Commande prédictive pour la réalisation de tâches d'asservissement visuel successives

Nicolas Cazy

ANNÉE 2016

UNIVERSITÉ DE RENNES 1

UNIVERSITE BRETAGNE LOIRE

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Bretagne Loire

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Informatique

Ecole doctorale Matisse

présentée par

Nicolas CAZY

préparée à l'unité de recherche Inria Rennes - Bretagne Atlantique
Institut National de Recherche
en Informatique et en Automatique

Commande prédictive pour la réalisation de tâches d'asservissement visuel successives

Thèse soutenue à Rennes
le 29 novembre 2016

devant le jury composé de :

Véronique PERDEREAU
Professeur de l'Université Pierre et Marie Curie, Paris / Présidente

Viviane CADENAT
Maître de conférence UPS, Toulouse / Rapporteuse

El Mustapha MOUADDIB
Professeur de l'Université de Picardie Jules Verne, Amiens / Rapporteur

Paolo ROBUFFO GIORDANO
Directeur de recherche CNRS Irisa, Rennes / Encadrant

Pierre-Brice WIEBER
Chargé de recherche Inria, Grenoble / Encadrant

François CHAUMETTE
Directeur de recherche Inria, Rennes / Directeur de thèse

Remerciements

Je tiens à remercier dans un premier temps les membres de mon jury de thèse pour l'attention qu'ils ont acceptée de porter à mon travail, ainsi que pour leurs nombreuses remarques et questions. Je remercie ainsi Véronique Perdereau pour avoir assuré la présidence de ce jury de thèse. Merci également aux rapporteurs Viviane Cadenat et El Mustapha Mouaddib pour toutes leurs remarques qui m'ont permis d'apprécier un regard extérieur extrêmement pertinent sur mes travaux, et d'améliorer la qualité de ce manuscrit.
Merci à François Chaumette, Paolo Robuffo Giordano et Pierre-Brice Wieber pour m'avoir permis de traiter ce sujet dont toutes les thématiques m'ont motivé à entreprendre ces travaux de recherche. Merci pour leurs aides précieuses, leurs conseils avisés, et pour toutes les notions qu'ils m'ont fait découvrir et que j'ai approfondies durant ces trois années de doctorat. Je les remercie finalement pour leurs supervisions et relectures indispensables durant la rédaction des articles, du manuscrit ou lors de la préparation de la soutenance.

Je tiens à remercier également l'ensemble de l'équipe Lagadic pour tout ce qu'elle m'a apporté professionnellement et personnellement. Merci à Céline d'avoir assurer mon intégration au sein de l'équipe et du laboratoire lors des premiers jours. Merci de l'aide précieuse apportée par Hélène pour la préparation des missions et de la soutenance. Je remercie également les permanents Marie, Eric, Alex, Vincent et Fabien qui ont toujours su se rendre disponible pour répondre à mes questions, quelles soient scientifiques, personnelles, ou professionnelles. Leur accompagnement lors de ces années de doctorat a été pour moi très important, et j'espère que les derniers arrivés chez Lagadic auront la chance d'en profiter. Je remercie l'ensemble des doctorants, des post-doctorants, des ingénieurs que j'ai pu croiser durant ces 3 ans pour l'agréable ambiance qu'ils ont su installer et faire perdurer dans l'équipe pendant tous les moments de travail ou de pauses café. Je remercie Aurélien pour son sens de la formule si particulier, son perfectionnisme, son ouverture d'esprit, ce roadtrip incroyable aux USA, et pour beaucoup d'autres choses que la pudeur nordique m'empêche de lister. Merci à Lucas pour sa bienveillance, son humour, son esprit, et pour m'avoir convaincu que Fougères, finalement, c'est pas si mal (même si Amiens c'est mieux). Merci à Quentin pour sa gaieté et son entrain permanents dans le travail, les discussions à Supelec, les activités en dehors du laboratoire, et pour se lancer avec moi dans cette nouvelle aventure que j'espère promise à de futurs beaux projets. Merci finalement à Pierre, Vishnu, Suman, Aly, Lesley, Jason,

Thomas, Giovanni, Souriya, Bryan, Firas, Ide Flore, Noël, Pedro, Fabrizio, Muhammad, Riccardo et Joven avec qui j'ai passé de formidables moments à l'Inria, aux fléchettes ou au billard.

Je remercie également les membres du laboratoire MIS de l'Université de Picardie Jules Verne à Amiens, et en particulier Guillaume Caron et Abdelhamid Rabhi pour m'avoir fait découvrir le monde de la recherche et m'avoir encouragé à y participer. Merci également à Frédéric Collet pour avoir su insuffler l'enthousiasme des métiers de l'électronique et de l'informatique à quelques Cazy.

Merci à Thibaut Griffart, Nicolas Kisgen, Léo Simonin et Jean-Baptiste Vilain pour tout ce qu'ils ont fait et été pour moi depuis toutes ces années, et sans qui je n'aurais certainement pas entrepris de telles études.

Je remercie ma famille pour leur soutien inébranlable durant ces années de doctorat, mais surtout durant mes jeunes années d'étudiants qui m'ont permis de réaliser un parcours qui semblait tellement inatteignable il y a encore quelques temps. Merci à mes parents et à mon frère pour cette saine complicité que nous partageons, et pour l'infinie fierté qu'ils m'offrent à être toujours auprès de moi.

Je tiens finalement à remercier Laurène Barabaray pour tout ce qu'elle m'a apporté durant ces années de hauts et de bas, et pour sa patience lors de mes instants de doute durant lesquels elle a toujours su m'accompagner de la meilleure des manières. Merci à ses mots qui ont su me rassurer et me motiver. Merci à elle pour continuer à me questionner, me confronter, et m'inspirer quotidiennement.

Table des matières

Introduction

Un robot est un système mécatronique équipé de capteurs, d'actionneurs, et d'un système de commande. Ce dernier exploite les informations des capteurs pour en déduire les actions que doivent réaliser les actionneurs afin que le système accomplisse une tâche désirée. Les nombreuses possibilités de conception des robots permettent à leurs utilisateurs de les employer dans d'innombrables applications qui ne nécessitent plus aucun recours à une intervention humaine. L'intérêt de ces systèmes est d'accomplir différentes tâches plus ou moins complexes à de mutliples échelles, de manière autonome, et de façon plus rapide, plus précise et plus sûre que n'importe quel être humain. Ces robots offrent de formidables perspectives dans différents domaines tels que l'industrie, la médecine, le service ou encore l'aide à la personne. Cependant, les développements de tels systèmes sont confrontés encore à des problématiques importantes sur tous les aspects de leur conception. Celles-ci s'articulent autour de trois domaines distincts : la perception, l'action et la commande.
Les travaux présentés dans ce document s'intéressent plus particulièrement au développement de méthodes permettant de commander un système robotique basée sur la perception de l'environnement via l'utilisation de capteurs de vision telles que les caméras.

Sujet

L'étude présentée dans ce document se concentre en grande partie sur la méthode de commande robotique utilisant des informations visuelles qui est la plus caractéristique, à savoir l'asservissement visuel.

L'asservissement visuel est le domaine de la robotique qui combine les travaux réalisés en vision par ordinateur avec les études menées en matière de commande robotique. Le principe consiste à utiliser les informations extraites d'une ou de plusieurs images capturées par une ou plusieurs caméra, et d'en générer une commande à appliquer aux différents actionneurs d'un robot. Cette méthode permet de réaliser diverses tâches avec le robot tel que le suivi d'objets mobiles pour qu'ils ne quittent pas le champ de vue de la caméra, ou le déplacement du robot vers une pose relative à un objet pour manipuler ou observer celui-ci depuis un point de vue désiré. Cette approche a déjà fait l'objet de nombreux travaux de recherche mais présente encore des problématiques intéressantes que nous traitons dans ce document. Un des aspects les plus significatifs de

l'asservissement visuel est le fait qu'un objet observé par une caméra ne puisse jamais quitter le champ de vision de celle-ci. Aucun événement qui empêcherait que la caméra puisse réaliser une observation de l'objet ne doit donc intervenir. Nous revenons dans ce document sur les méthodes de prédiction permettant à l'objet de ne plus être observé par la caméra, tout en assurant la réalisation de la tâche d'asservissement visuel. Nous proposons ensuite une méthode inédite qui permet d'assurer que les résultats de ces prédictions correspondent fidèlement à ce qu'observerait la caméra si les mesures étaient toujours disponibles.

En raison de la diminution des temps de calcul des ordinateurs embarqués, il est possible aujourd'hui de développer des algorithmes de plus en plus complexes qui permettent à la fois d'extraire de nouvelles informations des images, mais également d'intégrer des stratégies de commande plus efficaces. Les recherches qui sont menées aujourd'hui dans ce domaine tendent à obtenir une plus grande précision, une meilleure robustesse et une meilleure réactivité de la part des systèmes robotiques en les soumettant à des conditions de mesure, à des environnements, et à des tâches de plus en plus contraignantes. En ce sens, il est aujourd'hui possible d'utiliser des stratégies de commande qui exploitent habituellement d'autres types de capteurs que les caméras, mais qui offrent des outils et des perspectives intéressantes pour la réalisation de nouvelles tâches d'asservissement visuel. En vue d'apporter de nouvelles contributions au domaine de l'asservissement visuel, nous présentons dans ce document l'un de ces schémas de commande, à savoir la commande prédictive basée modèle.

La commande prédictive basée modèle est une méthode qui anticipe les futurs états d'un système robotique pour pouvoir générer une séquence de commandes à appliquer à ses actionneurs. Sa mise en œuvre requiert des mesures provenant de capteurs, un modèle dynamique qui permet de prédire les états du système sur un horizon de temps fini, une fonction objectif à atteindre, et des contraintes à satisfaire. Un algorithme d'optimisation est utilisé pour calculer la séquence de commandes optimales à appliquer aux actionneurs du robot de manière à atteindre l'objectif désiré tout en satisfaisant les contraintes imposées. Lorsque de nouvelles mesures des capteurs sont disponibles, l'état du système est mis à jour, et une nouvelle séquence de commandes est calculée. Cette méthode a pour intérêt de limiter le champ d'action du système à ce qui lui est demandé d'accomplir, et à ce qui lui est physiquement possible de faire par le biais de la fonction objectif et des contraintes. Elle permet également d'anticiper les variations brutales de l'état du système, et donc de calculer des séquences de commande qui ne brusquent pas les actionneurs. De plus, grâce aux mesures régulièrement collectées elle s'adapte efficacement aux variations sporadiques de l'état du système. Enfin, du fait qu'elle puisse prendre en compte un grand nombre de variables différentes, on retrouve cette commande employée dans de nombreux domaines différents, tel que les secteurs de l'industrie pétro-chimique, agro-alimentaire, automobile, ou encore métallurgique, qui impliquent de nombreuses contraintes et variables en entrée et sortie de leur système.

Aujourd'hui, plusieurs travaux exploitent la commande prédictive basée modèle pour

réaliser des tâches d'asservissement visuel sur différents types de systèmes robotiques. Par exemple, il est possible de trouver des stratégies qui génèrent des commandes à appliquer à un robot muni d'une caméra en prédisant les coordonnées des informations visuelles d'un objet sur l'image, et en posant comme objectif de déplacer la caméra vers une position désirée, tout en contraignant celle-ci à garder l'objet observé dans son champ de vue. D'autres types d'applications qui utilisent les spécificités de la commande prédictive pour réaliser des tâches d'asservissement visuel existent, et il est encore possible d'en imaginer une multitude d'autres pour des systèmes ou des tâches plus ou moins complexes. Chaque application apporte une spécificité qui se répercute dans le schéma de commande au travers du modèle dynamique, de l'objectif de fonctionnement et des contraintes à satisfaire. En ce sens, nous présentons dans ce document une application qui utilise la commande prédictive permettant de réaliser une succession de tâches d'asservissement visuel en forçant une caméra embarquée sur un drone à se déplacer d'un point à un autre pour assister des robots mobiles à accomplir des tâches de suivi de trajectoire.

Structure du document

L'organisation de ce document s'articule autour de trois chapitres :

- Dans le premier, nous revenons sur l'emploi de la vision en robotique. L'objectif est de présenter les schémas de commande que nous souhaitons utiliser dans la suite du manuscrit. Nous définissons dans un premier temps un aperçu de l'évolution de la robotique depuis les premiers automates jusqu'aux robots modernes. Nous présentons ensuite les outils de vision par ordinateur qui utilisent les données d'une ou de plusieurs caméras et permettent d'extraire des informations relatives à l'environnement dans lequel le système robotique évolue. Puis nous définissons les différentes méthodes classiques d'asservissement visuel, ainsi que les domaines d'application dans lesquels on peut les retrouver. Enfin, nous présentons le principe de la commande prédictive basée modèle, ainsi que des exemples d'applications où l'on peut la retrouver pour réaliser des tâches d'asservissement visuel.
- Dans le second chapitre, nous étudions les cas pouvant causer la perte d'informations visuelles lors d'une tâche d'asservissement visuel. Nous présentons ensuite deux méthodes de prédiction qui permettent à la tâche d'être réalisée malgré la perte de ces informations. Puis nous proposons deux méthodes inédites de correction qui permetttent d'obtenir de bons résultats de prédiction, comparées à la méthode classique. Des résultats obtenus en simulation viennent illustrer l'intérêt de ces méthodes tandis que des résultats obtenus en situation réelle démontrent leur efficacité lors d'une tâche d'asservissement visuel réalisée à l'aide d'un bras robotique.
- Dans le dernier chapitre, une application qui exploite les caractéristiques de la commande prédictive est présentée. Des robots réalisent des tâches de suivi de trajectoires, tandis que les observations d'une caméra embarquée sur un drone

sont utilisées pour les aider à corriger leurs erreurs de déplacement. L'objectif de la commande est de forcer la caméra à se diriger vers les robots pour les observer régulièrement. Nous présentons alors le modèle dynamique qui est utilisé, ainsi que l'objectif de fonctionnement et les contraintes à satisfaire. Finalement, des résultats obtenus en simulation sont présentés afin de visualiser l'influence de notre schéma de commande sur les déplacements de la caméra et du drone, ainsi que sur les trajectoires empruntées par les robots.

Chapitre 1

Schémas de commandes pour robots avec capteurs de vision

Ce chapitre est axé sur l'utilisation de la vision par ordinateur en robotique. Dans un premier temps, nous présentons un aperçu des avancées techniques dans le domaine de la robotique depuis l'apparition des premiers automates. Nous revenons ensuite sur les nombreux domaines d'application dans lesquels les robots peuvent être employés tels que l'industrie, la médecine, le service ou l'aide à la personne.

Dans un second temps, nous introduisons les différents capteurs qui existent aujourd'hui et qui permettent aux robots de percevoir ce qui les entoure. Nous revenons en particulier sur les caméras, éléments indispensables dans le cadre de la vision par ordinateur, qui ont pour but de convertir la lumière qu'ils reçoivent en données numériques. Nous décrivons alors comment il est possible d'extraire des informations visuelles géométriques depuis une image capturée par une caméra.

Enfin, nous décrivons deux schémas de commande qui permettent à un robot d'accomplir des tâches de manière autonome, à savoir l'asservissement visuel et la commande prédictive. Nous revenons en détail sur la manière dont sont élaborés ces schémas, et plus particulièrement comment ils permettent d'exploiter les informations provenant de la vision par ordinateur.

1.1 La robotique

Les travaux réalisés en robotique consistent en l'élaboration de machines qui ont la faculté de réaliser des tâches de manière autonome. Les différentes solutions de conception de tels systèmes sont multiples, et tendent à répondre à des objectifs plus ou moins complexes. En revenant en détail sur l'histoire de la robotique, il apparaît qu'un des enjeux les plus significatifs de la robotique depuis ses débuts est de suppléer l'être humain dans des activités répétitives, dangereuses ou qui lui sont impossibles à réaliser physiquement.

Il existe aujourd'hui une variété très importante de robots dont les conceptions dépendent à la fois des applications visées mais également des technologies disponibles. On peut ainsi présenter les différents systèmes robotiques qui ont pu être développés à travers l'histoire, en revenant dans un premier temps sur les prémices de la robotique moderne, à savoir les premiers automates utilisant les techniques avancées de la mécanique et de l'horlogerie.

1.1.1 Les premiers automates

À la fin du XVe siècle, Léonard de Vinci dessine en détail le premier exemple historique d'automate complet, le chevalier mécanique [Ros06]. Celui-ci devait être capable de se lever, bouger sa tête, sa mâchoire et ses bras par le moyen de poulies et d'engrenages. Bien qu'aucune trace d'un prototype fabriqué par Léonard de Vinci lui-même n'ait été répertorié, une reproduction réalisée d'après les croquis originaux est actuellement exposée au musée de Berlin (Figure 1.1).

FIGURE 1.1 – Un chevalier mécanique basé sur les croquis de Léonard de Vinci, exposé à Berlin en 2005.

Au XVIIIe siècle, l'inventeur grenoblois Jacques de Vaucanson crée plusieurs types d'automates. Le premier est un canard capable de simuler plusieurs actions (Figure 1.2). Celui-ci est capable de boire, de manger, de cancaner et de digérer comme un animal réel le ferait. Vaucanson crée ensuite plusieurs automates de taille humaine, dont un joueur de flûte capable d'interpréter plusieurs morceaux [dV38]. Ce dernier joue de la flûte traversière en modifiant la position de ses doigts, la forme de ses lèvres et son débit

d'air. Pour cela, un poids entraîne un mécanisme constitué de chaînes et de câbles, et des soufflets créent le souffle. Durant la même période, la famille suisse Jaquet-Droz conçoit trois automates dont tous les mouvements sont programmables et basés sur l'horlogerie [CLR79] : l'écrivain qui retranscrit quarante caractères sur papier pour former un texte préalablement choisi, la musicienne qui interprète cinq motifs musicaux différents en suivant des yeux le mouvement de ses mains et qui simule la respiration humaine, et le dessinateur qui est capable de réaliser les portraits de Louis XV, d'un couple royal, d'un chien et d'un Cupidon. Il peut également souffler sur le papier pour faire disparaître les éclats de mine de crayon (Figure 1.3).

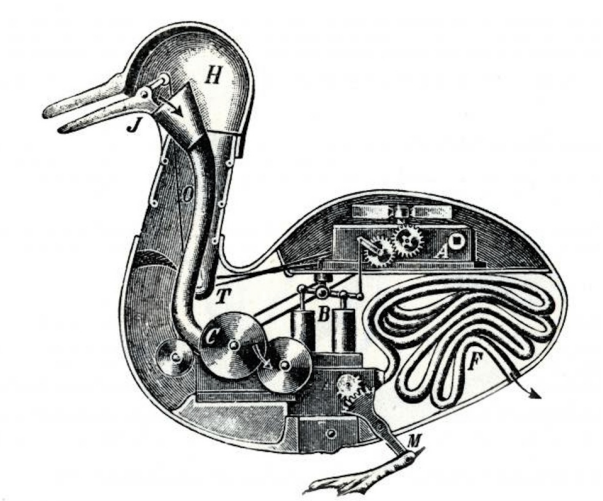


FIGURE 1.2 – Croquis du canard créé par Jacques de Vaucanson en 1738.

FIGURE 1.3 – Le dessinateur, la musicienne et l'écrivain de la maison Jaquet-Droz, exposés au musée d'Art et d'Histoire de Neuchâtel en Suisse.

On peut enfin citer les poupées Karakuri réalisées au Japon entre le XVIIIe et le XIXe siècle destinées à être utilisées soit au théâtre, soit durant des cérémonies religieuses, ou encore pour servir le thé chez des particuliers [Law97]. L'emploi de ressorts, de câbles, d'engrenages et de cames permettait à ces automates de se pencher ou de se déplacer en roulant d'une position à une autre et de faire demi-tour pour retourner à leur position initiale (Figure 1.4).

FIGURE 1.4 – Karakuri serveur de thé, exposé au musée national de la nature et des sciences de Tokyo.

Les automates présentés ci-dessus, dont les conceptions étaient basées sur les grandes avancées techniques en matière de mécanique et d'horlogerie, avaient un usage essentiellement récréatif et n'avaient pas la prétention d'apporter une aide particulière aux tâches accomplies par l'humain. Ils ont cependant été des inspirations certaines aux robots modernes développés depuis l'apparition des premières formes d'intelligences artificielles.

1.1.2 L'évolution de la robotique

Dès le début du XXe siècle, les premiers robots modernes combinant les technologies de la mécanique et de l'électronique sont apparus. L'objectif principal de ces machines est de réaliser des tâches plus ou moins complexes de manière totalement autonome, afin de remplacer ou d'assister les humains dans leurs tâches.

1.1.2.1 Les premiers robots autonomes

Suite aux travaux d'ingénieurs qui voulaient valider certaines hypothèses émises par des biologistes et des psychologues, les premières réalisations de systèmes robotiques sont apparues. Les comportements de ces derniers étaient alors fortement inspirés de ceux réalisés par l'animal ou l'humain. Le premier exemple de robot répertorié dans l'histoire capable de se déplacer de manière autonome est le chien électrique conçu par Hammond et Miessner en 1915 (Figure 1.5). Celui-ci était capable de suivre une source lumineuse [Oud09] sans qu'aucune intervention humaine ne soit nécessaire. Pour cela, il était équipé de roues, de deux moteurs et de deux capteurs photosensibles. Lorsque l'un des capteurs détectait une plus grande quantité de lumière que l'autre, les moteurs se mettaient à tourner et le robot s'orientait vers la source lumineuse. Ce robot est considéré comme le premier a être doté d'une intelligence artificielle puisqu'il est capable de prendre une décision de manière autonome. On voit apparaître depuis la création de ce dispositif la différence fondamentale entre les automates et les robots, à savoir la présence de capteurs capables d'interpréter d'une manière ou d'une autre les environnements dans lesquels évoluent les robots.

Dans les années suivantes, d'autres robots du même type ont été conçus. On peut citer le robot Philidog créé par le français Henry Piraux en 1928 [SS09], le robot Machina Speculatrix de l'américain Grey Walter créé en 1950 [Wal50] ou encore le robot Squee créé par son compatriote Edmund Berkeley en 1956 [Ber56]. Tous ces robots sont basés sur la même idée. Ils sont capables de suivre une source lumineuse grâce à des moteurs et à des capteurs photosensibles (Figure 1.6).

On doit également à l'américain Thomas Ross de grandes avancées en matière de commandes robotiques [Ros33]. Il crée tout d'abord un bras mécanique en 1933 capable de tracer un chemin pour sortir d'un labyrinthe, puis il réalise la même expérience en 1937, mais cette fois-ci avec un robot mobile qui évolue tout seul dans le labyrinthe (Figure 1.7). L'idée est que chaque robot est capable de tester et de se souvenir si un chemin mène ou non à la sortie, et ce à l'aide d'un dispositif mécanique constitué de disques utilisé comme mémoire par le robot.

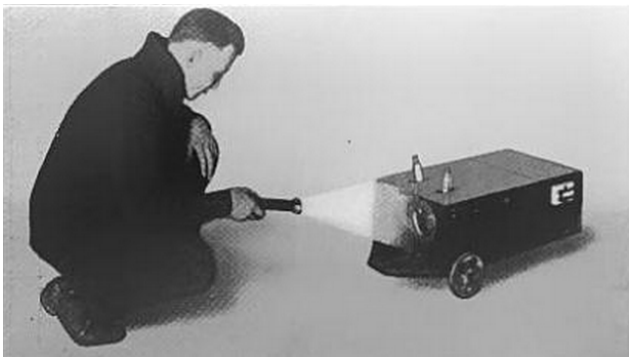
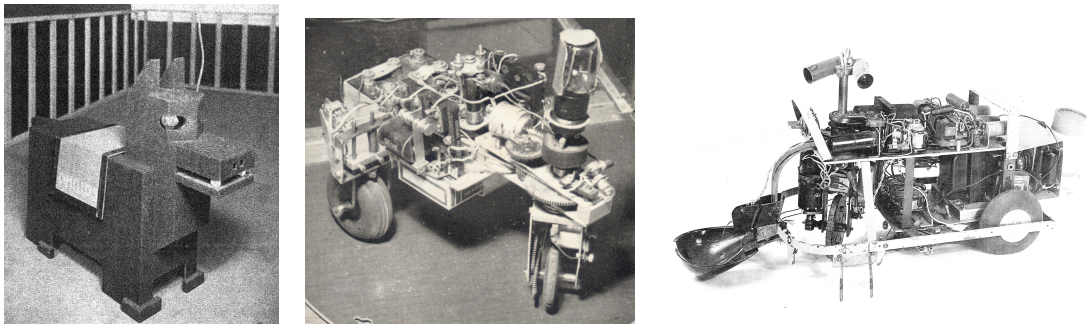


FIGURE 1.5 – Le chien électrique de Hammond et Miessner en 1915.



(a) (b) (c)

FIGURE 1.6 – Le début de la robotique. (a) Philidog de Henry Piraux en 1928. (b) Machina Speculatrix de Grey Walter en 1950. (c) Squee de Edmund Berkeley en 1956.

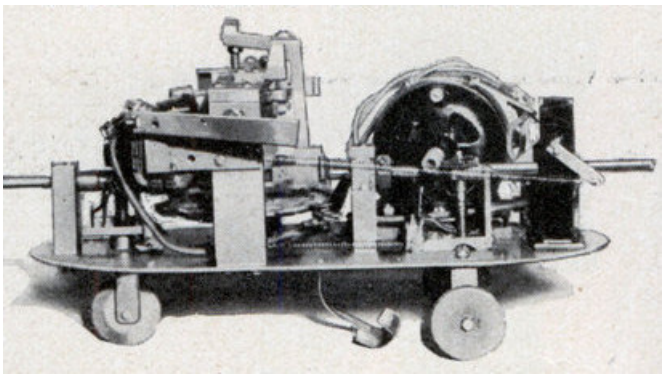


FIGURE 1.7 – Le robot mobile de Thomas Ross en 1937.

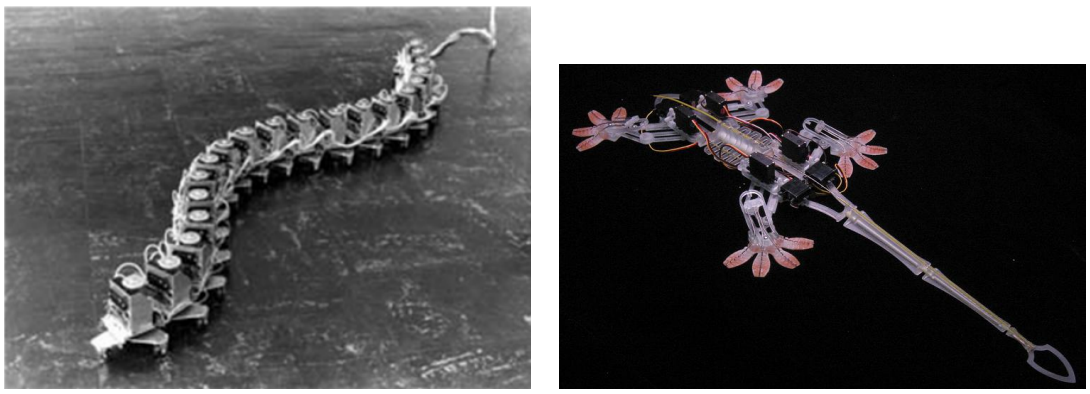
On voit apparaître dans les années suivantes des robots dotés de fonctionnalités de plus en plus avancées. Une des tendances suivie par beaucoup de chercheurs spécialisés dans la robotique est d'étudier ce qui peut être observé dans la nature, d'en retirer des intérêts pour leurs machines, et de trouver des solutions mécaniques ou électroniques afin d'intégrer de nouvelles fonctionnalités innovantes à leurs machines. Les robots issus de ces travaux de recherche composent l'ensemble de la robotique dite bio-inspirée.

1.1.2.2 La robotique bio-inspirée

En étudiant de manière détaillée certaines caractéristiques présentes chez les animaux, les chercheurs et ingénieurs ont développé des robots dotés de fonctionnalités originales, notamment en ce qui concerne leurs moyens de perception ou de locomotion.

Par exemple, en 1972, le japonais Shigei Hirose développe un robot dont le déplacement est inspiré par le serpent, l'ACM III (Figure 1.8a). Celui-ci est capable d'évoluer en suivant un mode de déplacement basé sur les ondulations latérales. Il est constitué de plusieurs modules articulés en contact avec le sol par l'intermédiaire de roues non

motorisées [Hir93]. Un autre moyen de locomotion issu du monde animal a été étudié par l'intermédiaire du robot Stickybot créé en 2006 au sein de l'Université de Stanford (Figure 1.8b). Les "pattes" de ce robot sont inspirées de celles du gecko, ce qui lui permet de grimper à la verticale sur des surfaces lisses telle qu'une vitre en verre [KST+07]. En ce qui concerne les nouveaux moyen de perception de l'environnement, on peut citer les travaux effectuées par Nicolas Franceschini du Centre National de la Recherche Scientifique (CNRS) de Marseille, qui utilisent des méthodes inspirées par les yeux des mouches sur des robots volants [FRS07]. Celles-ci permettent aux robots d'attérir ou d'éviter des obstacles en se basant sur le flux visuel, c'est-à-dire la vitesse de défilement des images comme le fait un insecte telle que la mouche.



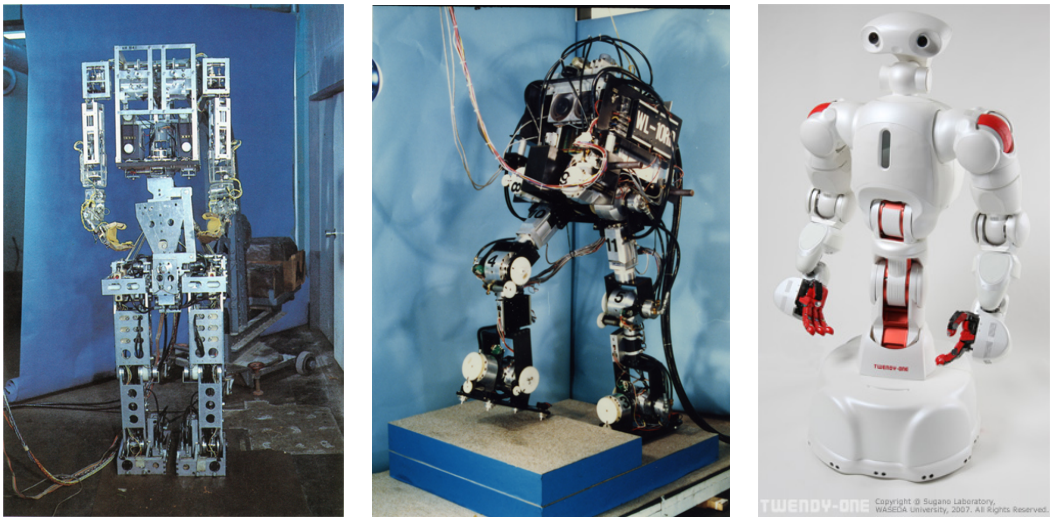
(a) (b)

FIGURE 1.8 – Les robots bio-inspirés. (a) L'ACM III de Shigei Hirose en 1972. (b) Le Stickybot de l'Université de Stanford en 2006.

