, nous obtenons le résultat suivant :

$$MC^{R_M} = \begin{pmatrix} D_x + C_x\cos(\vartheta) - C_y\sin(\vartheta) \\ C_x\sin(\vartheta) + C_y\cos(\vartheta) \\ h_r + C_z \end{pmatrix} \tag{A.10}$$

- **Calcul de $T_{C/R_O}^{R_M}$**

A l'aide des équations (A.3), (A.8) et (A.10), le torseur cinématique de la caméra par rapport au repère de la scène s'exprime dans R_M de la manière suivante :

$$T_{C/R_O}^{R_M} = \begin{pmatrix} 1 & -(C_x\sin(\vartheta) + C_y\cos(\vartheta)) & -(C_x\sin(\vartheta) + C_y\cos(\vartheta)) \\ 0 & D_x + C_x\cos(\vartheta) - C_y\sin(\vartheta) & C_x\cos(\vartheta) - C_y\sin(\vartheta) \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \end{pmatrix} \dot{q} \tag{A.11}$$

- **Calcul de $T_{C/R_O}^{R_C}$**

Il reste maintenant à projeter le torseur dans R_C. La matrice de passage entre R_M et R_C est classiquement définie par la relation suivante :

$$\mathcal{R}_{R_M/R_C} = \mathcal{R}_{R_M/R_P}\mathcal{R}_{R_P/R_C} \tag{A.12}$$

Le torseur cinématique de la caméra par rapport à R_O et exprimé dans R_C est donc donné par :

$$T_{C/R_O}^{R_C} = \begin{pmatrix} \mathcal{R}_{R_M/R_C}^T & 0_{(3,3)} \\ 0_{(3,3)} & \mathcal{R}_{R_M/R_C}^T \end{pmatrix} T_{C/R_O}^{R_M} \tag{A.13}$$

où $0_{(3,3)}$ est une matrice carrée et nulle de dimensions 3. Tous calculs faits, nous obtenons :

$$T_{C/R_O}^{R_C} = \begin{pmatrix} V_{\vec{x}_C} \\ V_{\vec{y}_C} \\ V_{\vec{z}_C} \\ \Omega_{\vec{x}_C} \\ \Omega_{\vec{y}_C} \\ \Omega_{\vec{z}_C} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ -\sin(\vartheta) & C_x + D_x\cos(\vartheta) & C_x \\ \cos(\vartheta) & -C_x + D_x\sin(\vartheta) & -C_y \\ 0 & -1 & -1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \dot{q} \tag{A.14}$$

ANNEXE B

EQUATIONS DE L'ESTIMATEUR POUR UNE COMMANDABILITÉ EN 2 COUPS

Dans cette annexe nous présentons les équations nécessaires à l'estimation de la profondeur à l'aide du prédicteur/correcteur pour une commandabilité en deux coups. La démarche menant aux équations étant identique à celle utilisée dans la section 3.2.2.3, nous ne donnons ici que les résultats. Dans un premier temps, nous rappelons que nous avons calculé une séquence de deux commandes équivalentes permettant d'atteindre un état $\chi(k)$ depuis un état initial $\chi(k-j)$. Ces commandes sont définies comme suit :

$$\dot{q}_{e1} = \begin{pmatrix} v_{e1} \\ \omega_{e1} \\ 0 \end{pmatrix} \qquad \dot{q}_{e2} = \begin{pmatrix} 0 \\ \omega_{e2} \\ \varpi_{e2} \end{pmatrix} \tag{B.1}$$

Une rapide analyse montre qu'il n'est pas nécessaire d'établir les équations de l'estimateur dans les seize cas comme nous l'avons mentionné dans la section 3.2.2.3. En effet, grâce à la forme particulière des vitesses équivalentes (B.1) nous pouvons limiter notre démarche aux cas suivants :

- $v_{e1} \neq 0$, $\omega_{e1} \neq 0$, $\omega_{e2} \neq 0$ et $\omega_{e2} \neq -\varpi_{e2}$
  Le robot est commandé en deux coups enchaînant successivement deux fois le cas n°1 (cf. section 3.1.1). Les équations de l'estimateur correspondantes ont été présentées dans la section 3.2.2.3.