Aujourd'hui, les robots bio-inspirés sont nombreux et variés. On trouve parmis eux certains dont les caractéristiques sont inspirées par celles de l'être humain, les robots humanoïdes.

1.1.2.3 La robotique humanoïde

On doit à l'université japonaise Waseda de grandes avancées en matière de robotique humanoïde (Figure 1.9), notamment WABOT-1 [KOK+74] en 1973, le premier modèle de robot bipède. Celui-ci était muni d'un système de conversation, de vision, de saisie d'objets, et était doté du premier système de marche statique sur deux jambes. La même équipe réalise en 1984 le système WL-10RD, pourvu uniquement de deux jambes [TIYK85]. Ce robot était le premier capable de réaliser une marche dynamique et de franchir un escalier. En 1999, la même université conçoit le robot Twendy-One qui se déplace sur des roues [IS09]. Ce robot est destiné à soutenir les personnes physiquement diminuées dans leurs activités quotidiennes. Il est capable par exemple de préparer un repas, ou d'aider une personne à se lever d'un lit et à s'installer sur une chaise roulante.



(a) (b) (c)

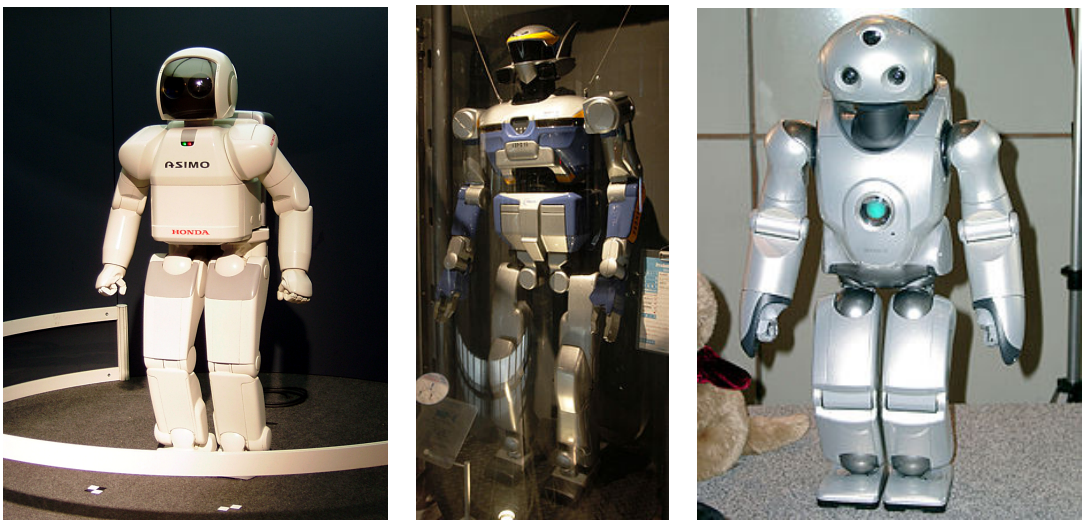
FIGURE 1.9 – Les robots de l'université Waseda (a) Le robot WABOT-1 en 1973. (b) Le système de marche WL-10RD en 1984. (c) Le robot Twendy-One en 1999.

À partir du début des années 2000, plusieurs grandes industries japonaises commencent à fabriquer leurs plateformes robotiques destinées à la recherche (Figure 1.10). Leur but est de développer des programmes complexes pour perfectionner leurs robots sur divers types d'aspects, tels que l'interaction avec l'homme ou avec l'environnement.
On peut citer en exemple le robot Asimo de l'entreprise HONDA créé en 2000 [Hon07], doté de fonctionnalités extrêmement avancées. Asimo est ainsi le premier robot humanoïde à pouvoir changer de trajectoire lors de sa marche, il est capable de danser, de garder son équilibre sur des surfaces instables et de manipuler des objets.
En 2003, la société Kawada industries développe le robot HRP-2 [KKK+02] vendu dans une dizaine de laboratoires à travers le monde, notamment en France sur le site du LAAS à Toulouse. Il sera suivi par les robots HRP-3 et HRP-4C développés respectivement en 2007 et 2009.
Entre 2003 et 2006, Sony développe plusieurs types de robot dont Qrio qui a comme caractéristique sa petite taille (58 cm de hauteur), ce qui lui permet de disposer d'une grande stabilité lors de ses mouvements [Ish04].

En France, la société Aldebaran est un acteur majeur dans l'élaboration des robots humanoïdes (Figure 1.11). En 2006, elle crée le robot Nao utilisé dans une grande quantité de laboratoires et d'événements grand public. On le retrouve lors de rassemblements populaires tels que les tournois internationaux de football robotique RoboCup [KAK+97]. Nao est également utilisé comme outil d'apprentissage. Sa programmation a été rendue assez accessible pour qu'il soit notamment utilisé dans la pédagogie en informatique [JLP+11].
En 2012, Aldebaran présente Romeo au public et aux laboratoires [Roba]. Ce robot est

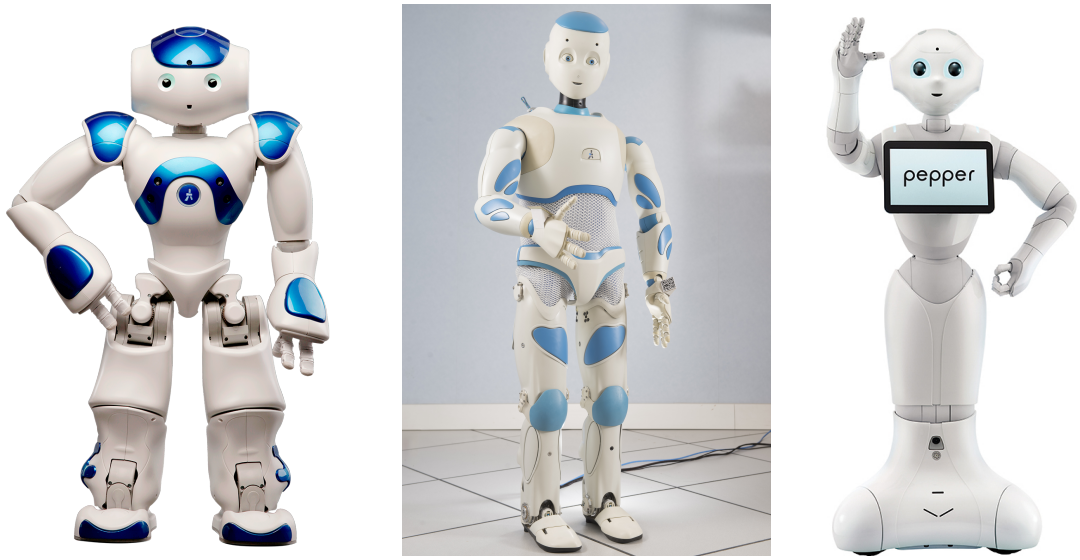
destiné à aider les personnes diminuées physiquement dans leurs tâches quotidiennes, et à être perçu comme un compagnon de vie par n'importe qui. Il a pour but, par exemple, de suivre la prise d'un traitement par un patient résidant dans son domicile, ou de lui rappeler un rendez-vous médical, ou encore d'appeler les secours en cas de malaise du patient. L'interaction sociale qu'il doit entretenir avec l'humain est un aspect très important. On peut citer les études menées au sujet de l'impact émotionnel de Romeo sur des sujets humains [DD12]. Celles-ci démontrent l'importance du comportement du robot en fonction de l'état émotionnel de la personne avec qui il intéragit.
En 2014, la société présente le robot Pepper se déplaçant sur roues [Robb]. Celui-ci est capable de percevoir et de reproduire des émotions et est destiné à intéragir socialement avec les êtres humains. Une des caractéristiques du robot est sa personnalisation par le biais d'applications téléchargeables, ainsi que sa faculté à évoluer avec son utilisateur selon ses goûts et habitudes. On le trouve déjà à l'usage dans des foyers japonais, ou dans les boutiques japonaises de l'entreprise SoftBank où il est chargé d'accueillir et de conseiller les clients de l'enseigne.

Nous avons présenté dans cette section un aperçu des évolutions de la robotique depuis ses débuts. Il existe toutefois encore un large panel de différents types de systèmes robotiques. La section qui suit propose de revenir en partie sur ces derniers en présentant les grands intérêts de la robotique pour différents domaines, qu'ils soient industriels, militaires ou médicaux.



(a) (b) (c)

FIGURE 1.10 – Les robots humanoïdes japonais (a) Asimo de Honda en 2000. (b) HRP-2 de Kawada industries en 2003. (c) Qrio de Sony en 2006.



(a) (b) (c)

FIGURE 1.11 – Les robots humanoïdes créés par la société Aldebaran (a) Nao en 2006. (b) Romeo en 2012. (c) Pepper en 2014.

1.1.3 Les domaines d'application de la robotique

Les robots présentés ci-dessus ont permis de grandes avancées scientifiques techniques. Mais d'autres domaines que la recherche sont aujourd'hui concernés par le développement des systèmes robotiques.

Les bras robotiques par exemple sont utilisés de manière massive dans l'industrie pour réaliser des tâches de soudage, de peinture ou d'assemblage. Ces robots sont capables de réaliser ces opérations de manière plus précise et plus rapide qu'un humain. On les retrouve employés pour intervenir dans des milieux à risques tels que les centrales nucléaires, ou sur différentes chaînes de montage, notamment celles en automobile. On peut ainsi citer le robot Unimate (Figure 1.12) vendu par la société Unimation au début des années 1960 et utilisé sur les lignes d'assemblage du groupe General Motors [DMA+11]. D'autre part, différents projets sont aujourd'hui développés pour intégrer des robots humanoïdes sur des chaînes de montage. On peut citer à titre d'exemple le projet H2020 Comanoide impliquant la société Airbus qui a pour objectif d'utiliser les robots humanoïdes pour réaliser des opérations de fabrication et de maintenance sur les fuselages des avions. Un des enjeux du projet est que ces opérations seront effectuées soit de manière autonome soit en collaboration avec des opérateurs humains classiques. On peut alors envisager que le déploiement intensif de robots humanoïdes dans le secteur industriel pourrait provoquer la disparition future de métiers difficiles ou dangereux.

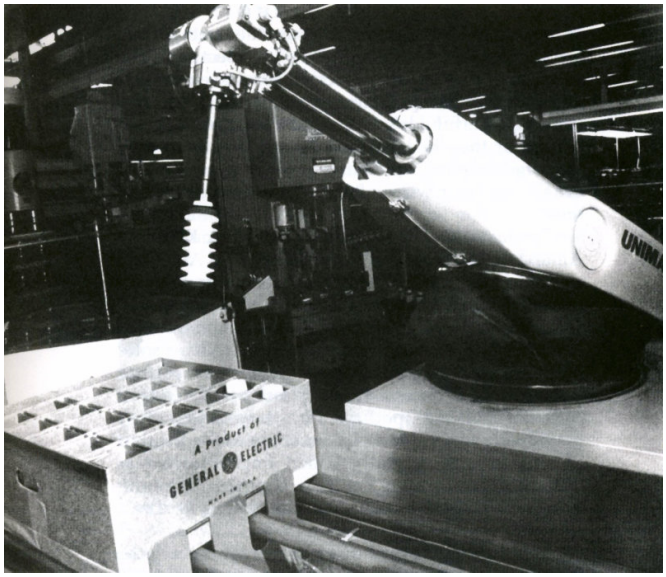


FIGURE 1.12 – Le robot Unimate de la société Unimation en 1961.

La médecine est également fortement concernée par le développement et l'utilisation des robots. Le plus populaire dans ce domaine aujourd'hui est certainement le robot Da Vinci (Figure 1.13a). Celui-ci est piloté par le chirurgien, mais corrige les approximations de ce dernier pour effectuer des gestes extrêmement précis. On le retrouve utilisé dans un grand nombre d'opérations différentes telles que celles effectuées sur la prostate [TPS+02]. D'autres applications dans le domaine médical sont également visées par la robotique. L'une d'elles a pour objectif de guider automatiquement des appareils de visualisation, tels que des sondes échographiques ou des caméras lors d'opérations d'endoscopie. D'autres types de systèmes sont destinés à apporter une aide active à la rééducation des patients par le biais de robots de types exosquelettes (Figure 1.13b).

Plusieurs modèles de robots peuvent également être employés dans le domaine de l'exploration. On retrouve ainsi des robots utilisés dans des milieus sous-marins (Figure 1.14a) ou aériens (Figure 1.14b). En dotant ces robots de différents moyens de navigation, il est possible de les faire évoluer dans des milieux inaccessibles aux humains et de réaliser des tâches de recherche, de cartographie ou d'interventions militaires.

Les robots humanoïdes sont quant à eux destinés à beaucoup de domaines d'applications différents. Dans un premier temps, le vieillissement rapide de la population japonaise a encouragé les laboratoires du pays à entreprendre d'importantes recherches en matière de robotique humanoïde. Ces robots ont pour objectif d'aider des personnes diminuées par leur âge ou leur handicap. Ils se doivent donc d'accomplir des tâches quotidiennes telle que la préparation d'un repas, mais également d'intéragir physiquement avec des personnes pour, par exemple, les aider à se déplacer. Une autre caractéristique, qui est directement liée au fait que le robot humanoïde soit présent dans les foyers du

grand public, est que le robot puisse être considéré comme un compagnon. Les interactions entre l'Homme et le robot sont alors étudiées pour que ce dernier se mêle de manière harmonieuse à la population. Dans le domaine militaire, les robots humanoïdes pourraient être utilisés pour des tâches de recherche et de sauvetage dans des environnements difficilement accessibles. Ils peuvent également être employés dans le transport de matériels lourds.

(a) (b)

FIGURE 1.13 – Les robots utilisés dans le domaine médical (a) Le robot chirurgien Da Vinci. (b) Un patient en rééducation avec un exosquelette médical.

(a) (b)

FIGURE 1.14 – Les robots utilisés dans le domaine militaire ou de l'exploration. (a) Le robot sous-marin Observer 3.1. (b) Le drone de reconnaissance Predator de l'armée américaine.

Les recherches menées depuis le début de la robotique s'articulent autour de grands thèmes impliquant beaucoup de domaines de recherches différents. Nous revenons dans la section suivante sur une de ces thématiques qui est la perception de l'environnement par le robot.

1.2 La perception de l'environnement

Dans l'histoire de la robotique, le comportement autonome d'une machine est directement lié à ce que celle-ci perçoit. Tout système robotique que l'on souhaite autonome doit être capable de percevoir son environnement d'une manière ou d'une autre. Ceci est rendu possible grâce aux différents capteurs qu'il est possible d'utiliser. Chaque capteur est conçu pour collecter une donnée spécifique de l'environnement, et la retranscrire en un signal analogique ou numérique transmis à l'ordinateur embarqué dans le robot. Le choix du capteur dépend donc de l'information que l'on souhaite extraire de l'environnement.

1.2.1 Les capteurs

Un capteur est un dispositif permettant d'obtenir une information spécifique de l'environnement dans lequel il est soumis. Il est donc impératif de définir précisemment la tâche que l'on souhaite accomplir avec le robot pour distinguer quelles sont les informations pertinentes à collecter. Par exemple, le vol stationnaire des drones quadrirotors peut être automatisé en se basant sur leur orientation dans l'espace 3D. Il est alors indispensable d'utiliser des centrales inertielles qui comprennent des accéléromètres et des gyroscopes qui eux-mêmes permettent d'estimer les angles de roulis de tangage et de cap de ces appareils.

On trouve également plusieurs capteurs qui permettent d'extraire la même information. Le choix du capteur dépend alors de la précision ainsi que des temps de mesure et de calcul avec lesquels on souhaite récolter les données. Il est par exemple possible de déterminer la position d'un robot dans son environnement via différents types de capteurs. Dans [Dru87], les informations fournies par un capteur à ultrason embarqué sur un robot sont utilisées pour déterminer sa localisation et son orientation dans une pièce. Dans [YZT10], un Lidar (Laser detection and ranging) est utilisé pour corriger les informations fournies par un GPS et une centrale inertielle, afin de localiser au mieux le robot dans une salle. Dans [AH93], des marqueurs visuels sont placés dans une salle et leur observation par une caméra permet de localiser le robot en temps réel. Ces trois méthodes utilisent donc des capteurs et des algorithmes différents pour extraire la même information. Cependant un même capteur peut être utilisé pour fournir tout un ensemble d'informations.

En effet, les tâches complexes que l'on souhaite réaliser avec un robot peuvent demander plusieurs types d'informations relatives à l'environnement. Il peut être utile, par exemple, de détecter la présence et la distance d'un obstacle lors du déplacement

d'un robot mobile. Il peut être également nécessaire de distinguer la nature d'un objet que l'on souhaite lui faire saisir, comme un outil placé au milieu d'autres. Dans ces cas là, les capteurs susceptibles de fournir au mieux toutes ces différentes informations sont les caméras. Utilisées avec des algorithmes de vision par ordinateur, elles permettent d'extraire une multitude d'informations relatives à l'environnement dans lequel le robot évolue.

Nous présentons dans la partie suivante comment les caméras et la vision par ordinateur permettent à un robot d'interpréter l'environnement qui l'entoure.

1.2.2 La vision par ordinateur

La caméra est un capteur capable de convertir l'intensité lumineuse qu'il reçoit en une image composée de pixels. Il lui est alors possible de restituer sur une image la forme d'un objet sur lequel se reflète la lumière, comme le fait l'oeil humain sur la rétine. Les données brutes obtenues par une caméra sont donc des informations numériques associées à chaque pixel. Grâce à l'utilisation de traitements d'images spécifiques, il est possible d'extraire différents types d'informations géométriques tels que des points, des droites, ou des ellipses, mais également de déterminer la nature de l'objet observé. Il est cependant primordial de définir comment un objet est géométriquement formé sur une image.

Pour cela on se réfèrera à [FP02, HZ03] issus de travaux accomplis dans le domaine de la vision par ordinateur. Comme il est présenté sur la Figure 1.15, nous illustrons ce principe à l'aide du cas de la projection perspective d'un point $\mathcal{X}$ de l'objet $\mathcal{O}$, situé dans l'espace 3D, sur le plan image d'une caméra monoculaire $\mathcal{C}$. Le modèle de la caméra est sténopé, c'est-à-dire que tous les rayons de lumière réfléchis par l'objet passent par le centre optique de la caméra C. De cette projection perspective résulte les coordonnées 2D du point sur le plan image de la caméra.

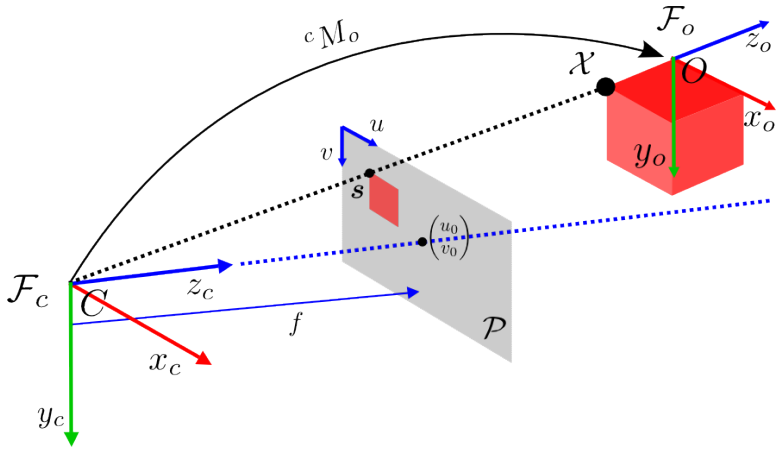


FIGURE 1.15 – Projection perspective d'un point sur le plan image d'une caméra.

1.2.2.1 Le changement de repère

Pour projeter le point 3D $\mathcal{X}$ sur le plan image de la caméra, il est essentiel de définir ses coordonnées 3D dans le repère de la caméra, et ce à l'aide d'un changement de repère. Pour cela nous introduisons dans un premier temps l'expression de la pose de l'objet dans le repère de la caméra :

$$^c\mathbf{P}_o = (^c\mathbf{t}_o, \theta\boldsymbol{u}) \in \mathbb{R}^6. \tag{1.1}$$

avec $^c\mathbf{t}_o = [t_x\ t_y\ t_z]^T$ le vecteur de translation sur les axes x, y, z, et $\theta\boldsymbol{u} \in \mathbb{S}^2 \times \mathbb{R}$ la représentation de la rotation entre la caméra et l'objet autour de l'axe décrit par le vecteur $\boldsymbol{u} = [u_x\ u_y\ u_z]^T$ avec un angle θ. Il est à noter que les singularités de la représentation d'une rotation par le vecteur $\theta\boldsymbol{u}$ sont données par $\theta = 2k\pi, k \in \mathbb{Z}^*$. C'est-à-dire que ces singularitées ne peuvent intervenir en pratique que pour des systèmes effectuant plusieurs tours, tel qu'un satellite par exemple. Elles n'interviennent donc dans aucune application classique de robotique.

Les paramètres de la pose (1.1) peuvent alors être exploités pour définir les coordonnées de l'objet dans le repère de la caméra. Soient les repères cartésiens $\mathcal{F}_c = (C, x_c, y_c, z_c)$ et $\mathcal{F}_o = (O, x_o, y_o, z_o)$ liés respectivement à la caméra et à l'objet. Le point $\mathcal{X}$ est défini par ses coordonnées homogènes $^o\boldsymbol{X} = [^oX\ ^oY\ ^oZ\ 1]^T$ dans le repère $\mathcal{F}_o$. Les coordonnées homogènes $^c\boldsymbol{X} = [^cX\ ^cY\ ^cZ\ 1]^T$ du point $\mathcal{X}$ dans le repère $\mathcal{F}_c$ peuvent être calculées à partir de $^o\boldsymbol{X}$ par le changement de repère suivant :

$$^c\boldsymbol{X} = {}^c\mathbf{M}_o\ {}^o\boldsymbol{X} \tag{1.2}$$

où $^c\mathbf{M}_o$ est la matrice de transformation homogène de $\mathcal{F}_o$ vers $\mathcal{F}_c$ et définie ainsi :

$$^c\mathbf{M}_o = \begin{bmatrix} ^c\mathbf{R}_o & ^c\mathbf{t}_o \\ \mathbf{0} & 1 \end{bmatrix} \tag{1.3}$$

avec le vecteur de translation $^c\mathbf{t}_o$ qui intervient dans l'expression de la pose (1.1) et $^c\mathbf{R}_o$ la matrice de rotation du repère $\mathcal{F}_o$ vers $\mathcal{F}_c$. Celle-ci peut être décomposée grâce à la représentation axe/angle ($\boldsymbol{u}$, θ) de l'expression de la pose (1.1) tel que :

$$^c\mathbf{R}_o(\boldsymbol{u}, \theta) = \boldsymbol{I}_3 + [\boldsymbol{u}]_\times \sin(\theta) + [\boldsymbol{u}]_\times^2 (1 - \cos(\theta)) \tag{1.4}$$

avec $\boldsymbol{I}_3 \in \mathbb{R}^{3\times3}$ la matrice identité d'ordre 3, et où $[\boldsymbol{u}]_\times$ est la matrice antisymétrique définie par :

$$[\boldsymbol{u}]_\times = \begin{bmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{bmatrix}. \tag{1.5}$$

L'expression des coordonnées 3D de $\mathcal{X}$ dans le repère $\mathcal{F}_c$ ainsi obtenue permet de définir les coordonnées 2D correspondantes sur le plan image de la caméra via la projection perspective.

1.2.2.2 La projection perspective

Le point $\mathcal{X}$ est projeté en un point $\bar{\boldsymbol{x}} = [\bar{x}\ \bar{y}\ f]$ du plan image $\mathcal{P}$ de la caméra, avec $\bar{z} = f$ où f est la distance focale de la caméra. En supposant dans la suite du document que la caméra est calibrée, on admet que la focale de la caméra est égale à 1. On obtient alors le projeté du point sur $\mathcal{P}$ en un point $\boldsymbol{x} = [x\ y\ 1]$ en utilisant les coordonnées homogènes du point dans le repère de la caméra ${}^c\boldsymbol{X}$:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} {}^cX/{}^cZ \\ {}^cY/{}^cZ \end{bmatrix} = f_s({}^c\mathbf{P}_o, {}^o\boldsymbol{X}). \tag{1.6}$$

De là, il est intéressant de définir comment ces coordonnées sont obtenues en coordonnées pixels par la caméra.

1.2.2.3 Les coordonnées pixels

Les données brutes d'une image capturée par la caméra sont des valeurs d'intensité de lumière associées à chaque pixel. Par le biais d'un traitement d'image, on peut déterminer dans un premier temps les coordonnées pixels u, v du point $\mathcal{X}$ dans l'image. Puis, dans un second temps, on peut retrouver les coordonnées $\boldsymbol{x}$ associées à $\mathcal{X}$.

On décrit ici comment passer des coordonnées sur le plan image $\boldsymbol{x}$ à des coordonnées pixels $(u,\ v)$:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} p_x & 0 & u_0 \\ 0 & p_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{1.7}$$

avec $p_x = \frac{1}{l_x}$, $p_y = \frac{1}{l_y}$ et l_x, l_y les dimensions d'un pixel en largeur et en hauteur, u_0 et v_0 les coordonnées pixels du point principal de l'image, c'est-à-dire le point par lequel passe l'axe optique. On peut extraire de (1.7) la matrice de calibration $\mathbf{K}$ des paramètres intrinsèques de la caméra :

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{1.8}$$

On supposera dans la suite du document que la caméra est correctement calibrée, et donc que les paramètres intrinsèques u_0, v_0, p_x, p_y sont connus. Cela permet d'obtenir les coordonnées $\boldsymbol{x}$ à partir des coordonnées pixels $(u,\ v)$ par :

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathbf{K}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}. \tag{1.9}$$

Nous avons décrit ici comment obtenir les coordonnées 2D d'un point en projetant ses coordonnées 3D sur le plan image 2D de la caméra. Nous avons également explicité comment cette projection se traduit en données pixels. La projection d'informations

visuelles géométriques plus complexes telles que des segments, des droites ou des cercles peut être obtenue en suivant le même principe. De là, un objet plus compliqué peut également être détecté et reconnu via différents outils de traitement d'images. La méthode de détection de contours par exemple, qui a été étudiée par David Marr en 1980 [MH80] et a fait l'objet par la suite de méthodes telles que le filtre de Canny [Can86], permet de déterminer efficacement la position de l'objet observé sur le plan image de la caméra.

Nous décrivons par la suite le schéma de commande classique qui exploite ces informations pour agir sur les actionneurs du robot, à savoir l'asservissement visuel.

1.3 Les schémas de commande

Un des aspects essentiel lors de la conception d'un robot autonome est le choix de la stratégie de commande qu'on lui attribue. Le but de celle-ci est d'établir le rapport entre le comportement désiré qui définit une ou plusieurs tâches à accomplir par le robot, les actionneurs qui lui permettent d'intéragir physiquement avec l'environnement, et les mesures fournies par ses capteurs que nous avons introduits dans la section 1.2.

Dans ce chapitre nous nous concentrons en particulier sur l'emploi de la vision par ordinateur dans la robotique. Dans cette optique, nous décrivons en premier lieu, l'asservissement visuel, stratégie de commande classique qui utilise des informations visuelles pour contrôler les actionneurs d'un robot afin qu'il effectue des tâches de suivi ou de positionnement.

1.3.1 L'asservissement visuel

Créé au début des années 1980, l'asservissement visuel est la stratégie qui exploite la vision par ordinateur dans une boucle de commande pour contrôler les mouvements d'un robot. Le principe consiste à utiliser les informations extraites d'images capturées par une ou plusieurs caméras, et de calculer une vitesse à appliquer aux différents actionneurs du robot. Cette méthode permet de réaliser diverses tâches avec le robot. Par exemple, celui-ci est capable de suivre des objets en mouvement pour qu'ils ne quittent pas le champ de vue de la caméra, ou d'atteindre une pose relative à un objet pour le manipuler ou l'observer depuis un point de vue désiré. Dans cette partie, on se réfère aux travaux présentés dans [CH06] qui reviennent de manière détaillée sur les différentes méthodes classiques d'asservissement visuel.

1.3.1.1 Les configurations eye-in-hand et eye-to-hand

L'asservissement visuel peut être appliqué à tout système composé d'un système robotique et d'une caméra. On distingue cependant deux configurations possibles de ce système :

- Si la caméra est montée sur un robot mobile ou sur l'effecteur d'un bras robotique, on appelle ce système eye-in-hand (Figure 1.17). Les vitesses appliquées dans le

repère du robot sont alors les mêmes que celles appliquées dans le repère de la caméra à un changement de repère près. La tâche consiste par exemple à déplacer la caméra pour qu'elle garde un objet au mileu de son champ de vue.

- Si la caméra observe le robot qui se déplace on dit que le système est eye-to-hand (Figure 1.17). Les vitesses appliquées dans le repère du robot sont alors liées à celles exprimées dans le repère de la caméra par un changement de repère qui est, cette fois-ci, variable dans le temps. La tâche consiste, par exemple, à positionner un bras robotique devant un objet pour le saisir. Dans ce cas, il est indispensable d'observer à la fois le robot en mouvement et l'objet considéré.

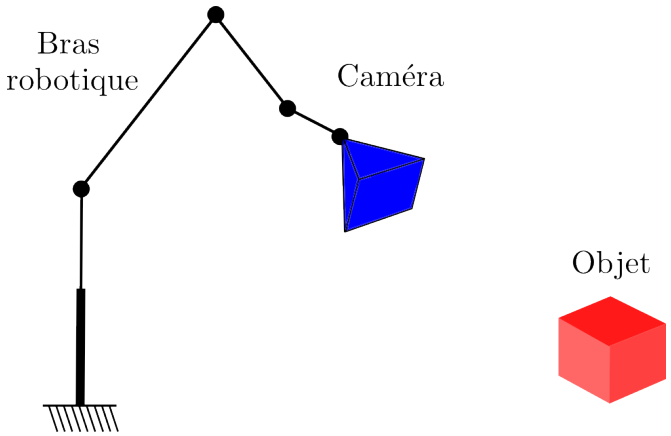


FIGURE 1.16 – Configuration eye-in-hand. La caméra est montée sur le robot et observe l'objet.

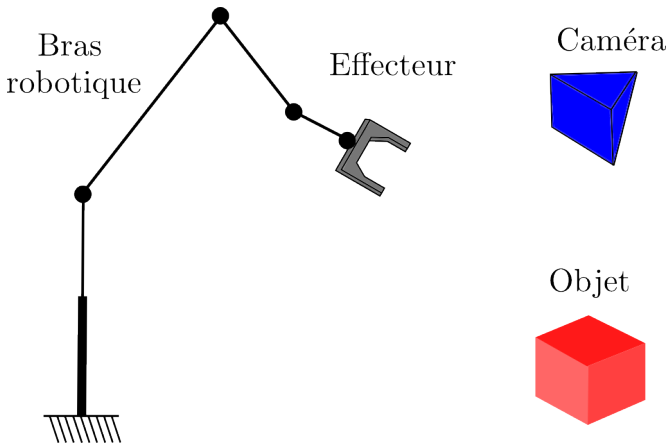


FIGURE 1.17 – Configuration eye-to-hand. La caméra observe le robot et l'objet depuis un point de vue extérieur.

Dans ces deux configurations, le traitement d'images permet d'extraire différents types d'informations. Celles-ci sont appelées primitives visuelles. Elles définissent un ensemble

représenté par le vecteur $\boldsymbol{s}$. Le rôle de la commande consiste alors à appliquer une vitesse $\mathbf{v}$ au robot afin que les primitives visuelles courantes atteignent des valeurs désirées définies par le vecteur $\boldsymbol{s}^*$. Cette vitesse est définie tel que :

$$\mathbf{v} = (\boldsymbol{v},\ \boldsymbol{\omega}) \tag{1.10}$$

où $\boldsymbol{v} = [v_x\ v_y\ v_z]$ est le vecteur des vitesses linéaires et $\boldsymbol{\omega} = [\omega_x\ \omega_y\ \omega_z]$ le vecteur des vitesses angulaires appliquées au robot.

Le principe de la commande consiste alors à minimiser la norme de l'erreur entre les primitives visuelles courantes et désirées :

$$\mathbf{e} = \boldsymbol{s} - \boldsymbol{s}^*. \tag{1.11}$$

Pour cela, le rapport entre l'évolution des primitives visuelles dans le temps $\dot{\boldsymbol{s}}$ et la vitesse appliquée au robot $\mathbf{v}$ est utilisée. Celui-ci dépend de la configuration choisie :

- Pour une configuration eye-in-hand, ce rapport est le suivant :

$$\dot{\boldsymbol{s}} = \boldsymbol{L}_s \mathbf{v} \tag{1.12}$$

  où $\boldsymbol{L}_s$ est appelée la matrice d'interaction liée à $\boldsymbol{s}$. Les caractéristiques de celle-ci dépendent du choix des informations visuelles et sont présentées dans les sections 1.3.1.2 et 1.3.1.3.
- Pour une configuration eye-to-hand :

$$\dot{\boldsymbol{s}} = -\boldsymbol{L}_s {}^c\boldsymbol{V}_r \mathbf{v} \tag{1.13}$$

  avec ${}^c\boldsymbol{V}_r$ la matrice de changement de repère des vitesses du robot vers la caméra. On rappelle que les mouvements du robot sont indépendants de ceux de la caméra. Ce changement de repère permet donc de calculer l'évolution de l'ensemble $\boldsymbol{s}$ du point de vue de la caméra en fonction des mouvements du robot. Il dépend de la matrice de rotation ${}^c\mathbf{R}_r$ définie dans (1.4) et du vecteur de translation ${}^c\mathbf{t}_r$ du robot vers la caméra :

$${}^c\boldsymbol{V}_r = \begin{bmatrix} {}^c\mathbf{R}_r & [{}^c\mathbf{t}_r]_\times {}^c\mathbf{R}_r \\ \mathbf{0} & {}^c\mathbf{R}_r \end{bmatrix}. \tag{1.14}$$

En utilisant soit la relation (1.12) soit (1.13) en fonction de la configuration du système, pour tenter d'obtenir une décroissance exponentielle de l'erreur $\dot{\boldsymbol{s}} = -\lambda(\boldsymbol{s} - \boldsymbol{s}^*)$, il suffit d'appliquer les vitesses suivantes à chaque nouvelle mesure de $\boldsymbol{s}$:

- Pour une configuration eye-in-hand, on applique la loi de commande suivante :

$$\mathbf{v} = -\lambda \boldsymbol{L}_s^\dagger (\boldsymbol{s} - \boldsymbol{s}^*), \tag{1.15}$$

  où $\boldsymbol{L}_s^\dagger$ est la pseudoinverse de Moore-Penrose [Pen55] de la matrice d'interaction $\boldsymbol{L}_s$, définie par $\boldsymbol{L}_s^\dagger = (\boldsymbol{L}_s^T \boldsymbol{L}_s)^{-1} \boldsymbol{L}_s^T$ quand $\boldsymbol{L}_s$ est de rang 6 plein. Étant

donné qu'en pratique il est impossible de déterminer parfaitement $\boldsymbol{L}_s$ ou $\boldsymbol{L}_s^\dagger$, une approximation doit être réalisée, et la loi de commande devient :

$$\mathbf{v} = -\lambda \widehat{\boldsymbol{L}_s^\dagger}(\boldsymbol{s} - \boldsymbol{s}^*), \tag{1.16}$$

avec $\widehat{\boldsymbol{L}_s^\dagger}$ l'approximation de la pseudo-inverse de la matrice d'interaction.

- Pour une configuration eye-to-hand :

$$\mathbf{v} = \lambda {}^r\boldsymbol{V}_c \widehat{\boldsymbol{L}_s^\dagger}(\boldsymbol{s} - \boldsymbol{s}^*), \tag{1.17}$$

avec ${}^r\boldsymbol{V}_c$ la matrice de changement de repère des vitesses de la caméra vers le robot.

Le choix des informations qui déterminent l'ensemble $\boldsymbol{s}$ et la matrice d'interaction $\boldsymbol{L}_s$ définit la technique d'asservissement visuel utilisée. On distingue deux méthodes classique d'asservissement visuel. La première est appelée "Image Based Visual Servoing" (IBVS) où l'ensemble $\boldsymbol{s}$ est basé sur des informations présentes sur l'image 2D. La seconde est appelée "Pose Based Visual Servoing" (PBVS) où cet ensemble est composé des informations 3D relatives aux poses courante et désirée de la caméra, du robot et de l'objet observé.

1.3.1.2 IBVS

L'asservissement visuel basé sur l'image est une méthode de commande robotique qui utilise des informations 2D extraites d'images capturées par une caméra.

Comme présenté dans la section 1.2.2, plusieurs types d'informations peuvent être extraites d'une image : des points, des segments, des droites, des ellipses. Ces informations définissent l'ensemble $\boldsymbol{s}$ et la pseudo-inverse de la matrice d'interaction $\boldsymbol{L}_s$ utilisés dans les lois de commande (1.16) et (1.17). Par exemple, dans le cas où les primitives visuelles sont les coordonnées 2D d'un point sur l'image, on utilise la projection perspective de ce point (1.6) pour définir l'ensemble $\boldsymbol{s} = [x\ y]^T$. La matrice d'interaction est alors la suivante :

$$\boldsymbol{L}_s(\boldsymbol{s}, {}^cZ) = \begin{bmatrix} -\frac{1}{{}^cZ} & 0 & \frac{x}{{}^cZ} & xy & -(1+x^2) & y \\ 0 & -\frac{1}{{}^cZ} & \frac{y}{{}^cZ} & 1+y^2 & -xy & -x \end{bmatrix} \in \mathbb{R}^{2\times 6}. \tag{1.18}$$

où ${}^cZ$ est le seul paramètre qui ne peut pas être mesuré directement dans l'image. Sa valeur est donc soit estimée soit approximée. On fixe enfin des coordonnées désirées relatives au point sur le plan image dans l'ensemble $\boldsymbol{s}^*$. Le robot se déplace ainsi pour minimiser l'erreur entre $\boldsymbol{s}$ et $\boldsymbol{s}^*$.

Il apparaît dans cet exemple que si un seul point est utilisé, la matrice $\boldsymbol{L}_s$ est de rang 2 seule. Si l'on souhaite contrôler les six degrés de liberté (ddl) d'un robot, il est nécessaire de disposer d'au moins six coordonnées dans $\boldsymbol{s}$ et donc d'au minimum trois points

projetés sur l'image. L'ensemble des primitives visuelles devient alors $\boldsymbol{s} = (\boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{s}_3)$, l'ensemble des coordonnées des profondeurs devient ${}^c\boldsymbol{Z} = [{}^cZ_1\ {}^cZ_2\ {}^cZ_3]^T$, et la matrice d'interaction :

$$\boldsymbol{L}_s({}^c\boldsymbol{Z}) = \begin{bmatrix} \boldsymbol{L}_s(\boldsymbol{s}_1, {}^cZ_1) \\ \boldsymbol{L}_s(\boldsymbol{s}_2, {}^cZ_2) \\ \boldsymbol{L}_s(\boldsymbol{s}_3, {}^cZ_3) \end{bmatrix} \in \mathbb{R}^{6\times 6}. \tag{1.19}$$

Le cas où trois points sont utilisés pour définir l'ensemble $\boldsymbol{s}$ est illustré sur la Figure 1.18 où les points rouges sont les coordonnées courantes des primitives visuelles sur le plan image de la caméra, tandis que les points verts sont leurs coordonnées désirées correspondantes.

Il est à noter qu'il existe des configurations où $\boldsymbol{L}_s$ est singulière [MR93]. C'est pourquoi il est indispensable de disposer de plus de trois points pour l'élaboration de la loi de commande.

Bien que la méthode IBVS soit susceptible de créer des perturbations dans la trajectoire du robot dûes aux approximations de la coordonnée ${}^cZ$ intervenant dans la matrice d'interaction, elle présente plusieurs avantages. Elle est relativement rapide à calculer. En effet, l'essentiel du temps de calcul est consacré au traitement d'images à chaque nouvelle mesure de la caméra. De plus, sa robustesse aux erreurs de traitement d'images et d'estimation de la profondeur est importante.

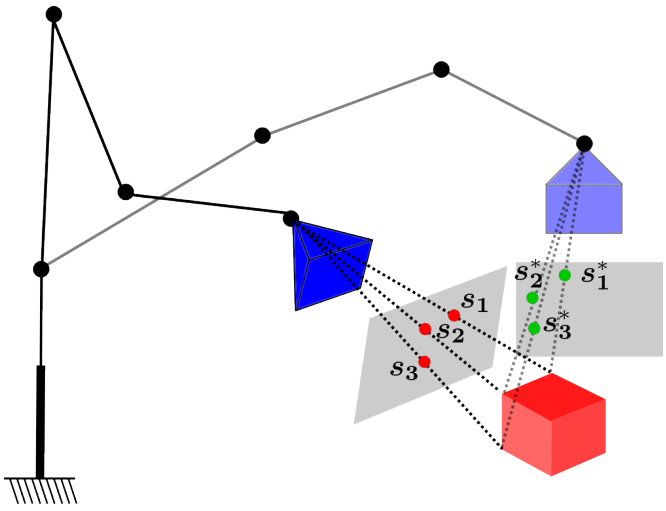


FIGURE 1.18 – IBVS : Le robot et la caméra se déplacent pour que les primitives visuelles dans l'image $\boldsymbol{s}$, representées en rouge, atteignent les coordonnées désirées $\boldsymbol{s}^*$, représentées en vert.

L'autre méthode classique d'asservissement visuel est basée sur la pose de l'objet dans le repère de la caméra dans le cas de la configuration eye-in-hand ou du robot et de l'objet dans le repère de la caméra dans le cas de la configuration eye-to-hand.

1.3.1.3 PBVS

L'asservissement visuel basé sur la pose est une technique qui utilise les informations 3D relatives au système composé de la caméra, du robot et de l'objet. Ces informations sont obtenues par l'expression de la pose (1.1). L'ensemble $\boldsymbol{s}$ est donc composé des caractéristiques de la pose qui sont la translation $\mathbf{t}$ et la rotation $\theta\boldsymbol{u}$.

Dans les configurations eye-in-hand et eye-to-hand, plusieurs poses peuvent être considérées : celle de l'objet dans le repère de la caméra, celle du robot dans le repère de la caméra, ou simplement la pose désirée de la caméra dans le repère courant de la caméra. À titre d'exemple, si l'on souhaite déplacer une caméra montée sur l'effecteur d'un robot vers une position désirée pour observer un objet comme illustré sur la Figure 1.19, on peut utiliser la pose de l'objet dans le repère courant de la caméra tel que :

$$\boldsymbol{s} = {}^{c}\mathbf{P}_{o} = ({}^{c}\mathbf{t}_{o}, \theta\boldsymbol{u}). \tag{1.20}$$

En déterminant la pose de l'objet dans le repère désiré de la caméra ${}^{c^*}\mathbf{P}_o$, on fixe $\boldsymbol{s}^* = ({}^{c^*}\mathbf{t}_o, 0)$ dans (1.16). De là, en minimisant l'erreur entre $\boldsymbol{s}$ et $\boldsymbol{s}^*$, la vitesse appliquée au robot va permettre d'atteindre la pose désirée de la caméra. Pour cela, la matrice d'interaction associée à $\boldsymbol{s}$ est la suivante :

$$\boldsymbol{L}_{s}(\boldsymbol{s}) = \begin{bmatrix} -\mathbf{I}_3 & [{}^{c}\mathbf{t}_{o}]_{\times} \\ \mathbf{0} & \boldsymbol{L}_{\omega}(\boldsymbol{u},\, \theta) \end{bmatrix} \tag{1.21}$$

avec $\mathbf{I}_3$ la matrice identité de dimension 3×3, la matrice antisymétrique du vecteur de translation ${}^{c}\mathbf{t}_{o}$ et avec :

$$\boldsymbol{L}_{\omega}(\boldsymbol{u},\, \theta) = \boldsymbol{I}_3 - \frac{\theta}{2}[\boldsymbol{u}]_{\times} + \left(1 - \frac{\mathrm{sinc}(\theta)}{\mathrm{sinc}^2\,(\theta/2)}\right)[\boldsymbol{u}]_{\times}^{2}, \tag{1.22}$$

où $\mathrm{sinc}(\theta) = \sin(\theta)/\theta$, [MCB99]. En utilisant la matrice d'interaction (1.21) dans la loi de commande (1.16), les 6 ddl du robot sont contrôlés et la trajectoire de la caméra permet à celle-ci d'atteindre sa pose désirée.

Cette méthode assure un bon comportement du robot dans l'espace 3D durant son déplacement. Elle requiert cependant une bonne estimation des poses pour permettre au robot d'atteindre avec précision sa pose désirée. Pour cela différentes méthodes de calcul de pose peuvent être utilisées. L'estimation de la pose d'un objet dans le repère de la caméra peut être accomplie par le biais de l'extraction dans l'image de points [HJL+89, DD92], de segments [DRLR89], des contours de l'objet [Low87, DC99], de cônes [DM93], ou de cylindres [DYL93]. Une autre solution consiste à combiner ces informations de différentes manières. Par exemple, dans [DG99] une combinaison de droites et de points est utilisée pour calculer la pose.

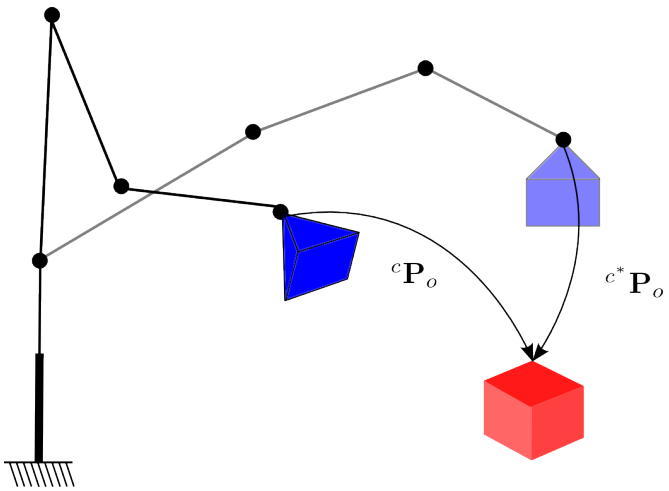


FIGURE 1.19 – PBVS : Le robot et la caméra se déplacent pour que la pose courante de l'objet dans le repère de la caméra $^{c}\mathbf{P}_{o}$ atteigne la pose désirée $^{c^*}\mathbf{P}_{o}$.

1.3.1.4 Approximation de la matrice d'interaction

Nous venons de voir que la matrice d'interaction $\boldsymbol{L}_s$ peut être associée à différents types d'informations 2D ou 3D. On peut ajouter à cela d'autres paramètres qui permettent de concevoir cette matrice. En effet, il est possible d'estimer $\widehat{\boldsymbol{L}_s^{\dagger}}$ dans (1.16) ou (1.17) en suivant trois types d'élaboration distinctes :

- La première spécifie qu'une nouvelle information est calculée à chaque fois qu'une nouvelle mesure provenant de la caméra est disponible. Par exemple, dans le cas de l'IBVS présenté dans la section 1.3.1.2, on a $\widehat{\boldsymbol{L}_s^{\dagger}} = \boldsymbol{L}_s^{\dagger}(\boldsymbol{s}, {}^{c}\boldsymbol{Z})$. La matrice d'interaction est ainsi mise à jour lorsque l'ensemble $\boldsymbol{s}$ est actualisée à chaque itération.
- La seconde est relative aux informations désirées. Dans le cas de l'IBVS, les coordonnées désirées des primitives visuelles dans l'image et de la profondeur ${}^{c}\boldsymbol{Z}$ sont considérées : $\widehat{\boldsymbol{L}_s^{\dagger}} = \boldsymbol{L}_s^{\dagger}(\boldsymbol{s}^*, {}^{c}\boldsymbol{Z}^*)$. Dans ce cas, la valeur de la matrice d'interaction est constante durant l'intégralité du déplacement du robot.
- La dernière combine les deux premières méthodes en prenant en compte la moyenne des deux matrices d'interaction : $\widehat{\boldsymbol{L}_s^{\dagger}} = 1/2\left(\boldsymbol{L}_s(\boldsymbol{s}, {}^{c}\boldsymbol{Z}) + \boldsymbol{L}_s(\boldsymbol{s}^*, {}^{c}\boldsymbol{Z}^*)\right)^{\dagger}$.

Ces trois solutions assurent la convergence des primitives visuelles vers leurs coordonnées désirées si la position courante $^{c}\mathbf{P}_{o}$ appartient à leur domaine de stabilité respectif. En effet, un système est globalement asymptotiquement stable si $\boldsymbol{L}_s\widehat{\boldsymbol{L}_s^{\dagger}} > 0$. Or, si dans le cas du PBVS cette condition est respectée, elle ne peut jamais l'être dans l'IBVS dans le cas où plus de trois points sont utilisés pour calculer la matrice d'interaction. Cependant, une stabilité locale peut être assurée si $\widehat{\boldsymbol{L}_s^{\dagger}}\boldsymbol{L}_s > 0$ et que la position courante $^{c}\mathbf{P}_{o}$ est à un voisinage proche de la position désirée $^{c}\mathbf{P}_{o}^{*}$.

Ces trois solutions ont chacune un impact différent sur les trajectoires 2D empruntées par les primitives visuelles dans l'image, ainsi que sur le comportement de la caméra et du robot dans l'espace 3D. Nous considérons à titre d'exemple le cas illustré sur la Figure 1.20 issu de [CH06]. En vert sont représentées les coordonnées désirées des primitives visuelles $\boldsymbol{s}$ sur le plan image de la caméra, et en rouge leurs coordonnées courantes. Le mouvement de la caméra pour passer de la configuration courante à la configuration désirée correspond à une rotation pure autour de son axe optique, mais les différentes approximations de la matrice d'interaction présentées ci-dessus et utilisées dans la loi de commande (1.16) vont provoquer chacune un déplacement spécifique de la caméra. Dans le cas où $\widehat{\boldsymbol{L}_s^\dagger} = \boldsymbol{L}_s^\dagger(\boldsymbol{s}, {}^c\boldsymbol{Z})$, les trajectoires 2D des primitives visuelles courantes vers leurs coordonnées désirées vont correspondre à des lignes droites dont les directions sont représentées par les flèches rouges. Pour obtenir de telles trajectoires 2D, la caméra doit réaliser une trajectoire 3D qui combine une rotation autour de son axe optique avec une translation arrière le long de ce même axe optique. On peut noter que cette approximation de $\boldsymbol{L}_s$ permet aux erreurs entre les coordonnées courantes et désirées de décroître du début à la fin du mouvement de la caméra. Un autre résultat est obtenu en considérant $\widehat{\boldsymbol{L}_s^\dagger} = \boldsymbol{L}_s^\dagger(\boldsymbol{s}^*, {}^c\boldsymbol{Z}^*)$ où les trajectoires 2D des coordonnées des primitives vont suivre les directions représentées en bleu. Les primitives visuelles vont alors décrire des courbes dans l'image pour atteindre leurs coordonnées désirées. Un mouvement de rotation de la caméra autour de son optique va alors être généré, mais également une translation vers l'avant le long du même axe. Il est alors possible d'observer une hausse des erreurs entre les primitives courantes et désirées définies dans (1.11) pendant le mouvement de la caméra (ce qui n'est pas le cas de l'exemple de la Figure 1.20), et ce même si elles sont destinées à décroître finalement. Enfin, dans le cas $\widehat{\boldsymbol{L}_s^\dagger} = 1/2\,(\boldsymbol{L}_s(\boldsymbol{s}, {}^c\boldsymbol{Z}) + \boldsymbol{L}_s(\boldsymbol{s}^*, {}^c\boldsymbol{Z}^*))^\dagger$, la moyenne des deux cas précédents génère les trajectoires 2D des primitives visuelles en suivant les directions représentées en violet. La caméra va alors suivre une trajectoire de rotation autour de son axe optique sans qu'aucune translation n'intervienne.

Le choix de l'approximation de la matrice d'interaction dépend donc des trajectoires que l'on souhaite obtenir des primitives visuelles dans l'image, mais également de la caméra et du robot. De plus, la manière dont est conçue la matrice d'interaction est également liée aux informations disponibles durant l'application de la loi de commande. Par exemple, si aucune information relative à la profondeur ${}^c\boldsymbol{Z}$ ne peut être mise à jour, il sera préférable d'adopter l'approximation relative aux informations désirées $\boldsymbol{s}^*$ et ${}^c\boldsymbol{Z}^*$.

L'asservissement visuel permet de prendre en compte un grand nombre d'informations différentes 2D ou 3D pour commander les déplacements d'un robot. C'est la raison pour laquelle on retrouve cette approche dans de nombreux domaines d'application présentés dans la partie suivante.

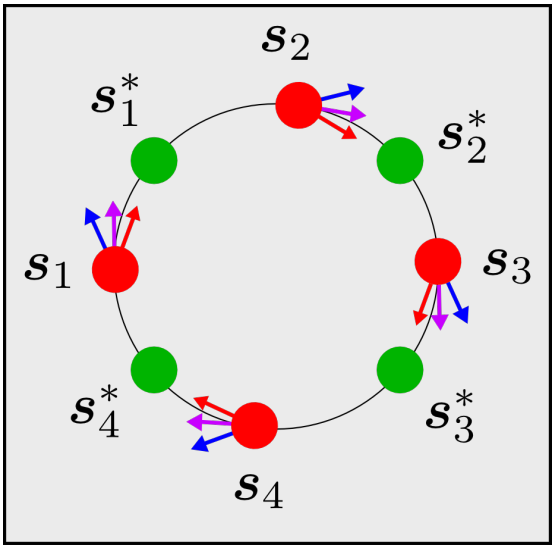


FIGURE 1.20 – Trajectoires 2D des primitives visuelles en fonction de l'approximation de $\boldsymbol{L}_s$. Les primitives visuelles courantes sont en rouge, leurs coordonnées désirées en vert.

1.3.1.5 Les domaines d'application

On retrouve de nombreuses méthodes basées sur l'asservissement visuel dans différents domaines d'application, notamment pour le pilotage autonome de véhicules. Dans [GHM08] une caméra embarquée sur un drone observe quatre points au sol. L'asservissement visuel permet alors de commander la position et l'orientation de l'appareil pour le stabiliser malgré la présence de perturbations atmosphériques. Dans le même esprit, un sous-marin soumis aux perturbations des courants est stabilisé par asservissement visuel dans [LLTC01]. Les travaux présentés dans [MT00] permettent à une voiture équipée d'une caméra de suivre une trajectoire de manière autonome grâce à l'obervation de la ligne blanche tracée au milieu de la route. On peut citer enfin [KDR+98] où les tâches répétitives d'un engin agricole sont effectuées à l'aide de différentes méthodes d'asservissement visuel basées sur les observations d'une seule caméra.

On trouve également l'asservissement visuel dans le domaine de l'imagerie médicale. Dans [CAL96] le positionnement d'une caméra est automatisé pour maintenir un instrument de chirurgie laparoscopique dans son champ de vue. Dans [MKC08] une sonde à ultrasons se déplace de manière autonome pour offrir des images utiles à un chirurgien pratiquant une intervention sur un patient.

Au vu des nombreuses possibilités d'implémentation de l'asservissement visuel dans tout type de système robotique, il est inévitable de le voir être utilisé dans le domaine de la robotique humanoïde. Une application présentée dans [TK01, VWA+08, AFP13] consiste à saisir et manipuler différents types d'objets avec une des mains du robot.

Les deux mains sont utilisées pour atteindre le même but dans [VBW+09]. Une autre application présentée dans [MCM13, DHM+11] consiste à réaliser des tâches de positionnement d'un robot humanoïde vers une pose désirée par la marche.

En raison de la diminution des temps de calcul des ordinateurs embarqués, il est possible aujourd'hui de développer des algorithmes de plus en plus complexes qui permettent à la fois d'extraire de nouvelles informations des images, mais également d'incorporer des stratégies de commande plus efficaces. Les recherches qui sont menées aujourd'hui dans le domaine de l'asservissement visuel tendent à obtenir une plus grande précision, une meilleure robustesse et une meilleure réactivité de la part des systèmes robotiques en les soumettant à des conditions de mesure, à des environnements, et à des tâches de plus en plus contraignantes. Dans ce sens, différentes stratégies de commande peuvent être utilisées pour réaliser une ou plusieurs tâches d'asservissement visuel par un robot. Le choix de la loi de commande dépend alors du comportement que l'on souhaite obtenir de la part du robot, de la nature de l'environnement dans lequel il évolue, et du nombre et des types de variables que l'on doit exploiter. La section suivante présente un des schémas de commande pouvant être utilisé pour réaliser un asservissement visuel, la commande prédictive basée modèle.

1.3.2 La commande prédictive basée modèle

Depuis le début des années 1980, les méthodes de commande prédictive ont été massivement utilisées pour le fonctionnement automatisé de systèmes industriels comprenant un grand nombre de variables à traiter. Ces méthodes ont comme point commun d'être basées sur la même philosophie : "Utiliser le modèle pour prédire le comportement du système et choisir la décision la meilleure au sens d'un certain coût tout en respectant les contraintes." On décrit ci-dessous les grandes étapes du développement de la commande prédictive basée modèle depuis sa naissance dans l'industrie.

1.3.2.1 L'historique

Les premières méthodes de commande prédictive appliquées en milieu industriel ont eu un impact conséquent sur les systèmes automatisés dans l'industrie. Elles sont affiliées en premier lieu aux travaux scientifiques de Shell, Cutler et Ramaker en 1980 [CR80], qui présentent le "Dynamic Matrix Control (DMC)" et aux travaux de Richalet *et al.* [RRTP78] en 1978, qui présentent le "Model Predictive Heuristic Control" (MPHD), également appelée "Model Algorithmic Control" (MAC), qui a débouché sur le développement de la solution logicielle "Identification and Command" (IDCOM). Ces deux approches ont pour but de minimiser des fonctions objectifs posées préalablement par l'utilisateur. Pour cela, les effets des futures actions générées par la loi de commande sur le système sont anticipés sur un horizon de prédiction par le biais de la dynamique du modèle du système considéré. L'opération est alors répétée à chaque pas d'itération, c'est-à-dire lorsque la connaissance de l'état du système est mise à jour grâce à de nouvelles mesures de capteurs. Les stabilités de ces deux méthodes n'ont pas été abordées

de manière théorique lors de leurs développements. Cependant, il a été observé qu'en choisissant un horizon de prédiction assez large dans le temps et en paramétrant des poids ajoutées à la fonction de coût, la stabilité peut être assurée. Ces méthodes ont l'avantage d'être algorithmique et heuristique et ont ainsi profité des capacités des ordinateurs de l'époque. Elles ont eu un impact conséquent sur la commande des procédés industriels et ont posé les bases de la définition de la commande prédictive basée modèle.

Les méthodes de commande prédictive suivantes sont apparues au début des années 1980 avec les travaux de Garcia et Morshedi, ingénieurs chez Shell également [GM86]. Ceux-ci proposent de prendre en compte de manière explicite des contraintes sur les commandes et sur les états du système en posant un problème quadratique "Quadratic/Dynamic Matrix Control" (QDMC). Le système considéré est linéaire, la fonction de coût est quadratique, et les contraintes en entrée et en sortie du système sont définies par des inégalités linéaires. La limite de ces méthodes est de réussir à obtenir une solution pour un problème quadratique qui ne peut être résolu. À la fin des années 1980 cette problématique est résolue par Grosdidier, Froisy, et Hammann qui développent la commande IDCOM-M [GFH88], et par les ingénieurs de Shell avec la commande "Shell Multivariable Optimizing Controller" (SMOC) [MB88].

Depuis le début des années 1990, une attention toute particulière est apportée à la stabilité et la robustesse des méthodes de commande prédictive basées modèle. Par exemple en 1995, les algorithmes PCT de Profimatics et RMPC de Honeywell ont permis de créer le "Robust Model Predictive Control Technology" (RMPCT) [Mac96].

Les travaux en matière de commande prédictive qui sont réalisés depuis s'articulent autour de plusieurs points importants : apporter des solutions explicites à des problèmes non-linéaires ; diminuer les temps de calculs pour pouvoir traiter en temps réel un grand nombre de variables ; proposer un ordre de priorité de résolution sur plusieurs problèmes d'optimisation. On présente dans la partie suivante les caractéristiques qui réunissent toutes les méthodes de commande prédictive basées modèle.

1.3.2.2 Le principe

La commande prédictive repose sur le principe qu'un nouveau problème d'optimisation est posé à chaque itération. Ce problème est composé de l'état actuel et prédit du système et d'une fonction objectif à atteindre en satisfaisant des contraintes. La mise en œuvre de ce type de schéma de commande requiert :

- Un modèle qui permet d'anticiper les futurs états $\boldsymbol{y}$ du système sur un horizon de prédiction fini N_p.
- Une fonction coût qui traduit mathématiquement les objectifs de commande du système et qui définit une consigne $\boldsymbol{r}$ à atteindre.
- Des contraintes de fonctionnement à satisfaire sur les entrées et les sorties du système.

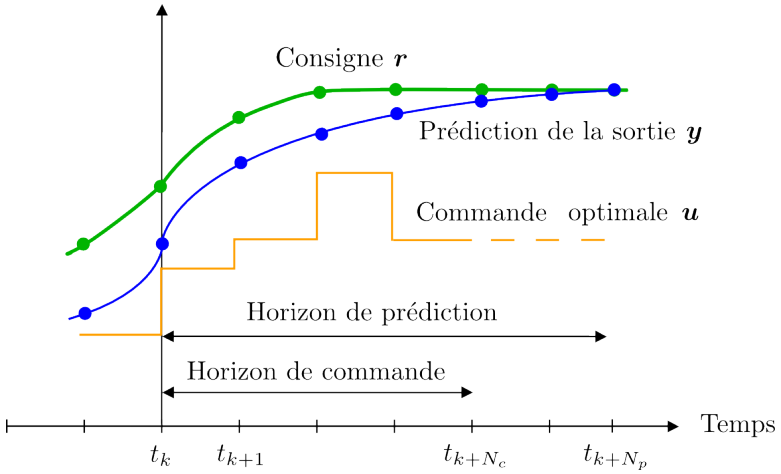


FIGURE 1.21 – Principe de la commande prédictive : à chaque instant t_k, l'algorithme de résolution calcule la séquence de commandes optimales $\boldsymbol{u}$ à appliquer au système jusqu'à l'instant t_{k+N_c} (représentée en orange). Pour cela, il se base sur le modèle du système qui permet de prédire la sortie $\boldsymbol{y}$ jusqu'à l'instant t_{k+N_p} (représenté en bleu) et sur la consigne $\boldsymbol{r}$ à atteindre (représentée en vert).