- $v_{e1} \neq 0$ et $\omega_{e2} = 0$
  Le robot est commandé en un coup avec $\dot{q}_{ec} = (v_{e1}\ \omega_{e1}\ \varpi_{e2})^T$. Nous sommes donc dans un des quatre cas pour lesquels les équations sont données dans la section 3.2.3.3.

- $v_{e1} = 0$
  Le robot est commandé en un coup avec $\dot{q}_{ec} = (0\ \omega_{e1} + \omega_{e2}\ \varpi_{e2})^T$. Nous sommes donc dans un des quatre cas pour lesquels les équations sont données dans la section 3.2.3.3.

- $v_{e1} \neq 0$, $\omega_{e1} = 0$, $\omega_{e2} \neq 0$ et $\omega_{e2} \neq -\varpi_{e2}$
  Le robot est commandé en deux coups enchaînant successivement le cas n°4 et le cas n°1.

Les équations de l'estimateur avec $\dot{q}(k-j) = \dot{q}_{e1}$ et $\dot{q}(v) = \dot{q}_{e2}$ sont alors les suivantes :

$$Num_P^j = \mu_3^2 + f^2\lambda^2 \tag{B.2}$$

$$Den_P^j = -\mu_3\gamma_3 + \mu_3\tilde{X}(k) - \left(\frac{f\tilde{Y}(k-j)\cos(A_2)}{f\mu_2} - \frac{f\sin(A_2)}{\mu_2} - \tilde{Y}(k)\right) f\lambda \tag{B.3}$$

avec

$$A_2 = (\omega(v) + \varpi(v))T_e$$
$$\gamma_1 = -v(k-j)\cos(\theta(k-j))T_e$$
$$\mu_1 = fv(k-j)\sin(\theta(k-j))T_e - \tilde{Y}(k-j)\gamma_1$$
$$\gamma_2 = \left(\tfrac{\mu_1}{f} - D_x\sin(\theta(v)) - \tfrac{v(v)}{\omega(v)}\cos(\theta(v)) + C_y\right)\sin(A_2)$$
$$+\left(D_x\cos(\theta(v)) - \tfrac{v(v)}{\omega(v)}\sin(\theta(v)) + C_x\right)\cos(A_2)$$
$$-D_x\cos(\theta(k)) + \tfrac{v(v)}{\omega(v)}\sin(\theta(k)) - C_x$$
$$\mu_2 = \frac{\tilde{Y}(k-j)}{f}\sin(A_2) + \cos(A_2)$$
$$\gamma_3 = \frac{\tilde{X}(k-j)}{\mu_2}$$
$$\mu_3 = -\frac{\gamma_2\tilde{X}(k-j)}{\mu_2} - \gamma_1\tilde{X}(k-j)$$
$$\lambda = \left(\tfrac{\mu_1}{f} - \tfrac{\tilde{Y}(k-j)\gamma_2}{f\mu_2} - D_x\sin(\theta(v)) - \tfrac{v(v)}{\omega(v)}\cos(\theta(v)) + C_y\right)\cos(A_2)$$
$$-\left(-\tfrac{\gamma_2}{\mu_2} + D_x\cos(\theta(v)) - \tfrac{v(v)}{\omega(v)}\sin(\theta(v)) + C_x\right)\sin(A_2)$$
$$+D_x\sin(\theta(k)) + \tfrac{v(v)}{\omega(v)}\cos(\theta(k)) - C_y$$

– $v_{e1} \neq 0$, $\omega_{e1} = 0$, $\omega_{e2} \neq 0$ et $\omega_{e2} = -\varpi_{e2}$

Le robot est commandé en deux coups enchaînant successivement le cas n°4 et le cas n°3. Les équations de l'estimateur avec $\dot{q}(k-j) = \dot{q}_{e1}$ et $\dot{q}(v) = \dot{q}_{e2}$ sont alors les suivantes :

$$Num_P^j = \mu_2^2 + \frac{\lambda^2}{\varpi^3(v)} \tag{B.4}$$

$$Den_P^j = \mu_2(\tilde{X}(k-j) - \tilde{X}(k)) - \frac{\lambda(\tilde{Y}(k-j) - \tilde{Y}(k))}{\varpi^2(v)} \tag{B.5}$$