Le principe de la commande est illustré sur la Figure 1.21. Celui-ci revient à optimiser la fonction coût à chaque itération t_k lorsque une nouvelle mesure est disponible, en prenant soin de satisfaire les contraintes de fonctionnement. Une séquence de N_c commandes est alors déterminée sur l'horizon de prédiction N_p. Finalement, seule la première composante de la séquence de commande est appliquée au système. Puis, lorsque une nouvelle mesure est disponible à t_{k+1}, l'état du système est mis à jour, et l'algorithme de résolution est à nouveau utilisé pour calculer une nouvelle séquence de N_c commandes à appliquer au système.

La caractéristique principale de cette loi de commande est son horizon fuyant. À chaque nouvelle mesure, l'horizon de prédiction N_p permet d'anticiper les futurs états du système jusqu'à un temps fini, ce qui permet de fournir à l'algorithme de résolution une meilleure qualité des informations relatives à l'évolution du système. Cependant, plus l'horizon est grand, plus la qualité de la prédiction est faible. En effet, des écarts apparaissent entre le modèle prédit et le système réel et augmentent à chaque pas de temps futur. Il n'existe pas encore de méthode générique permettant de déterminer la valeur de l'horizon de prédiction de manière optimale. Mais, un compromis peut toutefois être trouvé entre un horizon petit, qui offre peu d'informations sur l'évolution du système mais une meilleure qualité de prédiction, et un horizon grand, qui fournit une plus grande quantité d'informations mais des incertitudes plus importantes par rapport à l'état futur réel du système.
En ce qui concerne l'horizon de commande, le choix $N_c = 1$ est reconnu comme étant suffisant dans la plupart des cas. Cependant, une valeur plus élevée permet d'atteindre des objectifs plus complexes dans des systèmes présentant un grand nombre de degrés

de liberté.

En terme d'optimisation, le problème peut se résumer à la minimisation d'une fonction coût J. On peut l'exprimer tel que :

$$\min_{\boldsymbol{u}(t_k)...\boldsymbol{u}(t_{k+N_c-1}) \in \mathcal{K}} J = \sum_{j=t_{k+1}}^{j=t_{k+N_p}} (\boldsymbol{r}(j) - \boldsymbol{y}(j))^2 \tag{1.23}$$

où $\mathcal{K}$ est le domaine des contraintes de fonctionnement.

Les grands intérêts de la commande prédictive sont à la fois de : limiter le champ d'action du robot à ce qui lui est demandé d'accomplir et à ce qui lui est physiquement possible par le biais de la fonction objectif et des contraintes ; obtenir un comportement plus "souple" du robot en anticipant les variations brutales des variables considérées grâce aux prédictions du modèle du système ; s'adapter aux changements d'état du système en le mettant à jour à chaque nouvelle mesure. Grâce à ses nombreux points forts, on retrouve la commande prédictive employée dans différentes applications que nous présentons ci-dessous.

1.3.2.3 Les domaines d'application

La commande prédictive a été utilisée en premier lieu dans les secteurs de l'industrie pétro-chimique et métallurgique. Le point commun de ces domaines d'activité est qu'ils impliquent de nombreuses variables et contraintes en entrée et en sortie de leurs systèmes, ainsi qu'une imprévisibilité importante sur leurs états futurs. Par exemple, la solution présentée dans [WH05] permet d'équilibrer le compromis entre la qualité et le rendement d'une raffinerie. D'autre part, dans [BLS89], une méthode basée sur la commande prédictive est employée pour le fonctionnement d'un système utilisé en métallurgie qui comporte plusieurs unités montées en série.

On retrouve également la commande prédictive employée dans le domaine de l'automobile. Elle est notamment utilisée pour le contrôle de la transmission dans [AAT08], pour la régulation des émissions de gaz en contrôlant l'apport en oxygène utilisé pour la combustion du carburant dans [TDCBK09], pour le contrôle des suspensions dans [MAH+97], ou encore pour le contrôle de la direction d'un véhicule autonome dans [FBA+07]. Le domaine médical est également intéressé par la commande prédictive. Elle est par exemple utilisée dans [VHHP+07] pour le contrôle des doses d'insuline à injecter à un patient malade.

L'aptitude de cette méthode à prendre en compte un grand nombre de variables permet aujourd'hui à différents domaines de la robotique de l'utiliser pour des applications de plus en plus complexes. On peut citer en exemple la marche dynamique d'un robot humanoïde basé sur la commande prédictive dans [DSW11] où les déséquilibres qui apportent de fortes perturbations dans le déplacement du robot sont compensées par la

commande. D'autre part, la dynamique du corps du robot HRP-2 est prise en compte pour atteindre une pose désirée dans [KDPT+15]. Dans cet exemple, Le robot est capable de prendre appui sur une table pour atteindre la pose désirée tout en gardant son équilibre.

1.3.2.4 Le lien avec l'asservissement visuel

La commande prédictive est ouverte à l'utilisation de différents types de systèmes robotiques, notamment ceux utilisant la vision. On trouve dans [ACC10, HAKE+14] une tâche d'asservissement visuel IBVS exécutée à l'aide de la commande prédictive. Cette dernière est utilisée pour assurer que les primitives visuelles ne sortent pas du champ de vue de la caméra en ajoutant des contraintes relatives à leurs coordonnées dans le plan image de la caméra. Cette méthode apporte une solution à un problème majeur de l'asservissement visuel, à savoir la perte des informations visuelles dans l'image que nous aborderons de manière plus approfondie dans le chapitre suivant.
Dans le cadre d'applications plus concrètes, on trouve dans [GGdM+04] et [GGdM+05], une méthode permettant de compenser visuellement les mouvements des organes d'un patient qui doit être opéré. Cette méthode anticipe les futurs mouvements des poumons ou du cœur pour que l'image offerte au chirurgien soit la plus nette possible. Finalement, la commande prédictive et la vision sont également utilisées dans [SS04, GSH+15] pour le déplacement et l'évitement d'obstacle d'un robot humanoïde, et dans [HAEK+14] pour la navigation et la stabilisation d'un sous-marin par rapport à une cible.

1.4 Conclusion

La vision en robotique est un domaine de recherche qui offre de nombreuses perspectives sur la commande des systèmes robotiques. Au delà du choix des informations visuelles à utiliser dans une commande d'asservissement visuel, les travaux réalisés sur la loi de commande permettent d'envisager des comportements très intéressants pour des systèmes robotiques complexes.

Aussi, la combinaison de la commande prédictive et de l'asservissement visuel est un sujet qui reste encore très nouveau et dans lequel différentes pistes de recherche restent à explorer. Dans le chapitre suivant, nous constatons que la prédiction du modèle employée dans la commande prédictive est un outil qui offre des perspectives très intéressantes dans l'asservissement visuel soumis à des pertes d'informations visuelles. Dans le dernier chapitre de ce document, nous étudions comment les contraintes utilisées dans la commande prédictive permettent de réaliser une succession de tâches d'asservissement visuel.

Chapitre 2

Correction des modèles de primitives visuelles

Dans ce chapitre, nous nous concentrons sur le schéma de commande de l'asservissement visuel introduit précédemment dans la section 1.3.1. Nous nous penchons dans un premier temps sur un problème pouvant être rencontré lors de la mise en œuvre de la loi de commande, qui est la perte des primitives visuelles dans l'image. Aussi, nous présentons par la suite le modèle et la dynamique de ces primitives visuelles permettant à une commande de faire face à tout événement qui implique la perte de leur suivi dans l'image par le biais de leurs prédictions. Enfin, une méthode inédite de correction du modèle des primitives visuelles est présentée et comparée à d'autres méthodes classiques employées pour assurer une bonne correspondance entre ce modèle et la réalité. Des résultats de simulation et expérimentaux sont proposés pour illustrer l'intérêt de cette nouvelle méthode.

2.1 Perte des primitives visuelles

Dans le cadre de tâches d'asservissement visuel telles que le suivi ou le positionnement, l'observation continue par une caméra durant le déplacement du robot est primordiale. Cette observation permet de considérer l'ensemble $\boldsymbol{s}$ comme mesurable à chaque instant, et donc d'appliquer les lois de commande (1.16) et (1.17). Elle peut cependant être entravée par différents événements internes ou externes au système. Ces différents événements sont décrits ci-dessous.

2.1.1 Occultation

Le premier cas de perte des primitives visuelles, illustré sur la Figure 2.1, est rencontré lorsqu'un obstacle parasite provoque une occultation. Cet obstacle peut se glisser entre la caméra et l'objet observé, ou rester fixe alors que la caméra ou l'objet observé se déplace et provoque l'occultation. Dans cette situation, le traitement d'image ne peut pas retrouver les coordonnées pixels du point puis calculer les coordonnées des primitives

visuelles comme il est censé le faire avec (1.7). La mesure des primitives visuelles est donc perdue.

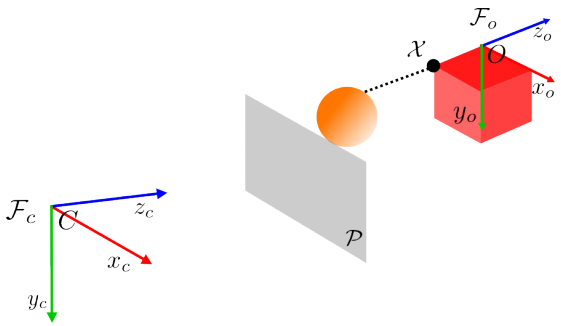


FIGURE 2.1 – Perte des informations visuelles : Le cas d'une occultation.

2.1.2 Sortie du champ de vue de la caméra

Le deuxième cas pouvant être rencontré est illustré sur la Figure 2.2. Cette situation intervient lorsque les coordonnées des primitives visuelles, issues de la projection du point sur le plan image de la caméra, sont en dehors du champ de vue de la caméra limité par les coordonnées $\boldsymbol{s}_{min} = [s_{min_x}\ s_{min_y}]$ et $\boldsymbol{s}_{max} = [s_{max_x}\ s_{max_y}]$. Le même problème rencontré dans le cas précédent intervient. Le traitement d'image ne peut pas identifier les coordonnées des primitives visuelles dans l'image qui s'en retrouvent perdues.

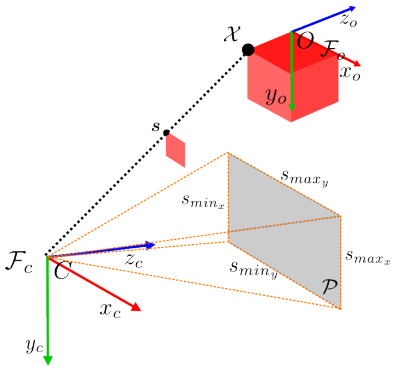


FIGURE 2.2 – Perte des informations visuelles : Le cas d'une sortie de l'objet en dehors du champ de vue de la caméra.

2.1.3 Erreur de traitement

Le dernier cas de perte de primitives visuelles est illustré sur la Figure 2.3. Cette situation peut être rencontrée si le résultat du traitement d'image utilisé est erroné. Ce

résultat peut laisser entendre que les coordonnées des primitives visuelles sont en dehors du champ de vue de la caméra comme pour le cas de la section 2.1.2, ou qu'elles sont à des coordonnées ne correspondant pas à la projection de l'information sur le plan image de la caméra, ou encore qu'elles sont simplement inexistantes.

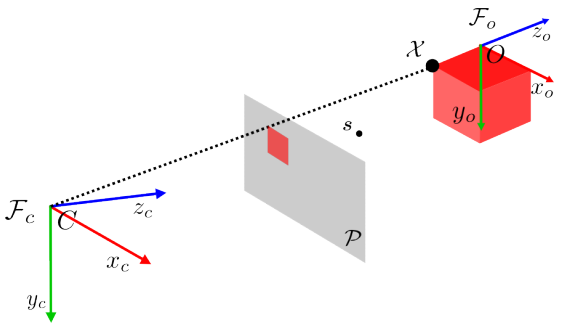


FIGURE 2.3 – Perte des informations visuelles : Le cas d'une erreur de traitement d'image.

L'ensemble de ces événements peuvent intervenir lors d'une tâche d'asservissement visuel. Des solutions existent cependant pour faire face à ces événements. Celles-ci sont décrites dans la partie suivante.

2.1.4 Faire face aux pertes

Dans la littérature de l'asservissement visuel, il existe deux grandes approches pour traiter les occultations ou les pertes de suivi des primitives visuelles en dehors du champ de vue de la caméra. Dans la première approche, des schémas de commande permettent d'éviter qu'un de ces évenements ne se produise. Dans la deuxième, d'autres schémas sont basés sur l'idée qu'un de ces événements peut arriver et proposent des stratégies pour accomplir la tâche d'asservissement quoi qu'il arrive.

De nombreuses techniques ont été proposées pour éviter toute perte d'information visuelle. L'évitement des occultations est exprimée comme une tâche secondaire associée à une tâche visuelle classique en utilisant la méthode de projection de gradient dans [MH98]. Cette méthode a également été utilisée et combinée avec des tâches séquencées pour permettre l'évitement d'obstacles dans [FC05] . Éviter que les primitives quittent le champ de vue de la caméra est pris en compte dans le choix des primitives visuelles dans [MLS+00]. Des méthodes explicites de changement de schémas de commande ont été proposées dans [CHPV04, GH06]. Une série de travaux présentée dans [MC02b, HNJ07, SG12, KGM13] a conduit à une planification de la trajectoire visuelle en satisfaisant plusieurs contraintes, telles que l'évitement des occultations et des pertes de suivi des primitives visuelles dans l'image. Enfin, comme évoqué dans la section 1.3.2, la commande prédictive basée modèle a été appliquée à l'asservissement visuel pour empêcher la sortie des primitives visuelles en dehors du champ de vue de

la caméra via les contraintes de fonctionnement [MYF06, SPDC06, LBC11, ACT08, ACC10, AJS14].

Une tâche d'asservissement visuel peut également être réalisée malgré la présence d'occultations ou de pertes d'informations si un nombre suffisant de primitives visibles est disponible pendant le mouvement du robot. La solution la plus simple consiste simplement à retirer de l'ensemble des primitives celles qui ne sont plus disponibles ou volontairement écartées car jugées mauvaises. Cependant, un changement dans l'ensemble des primitives implique un changement dans le nombre de lignes de la matrice d'interaction $\boldsymbol{L}_s$, et donc une discontinuité dans la sortie de la commande. Pour résoudre ce problème, un poids continu peut être associé à chaque primitive comme proposé dans [GAMASPV05] : ce poids varie entre 1, lorsque la fonction est située près du centre de l'image, et 0 lorsque la fonction atteint la limite de l'image. Cela garantit donc la continuité de la sortie du système de contrôle. Cette stratégie n'est cependant plus efficace lorsque le nombre de primitives visibles devient trop faible.
Un dernière solution consiste à remplacer efficacement les mesures manquantes des primitives visuelles à l'aide de différentes méthodes issues de la reconstruction d'images. En effet, l'image d'un objet observé par une caméra et suivi par un algorithme de traitement d'images peut être reconstruite efficacement lorsqu'elle devient flou [FS03] ou lorsqu'une occultation intervient [GVPG03, JYS04]. La prédiction des coordonnées des primitives visuelles utilisées dans la loi de commande peut alors être exploitée lorsque celles-ci ne sont plus disponibles, tel qu'il a été réalisé pour un système de contrôle impliquant une caméra embarquée sur un robot mobile dans [FC08]. À noter finalement que la prédiction permet également d'aider le traitement d'image à retrouver la primitive lorsque celle-ci redevient observable par la caméra.

Cette dernière stratégie nous permet de considérer la prédiction des primitives visuelles comme un point central pour la réalisation d'une tâche d'asservissement visuel soumise aux événements provoquant des pertes de primitives visuelles.

2.2 Prédiction des primitives visuelles

Cette section présente les méthodes consistant à prédire les coordonnées des primitives visuelles dans le futur. Le but est de s'affranchir de nouvelles observations dans le cas où les primitives visuelles sont perdues, tout en étant capable d'estimer leurs positions dans l'image.

À titre d'exemple, on considère un système qui suit une configuration eye-in-hand. Il est constitué d'une caméra monoculaire montée sur un bras robotique dont la position est définie par le repère $\mathcal{F}_c$, qui observe une cible définie en position par le repère $\mathcal{F}_o$ constituée de $\mathcal{N}$ points 3D. Cette configuration est illustrée sur la Figure 2.4, où la cible représentée en rouge est consituée de 4 points.
On définit alors l'ensemble des primitives visuelles $\boldsymbol{s}$ en fonction de la projection pers-

pective (1.6) de chaque point sur le plan image de la caméra :

$$\boldsymbol{s} = \begin{bmatrix} \boldsymbol{s}_1 \\ \vdots \\ \boldsymbol{s}_{\mathcal{N}} \end{bmatrix} \in \mathbb{R}^{2\mathcal{N}}, \tag{2.1}$$

où $\boldsymbol{s}_i$ correspond aux coordonnées des primitives visuelles du $i$ème point de la cible sur le plan image de la caméra. On rappelle par ailleurs que ces coordonnées peuvent être retrouvées en utilisant la connaissance des coordonnées 3D des points dans le repère $\mathcal{F}_o$ et la pose entre la caméra et la cible introduite dans (1.6)) :

$$\boldsymbol{s}_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} {}^cX_i/{}^cZ_i \\ {}^cY_i/{}^cZ_i \end{bmatrix} = f_s({}^c\mathbf{P}_o, {}^o\boldsymbol{X}_i). \tag{2.2}$$

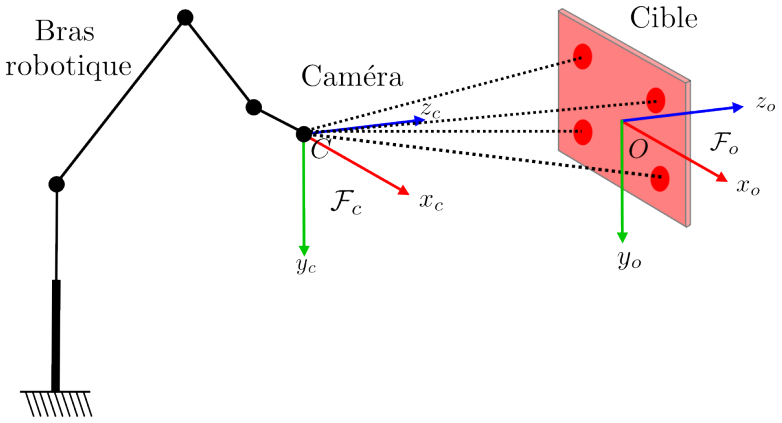


FIGURE 2.4 – Système considéré : Une caméra montée sur un bras robotique observe une cible représentée en rouge et constituée de 4 points.

L'objectif consiste à présent à prédire les coordonnées des primitives visuelles $\boldsymbol{s}$ via deux méthodes différentes présentées dans la section suivante.

2.2.1 Dynamique des primitives visuelles

Pour obtenir la prédiction des coordonnées des primitives visuelles dans l'image, deux méthodes de prédiction peuvent être utilisées. La première utilise la dynamique des primitives visuelles $\boldsymbol{s}$ directement dans l'image alors que la seconde utilise la dynamique de la pose de l'objet dans le repère de la caméra ${}^c\mathbf{P}_o$, ce qui permet de calculer les coordonnées des primitives visuelles comme dans l'expression (2.2). Ces deux méthodes sont basées sur l'expression de la vitesse de la caméra exprimée dans son repère :

$$\mathbf{v}_c = (\boldsymbol{v},\ \boldsymbol{\omega}) \in \mathbb{R}^6 \tag{2.3}$$

avec $\boldsymbol{v} = [v_x\ v_y\ v_z]$ la vitesse linéaire et $\boldsymbol{\omega} = [\omega_x\ \omega_y\ \omega_z]$ la vitesse angulaire appliquée à la caméra. Cette vitesse est considérée comme toujours connue, car elle est le résultat de

la loi de commande (1.16) appliquée à la caméra. En ce qui concerne l'objet observé, sa vitesse ne pouvant pas être connue dans un système eye-in-hand, il doit rester statique lors de l'asservissement visuel si l'on souhaite de bons résultats de prédiction.

Afin de représenter de manière explicite les coordonnées des primitives visuelles et la pose issues des méthodes de prédiction, nous utilisons l'expression de leurs modèles correspondants $\boldsymbol{s}_m$ et ${}^c\mathbf{P}_{om}$.

Dans la suite de cette partie, on considère que $\boldsymbol{s}(t)$ peut être mesuré jusqu'à un temps $t = t_{occ}$, après quoi apparaît un des événements décrits dans la section 2.1 et où $s(t)$ ne peut plus être mesuré durant une période $T > 0$. Les méthodes décrites ci-dessous sont donc appliquées à $\boldsymbol{s}_m$ et ${}^c\mathbf{P}_{om}$ et utilisées durant la période $[t_{occ},\ t_{occ} + T]$ discrétisée en H étapes de temps uniformes d'une durée de $\tau = \frac{T}{H}$. Ce temps τ correspond à la période d'échantillonnage de la caméra, afin que chaque pas de la prédiction corresponde au résultat de la loi de commande (1.16) qui dépend de chaque nouvelle mesure de la caméra.

2.2.1.1 Prédiction dans l'image

La prédiction dans l'image, qui a déjà été utilisée dans [FC08], consiste en une intégration simple du modèle des primitives visuelles $\boldsymbol{s}_m(t)$ via la matrice d'interaction $\boldsymbol{L}_s$ utilisée plus communément dans la commande par asservissement visuel présentée dans la section 1.3.1 et définie telle que $\dot{\boldsymbol{s}} = \boldsymbol{L}_s \mathbf{v}_c$. La matrice d'interaction associée à un point a été introduite dans (1.18). Celle-ci est obtenue en fonction des coordonnées x et y des primitives visuelles, ainsi que de la coordonnée de la profondeur ${}^cZ$ de chaque point dans le repère de la caméra. La matrice d'interaction associée à l'ensemble de $\mathcal{N}$ points est donc :

$$\boldsymbol{L}_s(\boldsymbol{s}_m, {}^c\boldsymbol{Z}_m) = \begin{bmatrix} \boldsymbol{L}_{s_1}(\boldsymbol{s}_{m_1}, {}^cZ_{m_1}) \\ \vdots \\ \boldsymbol{L}_{s_\mathcal{N}}(\boldsymbol{s}_{m_\mathcal{N}}, {}^cZ_{m_\mathcal{N}}) \end{bmatrix} \in \mathbb{R}^{2\mathcal{N}\times 6}, \tag{2.4}$$

avec la matrice d'interaction associée au $i$ème point :

$$\boldsymbol{L}_{s_i}(\boldsymbol{s}_{m_i}, {}^cZ_{m_i}) = \begin{bmatrix} -\frac{1}{{}^cZ_{m_i}} & 0 & \frac{x_{m_i}}{{}^cZ_{m_i}} & x_{m_i}y_{m_i} & -(1+x_{m_i}^2) & y_{m_i} \\ 0 & -\frac{1}{{}^cZ_{m_i}} & \frac{y_{m_i}}{{}^cZ_{m_i}} & 1+y_{m_i}^2 & -x_{m_i}y_{m_i} & -x_{m_i} \end{bmatrix} \in \mathbb{R}^{2\times 6}. \tag{2.5}$$

De là, on peut prédire les futurs modèles des coordonnées des $\mathcal{N}$ points tel que :

$$\boldsymbol{s}_m(t_{k+1}) = \boldsymbol{s}_m(t_k) + \tau \boldsymbol{L}_s(\boldsymbol{s}_m(t_k),\ {}^c\boldsymbol{Z}_m(t_k))\ \mathbf{v}_c(t_k), \tag{2.6}$$

avec t_k le temps courant considéré et t_{k+1} le temps futur après un temps τ. On précise que la vitesse $\mathbf{v}_c$ est supposée constante de t_k à t_{k+1}. On introduit ici ${}^c\boldsymbol{Z}_m =$

$[{}^cZ_{m_1}, \ldots, {}^cZ_{m_\mathcal{N}}]$, le modèle correspondant aux coordonnées ${}^cZ$ des points dans le repère de la caméra, qui peut être également prédit de la façon suivante :

$$ {}^c\boldsymbol{Z}_m(t_{k+1}) = {}^c\boldsymbol{Z}_m(t_k) + \tau \boldsymbol{L}_Z(\boldsymbol{s}_m(t_k), {}^c\boldsymbol{Z}_m(t_k))\, \mathbf{v}_c(t_k), \tag{2.7} $$

avec

$$ \boldsymbol{L}_Z(\boldsymbol{s}_m, {}^c\boldsymbol{Z}_m) = \begin{bmatrix} \boldsymbol{L}_{Z_1}(\boldsymbol{s}_{m_1}, {}^cZ_{m_1}) \\ \vdots \\ \boldsymbol{L}_{Z_\mathcal{N}}(\boldsymbol{s}_{m_\mathcal{N}}, {}^cZ_{m_\mathcal{N}}) \end{bmatrix} \in \mathbb{R}^{\mathcal{N}\times 6}, \tag{2.8} $$

et où :

$$ \boldsymbol{L}_{Z_i}(\boldsymbol{s}_{m_i}, {}^cZ_{m_i}) = \begin{bmatrix} 0 & 0 & -1 & -y_{m_i}{}^cZ_{m_i} & x_{m_i}{}^cZ_{m_i} & 0 \end{bmatrix}. \tag{2.9} $$

En utilisant (2.6) et (2.7), le comportement des coordonnées des modèles des primitives visuelles $\boldsymbol{s}_m(t)$ et de ${}^c\boldsymbol{Z}_m(t)$ peut être prédit sur l'intervalle de temps $[t_{occ},\, t_{occ}+T]$ sans recourir à l'utilisation de la pose ${}^c\mathbf{P}_{om}(t)$. La précision de cette prédiction dépend donc de la précision des valeurs initiales $\boldsymbol{s}_m(t_{occ})$ et ${}^c\boldsymbol{Z}_m(t_{occ})$, et donc de la connaissance du modèle 3D de l'objet à $t = t_{occ}$.

On note cependant que si seulement des mouvements de rotation de la caméra sont impliqués dans le système de commande, par exemple pour la commande d'une caméra de type pan-tilt, la profondeur n'intervient plus dans la matrice d'interaction. En effet ${}^cZ_{m_i}$ n'apparaît seulement que dans les trois premiers colonnes de (2.5) qui correspondent aux trois mouvements de translation. Dans ce cas, la prédiction de ${}^c\boldsymbol{Z}_m(t)$ via (2.7) s'avère inutile, et la connaissance du modèle 3D de l'objet également.

La seconde méthode de prédiction est basée sur le modèle 3D de l'objet et le modèle de la pose de l'objet dans le repère de la caméra ${}^c\mathbf{P}_{om}$.

2.2.1.2 Prédiction de la pose

La prédiction de la pose est basée sur la propagation du modèle ${}^c\mathbf{P}_{om}(t)$ tel que :

$$ {}^c\mathbf{P}_{om}(t_{k+1}) = {}^c\mathbf{P}_{om}(t_k) + \tau \boldsymbol{L}_P({}^c\mathbf{P}_{om}(t_k))\mathbf{v}_c(t_k), \tag{2.10} $$

où la matrice d'interaction liée à la pose a été introduite dans (1.21) et est définie ici tel que :

$$ \boldsymbol{L}_P = \begin{bmatrix} -\boldsymbol{I}_3 & [{}^c\mathbf{t}_o]_\times \\ \mathbf{0} & \boldsymbol{L}_\omega(\boldsymbol{u},\, \theta) \end{bmatrix}, \tag{2.11} $$

avec

$$ \boldsymbol{L}_\omega(\boldsymbol{u},\, \theta) = \boldsymbol{I}_3 - \frac{\theta}{2}[\boldsymbol{u}]_\times + \left(1 - \frac{\mathrm{sinc}(\theta)}{\mathrm{sinc}^2\,(\theta/2)}\right)[\boldsymbol{u}]_\times^2, \tag{2.12} $$

où $\mathrm{sinc}(\theta) = \sin(\theta)/\theta$, [MCB99]. On peut noter que la matrice $\boldsymbol{L}_P$ est inversible pour $\theta \neq 2k\pi$ avec $k \in \mathbb{Z}^*$, puisque $\det(\boldsymbol{L}_\omega(\boldsymbol{u},\, \theta)) = 1/\mathrm{sinc}^2\,(\theta/2)$, . Cette caractéristique signifie qu'elle est inversible dans n'importe quelle application réelle.

On peut obtenir alors la prédiction du modèle des primitives visuelles $\boldsymbol{s}_m(t)$ via (2.2) pendant l'intervalle $[t_{occ},\, t_{occ} + T]$, c'est-à-dire lorsque les mesures ne sont plus disponibles par la caméra. La différence avec le schéma de prédiction précédent est que l'évolution de la pose ${}^c\mathbf{P}_{om}(t)$ est estimée de manière continue alors que seulement $\boldsymbol{s}_m(t)$ et ${}^c\boldsymbol{Z}_m(t)$ le sont avec (2.6) et (2.7). De plus, cette méthode contrairement à la précédente, requiert de toujours connaître le modèle 3D de l'objet observé par la caméra même si seulement des mouvements de rotation sont impliqués, c'est-à-dire d'avoir de bonnes valeurs des coordonnées de chaqu'un des $\mathcal{N}$ points dans le repère $\mathcal{F}_o$. Tout comme précédemment, la précision de ce schéma de prédiction dépend de la précision de la valeur initiale ${}^c\mathbf{P}_{om}(t_{occ})$.

2.2.1.3 Comparaison des deux méthodes de prédiction

Les deux méthodes décrites ci-dessus offrent des résultats similaires. Une différence intervient cependant si la mise à jour de ${}^c\boldsymbol{Z}_m$ n'est pas effectuée via (2.7) lorsque la vitesse appliquée à la caméra n'est pas une rotation pure.

Pour illustrer cela, nous considérons qu'une vitesse constante $\mathbf{v}_c(t)$ est appliquée au robot, sur un temps $T = 4$ s discrétisé en $H = 100$ étapes avec un temps d'échantillonnage de $\tau = 40$ ms. La cible observée par la caméra est constituée de quatre points coplanaires formant un carré de 20 cm de longueur dont les coordonnées sont les suivantes :

$$\begin{cases} {}^o\boldsymbol{X}_0 &= \begin{pmatrix} -0.1 & -0.1 & 0 & 1 \end{pmatrix}^T \\ {}^o\boldsymbol{X}_1 &= \begin{pmatrix} 0.1 & -0.1 & 0 & 1 \end{pmatrix}^T \\ {}^o\boldsymbol{X}_2 &= \begin{pmatrix} 0.1 & 0.1 & 0 & 1 \end{pmatrix}^T \\ {}^o\boldsymbol{X}_3 &= \begin{pmatrix} -0.1 & 0.1 & 0 & 1 \end{pmatrix}^T \end{cases}.$$

Nous étudions alors deux cas différents. Dans le premier, une trajectoire de rotation pure appliquée au système présenté sur la Figure 2.4. Dans le second, une trajectoire complète faisant intervenir des modifications de ${}^c\boldsymbol{Z}_m$ est appliquée au même système.

Rotation pure : La vitesse appliquée au robot dans ce premier cas est la suivante :

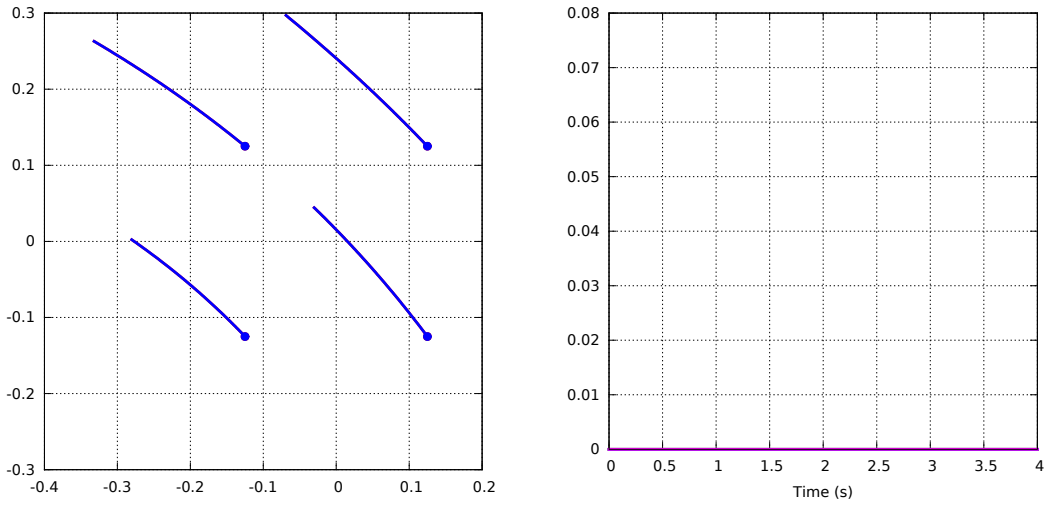
$$\mathbf{v}_c(t) = \begin{bmatrix} 0 & 0 & 0 & 0.04 & 0.04 & -0.04 \end{bmatrix}^T = const \tag{2.13}$$

Le mouvement de la caméra est une rotation autour des axes x, y, z et ne fait pas intervenir de modification de ${}^c\boldsymbol{Z}_m$ dans la prédiction (2.6). Sur la Figure 2.5a, les quatre points représentés en bleu correspondent aux coordonnées initiales des primitives visuelles sur le plan image de la caméra à $t = t_{occ}$. Les courbes représentent leurs prédictions jusqu'à $t = t_{occ} + T$. Les méthodes utilisées sont :

- La prédiction dans l'image (2.6) sans aucune connaissance de ${}^c\boldsymbol{Z}_m$ à $t = t_{occ}$ ni à $t > t_{occ}$.
- La prédiction dans l'image (2.6) avec la mise à jour de ${}^c\boldsymbol{Z}_m$ via (2.7). Le modèle 3D de l'objet est alors connu à $t = t_{occ}$.

• La prédiction de la pose (2.10). Le modèle 3D de l'objet est alors connu à $t = t_{occ}$.

On observe que les trois prédictions offrent exactement les mêmes résultats. En complément, la Figure 2.5b illustre les erreurs entre les prédictions des primitives visuelles et leurs valeurs réelles $e_i(t) = \|\boldsymbol{s}_i(t) - \boldsymbol{s}_{im}(t)\|$ qui sont nulles quelle que soit la méthode employée. Ces résultats permettent d'illustrer l'intérêt de la prédiction image (2.6) sans qu'aucune connaissance du modèle 3D de l'objet ne soit nécessaire, dans le cas particulier où la trajectoire appliquée au robot est une rotation pure.



(a) (b)

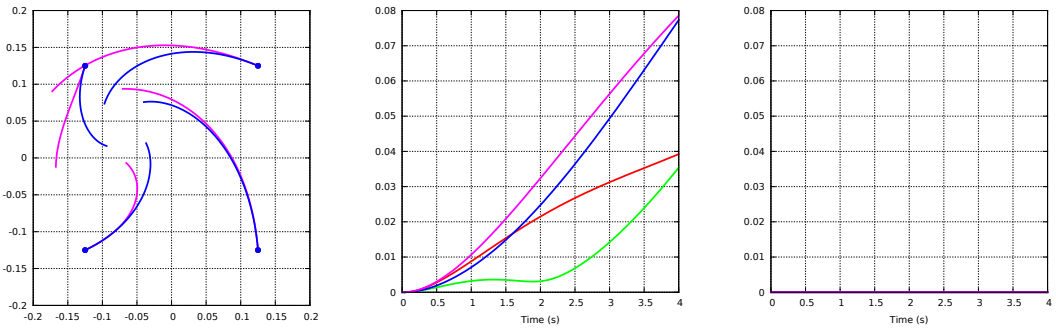
FIGURE 2.5 – Trajectoire de rotation pure : (a) Coordonnées prédites des modèles des primitives visuelles $\boldsymbol{s}_m$ dans l'image avec (2.6) sans la mise à jour de ${}^c\boldsymbol{Z}_m$, ou avec (2.6)–(2.7) ou (2.10). (b) Erreurs $e_i(t) = \|\boldsymbol{s}_i(t) - \boldsymbol{s}_{im}(t)\|$ nulles quelque soit la prédiction utilisée.

Trajectoire complète : Dans ce second cas, la vitesse appliquée au robot fait intervenir des modifications des valeurs de ${}^c\boldsymbol{Z}_m$:

$$\mathbf{v}_c(t) = \begin{bmatrix} -0.02 & 0 & -0.3 & 0.04 & 0.04 & -0.4 \end{bmatrix}^T = const \tag{2.14}$$

Sur la Figure 2.6a, les quatre points représentés en bleu correspondent aux coordonnées initiales des primitives visuelles sur le plan image de la caméra à $t = t_{occ}$. Les courbes représentent leurs prédictions jusqu'à $t = t_{occ} + T$. Les résultats de (2.6) sans la mise à jour de ${}^c\boldsymbol{Z}_m$ sont représentés en magenta, alors que les résultats obtenus soit grâce à (2.6) et la mise à jour de ${}^c\boldsymbol{Z}_m$ via (2.7) soit avec la seconde méthode (2.10) sont représentées en bleu. En complément, la Figure 2.6b illustre les erreurs entre les prédictions des primitives visuelles et leurs valeurs réelles $e_i(t) = \|\boldsymbol{s}_i(t) - \boldsymbol{s}_{im}(t)\|$ obtenues avec (2.6) sans la mise à jour de ${}^c\boldsymbol{Z}_m$. On peut comparer ces erreurs avec celles obtenues soit avec (2.6) et la mise à jour de ${}^c\boldsymbol{Z}_m$, soit avec (2.10) et représentées sur la Figure 2.6c.

On observe que pour une vitesse comportant des modifications des coordonnées ${}^c\boldsymbol{Z}_m$ durant le déplacement de la caméra, la méthode (2.6) sans la mise à jour de ${}^c\boldsymbol{Z}_m$ crée un écart important entre les prédictions et les trajectoires réelles des primitives visuelles sur le plan image de la caméra. En revanche avec la prédiction de pose (2.10) ou avec la mise à jour de ${}^c\boldsymbol{Z}_m$, les résultats des deux prédictions sont strictement identiques et correspondent aux trajectoires réelles des primitives visuelles. On préférera donc utiliser (2.6) avec la mise à jour de ${}^c\boldsymbol{Z}_m$ (2.7) ou (2.10) pour prédire efficacement les coordonnées des primitives visuelles dans l'image dans le cadre d'un asservissement visuel confronté à des pertes d'informations visuelles. Il apparaît ici que si les primitives visuelles redeviennent observable par la caméra à $t > t_{occ} + T$ le traitement d'image pourra retrouver plus facilement les primitives sur l'image capturée par la caméra.



(a) (b) (c)

FIGURE 2.6 – Trajectoire complète : (a) Coordonnées prédites des modèles des primitives visuelles $\boldsymbol{s}_m$ dans l'image avec (2.6) en magenta, et avec (2.6)–(2.7) ou (2.10) en bleu. (b) Erreurs $e_i(t) = \|\boldsymbol{s}_i(t) - \boldsymbol{s}_{im}(t)\|$ avec la prédiction (2.6) (c) Erreurs nulles avec les prédictions (2.6)–(2.7) ou (2.10).

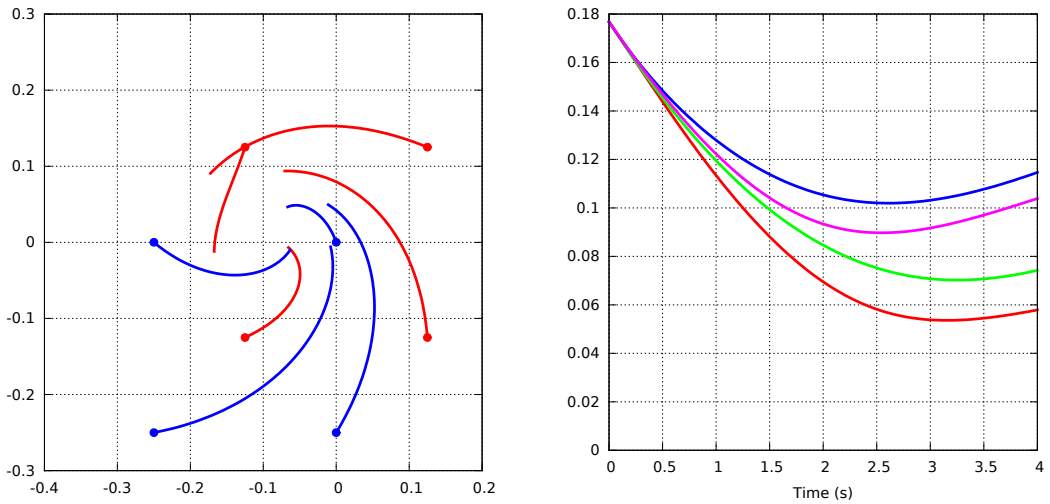
Nous avons présenté dans cette section deux méthodes différentes permettant de prédire les modèles des coordonnées des primitives visuelles dans l'image. Il est à noter que ces méthodes sont basées sur des résolutions du premier ordre d'équations différentielles. La qualité des résultats de prédiction produite est donc la moins bonne en comparaison avec celle pouvant être obtenue avec d'autres méthodes de résolutions plus précises. Cependant, le but de la section suivante a pour but de proposer de nouvelles méthodes permettant de corriger l'écart entre les modèles des primitives visuelles et leurs valeurs réelles. Il est alors judicieux d'utiliser les méthodes analytiques les plus simples, comme celles qui ont été présentées ci-dessus, afin d'obtenir les erreurs de prédiction les plus importantes qui peuvent être rencontrées dans un cas réel.

Dans la section suivante, nous proposons différentes méthodes permettant d'exploiter les mesures $\boldsymbol{s}$ pour corriger les modèles $\boldsymbol{s}_m$ et ${}^c\mathbf{P}_{om}$ à $t = t_{occ}$ de manière à obtenir de bons résultats de prédiction.

2.3 Correction des modèles

Les méthodes présentées ci-dessus ont pour but d'utiliser la prédiction pour se substituer aux mesures qui ne peuvent être obtenues à cause d'un événement provoquant la perte des primitives visuelles dans l'image. Elles doivent donc offrir des résultats équivalents aux comportements réels des primitives visuelles dans l'image. Pour cela, une correspondance exacte entre $\boldsymbol{s}$, ${}^c\boldsymbol{Z}$, ${}^c\mathbf{P}_o$ et leurs modèles correspondants $\boldsymbol{s}_m$, ${}^c\boldsymbol{Z}_m$, ${}^c\mathbf{P}_{om}$ doit être assurée lorsque un événement survient juste après la dernière mesure assurée à $t = t_{occ}$.

En effet, si les modèles sont mal estimés on peut obtenir des résultats de prédiction ne correspondant pas aux coordonnées réelles des primitives visuelles dans l'image. Par exemple, la vitesse définie dans (2.14) est une nouvelle fois appliquée au même système présenté dans la section 2.2.1.3, mais une erreur entre $\boldsymbol{s}$, ${}^c\boldsymbol{Z}$, ${}^c\mathbf{P}_o$ et leurs modèles $\boldsymbol{s}_m$, ${}^c\boldsymbol{Z}_m$, ${}^c\mathbf{P}_{om}$ est introduite à $t = t_{occ}$. Cette erreur se traduit simplement par une translation le long des axes x et y entre la pose réelle et modélisée. On observe sur la Figure 2.7a l'écart entre les coordonnées réelles des primitives visuelles représentées en rouge et les prédictions de leurs modèles en bleu. La Figure 2.7b représente l'évolution des erreurs entre les coordonnées réelles et prédites des primites visuelles.



(a) (b)

FIGURE 2.7 – Effets d'une erreur de modélisation sur la prédiction : (a) Coordonnées réelles des primitives visuelles en rouge, et prédites en bleu. (b) Erreurs $e_i(t) = \|\boldsymbol{s}_i(t) - \boldsymbol{s}_{im}(t)\|$ avec les prédictions (2.6)–(2.7) ou (2.10).

En pratique cette erreur peut intervenir après la fin d'un premier événement provoquant la perte des informations visuelles, c'est-à-dire au moment où les primitives visuelles réapparaissent dans l'image. Lors de cette événement, l'une des méthodes de prédiction (2.6)– (2.7) ou (2.10) a été utilisée mais a été sujette à des erreurs d'ap-

proximation. En effet, ces méthodes exploitent les vitesses du robot issues de la loi de commande qui, à cause de bruits, peuvent présentées des écarts plus ou moins importants avec les vitesses réelles appliquées au robot. Elles utilisent également le modèle 3D de l'objet observé qui, s'il n'est pas parfaitement connu, peut engendrer des erreurs entre les trajectoires réelles et prédites des coordonnées des primitives visuelles dans l'image. Finalement, l'erreur entre les coordonnées réelles et modélisées des primitives visuelles, qui apparaît après le premier événement, doit absolument être corrigée dans le cas où un second événement interviendrait et qu'une nouvelle phase de prédiction doit être réalisée.

Dans la suite de cette section, on se place dans le cas décrit ci-dessus, à savoir qu'une première phase prédiction a été réalisée et a engendré des erreurs entre les primitives visuelles réelles et modélisées, et qu'un second événement provoquant la perte des informations visuelles intervient juste après une mesure effectuée à $t = t_{occ}$.

Nous décrivons, par la suite, comment cette erreur peut être corrigée à partir des mesures $\boldsymbol{s}$ effectuées à $t = t_{occ}$. Pour cela, nous présentons plusieurs méthodes de correction des modèles des primitives visuelles et de la pose.

2.3.1 Correction dans l'image

La méthode classique de correction des modèles des primitives proposée dans [ACC10] agit directement sur le modèle des primitives visuelles. Elle vise à corriger le modèle $\boldsymbol{s}_{m_i}(t_k)$ en utilisant la mesure $\boldsymbol{s}_i(t_k)$ de chaque point. Elle permet ensuite de corriger les coordonnées ${}^cX_{m_i}$ et ${}^cY_{m_i}$ dans le repère de la caméra, mais n'apporte aucune modification à ${}^cZ_{m_i}$. Cela se traduit par la formulation suivante :

$$\begin{cases} \boldsymbol{s}_{m_i}(t_k) & \leftarrow & \boldsymbol{s}_i(t_k), \\ {}^cX_{m_i}(t_k) & \leftarrow & x_{m_i}(t_k){}^cZ_{m_i}(t_k), \\ {}^cY_{m_i}(t_k) & \leftarrow & y_{m_i}(t_k){}^cZ_{m_i}(t_k), \\ {}^cZ_{m_i}(t_k) & \leftarrow & {}^cZ_{m_i}(t_k). \end{cases} \tag{2.15}$$

Cette méthode garantit clairement que $\boldsymbol{s}_{m_i}(t_{occ}) = \boldsymbol{s}_i(t_{occ})$ lorsque la mesure est disponible pour la dernière fois. Elle est par contre incapable de corriger les potentielles erreurs dans le modèle des profondeurs ${}^cZ_{m_i}(t_{occ})$, ce qui peut provoquer une erreur entre $\boldsymbol{s}_i(t)$ et $\boldsymbol{s}_{m_i}(t)$ pour $t > t_{occ}$ En effet cette correction n'a aucune incidence sur le modèle de la pose ${}^c\mathbf{P}_{om}(t)$ qui présente toujours un écart potentiel avec la pose réelle ${}^c\mathbf{P}_o(t)$. Ce problème est justement traité à l'aide la méthode qui suit.

2.3.2 Correction du modèle de pose

La correction du modèle de pose propose d'exploiter l'écart entre le modèle $\boldsymbol{s}_m(t_k)$ et la mesure $\boldsymbol{s}(t_k)$ pour obtenir la différence de pose entre les poses réelle et modélisée $\Delta{}^c\mathbf{P}_o$, puis utiliser cette différence pour corriger la pose ${}^c\mathbf{P}_{om}(t_k)$.

Pour cela, on prend en compte la vitesse linéaire et angulaire exprimée dans $\mathcal{F}_C$, pour exprimer la loi de cinématique standard qui donne :

$$ {}^c\dot{\mathbf{P}}_o = \boldsymbol{L}_x({}^c\mathbf{P}_o)\mathbf{v}_c, \qquad \boldsymbol{L}_x \in \mathbb{R}^{6\times 6}, \tag{2.16} $$

où $\boldsymbol{L}_x$ permet de lier la dynamique de la pose à la vitesse appliquée à la caméra tel que :

$$ \boldsymbol{L}_x = \begin{bmatrix} {}^c\mathbf{R}_o(\boldsymbol{u},\, \theta) & \mathbf{0} \\ \mathbf{0} & \boldsymbol{L}_\omega(\boldsymbol{u},\, \theta) \end{bmatrix} \tag{2.17} $$

avec ${}^c\mathbf{R}_o(\boldsymbol{u},\, \theta)$ la matrice de rotation du repère de la caméra vers le repère de l'objet définie dans (1.4) et $\boldsymbol{L}_\omega$ définie dans (2.12).
D'autre part, on a déjà vu que les modèles standards de la dynamique des primitives visuelles sont donnés par [CH06] :

$$ \dot{\boldsymbol{s}} = \boldsymbol{L}_s(\boldsymbol{s},\, {}^c\boldsymbol{Z})\mathbf{v}_c, \qquad \boldsymbol{L}_s \in \mathbb{R}^{2\mathcal{N}\times 6} \tag{2.18} $$

En utilisant les deux expressions précédentes, on obtient :

$$ \dot{\boldsymbol{s}} = \boldsymbol{L}_s\boldsymbol{L}_x^{-1}{}^c\dot{\mathbf{P}}_o. \tag{2.19} $$

On obtient ainsi un développement de Taylor du premier ordre que l'on peut discrétiser tel que :

$$ \boldsymbol{s}(t_k) - \boldsymbol{s}_m(t_k) \approx \boldsymbol{L}_s(\boldsymbol{s}_m(t_k),\, {}^c\boldsymbol{Z}_m(t_k))\boldsymbol{L}_x^{-1}({}^c\mathbf{P}_{om}(t_k))\Delta^c\mathbf{P}_o. \tag{2.20} $$

On rappelle que l'on souhaite obtenir la différence entre les poses réelle et modélisée $\Delta^c\mathbf{P}_o$ à l'aide des mesures $\boldsymbol{s}$ à chaque instant t_k, que l'on peut exprimer à l'aide de la méthode des moindres carrés :

$$ \min_{\Delta^c\mathbf{P}_o} \|\boldsymbol{s}_m(t_k) - \boldsymbol{s}(t_k)\|^2 \tag{2.21} $$

qui peut être reformulée en utilisant l'expression (2.20), tel que :

$$ \min_{\Delta^c\mathbf{P}_o} \|\boldsymbol{L}_s\boldsymbol{L}_x^{-1}\Delta^c\mathbf{P}_o + \boldsymbol{s}_m(t_k) - \boldsymbol{s}(t_k)\|^2 \tag{2.22} $$

ce qui permet d'obtenir :

$$ \begin{aligned} \Delta^c\mathbf{P}_o(t_k) &= (\boldsymbol{L}_s\boldsymbol{L}_x^{-1})^\dagger(\boldsymbol{s}(t_k) - \boldsymbol{s}_m(t_k)) \\ &= \boldsymbol{L}_x({}^c\mathbf{P}_{om}(t_k))\ \boldsymbol{L}_s^\dagger(\boldsymbol{s}_m(t_k),\, {}^c\boldsymbol{Z}_m(t_k))\ (\boldsymbol{s}(t_k) - \boldsymbol{s}_m(t_k)) \end{aligned} \tag{2.23} $$

avec $\boldsymbol{L}_s^\dagger$ la pseudoinverse de Moore-Penrose de $\boldsymbol{L}_s$ [Pen55].

La différence de poses $\Delta^c\mathbf{P}_o(t_k)$ peut alors être utilisée pour implémenter la correction de pose. Le nouveau modèle de pose ainsi évalué permet de calculer les modèles des

coordonnées des primitives visuelles et des coordonnées 3D ${}^cX_{m_i}$, ${}^cY_{m_i}$, ${}^cZ_{m_i}$ de chaque point dans le repère de la caméra :

$$\left\{\begin{array}{rcl} {}^c\mathbf{P}_{om}(t_k) & \leftarrow & {}^c\mathbf{P}_{om}(t_k) \oplus \Delta{}^c\mathbf{P}_o(t_k), \\ \boldsymbol{s}_{m_i}(t_k) & \leftarrow & f_s({}^c\mathbf{P}_{om}(t_k), {}^o\boldsymbol{X}_i), \\ {}^cX_{m_i}(t_k) & \leftarrow & f_X({}^c\mathbf{P}_{om}(t_k), {}^o\boldsymbol{X}_i), \\ {}^cY_{m_i}(t_k) & \leftarrow & f_Y({}^c\mathbf{P}_{om}(t_k), {}^o\boldsymbol{X}_i), \\ {}^cZ_{m_i}(t_k) & \leftarrow & f_Z({}^c\mathbf{P}_{om}(t_k), {}^o\boldsymbol{X}_i). \end{array}\right. \tag{2.24}$$

Grâce à la correction du modèle de la pose, cette méthode permet de corriger l'ensemble des modèles des primitives visuelles même si un seul point est visible dans l'image. Ici, $f_X(\cdot)$, $f_Y(\cdot)$, et $f_Z(\cdot)$ fournissent les coordonnées ${}^cX_{m_i}$, ${}^cY_{m_i}$, ${}^cZ_{m_i}$ obtenues directement depuis le modèle de la pose et des coordonnées du $i$ème point 3D dans le repère de l'objet. En ce qui concerne l'opération ${}^c\mathbf{P}_{om}(t_k) \leftarrow {}^c\mathbf{P}_{om}(t_k) \oplus \Delta{}^c\mathbf{P}_o(t_k)$, elle est effectuée de la manière suivante : en utilisant la relation entre $\theta\boldsymbol{u}$ et la matrice de rotation (1.4), ainsi que le vecteur de translation, il est possible de retrouver la matrice de transformation homogène définie dans (1.3) à partir de chaque pose. Ainsi la matrice ${}^{om1}\mathbf{M}_{om}$ peut être obtenue de $\Delta{}^c\mathbf{P}_o$ et ${}^c\mathbf{M}_{om1}$ de la pose ${}^c\mathbf{P}_{om}$ à droite de l'équation. L'opération consiste alors à utiliser l'expression suivante :

$${}^c\mathbf{M}_{om} = {}^c\mathbf{M}_{om1}\,{}^{om1}\mathbf{M}_{om}, \tag{2.25}$$

et de retrouver la nouvelle pose corrigée ${}^c\mathbf{P}_{om}$ à gauche de l'équation grâce à ${}^c\mathbf{M}_{om}$.

Contrairement à la correction dans l'image (2.15), cette correction (2.24) réajuste le modèle de la pose de l'objet dans le repère de la caméra en exploitant les mesures des primitives visuelles $\boldsymbol{s}(t_k)$. Cependant, elle ne garantit pas que $\boldsymbol{s}_m(t_{occ}) = \boldsymbol{s}(t_{occ})$. En effet, l'expression utilisée (2.23) ne donne qu'une simple solution au problème des moindres carrés (2.21).

Il est à noter que la méthode de correction proposée ci-dessus correspond à l'application d'une seule itération d'une estimation de la pose par asservissement visuel virtuel décrite dans [MC02a]. Aussi, réaliser plusieurs itérations de la même méthode à chaque mesure disponible est possible, mais requerrait davantage de temps de calcul.

De plus, dans l'équation (2.23), la matrice $\boldsymbol{L}_s$ est évaluée par rapport à $\boldsymbol{s}_m(t_k)$ et ${}^c\boldsymbol{Z}_m(t_k)$ qui ne sont que des approximations. Comme une mesure $\boldsymbol{s}(t_k)$ est disponible, une autre possibilité consisterait à remplacer $\boldsymbol{s}_m(t_k)$ par $\boldsymbol{s}(t_k)$ en évaluant $\boldsymbol{L}_s(\boldsymbol{s}(t_k), {}^c\boldsymbol{Z}_m(t_k))$ afin de réduire le niveau d'approximation.

Finalement, en calculant le produit des expressions (2.20)–(2.23) tel que :

$$\boldsymbol{s}_{mc}(t) - \boldsymbol{s}_m(t) \approx \boldsymbol{L}_s\boldsymbol{L}_s^{\dagger}(\boldsymbol{s}(t) - \boldsymbol{s}_m(t)), \tag{2.26}$$

où $\boldsymbol{s}_{mc}$ représente le modèle corrigé des primitives visuelles, ce qui donne :

$$\boldsymbol{s}_{mc}(t) - \boldsymbol{s}_m(t) \approx \boldsymbol{L}_s \boldsymbol{L}_s^\dagger \boldsymbol{s}(t) - \boldsymbol{L}_s \boldsymbol{L}_s^\dagger \boldsymbol{s}_m(t), \tag{2.27}$$

et en exploitant la propriété de la pseudoinverse de Moore-Penrose suivante :

$$\boldsymbol{L}_s^\dagger \boldsymbol{L}_s \boldsymbol{L}_s^\dagger = \boldsymbol{L}_s^\dagger, \tag{2.28}$$

on fait le produit de (2.27) avec $\boldsymbol{L}_s^\dagger$ et on obtient :

$$\boldsymbol{L}_s^\dagger \boldsymbol{s}_{mc}(t) - \boldsymbol{L}_s^\dagger \boldsymbol{s}_m(t) \approx \boldsymbol{L}_s^\dagger \boldsymbol{s}(t) - \boldsymbol{L}_s^\dagger \boldsymbol{s}_m(t). \tag{2.29}$$

Ce qui nous donne finalement :

$$\boldsymbol{L}_s^\dagger \boldsymbol{s}_{mc}(t) \approx \boldsymbol{L}_s^\dagger \boldsymbol{s}(t) \tag{2.30}$$

Par conséquent, la loi de commande classique de l'asservissement visuel (1.16) donnera une vitesse de caméra similaire dans les cas où $\boldsymbol{s}(t)$ ou $\boldsymbol{s}_{mc}(t)$ sont utilisés :

$$\mathbf{v} = -\lambda \boldsymbol{L}_s^\dagger (\boldsymbol{s}(t) - \boldsymbol{s}^*) \approx -\lambda \boldsymbol{L}_s^\dagger (\boldsymbol{s}_{mc}(t) - \boldsymbol{s}^*) \tag{2.31}$$

L'asservissement visuel classique basé sur l'image ne sera donc que peu perturbé par cette méthode de correction de pose. Ceci est d'un intérêt très important pour $t > t_{occ}$ quand les mesures $\boldsymbol{s}(t)$ ne sont plus disponibles.

La dernière méthode, présentée dans la partie suivante, consiste à exploiter les caractéristiques des deux précédentes.

2.3.3 Correction combinée

Cette troisième méthode de correction consiste à combiner la correction (2.24) avec (2.15) afin d'appliquer une correspondance parfaite des modèles $\boldsymbol{s}_m(t_k)$ avec la mesure $\boldsymbol{s}(t_k)$, ce qui permet de corriger les coordonnées 3D de chaque point ${}^cX_{m_i}$, ${}^cY_{m_i}$ en utilisant la correction de ${}^cZ_{m_i}$ obtenue par la correction de pose. Pour cela la correction suivante est appliquée :

$$\begin{cases} {}^c\mathbf{P}_{om}(t_k) & \leftarrow & {}^c\mathbf{P}_{om}(t_k) \oplus \Delta {}^c\mathbf{P}_o(t_k), \\ \boldsymbol{s}_m(t_k) & \leftarrow & \boldsymbol{s}(t_k), \\ {}^cX_{m_i}(t_k) & \leftarrow & x_{m_i}(t_k) {}^cZ_{m_i}(t_k), \\ {}^cY_{m_i}(t_k) & \leftarrow & y_{m_i}(t_k) {}^cZ_{m_i}(t_k), \\ {}^cZ_{m_i}(t_k) & \leftarrow & f_Z({}^c\mathbf{P}_{om}(t_k), {}^o\boldsymbol{X}_i)). \end{cases} \tag{2.32}$$

L'intérêt de cette méthode est d'obtenir une correction du modèle de la pose de l'objet dans le repère de la caméra, tout en s'assurant que $\boldsymbol{s}_m(t_{occ}) = \boldsymbol{s}(t_{occ})$. Il est à noter finalement un dernier avantage important au sujet des méthodes proposées décrites ci-dessus. Celles-ci demandent en effet des temps de calcul très faibles, qui sont de plus

parfaitement négligeable face au temps requis par la plupart des algorithmes de traitement d'images permettant de fournir les mesures des primitives visuelles à chaque nouvelle image capturée par la caméra. L'intégration de ces méthodes est donc très facile à mettre en place dans des applications réelles et n'impacte pas leurs réalisations.

Le but des sections qui suivent est de présenter différents résultats obtenus en simulation et en situation réelle afin de revenir de manière plus tangible sur l'étude des prédictions introduites dans 2.2.1 et des méthodes de corrections présentées dans 2.3. Les résultats de simulation s'attardent sur l'impact des différentes corrections sur la prédiction, et ce pour un système en boucle ouverte. Les résultats expérimentaux quant à eux sont une intégration de ces méthodes dans une loi de commande classique d'asservissement visuel confrontée à des pertes d'informations visuelles.