avec

$$c_1 = \cos(\theta(k)) - \cos(\theta(v))$$
$$c_2 = \sin(\theta(k)) - \sin(\theta(v))$$
$$\gamma_1 = -v(k-j)\cos(\theta(k-j))T_e$$
$$\mu_1 = fv(k-j)\sin(\theta(k-j))T_e - \tilde{Y}(k-j)\gamma_1$$
$$\gamma_2 = -\frac{v(v)}{\varpi(v)}c_2 - D_xc_1$$
$$\mu_2 = (\gamma_1 + \gamma_2)\tilde{X}(k-j)$$
$$\lambda = -fv(v)c_1 - f\varpi(v)D_xc_2 + \varpi(v)\gamma_1 - \varpi(v)\tilde{Y}(k-j)\mu_2$$

ANNEXE C

LISTE DES PUBLICATIONS

V. Cadenat, D. Folio, A. Durand Petiteville *A comparaison of two sequencing techniques to perform a vision-based task in a cluttered environment* Advanced Robotics Journal 2012.

A. Durand Petiteville, S. Hutchinson, V. Cadenat, M. Courdesses *2D visual servoing for long range navigation in cluttered environment* 50th IEEE Conference on Decision and Control and European Control Conference, Orlando (USA), 12-15 December 2011, 6p.

A. Durand Petiteville, M. Courdesses, V. Cadenat, P. Baillion *On-line estimation of the reference visual features. Application to a vision based long range navigation task* IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2010), Taipei (Taiwan), 18-22 October 2010, 8p.

A. Durand Petiteville, V. Cadenat, M. Courdesses *A unified initialization phase to improve visual servoing in an unknown environment* IFAC Symposium on Intelligent Autonomous Vehicles (IAV 2010), Lecce (Italy), 6-8 September 2010, 6p.

A. Durand Petiteville, M. Courdesses, V. Cadenat *A new predictor/corrector pair to estimate the visual features depth during a vision-based navigation task in an unknown environment* International Conference on Informatics in Control, Automation and Robotics (ICINCO 2010), Funchal (Portugal), 15-18 June 2010, pp.268-274.

A. Durand Petiteville, V. Cadenat, M. Courdesses, F. Delpech de Frayssinet, A. Magassouba *A comparison of several approaches to perform a vision-based long range navigation* 10th IEEE International Workshop on Electronics, Control, Modelling, Measurement and Signals (ECMS 2011), Liberec (Czech Republic), 1-3 June 2011, 10p.

A. Durand Petiteville, M. Courdesses, V. Cadenat *Reconstruction of the features depth to improve the execution of a vision-based task* 9th International Workshop on Electronics, Control, Modelling, Measurement and Signals (ECMS 2009), Mondragon (Spain), 8-10 July 2009, 6p.

BIBLIOGRAPHIE

[Astolfi & Ortega 2003] A. Astolfi et R. Ortega. *Immersion and Invariance : A New Tool for Stabilization and Adaptive Control of Nonlinear Systems*. IEEE Trans. Automat. Contr., vol. 48, no. 4, pages 590 – 606, 2003.

[Bay *et al.* 2006] H. Bay, T. Tuytelaars et L. Van Gool. *SURF : Speeded Up Robust Features*. In 9th European Conference on Computer Vision, May 2006.

[Bellot 2002] D. Bellot. *Contribution à l'analyse et à la synthèse de schémas de commande référencée vision*. PhD thesis, Université Paul Sabatier - Toulouse III, 2002.

[Blanc *et al.* 2005] G. Blanc, Y. Mezouar et P. Martinet. *Indoor navigation of a wheeled mobile robot along visual routes*. In International Conference on Robotics and Automation, Barcelona, Spain, 2005.

[Bonin-Font *et al.* 2008] F. Bonin-Font, F. Ortiz et G. Oliver. *Visual navigation for mobile robots : a survey*. Journal of intelligent and robotic systems, vol. 53, no. 3, page 263, 2008.

[Booij *et al.* 2007] O. Booij, B. Terwijn, Z. Zivkovic et B. Krose. *Navigation using an appearance based topological map*. In IEEE Int. Conf. on Robotics and Automation, pages 3927– 3932, Rome, Italy, 2007.