2.4 Résultats de simulation

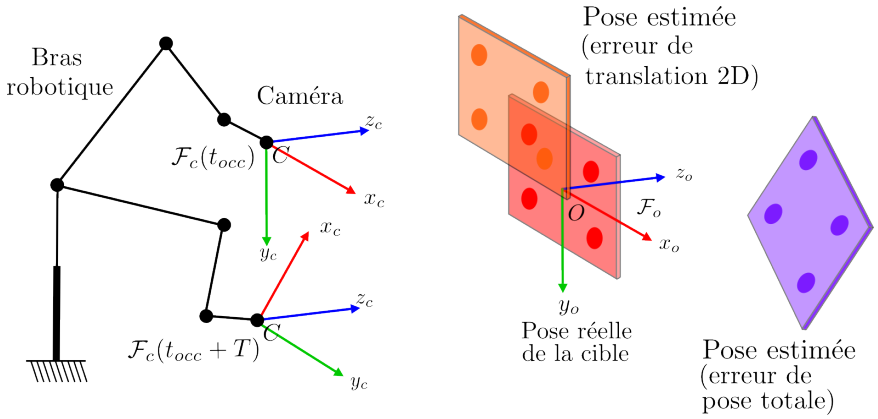


FIGURE 2.8 – Les deux positions de la caméra au début et à la fin de son mouvement, à $t = t_{occ}$ et à $t = t_{occ} + T$ sont représentées. La pose réelle de la cible est représentée en rouge. L'erreur de pose de translation 2D est représentée en orange (Sect. 2.4.1). L'erreur de pose complète est représentée en violet (Sect. 2.4.2).

Afin de visualiser les effets des corrections (2.15), (2.24) ou (2.32), nous reprenons le système introduit dans la section 2.2.1.3, à savoir une caméra monoculaire montée sur un bras robotique se déplaçant avec une vitesse constante sur un temps $T = 4$ s discrétisé en $H = 100$ étapes avec un temps d'échantillonnage de $\tau = 40$ ms, une cible constituée de quatre points coplanaires formant un carré de 20 cm de longueur. La pose de $\mathcal{F}_O$ dans $\mathcal{F}_C$ est initialisée tel que

$$ {}^c\mathbf{P}_o(t_{occ}) = \begin{pmatrix} 0 & 0 & 0.8 & 0 & 0 & 0 \end{pmatrix}^T, \tag{2.33} $$

qui correspond à un simple déplacement de 0.8 m le long de l'axe z_c, et la vitesse de la caméra est fixée telle que :

$$\mathbf{v}_c(t) = \begin{bmatrix} -0.02 & 0 & -0.03 & 0.04 & 0.04 & -0.4 \end{bmatrix}^T = const$$

sur l'intervalle de temps $t \in [t_{occ},\, t_{occ} + T]$. Enfin, on rappelle que la prédiction (2.6)–(2.7) offre les mêmes résultats que ceux obtenus par la prédiction (2.10). C'est pourquoi, on utilise cette dernière dans toutes les simulations qui suivent.

2.4.1 Erreur de translation 2D

Comme montré sur la Figure 2.8 en orange, nous considérons tout d'abord le cas où l'erreur de la pose de l'objet est juste une translation le long des axes x_o et y_o. Le modèle de la pose est ainsi considéré tel que ${}^c\mathbf{P}_{om}(t_{occ}) = \begin{bmatrix} -0.1 & -0.1 & 0.8 & 0 & 0 & 0 \end{bmatrix}^T$ contrairement à la pose réelle $\mathbf{P}(t_{occ})$ définie dans (2.33). Les coordonnées ${}^c\boldsymbol{Z}_m$ ne sont ainsi pas impactées par l'erreur de pose.

La Figure 2.9a montre les résultats de la prédiction (2.10) sur le plan image de la caméra sans effectuer aucune correction.. Les lignes rouges représentent les trajectoires des mesures réelles des primitives visuelles $\boldsymbol{s}(t)$, alors que les lignes bleus représentent les trajectoires des modèles des primitives visuelles $\boldsymbol{s}_m(t)$ après la prédiction. Les deux trajectoires ne correspondent en rien en raison de l'absence de correction des modèles des primitives visuelles ou de la pose de la cible dans le repère de la caméra. En complément, la Figure 2.10a illustre l'erreur entre la prédiction des primitives visuelles et leurs valeurs réelles $e_i(t) = \|\boldsymbol{s}_i(t) - \boldsymbol{s}_{im}(t)\|$.

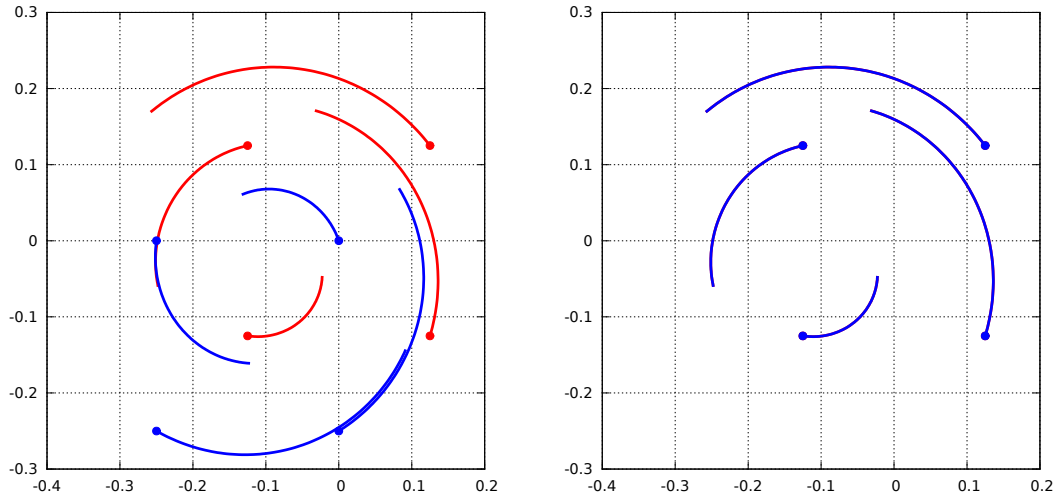
La Figure 2.9b montre les résultats de prédiction après que les trois corrections (2.15), (2.24), et (2.32) aient été utilisées à $t = t_{occ}$. On peut y observer la correspondance parfaite entre les coordonnées réelles $\boldsymbol{s}(t)$ et prédites $\boldsymbol{s}_m(t)$. De façon similaire, la Figure 2.10b représente l'évolution des erreurs des modèles des primitives visuelles qui sont égales à zéro durant toute la prédiction. On peut expliquer ces résultats strictement identiques par l'absence d'erreur initiale sur les coordonnées ${}^c\boldsymbol{Z}_m$.

Nous présentons dans la simulation suivante une erreur de pose qui impacte l'ensemble des coordonnées, dont les ${}^c\boldsymbol{Z}_m$, afin de mettre en valeur l'intérêt de la correction (2.10) sur les autres méthodes.

2.4.2 Erreur de pose complète

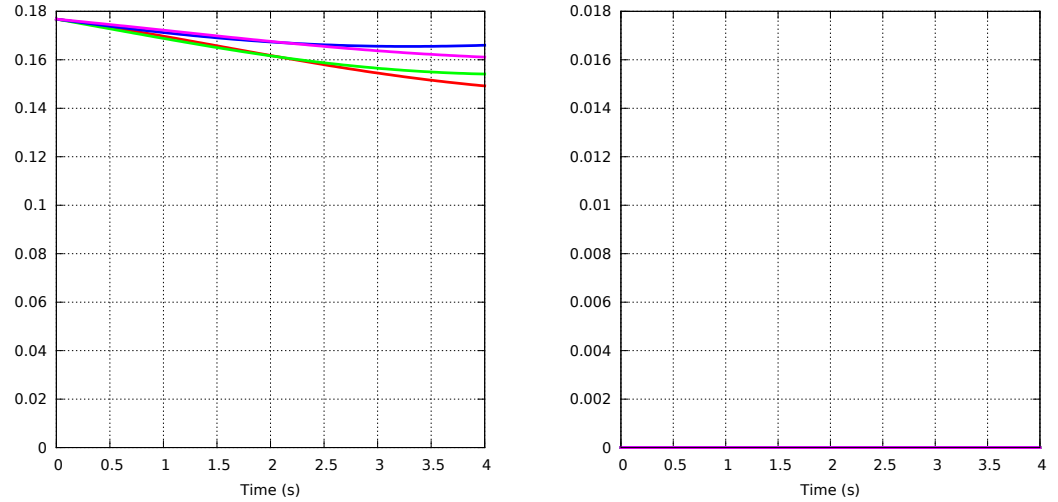
Comme montré sur la Figure 2.8 en bleu, nous considérons maintenant un cas plus complexe impliquant à la fois une erreur de translation et une erreur de rotation dans la pose de l'objet en fixant ${}^c\mathbf{P}_{om}(t_{occ}) = \begin{bmatrix} 0 & 0 & 1.5 & 0.05 & 0 & -0.35 \end{bmatrix}^T$ face à la pose réelle ${}^c\mathbf{P}_o(t_{occ})$ définie dans (2.33)).

En tant que référence, la Figure 2.11a rapporte les résultats de la prédiction (2.10)



(a) (b)

FIGURE 2.9 – Erreur de pose 2D : représentation des primitives visuelles sur le plan image de la caméra. En rouge, les trajectoires des primitives visuelles réelles $\boldsymbol{s}(t)$ et en bleu celles prédites $\boldsymbol{s}_m(t)$. La prédiction de la pose (2.10) est employée dans les deux cas. (a) Aucune correction n'est réalisée. (b) Les corrections (2.15), (2.24), et (2.32) sont utilisées avant la prédiction. On note comment la prédiction dans les trois cas coïncide parfaitement avec le comportement réel $\boldsymbol{s}(t)$.



(a) (b)

FIGURE 2.10 – Erreur de pose 2D : comportement des erreurs de prédiction $e_i(t) = \|\boldsymbol{s}_i(t) - \boldsymbol{s}_{im}(t)\|$, $i = 1 \ldots 4$. (a) Aucune correction n'est réalisée. (b) Les corrections (2.15), (2.24), et (2.32) sont utilisées avant la prédiction.

dans le plan image de la caméra sans qu'aucune correction ne soit apportée, et la Figure 2.12a montre les erreurs des primitives visuelles correspondantes.

Les Figures 2.11b–2.12b montrent les résultats de prédiction après la correction dans le plan image (2.15) : on peut noter la correspondance parfaite entre $\boldsymbol{s}_m(t_{occ})$ et $\boldsymbol{s}(t_{occ})$, mais aussi comment $\boldsymbol{s}_m(t)$ diverge sensiblement de $\boldsymbol{s}(t)$ au fil du temps à cause de la non-correction de la pose ${}^c\mathbf{P}_{om}(t_{occ})$, en particulier à cause de l'erreur initiale induite par une mauvaise valeur de ${}^c\boldsymbol{Z}_m(t_{occ})$.

Le comportement de la prédiction des primitives visuelles est sensiblement amélioré lors de l'utilisation de la correction (2.24) qui tente d'ajuster directement la pose ${}^c\mathbf{P}_{om}(t_{occ})$. Les Figures 2.11c–2.12c en rapportent les résultats : on note comment, dans l'ensemble, les primitves prédites $\boldsymbol{s}_m(t)$ correspondent mieux avec $\boldsymbol{s}(t)$ par rapport au cas précédent, bien que $\boldsymbol{s}_m(t_{occ}) \neq \boldsymbol{s}(t_{occ})$.

Finalement, les Figures 2.11d–2.12d montrent les résultats lorsque la correction (2.32) est utilisée : cela permet d'obtenir de bien meilleures performances, étant donné que maintenant $\boldsymbol{s}_m(t_{occ}) = \boldsymbol{s}(t_{occ})$ en plus de la correction de la pose ${}^c\mathbf{P}_{om}(t_{occ})$. On note qu'il subsiste encore une erreur qui se propage qui est dûe à l'approximation résiduelle de la pose corrigée ${}^c\mathbf{P}_{om}(t_{occ})$ et donc des coordonnées ${}^c\boldsymbol{Z}_m(t_{occ})$. En effet, une seule itération de l'expression (2.23) ayant été effectuée, une approximation subsiste dans le résultat du problème des moindres carrés (2.21).

Cette étude en simulation permet de revenir explicitement sur plusieurs points : pour des erreurs n'impliquant pas de différence de profondeurs dans le modèle de pose de la caméra ${}^c\mathbf{P}_{om}(t_{occ})$ (comme dans la section 2.4.1), les trois corrections (2.15)–(2.24)–(2.32) sont en mesure de faire face de manière équivalente à l'erreur initiale, et permettent donc de calculer la prédiction "parfaite" des primitives $\boldsymbol{s}_m(t)$ dans le temps. Cependant, en présence d'erreurs complètes dans le modèle de la pose (section 2.4.2), la simple correction (2.15) offre une performance très faible, alors que les nouvelles corrections (2.24), et (2.32) donnent de bien meilleurs résultats grâce à leur capacité de prendre en compte la pose de l'objet à l'aide de l'erreur entre les primitives visuelle modélisées et mesurées $\boldsymbol{s}_m(t_{occ}) - \boldsymbol{s}(t_{occ})$. Ceci confirme donc l'analyse proposée dans les sections précédentes et l'importance de développer de meilleures méthodes de correction pour améliorer la prédiction des modèles de primitives visuelles dans le contexte de l'asservissement visuel confronté à des pertes d'informations visuelles.

La section suivante propose d'introduire ces méthodes simulées dans différents cas expérimentaux.

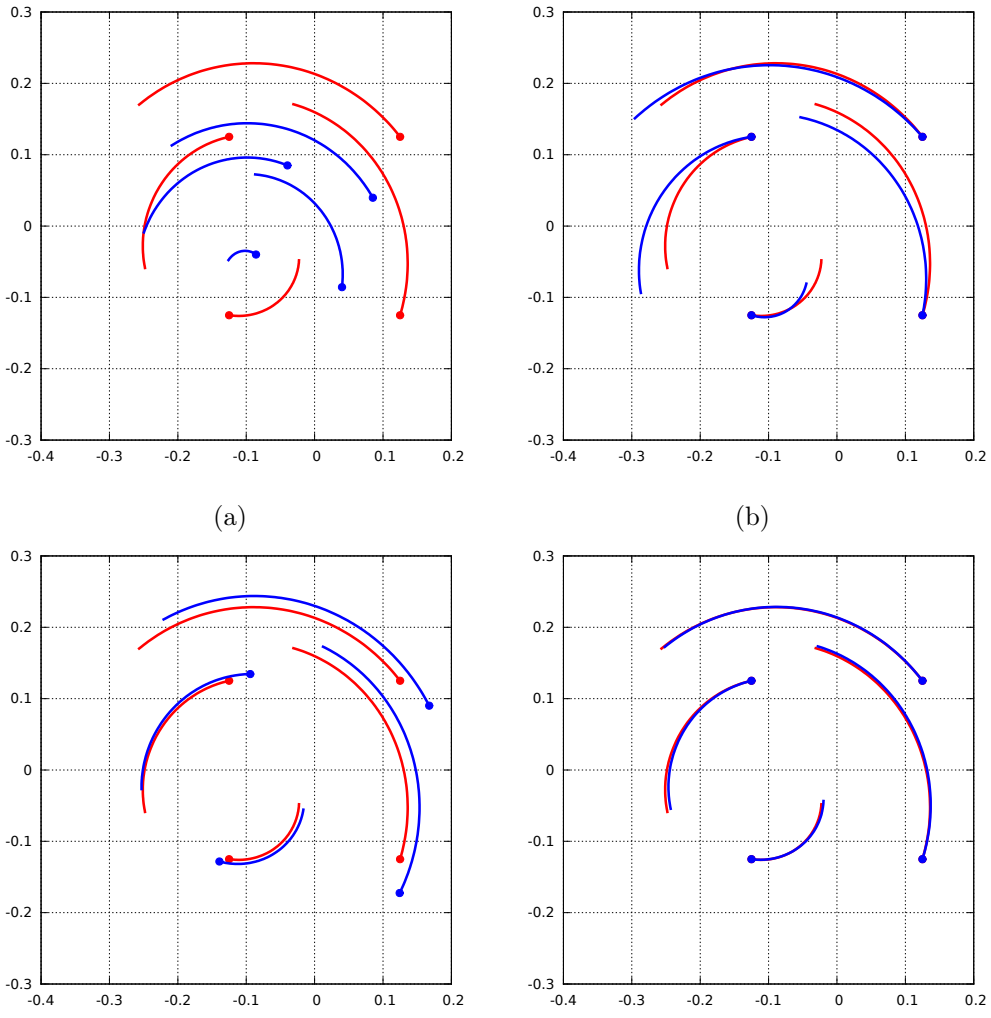


FIGURE 2.11 – Erreur de pose totale : les primitives visuelles sur le plan image de la caméra. En rouge, les trajectoires des primitives visuelles réelles $\boldsymbol{s}(t)$ et en bleu celles des primitives visuelles prédites $\boldsymbol{s}_m(t)$. (a) Aucune correction n'est réalisée. (b) La correction (2.15) est utilisée. (c) La correction (2.24) est utilisée. (d) La correction (2.32) est utilisée, et permet d'obtenir les meilleurs performances.

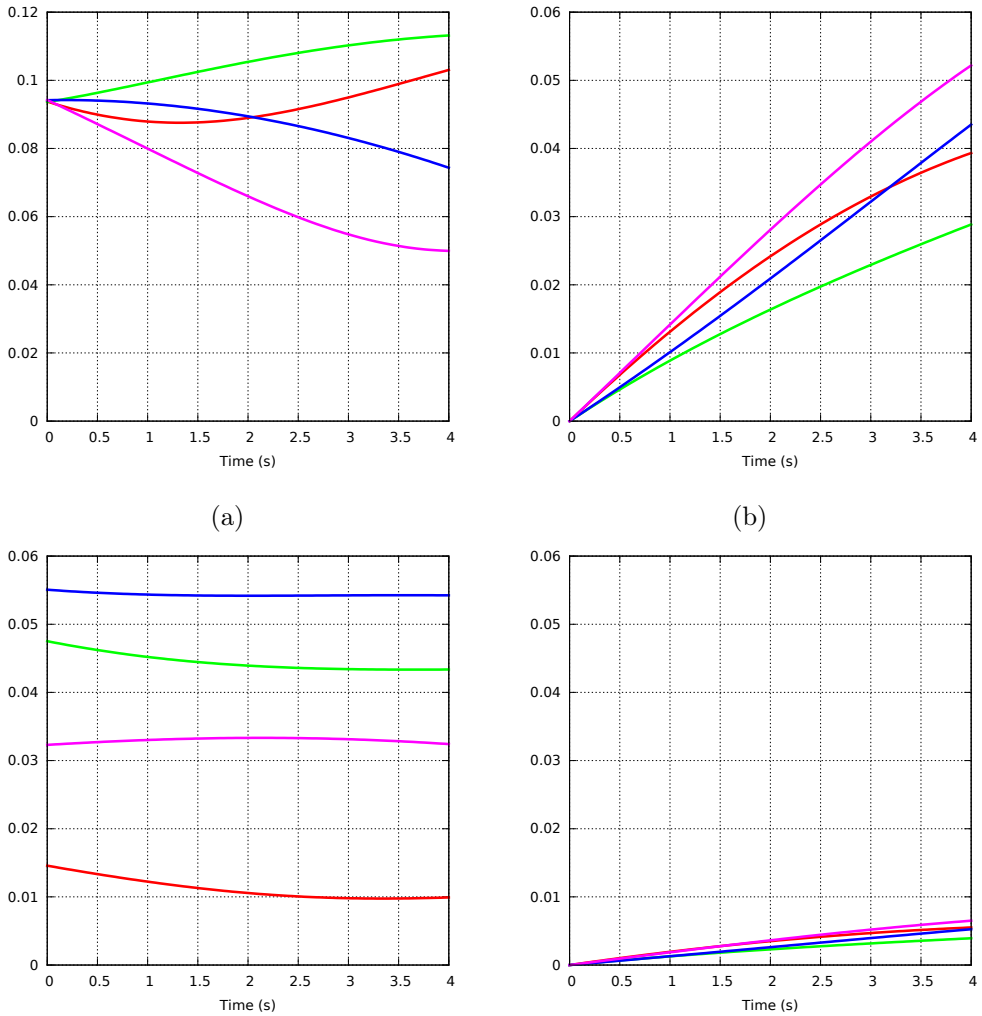


FIGURE 2.12 – Erreur de pose totale : comportement de l'erreur de prédiction $e_i(t) = \|\boldsymbol{s}_i(t) - \boldsymbol{s}_{im}(t)\|$, $i = 1 \ldots 4$. (a) : aucune correction n'est réalisée. (b) : la correction (2.15) est utilisée. (c) : la correction (2.24) est utilisée. (d) : la correction (2.32) est utilisée.

2.5 Résultats expérimentaux

Dans cette section, on souhaite valider expérimentalement les résultats obtenus dans les simulations de la section précédente. Cette partie présente les résultats expérimentaux d'une tâche d'asservissement visuel confrontée à des occultations. Des phases de prédiction remplacent les mesures manquantes. Les modèles des primitives visuelles sont prédits à partir de la méthode (2.10). Les corrections (2.15)–(2.32) sont comparées pour définir laquelle est la plus adéquate pour que les primitives visuelles réelles puissent être retrouvées par le traitement d'image après l'occulation, c'est-à-dire à $t = t_{occ} + T$. On ne traitera pas ici la correction (2.24) en raison du nombre de mesures disponibles avant l'occultation. En effet, les résultats de simulation présentés précédemment sont valables pour une seule opération de correction. En multipliant les mesures, cette correction revient à faire un calcul de pose à partir de plusieurs mesures et donc le modèle de pose initial est considéré comme bon. Les résultats sont alors sensiblement les mêmes qu'avec la correction (2.32).

Deux configurations du même système défini par le bras, la caméra et la cible sont présentées. La première est une configuration eye-in-hand, c'est-à-dire que la caméra est montée sur le bras comme pour les simulations précédentes. La deuxième en revanche est une configuration eye-to-hand, c'est-à-dire que la caméra est fixe et observe le bras et la cible.

2.5.1 Configuration eye-in-hand

Comme dans les parties précédentes, nous considérons dans un premier temps un système composé d'une caméra montée sur un bras robotique qui observe une cible fixe composée de quatre points. Ce système correspond à une configuration eye-in-hand, c'est-à-dire que la vitesse appliquée au bras est la même que celle appliquée à la caméra dans le repère de la caméra. On rappelle alors la loi de commande mise en œuvre :

$$\mathbf{v}_c = -\lambda \widehat{\boldsymbol{L}_s}^{\dagger} (\boldsymbol{s} - \boldsymbol{s}^*), \tag{2.34}$$

où $\widehat{\boldsymbol{L}_s}$ est une estimation de la matrice d'interaction $\boldsymbol{L}_s$ associée aux points qui a été définie dans (2.4) et (2.5). Le but de cette commande est d'entrainer l'erreur entre les primitives visuelles courantes $\boldsymbol{s}$ et leurs coordonnées désirées $\boldsymbol{s}^*$ vers zéro.

2.5.1.1 Paramètres

Les deux premières expérimentations suivantes ont été réalisées avec une caméra perspective monoculaire montée sur un bras robotique à 6 degrés de liberté (ddl).

La cible est constituée de quatre points coplanaires formant un carré d'environ 10 cm

de côté dont les coordonnées dans le repère $\mathcal{F}_o$ sont les suivantes :

$$\begin{cases} {}^o\boldsymbol{X}_0 &= \begin{bmatrix} -0.05 & -0.05 & 0 & 1 \end{bmatrix}^T \\ {}^o\boldsymbol{X}_1 &= \begin{bmatrix} 0.05 & -0.05 & 0 & 1 \end{bmatrix}^T \\ {}^o\boldsymbol{X}_2 &= \begin{bmatrix} 0.05 & 0.05 & 0 & 1 \end{bmatrix}^T \\ {}^o\boldsymbol{X}_3 &= \begin{bmatrix} -0.05 & 0.05 & 0 & 1 \end{bmatrix}^T \end{cases} .$$

Le logiciel ViSP [MSC05] a été utilisé pour extraire et suivre les coordonnées des centres des points dans l'image qui composent l'ensemble des primites visuelles $\boldsymbol{s}$.

La pose du repère de l'objet $\mathcal{F}_o$ dans $\mathcal{F}_c$ au début de la tâche est la suivante :

$${}^c\mathbf{P}_o(t_0) = \begin{bmatrix} -0.072 & -0.035 & 1.21 & -0.35 & 0.34 & 0.050 \end{bmatrix}^T .$$

Celle-ci est calculée à l'aide d'un asservissement visuel virtuel [MC02a]. Le modèle initial correspondant à cette pose est initialisé de manière incorrecte pour mettre en évidence l'intérêt de la correction (2.32), tel que

$${}^c\mathbf{P}_{om}(t_0) = \begin{bmatrix} 0 & 0 & 2 & 0 & 0 & 0.79 \end{bmatrix}^T .$$

Ce modèle est alors utilisé pour calculer les valeurs initiales des modèles des primitives visuelles $\boldsymbol{s}_{m_i}(t_0) = f_s({}^c\mathbf{P}_{om}(t_0), {}^o\boldsymbol{X}_i)$ et les modèles des profondeurs ${}^cZ_{m_i}(t_0) = f_Z({}^c\mathbf{P}_{om}(t_0), {}^o\boldsymbol{X}_i)$.

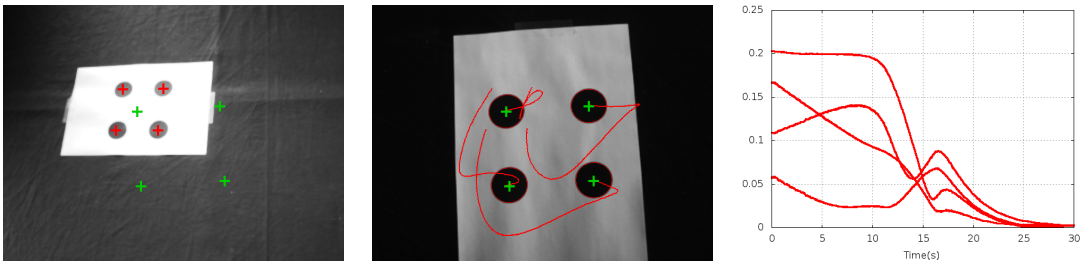
Nous rappelons que nous désirons illustrer l'intérêt des corrections des modèles $\boldsymbol{s}_m$ et ${}^c\boldsymbol{Z}_m$ et leurs prédictions au sein de la loi de commande (2.34). Or, celle-ci dépend de la matrice d'interaction $\boldsymbol{L}_s$, qui comme présenté dans la section 1.3.1.4 peut être approximée selon plusieurs types d'élaboration. Nous choisissons ici la matrice d'interaction calculée en fonction des primitives visuelles et des profondeurs désirées $\widehat{\boldsymbol{L}_s} = \boldsymbol{L}_s(\boldsymbol{s}^*, {}^c\boldsymbol{Z}^*)$. Cette matrice reste constante durant tout le déplacement du robot et permet de démontrer que même si la profondeur ${}^c\boldsymbol{Z}_m(t)$ n'est pas impliquée dans le système de commande, son utilisation dans l'étape de prédiction est cruciale. Une autre raison pour laquelle nous utilisons cette approximation est liée au fait que les trajectoires 2D des primitives visuelles pour atteindre leurs coordonnées désirées ne décrivent pas des lignes droites mais des courbes susceptibles de sortir du champ de vue de la caméra. En effet, on a vu dans l'exemple présenté dans la section 1.3.1.4 que dans le cas où une rotation autour de l'axe optique de la caméra est considérée, un mouvement de translation vers l'avant s'effectue de la part de la caméra, et génère un décalage des primitives visuelles vers les limites du champ de vue de la caméra. Ce comportement nous permet alors de provoquer volontairement des sorties des primitives visuelles en dehors du champ de vue de la caméra et ainsi démontrer l'efficacité des méthodes de prédiction qui se substituent aux mesures manquantes.

Les résultats présentées ci-dessous peuvent être visualisés en vidéo en suivant l'URL suivant : `https://youtu.be/6OrrGw0VpeM`.

2.5.1.2 Trajectoire de référence

La Figure 2.13a montre l'image initiale obtenue par la caméra avant que la loi de commande (2.34) ne soit appliquée. Les croix vertes représentent les positions désirées des primitives visuelles $\boldsymbol{s}^*$ et les croix rouges leurs positions initiales mesurées $\boldsymbol{s}(t_0)$.

Cette première expérimentation peut être visualisée dans la première partie de la vidéo `https://youtu.be/6OrrGw0VpeM`. Elle ne comprend pas d'occultation ni de perte de suivi et est uniquement destinée à fournir une référence pour comparer les trajectoires obtenues par la suite. Les courbes rouges dans la Figure 2.13c représentent les chemins empruntés par chaque point dans l'image pendant le mouvement de la caméra, tandis que la Figure 2.13c montre l'évolution de l'erreur $e_i(t) = \|\boldsymbol{s}_i(t) - \boldsymbol{s}_i^*(t)\|$ entre les positions courantes et désirées, qui converge vers zéro comme il est attendu avec l'utilisation de la loi de commande (2.34).



(a) (b) (c)

FIGURE 2.13 – Trajectoire de référence : (a) État initial du système. Les positions des primitives visuelles désirées $\boldsymbol{s}^*$ sur le plan image de la caméra sont en vert et leurs positions mesurées $\boldsymbol{s}(t_0)$ sont en rouge. (b) État final du système. Les trajectoires suivies par chaque primitive mesurée sont en rouge, alors que leurs positions désirées sont en vert. (c) Évolution du comportement de l'erreur $e_i(t) = ||\boldsymbol{s}_i(t) - \boldsymbol{s}_i^*(t)||$ de chaque primitive visuelle pendant l'asservissement.

2.5.1.3 Champ de vue limité

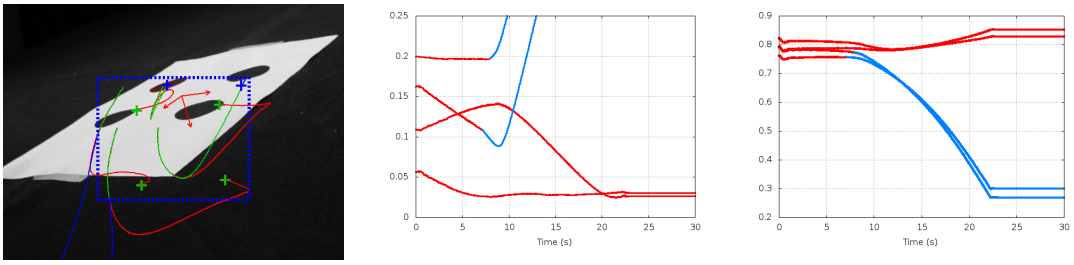
En reprenant la configuration initiale présentée sur la Figure 2.13a, nous considérons à présent le cas où les primitives visuelles quittent le champ de vue de la caméra, qui est artificiellement limité sur une plus petite partie que la vue réelle. Ce nouveau champ de vue est représenté dans les Figures suivantes par un quadrillage bleu.