[Cadenat *et al.* 2012] V. Cadenat, D. Folio et A. Durand Petiteville. *A comparison of two sequencing techniques to perform a vision-based navigation task in a cluttered environment*. Advanced Robotics, 2012.

[Cadenat 1999] V. Cadenat. *Commande référencée multi-capteurs pour la navigation d'un robot mobile*. PhD thesis, Université Paul Sabatier - Toulouse III, 1999.

[Canudas de Wit *et al.* 1993] C. Canudas de Wit, H. Khennouf, C. Samson et O. J. Sordalen. Nonlinear control design for mobile robots, volume 11. World Scientific Publishing, 1993.

[Chaumette & Hutchinson 2006] F. Chaumette et S. Hutchinson. *Visual Servo Control, Part 1 : Basic Approaches*. IEEE Robotics and Automation Magazine, vol. 13, no. 4, 2006.

[Chaumette *et al.* 1994] F. Chaumette, S. Boukir, P. Bouthemy et D. Juvin. *Optimal estimation of 3D structures using visual servoing*. In IEEE Conf. Computer Vision and Pattern Recognition, CVPR'94, pages 347–354, Seattle, Washington, June 1994.

[Chaumette *et al.* 1996] F. Chaumette, S. Boukir, P. Bouthemy et D. Juvin. *Structure from controlled motion*. IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 18, no. 5, pages 492–504, May 1996.

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

[Chaumette 1998] F. Chaumette. *Potential problems of stability and convergence in image-based and position-based visual servoing.* In D. Kriegman, G . Hager et A.S. Morse, editeurs, The Confluence of Vision and Control, pages 66–78. LNCIS Series, No 237, Springer-Verlag, 1998.

[Chaumette 2002] Chaumette. La commande des robots manipulateurs, chapitre Asservissement visuel. Hermés, 2002.

[Chaumette 2004] F. Chaumette. *Image moments : a general and useful set of features for visual servoing.* Robotics, IEEE Transactions on, vol. 20, no. 4, pages 713 – 723, aug. 2004.

[Chen & Birchfield 2006] Zhichao Chen et S.T. Birchfield. *Qualitative vision-based mobile robot navigation.* In Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on, pages 2686 –2692, may 2006.

[Cheng *et al.* 2006] Yang Cheng, M.W. Maimone et L. Matthies. *Visual odometry on the Mars exploration rovers - a tool to ensure accurate driving and science imaging.* Robotics Automation Magazine, IEEE, vol. 13, no. 2, pages 54 –62, june 2006.

[Cherubini & Chaumette 2009] A. Cherubini et F. Chaumette. *Visual navigation with a time-independent varying reference.* In IEEE Int. Conf. on Intelligent Robots and Systems, IROS'09, pages 5968–5973, St Louis, USA, October 2009.

[Cherubini & Chaumette 2010] A. Cherubini et F. Chaumette. *A redundancy-based approach to obstacle avoidance applied to mobile robot navigation.* In Proc. of IEEE Int. Conf. on Intelligent Robots and Systems, Taipei, Taiwan, 2010.

[Cherubini *et al.* 2011] A. Cherubini, F. Spindler et F. Chaumette. *A Redundancy-Based Approach for Visual Navigation With Collision Avoidance.* In ICVTS proceedings, 2011.

[Choset *et al.* 2005] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki et S Thrun. Principles of robot motion. MIT Press, Boston, 2005.

[Cobzas & Zhang 2001] D. Cobzas et H. Zhang. *Mobile robot localization using planar patches and a stereo panoramic model.* In Vision Interface, pages 04–99, Ottawa, Canada, June 2001.

[Comport *et al.* 2004] A Comport, E. Marchand et F. Chaumette. *Robust model-based tracking for robot vision.* In IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, Sendai, Japan, Octobre 2004.

[Comport *et al.* 2010] A.I. Comport, E. Malis et P. Rives. *Real-time Quadrifocal Visual Odometry.* International Journal of Robotics Research, Special issue on Robot Vision, vol. 29, 2010.

[Corke & Hutchinson 2001] P.I. Corke et S.A. Hutchinson. *A new partitioned approach to image-based visual servo control.* Robotics and Automation, IEEE Transactions on, vol. 17, no. 4, pages 507 –515, aug 2001.

[Courbon 2009] Jonathan Courbon. *Navigation de robots mobiles par mémoire sensorielle.* PhD thesis, Université Blaise Pascal, Clermont-Ferrand, France, December 2009.

[Crisan & Doucet 2002] D. Crisan et A. Doucet. *A survey of convergence results on particle filtering methods for practitioners.* IEEE Transactions on Signal Processing, vol. 50, 2002.

[De Luca *et al.* 2008] De Luca, Oriolo et Giordano. *Features depth observation for image based visual servoing : theory and experiments.* Int. Journal of Robotics Research, vol. 27, no. 10, 2008.

[Deguchi 1998] K. Deguchi. *Optimal motion control for image-based visual servoing by decoupling translation and rotation.* In Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on, volume 2, pages 705 –711 vol.2, oct 1998.

[Desouza & Kak 2002] G.N. Desouza et A.C. Kak. *Vision for mobile robot navigation : a survey.* Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 24, no. 2, pages 237 –267, feb 2002.

[Dijkstra 1971] E. W. Dijkstra. A short introduction to the art of programming. Aug 1971.

[Djeridane 2004] Djeridane. *Sur la commandabilité des systèmes non linéaires à temps discret.* PhD thesis, Université Paul Sabatier - Toulouse III, 2004.

[Dormand & Prince 1980] J.. Dormand et P.J. Prince. *A family of embedded Runge-Kutta formulae.* Journal of Computational and Applied Mathematics, vol. 6, pages 19 – 26, 1980.

[Folio & Cadenat 2008] Folio et Cadenat. Computer vision, chapitre 4. Xiong Zhihui ; INTECH, 2008.

[Folio 2007] D. Folio. *Stratégies de commandes référencées multi-capteurs et gestion de la perte du signal visuel pour la navigation d'un robot mobile.* PhD thesis, Université Paul Sabatier - Toulouse III, 2007.

[Gans *et al.* 2003] N. Gans, S. Hutchinson et P. Corke. *Performance Tests for Visual Servo Control Systems, with Application to Partitioned Approaches to Visual Servo Control.* International Journal of Robotics Research, no. 10-11, pages 955–981, oct-nov 2003.

[Garcia-Aracil *et al.* 2005] N. Garcia-Aracil, E. Malis, R. Aracil-Santonja et C. Perez-Vidal. *Continuous visual servoing despite the changes of visibility in image features.* IEEE Transactions on Robotics and Automation, vol. 21, 2005.

[Gaspar *et al.* 2000] J. Gaspar, N. Winters et J. Santos-Victor. *Vision-based navigation and environmental representations with an omni-directional camera.* IEEE transactions on robotics and automation, vol. 6, no. 6, pages 890–898, 2000.

[Geraerts & Overmars 2002] R. Geraerts et M. H. Overmars. *A comparative study of probabilistic roadmap planners.* In Proc. Workshop on the Algorithmic Foundations of Robotics (WAFR'02), pages 43–57, Nice, France, December 2002.

[Gibson & Marques 2008] J. Gibson et O. Marques. *Stereo Depth with a Unified Architecture GPU.* In IEEE CVPR Workshops, Anchorage, USA, June 2008.

[Goedemé *et al.* 2007] Toon Goedemé, Marnix Nuttin, Tinne Tuytelaars et Luc Van Gool. *Omnidirectional Vision based Topological Navigation.* International Journal of Computer Vision, vol. 74, no. 3, pages 219–236, 2007.

[Harris & Stephens 1988] C. Harris et M. Stephens. *A combined corner and edge detector.* In Proceedings of the 4th Alvey Vision Conference, Manchester, UK, 1988.

[Hart *et al.* 1968] P. E. Hart, N. J. Nilsson et B. Raphael. *A Formal Basis for the Heuristic Determination od Minimum Cost Paths.* Systems Science and Cybernetics, IEEE Transactions on, vol. 4, no. 2, pages 100–107, 1968.

[Horaud & Monga 1995] R. Horaud et O. Monga. Vision par ordinateur, outils fondamentaux (deuxième édition). Hermès, 1995.

[Horaud 1987] Radu Horaud. *New Methods for Matching 3-D Objects with Single Perspective Views.* Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. PAMI-9, no. 3, pages 401 –412, may 1987.