Les Figures 2.14a et 2.15a montrent les images capturées par la caméra à la fin de l'asservissement, en utilisant respectivement la correction (2.15) et la correction (2.32). Ces deux expérimentations peuvent être visualisées sur la seconde et la troisième partie de la vidéo `https://youtu.be/6OrrGw0VpeM`. Les courbes rouges représentent les trajectoires de référence des primitives visuelles dans l'image issues de l'expérimentation précédente. Les courbes vertes représentent les chemins empruntés par les primitives

lorsque les mesures sont disponibles, alors que les bleues représentent les chemins prédits à l'aide de (2.10) lorsque les mesures ne sont plus disponibles.

Les Figures 2.14b et 2.15b montrent les erreurs entre les positions courantes et désirées des points $e_i(t)$, et les Figures 2.14c et 2.15c l'erreur d'estimation de chaque profondeur $Z_i(t)$ qui a été mesurée pour les biens de l'expérimentation à l'aide d'un calcul de pose introduit dans la section 1.3.1.3. Dans chaque graphique, les parties bleues des courbes correspondent à des phases de prédiction.

Dans ces deux expérimentations, toutes les primitives visuelles sont visibles par la caméra à $t = 0$ et sont donc utilisées dans la loi de commande (2.34). On remarque sur la Figure 2.14c que le modèle incorrecte de la pose conduit à des erreurs importantes entre les profondeurs réelles Z_i et leurs modèles Z_{m_i}, mais également sur la Figure 2.15c que l'emploi de (2.32) permet de les corriger dès la première mesure.

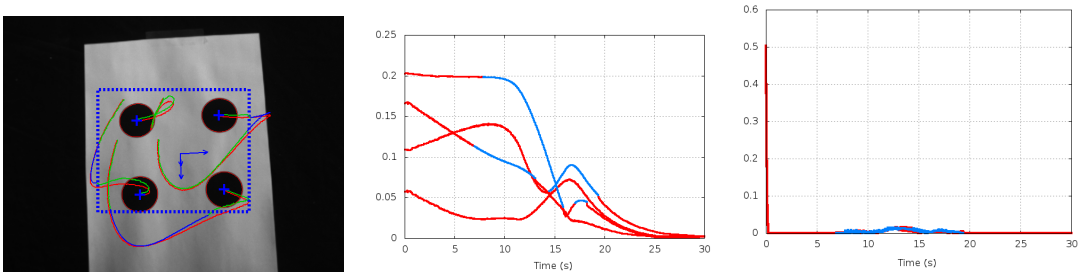


(a) (b) (c)

FIGURE 2.14 – Champ de vue limité avec la correction (2.15). À cause des performances limitées de la correction (2.15), les trajectoires prédites divergent fortement de celles souhaitées (représentées en rouge), causant un échec total de l'asservissement : (a) État final. Les courbes vertes représentent le chemin des primitives visuelles mesurées alors que les courbes bleues représentent les trajectoires des primitives visuelles prédites, lorsque les mesures ne sont pas disponibles. (b) Évolution de l'erreur $e_i(t) = ||\boldsymbol{s}_i(t) - \boldsymbol{s}_i^*||$ en rouge et de $e_i(t) = ||\boldsymbol{s}_m(t) - \boldsymbol{s}_i^*||$ en bleu. (c) Évolution de $\|Z_{m_i}(t) - Z_i(t)\|$.

À $t \approx 7$s, deux points sortent du quadrillage bleu et sont considérés comme perdus. La loi de commande exploite alors les mesures des points visibles et les prédictions des points invisibles via (2.10) pour calculer la vitesse à appliquer au robot. Malgrés le fait que deux points soient devenus invisibles, il est toujours possible de corriger le modèle de la pose avec (2.32) et, par conséquent, les profondeurs $Z_i(t)$. Comme attendu, le fait de ne pas corriger la pose ${}^c\mathbf{P}_{om}(t)$ dans le cas de la correction image (2.15) se traduit par une dérive importante de la prédiction dans la Figure 2.14a jusqu'à ce que le robot atteigne une position limite à $t \approx 22$s, tandis que la correction (2.32) conduit à une prédiction presque parfaite du comportement comme on peut l'observer sur la Figure 2.15a, malgré la perte d'un troisième point à $t \approx 17$s. Par conséquent, la caméra converge vers une pose incorrecte dans le premier cas comme on peut l'observer au travers de l'évolution de l'erreur e_i sur la Figure 2.14b, tandis qu'elle converge vers la pose désirée dans le

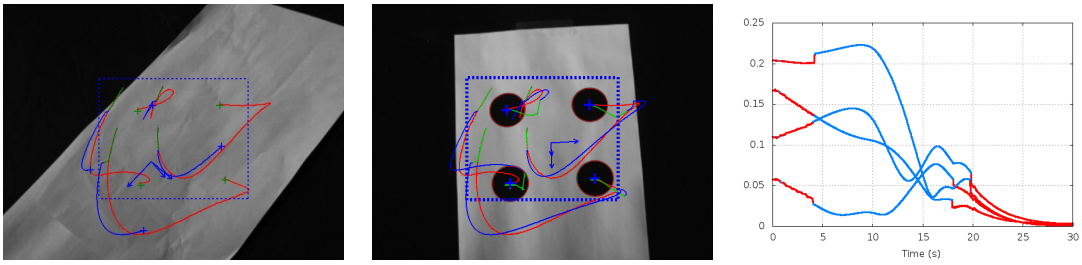
second cas (Figure 2.15b). On note comment la prédiction correcte des coordonnées des points dans l'image permet au traitement d'image de suivre à nouveau les primitives visuelles lorsqu'elles se retrouvent à nouveau dans le champ de vue de la caméra.



(a) (b) (c)

FIGURE 2.15 – Champ de vue limité avec la correction (2.32) : (a) État final. (b) Évolution de l'erreur $e_i(t) = ||\boldsymbol{s}_i(t) - \boldsymbol{s}_i^*||$ en rouge et de $e_i(t) = ||\boldsymbol{s}_m(t) - \boldsymbol{s}_i^*||$ en bleu. (c) Évolution de $\|Z_{m_i}(t) - Z_i(t)\|$. On note comment, comparé à la Figure 2.14 où l'asservissement vers la pose désirée n'a pas été correctement réalisée, les performances supérieures de la correction (2.32) permettent une plus grande précision de la prédiction $\boldsymbol{s}_m(t)$ lorsque les mesures ne sont plus disponibles.

En complément, la Figure 2.16 propose le cas d'une occultation plus longue que dans le cas précédent. Cette expérimentation peut être visualisée sur la quatrième partie de la vidéo `https://youtu.be/6OrrGw0VpeM`. On observe sur la Figure 2.16a qu'en plus du champ de vue artificiellement plus petit que la véritable vision de la caméra, une simple feuille blanche est placée entre la caméra et l'ensemble des points observés durant la tâche d'asservissement visuel, ce qui prolonge la phase d'occultation. Une nouvelle fois, la correction (2.32) est utilisée lorsque les points sont observables et la prédiction (2.10) est utilisée lorsqu'ils ne le sont plus. Finalement, bien que la phase de prédiction soit bien plus longue que pour les autres cas et concerne l'ensemble des primitives visuelles, le traitement d'image est toujours en mesure de retrouver les primitives visuelles dans l'image lorsque celles-ci sont à nouveau observables. La caméra converge ainsi vers sa position désirée comme le montrent la Figure 2.16b et la Figure 2.16c qui représente l'évolution des erreurs e_i des primitives visuelles. On peut noter sur ces graphiques de légers décalages qui apparaissent lorsque les points sont à nouveau visibles à $t \approx 20$s. Ils témoignent à la fois de la présence de bruits de commande qui ont pu intervenir lors du déplacement du robot, mais également d'erreurs d'approximations qui ont pu être introduites lors de l'élaboration du modèle 3D de l'objet. Ces erreurs ont ainsi généré de légères imprécisions dans la prédiction des modèles des primites visuelles, mais n'ont pas contrarié la réalisation de la tâche.



(a) (b) (c)

FIGURE 2.16 – Champ de vue limité et objet parasite avec la correction (2.32) : (a) Apparition de l'occultation. (b) État final. (c) Évolution de l'erreur $e_i(t) = ||\boldsymbol{s}_i(t) - \boldsymbol{s}_i^*||$ en rouge et de $e_i(t) = ||\boldsymbol{s}_m(t) - \boldsymbol{s}_i^*||$ en bleu. Comme dans les résultats présentés dans la Figure 2.15, la correction (2.32) permet une plus grande précision de la prédiction $\boldsymbol{s}_m(t)$ lorsque les mesures ne sont plus disponibles.

2.5.2 Configuration eye-to-hand

Afin d'illustrer la façon dont la correction proposée (2.32) peut être appliquée à différents scenarios, nous considérons à présent un système eye-to-hand dans lequel une caméra statique observe une cible fixée à l'effecteur au bout du bras robotique, présenté dans [HDE98]. La loi de commande résultante est légèrement différente de (2.34). On rappelle que dans cette configuration :

$$\mathbf{v}_e = \lambda {}^e\boldsymbol{V}_c \widehat{\boldsymbol{L}_s}^{\dagger} (\boldsymbol{s} - \boldsymbol{s}^*), \tag{2.35}$$

où $\mathbf{v}_e$ est la vitesse de l'effecteur du robot exprimée dans son repère, et ${}^e\boldsymbol{V}_c$ est la matrice de changement de repère des vitesses définie par :

$${}^e\boldsymbol{V}_c = \begin{bmatrix} {}^e\mathbf{R}_c & [{}^e\mathbf{t}_c]_\times {}^e\mathbf{R}_c \\ \mathbf{0} & {}^e\mathbf{R}_c \end{bmatrix}.$$

On notera que la vitesse mesurée du robot doit être exprimée dans le repère de la caméra afin d'utiliser les prédictions (2.6)–(2.7) et (2.10). Ceci est simplement obtenu par :

$$\mathbf{v}_c = {}^c\boldsymbol{V}_e \mathbf{v}_e. \tag{2.36}$$

La difficulté principale de la configuration eye-to-hand est la connaissance inexacte de la pose de l'effecteur dans le repère de la caméra, qui est introduit dans ${}^c\boldsymbol{V}_e$ dans (2.36). En effet, même si le modèle de la pose ${}^c\mathbf{P}_{om}$ est parfaitement corrigé, la pose objet/effecteur doit être parfaitement calibrée pour éviter les dérives entre les mouvements réels et les mouvements prédits.

2.5.2.1 Paramètres

Cette expérimentation a été réalisée avec une caméra monoculaire qui observe une cible embarquée sur l'effecteur d'un bras robotique à 6 ddl qui a comme tâche de saisir un

cube de 4 cm de côté à l'aide d'une pince. La pose du cube dans le repère de la caméra est calculée au début de l'expérimentation et est utilisée pour générer les positions désirées des primitives visuelles de la cible dans le plan image de la caméra.

La cible est composée de 4 points dont les coordonnées homogènes dans le repère $\mathcal{F}_o$ sont :

$$\begin{cases} {}^o\boldsymbol{X}_0 &= \begin{bmatrix} 0 & 0.023 & -0.134 & 1 \end{bmatrix}^T \\ {}^o\boldsymbol{X}_1 &= \begin{bmatrix} 0 & 0.023 & -0.247 & 1 \end{bmatrix}^T \\ {}^o\boldsymbol{X}_2 &= \begin{bmatrix} -0.057 & 0 & -0.132 & 1 \end{bmatrix}^T \\ {}^o\boldsymbol{X}_3 &= \begin{bmatrix} -0.057 & -0.113 & -0.13 & 1 \end{bmatrix}^T \end{cases}.$$

La pose réelle de la cible dans le repère de la caméra au début de la tâche est :

$${}^c\mathbf{P}_o(t_0) = \begin{bmatrix} 0.307 & -0.522 & 0.753 & -1.155 & -0.776 & -0.977 \end{bmatrix}^T$$

alors que le modèle de la pose initiale est une nouvelle fois volontairement mauvaise :

$${}^c\mathbf{P}_{om}(t_0) = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}^T.$$

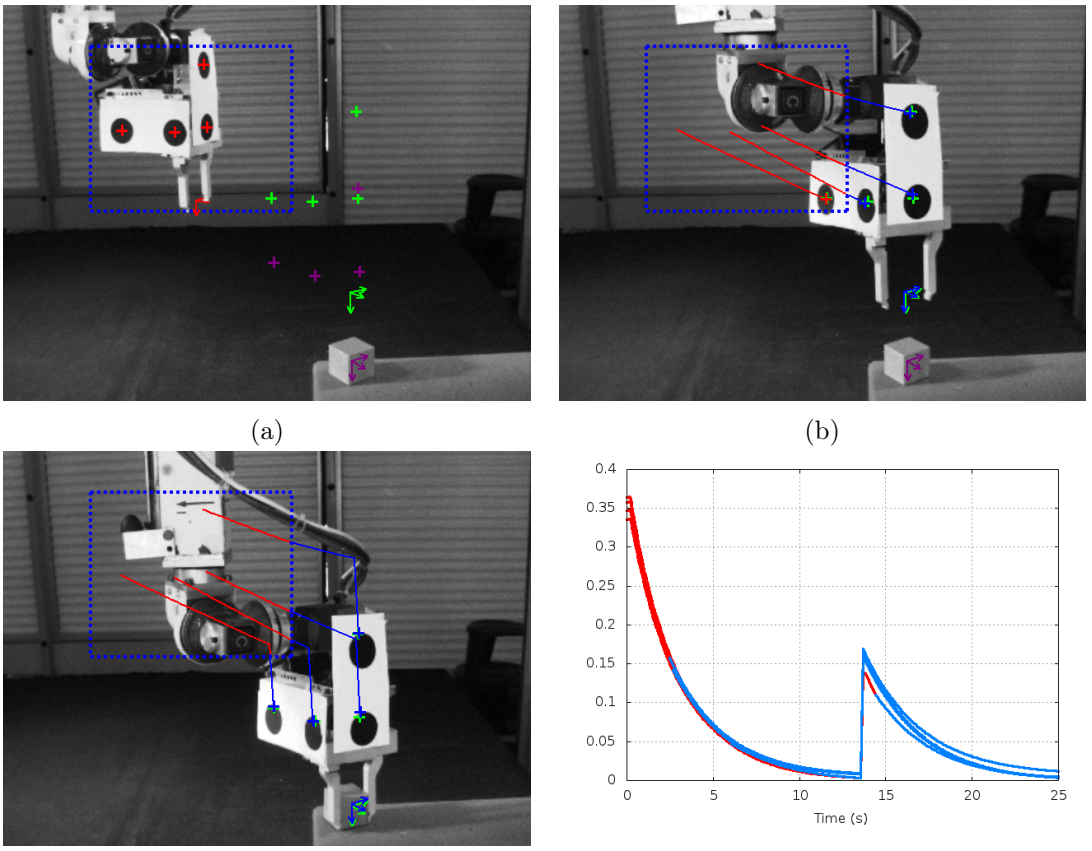
L'asservissement visuel doit permettre de réaliser 2 tâches disctinctes successives en utilisant la loi de commande (2.35). La première consiste à bouger la pince de l'effecteur au dessus du cube pour éviter toute collision, et la seconde consiste à positionner la pince autour du cube pour le saisir. Enfin, et comme pour les expérimentations précédentes, nous avons choisi $\widehat{\boldsymbol{L}_s} = \boldsymbol{L}_s(\boldsymbol{s}^*, \boldsymbol{Z}^*)$ appliquée à la loi de commande (2.35) soit la valeur de la matrice d'interaction calculée en fonction de la pose désirée.

Cette expérimentation peut être visualisée sur la cinquième et dernière partie de la vidéo `https://youtu.be/6OrrGw0VpeM`.

2.5.2.2 Champ de vue limité

Pour cette expérimentation, le champ de vue de la caméra est artificiellement limité sur une plus petite section de l'image fournie par la caméra, ce qui cause des pertes de mesures remplacées par les prédictions (2.6)–(2.7).

La Figure 2.17a montre l'image capturée par la caméra à $t = t_0$, la Figure 2.17b l'image capturée lorsque la première tâche est terminée, et 2.17c l'image capturée à la fin de l'expérimentation. La Figure 2.17d montre les erreurs correspondantes à chaque point $e_i(t)$. Cette expérimentation démontre l'efficacité de la correction (2.32) pour un système avec une configuration eye-to-hand. En effet, les deux tâches successives pour saisir le cube sont réalisées même si les primitives visuelles quittent le champ de vue de la caméra durant la première tâche. On peut noter une légère différence entre les positions des primitives visuelles réelles et prédites à la fin de l'expérimentation, qui est due à l'estimation de la pose effecteur/cible qui n'est pas parfaite.



(a) (b)

(c) (d)

FIGURE 2.17 – Champ de vue limité : Utilisation de la correction (2.32) appliquée dans une configuration eye-to-hand. Les deux tâches d'asservissement sont correctement réalisées. Ceci confirme l'effacité de la correction (2.32) qui permet de traiter rapidement n'importe quelle erreur initiale dans le modèle de la pose caméra/objet en permettant de prédire avec précision $\boldsymbol{s}_m(t)$ pendant la phase de sortie des primitives visuelles en dehors du champ de vue de la caméra. (a) État initial. Les primitives désirées sont en vert pour la première tâche et en violet pour la seconde. (b) État intérmédiaire à la fin de la première tâche. (c) État final. (d) Évolution de l'erreur $e_i(t) = ||\boldsymbol{s}_i(t) - \boldsymbol{s}_i^*||$ en rouge et de $e_i(t) = ||\boldsymbol{s}_m(t) - \boldsymbol{s}_i^*||$ en bleu.

2.6 Contribution

Les travaux qui ont été présentés dans ce chapitre ont fait l'objet de deux articles présentés respectivement lors des conférence International Conference on Intelligent Robots and Systems (IROS) de 2014 à Chicago [CDW+14] et International Conference on Robotics and Automation (ICRA) de 2015 à Seattle [CWRGC15].

2.7 Conclusion

Nous avons exposé dans ce chapitre des notions très importantes pour faire face à des pertes d'informations visuelles dans le cadre d'une tâche d'asservissement visuel.

Dans un premier temps, deux méthodes de prédiction ont été définies et comparées pour permettre à une commande de s'affranchir de nouvelles mesures en les remplaçant par des modèles équivalents.

Dans un second temps, nous avons présenté une nouvelle méthode pour corriger le modèle de la pose entre un objet et une caméra. Cela permet de prédire de manière plus efficace les coordonnées des primitives visuelles dans l'image. Dans le cadre d'un asservissement visuel classique, cette méthode se révèle très intéressante et ouvre donc la voie à d'autres types de commandes faisant face à des cas de pertes de primitives visuelles.

Le prochain chapitre propose d'utiliser la commande prédictive et l'asservissement visuel en collaboration et d'exploiter l'idée selon laquelle la prédiction est en mesure de remplacer les mesures efficacement, et qu'il est même possible de provoquer volontairement des pertes d'informations visuelles pour réaliser plusieurs tâches d'asservissement visuel successives.

Chapitre 3

Commande prédictive et tâches d'asservissement visuel successives

Ce chapitre présente les travaux effectués afin de combiner la commande prédictive introduite dans la section 1.3.2 avec l'asservissement visuel introduit dans la section 1.3.1. Le but de la méthode présentée est de réaliser plusieurs tâches d'asservissement visuel successives à l'aide d'un robot embarquant une caméra.

Comme il a été abordé dans le chapitre précédent, en utilisant des méthodes pour la correction et la prédiction des modèles des primitives visuelles dans l'image, des pertes d'informations visuelles peuvent être tolérées par la caméra. En effet, on a vu que ces pertes peuvent survenir durant le mouvement de la caméra ou de l'objet et que la prédiction est capable de remplacer les véritables coordonnées des primitives visuelles dans l'image, qui peuvent être récupérées par le traitement d'image lorsqu'elles se retrouvent à nouveau sur le plan image de la caméra. Il est donc possible de réaliser plusieurs tâches disctinctes d'asservissement visuel par plusieurs robots suivant la configuration eye-to-hand tel qu'il a été décrit dans la section 2.5.2, et ce à l'aide d'une seule caméra sans avoir à la contraindre de garder en permanence toutes les primitives visuelles dans son champ de vue limité pendant toute la durée des tâches d'asservissement.

Cependant, même si une excellente correction et/ou prédiction est appliquée aux modèles des primitives visuelles, des bruits provoquant des dérives entre les modèles et les véritables coordonnées peuvent intervenir. Ces dérives sont évidemment destinées à s'accroître dans le temps si aucune correction n'est apportée par le biais d'une nouvelle mesure provenant de la caméra. Il est alors indispensable de forcer la caméra à mesurer régulièrement les coordonnées des primitives visuelles afin de corriger les dérives de leurs modèles. Pour cela, nous décrivons dans ce chapitre une méthode de perception active permettant de commander la caméra pour qu'elle retrouve les coordonnées des primitives visuelles dans son champ de vue lorsque cela s'avère nécessaire. Nous exprimons dans un premier temps les dérives entre les coordonnées réelles et modélisées par des incertitudes que nous propageons dans le temps. Puis nous fixons des seuils à ne pas

dépasser sur ces incertitudes. Enfin, nous définissons des contraintes à satisfaire sur les coordonnées des primitives visuelles dans l'image, qui dépendent des incertitudes et de leurs seuils à ne pas dépasser, en utilisant un schéma de commande prédictive adéquat.

Afin d'illustrer la méthode proposée, nous considérons un nombre $\mathcal{N}_r$ de robots mobiles au sol qui doivent suivre une trajectoire désirée chacun. Une caméra monoculaire perspective est embarquée sur un drone dont la hauteur par rapport au sol est constante. Enfin, un amer dont la position est connue dans le repère monde est fixé au sol.

Les modèles des positions des robots et de la caméra sont calculés en fonction de leurs vitesses, mais chaque position réelle dérive de son modèle au cours du temps. Ces dérives sont liées à des bruits qui viennent s'ajouter à la commande, et représentent le fait que les robots et le drone/caméra ne réagissent pas parfaitement au schéma de commande qui leur est appliqué. Ces bruits sont présents en raison de l'imperfection de la modélisation des dynamiques des robots et de la caméra, ou de retards entre la commande appliquée et le déplacement réalisé.

Dans le système que nous venons de décrire, la seule façon de remédier aux dérives entre les positions réelles et modélisées est de corriger ces dernières grâce aux mesures de la caméra utilisée pour observer les robots (pour corriger les modèles des positions des robots) et pour observer l'amer (pour corriger le modèle de la position du drone/caméra). Cependant de manière générale, un seul robot ou l'amer peut se trouver dans le champ de vue de la caméra à la fois. Pour déterminer quel modèle doit être corrigé, l'incertitude de chaque position est propagée afin de quantifier la valeur des dérives. La commande de la caméra doit être capable de gérer quel modèle doit être corrigé en fonction de ces incertitudes.

3.1 Une caméra embarquée et des robots mobiles

Deux types de représentation sont utilisés dans ce système. Les coordonnées des robots, de l'amer et du drone/caméra dans le repère monde pour définir la position de tous les éléments les uns par rapport aux autres, et les coordonnées de l'amer et des robots dans le repère de la caméra pour définir la position de chaque élément dans ou en-dehors du champ de vue de la caméra.
Nous considérons que la caméra et les robots se déplacent sur 2 degrés de liberté chacun, c'est-à-dire que leurs coordonnées cartésiennes sont définies sur un plan :

$$\left\{ \begin{array}{rcll} \boldsymbol{c} & = & \big(c_x\,,\,c_y\big) & \\ \boldsymbol{r}_i & = & \big(r_{ix}\,,\,r_{iy}\big) & \forall i = 1, ..., \mathcal{N}_r \\ \boldsymbol{m} & = & \big(m_x\,,\,m_y\big) & \end{array} \right. \tag{3.1}$$

où $\boldsymbol{c}$, $\boldsymbol{r}_i$ et $\boldsymbol{m}$ sont respectivement les coordonnées de la caméra, les coordonnées du $i$ème robot et les coordonnées de l'amer dans le repère monde. Les coordonnées de la caméra et des robots sont destinées à évoluer dans le temps alors que celles de l'amer

sont fixées et connues. La Figure 3.1 représente la position de la caméra en orange, les positions de deux robots en rouge et la position de l'amer en noir.

On considère $\boldsymbol{c}_m$ et $\boldsymbol{r}_{im}$ les modèles des coordonnées de la caméra et du $i$ème robot, représentées en bleu sur la Figure 3.1. Ces modèles sont introduits pour deux raisons :
(i) La première est liée au développement du schéma de commande prédictive qui sera exposé dans la section 3.5. Dans ce schéma, les positions futures des robots et de la caméra sont déterminées en utilisant le modèle de leurs dynamiques.
(ii) La seconde est liée au fait que, quand les robots et l'amer sont en dehors du champ de vue de la caméra, aucune mesure n'est disponible pour déterminer précisément la position des robots et de la caméra. Utiliser le modèle permet de garder une information approximée de la position en permanence malgré le manque de mesures. Évidemment, lorsqu'une mesure est disponible, le modèle correspondant peut être corrigé en conséquence, comme il le sera présenté dans la section 3.4.

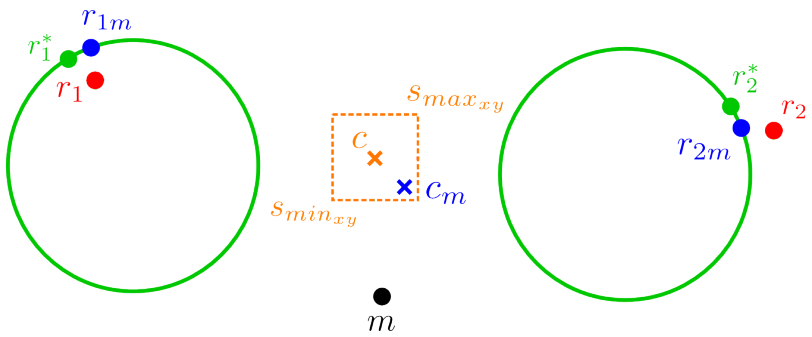


FIGURE 3.1 – Caméra avec un champ de vue limité qui assiste le suivi de trajectoire de deux robots sur le sol (vue de dessus).

On peut exprimer à présent les coordonnées de l'amer dans le repère de la caméra telles que :

$$\boldsymbol{s}_a = \boldsymbol{m} - \boldsymbol{c}, \tag{3.2}$$

et les coordonnées des robots dans le repère de la caméra telles que :

$$\boldsymbol{s}_i = \boldsymbol{r}_i - \boldsymbol{c}. \tag{3.3}$$

Ces coordonnées représentent les primitives visuelles associées à chaque élément. Comme il a été introduit au début de ce chapitre, les commandes appliquées aux robots et à la caméra sont sujettes à des bruits additifs dont les valeurs exactes ne peuvent être prédites. Il est donc nécessaire dans la partie suivante de distinguer la dynamique réelle des robots et de la caméra de leur dynamique modélisée.

3.2 Dynamiques

On décrit dans cette partie l'évolution des positions réelles et modélisées des robots et de la caméra dans le repère monde en fonction des vitesses qui leurs sont appliquées.

Les dynamiques réelles comportent des bruits additifs qui ne sont pas présents dans les dynamiques des modèles.

3.2.1 Dynamiques bruitées

Les dynamiques réelles des robots et de la caméra ne peuvent pas être connues avec exactitude. Il est toutefois possible de les exprimer en introduisant des bruits de commande tel que :

$$\begin{cases} \dot{\boldsymbol{c}} &= \mathbf{v}_c + n_c \\ \dot{\boldsymbol{r}}_i &= \mathbf{v}_i + n_i \end{cases} ; \tag{3.4}$$

où :

- les vitesses appliquées à la caméra lui permettent de se déplacer parallèlement au sol le long des axes x_c et y_c de son repère à une hauteur constante : $\mathbf{v}_c = \big(v_{cx}\,,\, v_{cy}\big)$.
- les vitesses appliquées à chaque robot leurs permettent de se déplacer sur le sol le long des axes x_r et y_r de leurs repères : $\mathbf{v}_i = \big(v_{ix}\,,\, v_{iy}\big)$.
- n_c et n_i sont les bruits additifs gaussiens appliqués aux vitesses des robots et de la caméra qui suivent la loi normale suivante :

$$n \sim \mathcal{N}(\mu, \sigma^2), \tag{3.5}$$

avec μ la moyenne et σ^2 la variance du bruit n. Ils définissent l'erreur entre la vitesse appliquée et la vitesse réelle. Il est cependant possible d'éliminer la dérive introduite par la moyenne μ par une calibration appropriée des robots et du drone, et d'utiliser une méthode statistique qui exploite un grand nombre de résultats obtenus en condition réelle pour estimer la variance σ^2. Nous supposons donc pour la suite que μ sera toujours nulle, et que σ^2 sera toujours connue.

3.2.2 Dynamiques des modèles

Comme discuté dans la section 3.1, les positions modélisées de la caméra et des robots sont les seules informations toujours disponibles. Il est donc primordial de définir leurs dynamiques. En reprenant l'expression des dynamiques bruitées (3.4), les dynamiques des modèles des coordonnées de la caméra et des robots sont définies sous forme discrète à chaque instant t_k tels que :

$$\begin{cases} \boldsymbol{c}_m(t_{k+1}) &= \boldsymbol{c}_m(t_k) + \tau \mathbf{v}_c(t_k) \\ \boldsymbol{r}_{im}(t_{k+1}) &= \boldsymbol{r}_{im}(t_k) + \tau \mathbf{v}_i(t_k) \end{cases} \tag{3.6}$$

avec τ la période d'échantillonnage. On note que les bruits présents dans (3.4) n'interviennent pas ici, ce qui provoque les dérives entre les positions réelles et modélisées.

Nous décrivons maintenant la manière dont les vitesses des robots sont calculées pour leur permettre de suivre leurs trajectoires désirées.

3.3 Contrôle des robots

Les trajectoires désirées des robots sont représentées par des positions désirées qui varient dans le temps $\boldsymbol{r}_i^*(t)$, illustrées en vert sur la Figure 3.1 où les cercles représentent les trajectoires dans leur globalité. Les robots suivent leurs trajectoires désirées en suivant le schéma de contrôle suivant :

$$\mathbf{v}_i = \gamma(\boldsymbol{r}_i^* - \boldsymbol{r}_{im}) \tag{3.7}$$

avec le gain scalaire $\gamma > 0$. Ici, les modèles des positions des robots sont utilisés pour pouvoir appliquer cette loi de commande même en cas de sorties des robots en dehors du champ de vue de la caméra. Il apparaît ici que les dérives qui interviennent entre les positions réelles et modélisées des robots durant leurs déplacements vont également se manifester entre les positions réelles et désirées. Nous présentons maintenant comment corriger les positions modélisées grâce aux mesures de la caméra.