[Hutchinson *et al.* 1996] S. Hutchinson, G.D. Hager et P.I. Corke. *A tutorial on visual servo control.* IEEE Trans. on Rob. and Automation, vol. 12, no. 5, pages 651–670, 1996.

[Jackson *et al.* 2004] J. Jackson, A. Yezzi et S. Soatto. *Tracking deformable objects under severe occlusions.* In IEEE International Conference on Decision and Control, Atlanta, USA, Decembre 2004.

[Jerian & Jain 1991] C. Jerian et R. Jain. *Structure from motion : a critical analysis of methods.* IEEE Transactions on systems, Man, and Cybernetics, vol. 21, no. 3, pages 572–588, 1991.

[Jones *et al.* 1997] S.D. Jones, C. Andresen et J.L. Crowley. *Appearance based process for visual navigation.* In Intelligent Robots and Systems, 1997. IROS '97., Proceedings of the 1997 IEEE/RSJ International Conference on, volume 2, pages 551 –557 vol.2, sep 1997.

[Kalman 1960] R. Kalman. *A new approach to linear filtering and prediction problems.* Transactions of the ASME-Journal of Basic Engineering, vol. 82, 1960.

[Kavraki *et al.* 1996] L.E. Kavraki, P. Svestka, J.-C. Latombe et M.H. Overmars. *Probabilistic roadmaps for path planning in high-dimensional configuration spaces.* Robotics and Automation, IEEE Transactions on, vol. 12, no. 4, pages 566 –580, aug 1996.

[Krajnìk & Přeučil 2008] T. Krajnìk et L. Přeučil. A simple visual navigation system with convergence property. H. Bruyninckx et al. (Eds.), 2008.

[Krösea *et al.* 2001] B.J.A. Krösea, N. Vlassisa, R. Bunschotena et Y. Motomura. *A probabilistic model for appearance-based robot localization.* Image and Vision Computing, vol. 19, pages 381–391, 2001.

[Lamiraux *et al.* 2004] F. Lamiraux, D. Bonnafous et O. Lefebvre. *Reactive path deformation for nonholonomic mobile robots.* Robotics, IEEE Transactions on, vol. 20, no. 6, pages 967 – 977, dec. 2004.

[LaValle 1998] S M LaValle. *Rapidly-Exploring Random Trees : A New Tool for Path Planning.* In, vol. TR 98-11, no. 98-11, pages 98–11, 1998.

[Lepetit & Fua 2006] V. Lepetit et P. Fua. *Monocular based-model 3d tracking of rigid objects.* found. Trends. Comput. Graph. Vis., vol. 1, 2006.

[Levine 1996] William S. Levine. The control handbook. CRC Press Handbook, 1996.

[Lowe 1999] D.G. Lowe. *Object recognition from local scale-invariant features.* In Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on, volume 2, pages 1150 –1157 vol.2, 1999.

[Malis *et al.* 1999] E. Malis, F. Chaumette et S. Boudet. *2D 1/2 visual servoing.* IEEE Trans. on Rob. and Automation, vol. 15, no. 2, pages 234–246, 1999.

[Mansard & Chaumette 2007] N. Mansard et F. Chaumette. *Task sequencing for sensor-based control.* IEEE Trans. on Robotics, vol. 23, no. 1, pages 60–72, February 2007.

[Matsumoto *et al.* 1996] Y. Matsumoto, M. Inaba et H. Inoue. *Visual navigation using view-sequenced route representation.* In IEEE Int. Conf. on Robotics and Automation, pages 83–88 –2692, Minneapolis, USA, 1996.

[Matthies *et al.* 1989] Matthies, Kanade et Szeliski. *Kalman filter-based algorithms for estimating depth in image sequences.* Int, Journal of Computer Vision, vol. 3, no. 3, pages 209–238, 1989.

[Morbidi & Prattichizzo 2009] F. Morbidi et D. Prattichizzo. *Range estimation from moving camera : an Immersion and Invariance approach.* In IEEE 2009 International Conference on Robotics and Automation, Kobe, Japan, May 2009.

[Nister *et al.* 2004] D. Nister, O. Naroditsky et J. Bergen. *Visual odometry.* In Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on, volume 1, pages I–652 – I–659 Vol.1, june-2 july 2004.

[Okabe *et al.* 2000] A. Okabe, B. Boots, K. Sugihara et S. N. Chiu. Spatial tessellations - concepts and applications of voronoi diagrams. John Wiley, 2000.

[Paletta *et al.* 2001] L. Paletta, S. Frintrop et J. Hertzberg. *Robust localization using context in omnidirectional imaging.* In Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on, volume 2, pages 2072 – 2077 vol.2, 2001.

[Pissard-Gibollet & Rives 1995] Pissard-Gibollet et Rives. *Applying visual servoing techniques to control a mobile handeye system.* In IEEE Int., Conf. on Robotics and Automation, Nagoya, Japan, 1995.

[Pissard-Gibollet 1993] Pissard-Gibollet. *Conception et commande par asservissement visuel d'un robot mobile.* PhD thesis, Ecole des mines de Paris, 1993.

[Remazeilles 2004] A. Remazeilles. *Navigation à partir d'une mémoire d'images.* PhD thesis, Université de Rennes I, 2004.

[Royer *et al.* 2007] E. Royer, M. Lhuillier, M. Dhome et J-M. Lavest. *Monocular Vision for Mobile Robot Localization and Autonomous Navigation.* International Journal of Computer Vision, vol. 74, no. 3, pages 237–260, 2007.

[Samson *et al.* 1991] Samson, Borgne et Espiau. *Robot Control : The task function approach.* Oxford science publications, 1991.

[Samson 1992] C. Samson. *Path following and time-varing feedback statbilization of a wheeled mobile robot.* In ICARCV, Singapore, Septembre 1992.

[Siegwart & Nourbakhsh 2004] R. Siegwart et I.R. Nourbakhsh. Introduction to autonomous mobile robots. A bradford book, Intelligent robotics and autonomous agents series. The MIT Press, 2004.

[Sim & Dudek 1999] R. Sim et G. Dudek. *Learning visual landmarks for pose estimation.* In Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on, volume 3, pages 1972 –1978 vol.3, 1999.

[Soatto & Perona 1998a] S. Soatto et P. Perona. *Reducing "structure from motion" : A general framework for dynamic vision part 1 : Modelling.* IEEE Transactions on PAttern Analysis and Machine Intelligence, vol. 9, 1998.

[Soatto & Perona 1998b] S. Soatto et P. Perona. *Reducing "structure from motion" : A general framework for dynamic vision part 2 : Implementation and experimental assessment.* IEEE Transactions on PAttern Analysis and Machine Intelligence, vol. 9, 1998.

[Souères & Cadenat 2003] P. Souères et V. Cadenat. *Dynamical sequence of multi-sensor based tasks for mobile robots navigation.* In SYROCO, Wroclaw, Poland, September 2003.

[Souères *et al.* 1998] P. Souères, T. Hamel et V. Cadenat. *A path following controller for wheeled robots wich allows to avoid obstacles during the transition phase.* In IEEE, Int. Conf. on Robotics and Automation, Leuven, Belgium, May 1998.

[Tahri & Chaumette 2005] O. Tahri et F. Chaumette. *Point-based and region-based image moments for visual servoing of planar objects.* IEEE Trans. on Robotics, vol. 21, no. 6, pages 1116–1127, December 2005.

[Thrun *et al.* 2000] S. Thrun, W. Burgard et D. Fox. *A real time algorithm for mobile robot mapping with applications to multi-robot and 3d mapping.* In IEEE International Conference on Robotics and Automation, San Francisco, CA, USA, April 2000.

[Thrun 2002] S. Thrun. Robotic mapping : A survey. Feb 2002.

[Triggs *et al.* 2000] Bill Triggs, Philip McLauchlan, Richard Hartley et Andrew Fitzgibbon. *Bundle Adjustment : A Modern Synthesis.* In Bill Triggs, Andrew Zisserman et Richard Szeliski, editeurs, Vision Algorithms : Theory and Practice, volume 1883, pages 153–177. Springer Berlin / Heidelberg, 2000.

[Vassalo *et al.* 2000] R. Vassalo, H. Schneebeli et J. Santos-Victor. *Visual servoing and appearance for navigation.* Robotics and autonomous systems, 2000.

[Victorino & Rives 2004] A.C. Victorino et P. Rives. *An hybrid representation well-adapted to the exploration of large scale indoors environments.* In IEEE International Conference on Robotics and Automation, pages 2930–2935, New Orleans, USA, 2004.

[Wolf *et al.* 2002] J. Wolf, W. Burgard et H. Burkhardt. *Robust vision-based localization for mobile robots using an image retrieval system based on invariant features.* In Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on, volume 1, pages 359 – 365 vol.1, 2002.

[Yagi *et al.* 2005] Yasushi Yagi, Kousuke Imai, Kentaro Tsuji et Masahiko Yachida. *Iconic Memory-Based Omnidirectional Route Panorama Navigation.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, pages 78–87, 2005.

Navigation référencée multi-capteurs d'un robot mobile en environnement encombré

Dans ce travail, nous nous intéressons à la navigation référencée vision d'un robot mobile équipé d'une caméra dans un environnement encombré d'obstacles possiblement occultants. Pour réaliser cette tâche, nous nous sommes appuyés sur l'asservissement visuel 2D. Cette technique consiste à synthétiser une loi de commande basée sur les informations visuelles renvoyées par la caméra embarquée. Le robot atteint la situation désirée lorsque les projections dans l'image de l'amer d'intérêt, appelés indices visuels, atteignent des valeurs de consigne prédéfinies. La navigation par asservissement visuel 2D nécessite de s'intéresser à trois problèmes : garantir l'intégrité du robot vis-à-vis des obstacles, gérer les occultations des amers d'intérêts et réaliser de longs déplacements.

Nos contributions portent sur les deux derniers problèmes mentionnés. Dans un premier temps nous nous sommes intéressés à l'estimation des indices visuels lorsque ceux-ci ne sont plus disponibles à cause d'une occultation. La profondeur étant un paramètre déterminant dans ce processus, nous avons développé une méthode permettant de l'estimer. Celle-ci est basée sur une paire prédicteur/correcteur et permet d'obtenir des résultats exploitables malgré la présence de bruits dans les mesures. Dans un second temps, nous nous sommes attachés à la réalisation de longs déplacements par asservissement visuel. Cette technique nécessitant de percevoir l'amer d'intérêt dès le début de la tâche, la zone de navigation est limitée par la portée de la caméra. Afin de relaxer cette contrainte, nous avons élaboré un superviseur que nous avons ensuite couplé à une carte topologique intégrant un ensemble d'amers caractéristiques de l'environnement. La tâche de navigation globale peut alors être décomposée sous la forme d'une séquence d'amers à atteindre successivement, la sélection et l'enchainement des mouvements nécessaires étant effectués au sein du superviseur. Les travaux ont été validés par le biais de simulations et d'expérimentations, démontrant la pertinence et l'efficacité de l'approche retenue.

Mots-clés : robotique mobile, asservissement visuel, gestion des occultations, estimation de la profondeur, navigation au long court, carte topologique

Multi sensor based navigation in cluttered environment

This work focuses on the navigation of a mobile robot equipped with a camera in a cluttered environment. To perform such a task, we propose to use the image based visual servoing (IBVS). This method consists in designing a control law using visual features provided by the camera. These features are defined by the projection of a characteristic landmark on the image plane. The IBVS based navigation requires to address three issues : the robot security with respect to the obstacles, the management of the occlusions and the long range navigation realization.

Our contributions are mainly focused on the two last mentioned problems. First, we have dealt with the visual features estimation problem during occlusions. As the visual features depth is an important parameter in this process, we have developed a predictor/corrector pair able to estimate its value on-line. This method has provided nice results, even when the used measures are noisy. Second, we have considered the problem of performing a long range navigation with an IBVS. However, classically, using this kind of controller greatly limits the realizable displacement because the reference landmark must be seen from the beginning to the end of the mission. To relax this constraint, we have developed a topological map and a supervision algorithm which have then been coupled. The first one contains the most characteristic landmarks of the environment. Using this information, it is possible to divide the global navigation task into a sequence of landmarks which must be successively reached. The supervision algorithm then allows to select the right task at the right instant and to guarantee a smooth switch between the different motions. Our works have been validated by simulations and experimentations, demonstrating the efficiency of our approach.

Keywords : mobile robotics, visual servoing, occlusion management, depth estimation, long range navigation, topological mapping