3.4 Correction des modèles

Les mesures de la caméra doivent en principe fournir les coordonnées de l'amer et des robots dans le repère de la caméra telles qu'elles ont été décrites dans (3.2) et (3.3). Cependant, en raison d'erreurs de traitement d'image, les mesures de la caméra sont soumises à des bruits additifs n_s qui suivent la même loi normale que les bruits de commande (3.5). Ceci implique une légère erreur sur chaque mesure $\bar{\boldsymbol{s}_a}$ pour l'amer et $\bar{\boldsymbol{s}_i}$ pour le $i$ème robot telle que :

$$\left\{ \begin{array}{rcl} \bar{\boldsymbol{s}_a} & = & \boldsymbol{s}_a + n_s \\ \bar{\boldsymbol{s}_i} & = & \boldsymbol{s}_i + n_s \end{array} \right. . \tag{3.8}$$

Ces coordonnées sont disponibles comme mesures seulement lorsque l'amer et les robots apparaissent dans le champ de vue de la caméra délimité par le champ de vue défini par $\boldsymbol{s}_{min} = (s_{min_x},\ s_{min_y})$ et $\boldsymbol{s}_{max} = (s_{max_x},\ s_{max_y})$ représentées par les pointillés oranges sur la Figure 3.1.

Une mesure permet de corriger soit les coordonnées du modèle de la caméra $\boldsymbol{c}_m$ soit les coordonnées du modèle d'un robot $\boldsymbol{r}_{im}$ dans le repère monde. De là, en utilisant (3.2), (3.3), et (3.8), on peut écrire les expressions de correction suivantes :

$$\left\{ \begin{array}{rcll} \boldsymbol{c}_m & \leftarrow & \boldsymbol{m} - \bar{\boldsymbol{s}_a} & \text{si} \quad \boldsymbol{s}_{min} \leqslant \bar{\boldsymbol{s}_a} \leqslant \boldsymbol{s}_{max} \\ \boldsymbol{r}_{im} & \leftarrow & \bar{\boldsymbol{s}_i} + \boldsymbol{c}_m & \text{si} \quad \boldsymbol{s}_{min} \leqslant \bar{\boldsymbol{s}_i} \leqslant \boldsymbol{s}_{max} \end{array} \right. \tag{3.9}$$

La dérive du modèle de la caméra peut donc être corrigée par l'observation de l'amer, alors que les dérives des modèles des robots et, par conséquent, les trajectoires désirées des véritables robots peuvent être corrigées par l'observation des robots. On note toutefois que la correction des modèles des robots s'effectue dans le repère de la caméra et non dans un repère dont la position est parfaitement connue. La précision des positions

modélisées des robots dépend donc également de la dérive du modèle de la position de la caméra dans le repère de l'amer. C'est pourquoi la correction de la dérive de la caméra est aussi importante que celle des dérives des robots pour assurer leurs suivis de trajectoires.

Nous avons présenté dans cette section les positions et les dynamiques de la caméra, de l'amer et des robots dans le repère monde et dans le repère caméra. Nous avons également introduit les bruits intervenant dans les commandes et les mesures. Nous avons souligné l'importance d'obtenir une mesure via la caméra pour corriger les modèles des positions de la caméra et des robots et, par conséquent, améliorer les performances de suivi de trajectoire des robots. Le point principal qu'il reste à développer est le système de commande approprié pour que la caméra se déplace vers les robots ou l'amer lorsque cela s'avère nécessaire, ce qui est le but de la section suivante.

3.5 Stratégie de commande prédictive

Nous décrivons dans cette section une stratégie de commande prédictive qui permet à la caméra de se déplacer soit vers l'amer soit vers l'un des robots. L'objectif est de forcer la caméra à se focaliser sur une cible puis sur une autre en fonction de l'évolution des incertitudes des positions de la caméra et des robots. Dans le cas décrit ici, ces incertitudes sont déterminées à partir du modèle des bruits qui caractérisent l'évolution de leurs dérives au cours du temps. La stratégie présentée ici peut être considérée comme un exemple particulier de perception active [Baj88]. Dans le sens où le mouvement de la caméra doit être optimisé en ligne afin de recueillir des informations relatives au système durant son évolution dans le temps.

L'utilisation de cette approche est justifiée par le fait que la caméra ne peut pas se déplacer de manière instantanée d'un point à un autre. Il est donc nécessaire d'anticiper à la fois les positions de la caméra et des robots, et de prédire l'évolution de leurs incertitudes liées aux dérives entre leurs positions réelles et modélisées.

Dans le but de développer cette stratégie de commande, nous avons besoin de définir la dynamique du système sur un horizon de prédiction, une fonction de coût appropriée en mesure de répondre à un objectif de commande, et des contraintes à satisfaire. L'usage de ces constraintes permet d'obtenir le comportement souhaité de la part de la caméra, à savoir qu'elle se déplace d'une cible à une autre en fonction des incertitudes.

3.5.1 Prédiction des positions

En suivant les dynamiques des robots et de la caméra décrites dans (3.6), les coordonnées modélisées peuvent être prédites sur un horizon de prédiction N_p comme suit :

$$\begin{cases} \overrightarrow{\boldsymbol{c}_m} &= \mathbf{A}\boldsymbol{c}_m(t_k) + \mathbf{B}\overrightarrow{\mathbf{v}_c} \\ \overrightarrow{\boldsymbol{r}_{1m}} &= \mathbf{A}\boldsymbol{r}_{im}(t_k) + \mathbf{B}\overrightarrow{\mathbf{v}_i} \end{cases} \tag{3.10}$$

où :

- $\overrightarrow{\mathbf{v}_c}$ et $\overrightarrow{\mathbf{v}_i}$ sont les ensembles des entrées de commande de la caméra et du $i$ème robot à partir de l'instant courant t_k, jusqu'à l'instant t_{k+N_p-1} correspondant à un temps d'échantillonnage avant la fin de l'horizon de prédiction :

$$\begin{cases} \overrightarrow{\mathbf{v}_c} &= \big(\mathbf{v}_c(t_k)\,,\,\ldots\,,\,\mathbf{v}_c(t_{k+N_p-1})\big) \\ \overrightarrow{\mathbf{v}_i} &= \big(\mathbf{v}_i(t_k)\,,\,\ldots\,,\,\mathbf{v}_i(t_{k+N_p-1})\big) \end{cases} \tag{3.11}$$

- $\overrightarrow{\boldsymbol{c}_m}$ et $\overrightarrow{\boldsymbol{r}_{im}}$ sont les ensembles des modèles des positions de la caméra et du $i$ème robot à partir de l'instant correspondant au temps d'échantillonnage suivant l'instant courant t_{k+1}, jusqu'à l'instant t_{k+N_p} correspondant à la fin de l'horizon de prédiction :

$$\begin{cases} \overrightarrow{\boldsymbol{c}_m} &= \big(\boldsymbol{c}_m(t_{k+1})\,,\,\ldots\,,\,\boldsymbol{c}_m(t_{k+N_p})\big) \\ \overrightarrow{\boldsymbol{r}_{im}} &= \big(\boldsymbol{r}_{im}(t_{k+1})\,,\,\ldots\,,\,\boldsymbol{r}_{im}(t_{k+N_p})\big) \end{cases} \tag{3.12}$$

- et

$$\mathbf{A} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ \vdots & \vdots \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \in \mathbb{R}^{2N_p \times 2}, \qquad \mathbf{B} = \begin{pmatrix} \tau & 0 & 0 & 0 & \ldots & 0 \\ 0 & \tau & 0 & 0 & \ldots & 0 \\ \tau & 0 & \tau & 0 & \ldots & 0 \\ 0 & \tau & 0 & \tau & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \tau & 0 & \tau & \ldots & \tau \end{pmatrix} \in \mathbb{R}^{2N_p \times 2N_p}. \tag{3.13}$$

La prédiction des positions sur l'horizon de prédiction N_p va permettre à la loi de commande d'anticiper où la caméra doit se déplacer pour retrouver les robots ou l'amer dans son champ de vue.

3.5.2 Fonction de coût

Nous avons défini l'objectif à atteindre pour le comportement de la caméra au début de ce chapitre. La caméra doit se déplacer d'une cible à une autre durant l'exécution des tâches de suivi de trajectoire. Or, ce comportement est obtenu grâce aux contraintes à satisfaire que nous développons dans la section 3.5.3.

La fonction de coût que nous souhaitons minimiser a donc un impact limité sur le comportement. Celle que nous avons choisie assure que la caméra demeure à égale distance de l'amer et des robots. Elle est donnée par l'expression suivante :

$$\min_{\overrightarrow{\mathbf{v}_c} \in \mathcal{K}} J(\overrightarrow{\mathbf{v}_c}) \tag{3.14}$$

où $\mathcal{K}$ est le domaine de contraintes décrit dans la prochaine section 3.5.3, et

$$J(\overrightarrow{\mathbf{v}_c}) = ||\overrightarrow{\boldsymbol{s}_{am}} - \overrightarrow{\boldsymbol{s}^*_{am}})||^2 + \sum ||\overrightarrow{\boldsymbol{s}_{im}} - \overrightarrow{\boldsymbol{s}^*_{im}}||^2 \tag{3.15}$$

où $\boldsymbol{s}_{am}$ et $\boldsymbol{s}_{im}$ sont les modèles des coordonnées de l'amer et du $i$ème robot définis tels que :

$$\begin{cases} \boldsymbol{s}_{am} &= \boldsymbol{m} - \boldsymbol{c}_m \\ \boldsymbol{s}_{im} &= \boldsymbol{r}_{im} - \boldsymbol{c}_m \end{cases}, \tag{3.16}$$

et $\overrightarrow{\boldsymbol{s}_{am}}$ et $\overrightarrow{\boldsymbol{s}_{im}}$ sont leurs ensembles correspondant à partir de l'instant du temps d'échantillonnage suivant l'instant courant t_{k+1}, jusqu'à l'horizon de prédiction t_{k+N_p}.

Enfin, $\overrightarrow{\boldsymbol{s}^*_{am}}$ et $\overrightarrow{\boldsymbol{s}^*_{im}}$ sont les ensembles des positions désirées de l'amer et des robots dans le repère de la caméra sur l'horizon de prédiction. Afin que l'amer et les robots soient dans le centre du champ de vue de la caméra, on fixe $\boldsymbol{s}^*_{am}$ et $\boldsymbol{s}^*_{im}$ à zéro constamment. Cette fonction de coût a été définie afin que la caméra reste à égale distance de l'amer et des robots. Mais, cette distance est fortement perturbée par les contraintes (3.22). En effet, ces contraintes forcent la caméra à bouger vers l'amer ou vers un robot.

3.5.3 Contraintes

Un des points essentiels de la commande prédictive est la possibilité de prendre en compte des contraintes. Nous allons utiliser cette caractéristique pour influencer le mouvement de la caméra tel qu'il est souhaité. Deux types de contraintes sont à distinguer ici, celles liées aux limites physiques du système (limites en vitesse du drone sur lequel la caméra est montée), et celles relatives aux positions de l'amer et des cibles dans le champ de vue de la caméra qui vont nous permettre de déplacer la caméra d'une cible à une autre.

3.5.3.1 Contraintes physiques

Dans un premier temps, les constraintes liées aux entrées de commande de la caméra sont inclues dans le domaine $\mathcal{K}$ tel que :

$$\mathbf{v}_c^- \leqslant \mathbf{v}_c \leqslant \mathbf{v}_c^+ \tag{3.17}$$

pour assurer que la vitesse de la caméra ne dépasse pas les limites physiques imposées par le drone, avec $\mathbf{v}_c^- = (v_{cx}^-,\, v_{cy}^-)$ et $\mathbf{v}_c^+ = (v_{cx}^+,\, v_{cy}^+)$ les vitesses minimales et maximales de la caméra le long des axes x_c et y_c. Ces contraintes sont appliquées sur l'ensemble des entrées $\overrightarrow{\mathbf{v}_c}$ à partir de l'instant courant t_k, jusqu'à l'instant t_{k+N_p-1}.

3.5.3.2 Contraintes relatives aux positions

Dans cette partie, nous allons modéliser la propagation des incertitudes de position entre les positions réelles et modélisées sur l'horizon de prédiction. Les évolutions de ces incertitudes vont ensuite être comparées pour sélectionner quel modèle doit être corrigé en priorité. Puis, pour forcer la caméra à se déplacer vers le modèle sélectionné, des contraintes sur les positions modélisées de l'amer et des robots dans le repère de la caméra sont inclues dans le domaine $\mathcal{K}$.

Incertitudes des positions : Les vitesses appliquées à la caméra et aux robots sont sujettes à des bruits (3.4), ce qui cause une dérive entre leurs positions réelles et modélisées. Ces bruits suivent la loi normale (3.5), mais comme il a été introduit dans la section 3.2, les moyennes μ de ces bruits peuvent être considérées comme nulles. En revanche, en connaissant les variances σ^2 de ces bruits, il est possible de modéliser la propagation des dérives dans le temps. Pour cela, nous introduisons l'incertitude p_j de chaque position où $j = 0, ..., \mathcal{N}_r$ et où p_0 représente l'incertitude sur la position de la caméra et $p_i,\ i = 1, ..., \mathcal{N}_r$ celle sur les positions des robots.

Deux situations apparaissent pour déterminer la propagation des incertitudes. Premièrement, lorsqu'une cible se trouve dans le champ de vue de la caméra, le modèle correspondant est corrigé. Une valeur minimale est donc affectée à son incertitude :

$$p_j(t_k) = \sigma_s^2 \qquad \text{si} \quad \boldsymbol{s}_{min} \leqslant \bar{\boldsymbol{s}}_j(t_k) \leqslant \boldsymbol{s}_{max}, \tag{3.18}$$

qui correspond à la variance σ_s^2 de l'erreur de mesure de la caméra (3.8). En effet, la seule incertitude qui apparaît lorsque l'amer ou un des robots se trouve dans le champ de vue de la caméra est liée à l'imprécision de la mesure de la caméra.

Deuxièmement, lorsqu'une cible ne se trouve pas dans le champ de vue de la caméra, le modèle de sa position n'est pas corrigé. La propagation de son incertitude est modélisée telle que [TBF05] :

$$p_j(t_{k+1}) = p_j(t_k) + \tau^2 \sigma_j^2. \tag{3.19}$$

En prenant en compte les deux situations décrites par les expressions (3.18) et (3.19), on propage l'incertitude p_j sur l'horizon de prédiction N_p :

$$\begin{cases} p_j(t_{k+N_p}) &= \sigma_s^2 \ \text{ si } \ \boldsymbol{s}_{min} \leqslant \bar{\boldsymbol{s}}_j(t_k) \leqslant \boldsymbol{s}_{max} \\ p_j(t_{k+N_p}) &= p_j(t_k) + N_p \tau^2 \sigma_j^2 \ \text{ sinon} \end{cases} . \tag{3.20}$$

Nous avons décrit comment les incertitudes des robots et de la caméra évoluent sur l'horizon de prédiction N_p. En comparant les incertitudes les unes avec les autres, il est possible de déterminer quel modèle doit être corrigé en priorité. Pour cela on introduit une condition associée à chaque élément j du système.

Priorité de la correction : En prédisant les incertitudes de position sur l'horizon de prédiction, on peut définir quel modèle doit être corrigé. Pour cela, une condition C_j est associé à chaque incertitude telle que :

$$\begin{array}{l} C_j \Leftrightarrow \\ \left(\begin{array}{lcll} & p_j(t_{k+N_p}) & \geqslant & p_{max_j} \\ \text{et} & p_j(t_{k+N_p}) - p_{max_j} & > & p_l(t_{k+N_p}) - p_{max_l} \ , \ \forall l \neq j \end{array} \right) \\ \text{ou} \\ \left(\begin{array}{lcl} & p_l(t_{k+N_p}) & < \ p_{max_l} \\ \text{et} & p_j(t_{k+N_p}) & < \ p_l(t_{k+N_p}) \end{array} \right) \forall l \neq j \\ \forall j = 0, ..., \mathcal{N}_r \end{array} \tag{3.21}$$

On introduit ici le seuil d'incertitude p_{max_j} correspondant à la valeur que l'incertitude p_j ne doit pas dépasser pour que le modèle correspondant puisse être corrigé malgré le champ de vue limité de la caméra. La manière dont ce seuil doit être calculé est décrite dans le paragraphe "Seuil d'incertitude". On décrit ici comment la condition C_j se comporte en détails :

- Si l'incertitude p_j est plus grande que le seuil p_{max_j} sur l'horizon de prédiction et si la différence entre p_j et p_{max_j} est la plus grande parmi toutes les autres incertitudes, la condition est respectée. Cela garantit que le modèle, dont l'incertitude dépasse le plus son seuil sur l'horizon de prédiction, doit être corrigé.
- Ou si les incertitudes sont toutes en-dessous de leurs seuils, et si l'incertitude p_j est plus petite que toutes les autres sur l'horizon de prédiction, la condition est respectée. Cela garantit que soit le modèle de l'amer soit de l'un des robots est toujours corrigé même si aucune incertitude ne dépasse son seuil sur l'horizon de prédiction. Ce choix réduit au minimum le déplacement de la caméra, puisque elle reste focalisée sur la même cible tant que les autres n'ont pas à être corrigées.

On peut noter que dans (3.20) si l'amer ou un robot est observé à $t = t_k$, on considère que ce dernier sera visible sur l'ensemble de l'horizon de prédiction jusqu'à $t = t_{k+N_p}$. Cette hypothèse va inciter la condition (3.21) à apporter une priorité sur un des modèles qui n'est pas corrigé.

Les prédictions des incertitudes et leurs comparaisons les unes avec les autres permettent à la stratégie de commande de déterminer quel modèle doit être corrigé en priorité par les observations de la caméra. Nous décrivons maintenant comment inclure ces informations dans la commande prédictive par le biais de contraintes à satisfaire.

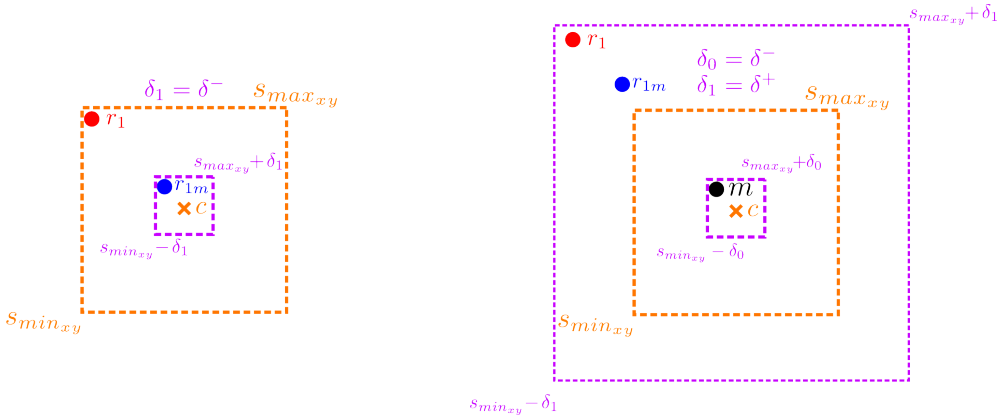
Tolérances : On appelle tolérances un ensemble de distances $\overrightarrow{\delta_j}$ liées aux limites du champ de vue de la caméra $\boldsymbol{s}_{min}$, $\boldsymbol{s}_{max}$ et aux modèles des coordonnées de l'amer et des robots dans le repère de la caméra sur l'horizon de prédiction N_p. Ces distances interviennent pour forcer la caméra à se diriger vers l'amer ou l'un des robots. Elles sont donc introduites au travers des contraintes ajoutées au domaine $\mathcal{K}$:

$$\boldsymbol{s}_{min} - \overrightarrow{\delta_j} \leqslant \overrightarrow{\boldsymbol{s}_{jm}} \leqslant \boldsymbol{s}_{max} + \overrightarrow{\delta_j} \ , \quad \forall j = 0, ..., \mathcal{N}_r. \tag{3.22}$$

Ainsi, les modèles des coordonnées de l'amer et des robots dans le repère de la caméra $\boldsymbol{s}_{jm}$ doivent rester entre des limites virtuelles du champ de vue de la caméra qui sont soit étendues soit réduites par rapport aux limites réelles en fontion de la valeur de $\overrightarrow{\delta_j}$.

Nous posons δ^- et δ^+ comme étant respectivement une valeur basse et une valeur haute de la tolérance. Celles-ci sont affectées aux tolérances $\overrightarrow{\delta_j}$ en suivant les conditions définies dans (3.21) :

$$\begin{cases} \overrightarrow{\delta_j} & = & \delta^- & \text{si} & C_j \\ \overrightarrow{\delta_j} & = & \delta^+ & \text{sinon} & \end{cases} \tag{3.23}$$



(a) (b)

FIGURE 3.2 – Principe de la tolérance : Le champ de vue de la caméra est représenté par les pointillés oranges. (a) La tolérance δ_1 representée en violet est négative. La caméra $\boldsymbol{c}$ retrouve le robot $\boldsymbol{r}_1$ dans son champ de vue malgré la dérive du modèle $\boldsymbol{r}_{1m}$. (b) La tolérance δ_0 est négative, et δ_1 est grande. La caméra retrouve l'amer dans son champ de vue et son modèle est corrigé. La caméra tolère que le robot reste loin de son champ de vue.

L'idée consiste à avoir une valeur basse δ^- pour forcer la caméra à se déplacer vers le modèle dont la correction est estimée comme prioritaire par la condition C_j. Par exemple, sur la Figure 3.2a, la tolérance δ_1 est égale à δ^-, la caméra se déplace donc vers le premier robot. Cependant, si la valeur de δ^- était égale à 0, on ne pourrait pas garantir que la position réelle du robot serait dans le champ de vue de la caméra à la fin du déplacement de la caméra à cause de sa dérive par rapport à son modèle. C'est pourquoi la valeur δ^- doit être négative pour permettre au robot et à son modèle d'être dans le champ de vue de la caméra malgré les dérives.

La caméra ne peut se déplacer que vers un robot ou l'amer à la fois. C'est pourquoi une valeur haute de tolérance δ^+ est attribuée aux modèles qui n'ont pas la priorité de correction fournie par la condition C_j afin d'assurer que le système tolère que la caméra reste loin des autres cibles. Ainsi, sur la Figure 3.2b, alors que la tolérance appliquée à l'amer δ_0 est négative, la tolérance attribuée au robot δ_1 est grande, ce qui permet à la caméra de se déplacer vers l'amer et de rester loin du robot $\boldsymbol{r}_1$.

Les contraintes définies par (3.22) permettent au schéma de commande de forcer la caméra à se déplacer vers l'amer ou vers un robot lorsque c'est nécessaire. δ^- et δ^+ sont attribués en fonction des conditions (3.21) qui dépendent de l'évolution des incertitudes p_j et du seuil d'incertitude p_{maxj} que l'on décrit maintenant.

Seuil d'incertitude : Comme décrit ci-dessus, la valeur δ^- doit être négative pour assurer au robot ou à l'amer d'être dans le champ de vue de la caméra malgré les dé-

rives des modèles. Cependant, quelle que soit la valeur choisie pour δ^-, la distance de la dérive entre la pose réelle et modélisée peut dépasser la distance décrite par $|\delta^-|$ après plusieurs itérations. Si cela se produit, la position réelle de l'amer ou du robot se retrouve en dehors du champ de vue de la caméra, et le modèle correspondant ne peut pas être corrigé. La solution consiste à corriger le modèle avant que la distance $|\delta^-|$ ne soit atteinte par la distance de dérive.

On définit N_{min_j} comme le nombre d'itérations requis pour que la dérive atteigne la distance $|\delta^-|$ depuis une valeur nulle. On modélise alors l'évolution de la distance de dérive. Pour cela, on utilise l'évolution de l'incertitude introduite dans (3.20). Enfin, on calcule l'incertitude p_{max_j} atteinte après N_{min_j} itérations. Cette incertitude est donc le seuil à ne pas dépasser par l'incertitude p_j pour que la dérive ne dépasse pas la distance limitée par $|\delta^-|$.

En propageant l'incertitude p_j sur N_{min_j} itérations à partir de l'incertitude correspondante au modèle corrigé (3.18), nous obtenons le seuil d'incertitude p_{max_j}. Pour cela, on utilise (3.20) tel que :

$$p_{max_j} = \sigma_s^2 + N_{min_j}\tau^2\sigma_j^2. \tag{3.24}$$

En utilisant ce seuil dans (3.21), nous assurons que le modèle est corrigé avant que la dérive n'atteigne la distance $|\delta^-|$. Il est cependant nécessaire de définir le nombre minimum d'itérations N_{min_j} qui intervient dans (3.24).

Nombre minimum d'itérations : On introduit dans un premier temps les distances de dérive entre les positions réelles et modélisées :

$$\left\{ \begin{array}{rcl} e_0 & = & ||\boldsymbol{c} - \boldsymbol{c}_m|| \\ e_i & = & ||\boldsymbol{r}_i - \boldsymbol{r}_{im}|| \end{array} \right. \quad \forall i = 1, \ldots, \mathcal{N}_r \tag{3.25}$$

avec e_0 la distance entre la caméra et son modèle et e_i la distance entre le $i$ème robot et son modèle. L'effet de ces dérives sur les positions réelles d'un robot $\boldsymbol{r}_1$ et de la caméra $\boldsymbol{c}$ est illustré sur la Figure 3.3. Dans cet exemple $\delta_1 = \delta^-$, la caméra doit donc se déplacer vers le robot. Entre t_k et $t_k + 1$, des dérives apparaissent entre $\boldsymbol{c}$ et $\boldsymbol{c_m}$ et entre $\boldsymbol{r}_1$ et $\boldsymbol{r}_{1m}$. Ces dérives se propagent jusqu'à ce que la somme de leurs distances $e_0 + e_1$ atteigne la distance définie par $|\delta^-|$ après N_{min_1} itérations. Finalement, malgré ces dérives, la caméra est toujours capable de retrouver le robot dans son champ de vue. D'où l'importance de retrouver ce nombre minimum d'itérations N_{min_j} pour la caméra et pour chaque robot afin d'assurer que le seuil d'incertitude (3.24) corresponde à l'incertitude atteinte lorsque les distances de dérive égalent la distance $|\delta^-|$.

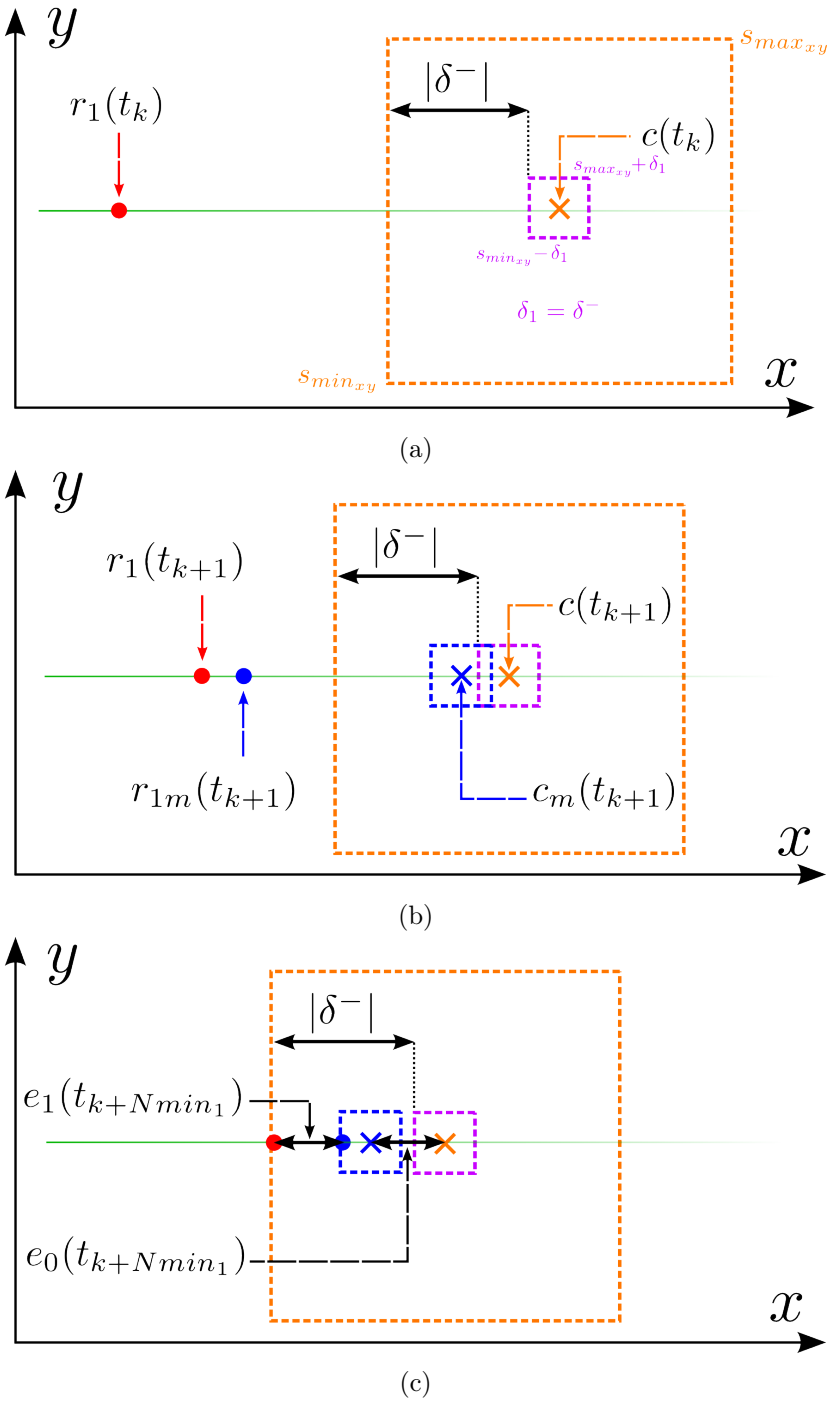


FIGURE 3.3 – Distances de dérive : (a) À t_k, le robot $\boldsymbol{r}_1$ suit la trajectoire représentée en vert, la caméra $\boldsymbol{c}$ doit se déplacer vers le robot pour satisfaire la contrainte (3.22). (b) À t_{k+1}, on observe des dérives entre $\boldsymbol{c}$ et $\boldsymbol{c}_m$ et entre $\boldsymbol{r}_1$ et $\boldsymbol{r}_{1m}$. (c) à t_{k+Nmin_1} la somme des dérives $e_0 + e_1$ atteint la distance $|\delta^-|$, et le robot se trouve dans le champ de vue de la caméra.

Pour déterminer N_{min_j}, il est nécessaire de modéliser les distances de dérive (3.25). Pour cela, on utilise la densité de probabilité de la loi normale que suit chaque bruit (3.5). Cette densité établit que 99.7% des valeurs du bruit sont distribuées entre $-3\sigma_j$ et $+3\sigma_j$ où σ_j est l'écart type du bruit [Pro53]. On utilise donc cette caractéristique de manière à estimer efficacement les distances de dérive, en considérant qu'une distance de dérive e_j égale à $3\sigma_j$ se propage à chaque itération.

De là, l'incertitude correspondante est le carré de cette distance, soit $9\sigma_j^2$ à chaque itération. La valeur de N_{min_j} correspond donc au rapport entre le carré de la distance "limite" définie par $|\delta^-|^2$ et l'incertitude $9\sigma_j^2$.

Cependant, deux N_{min_j} différents sont calculés, le premier N_{min_i} est lié au $i$ème robot, le second N_{min_0} est lié à la caméra.

Comme il a été rappelé au début de ce paragraphe, le nombre d'itérations N_{min_i} dépend de la somme des distances de dérives du $i$ème robot et de la caméra, et est donc le résultat du rapport entre le carré de la distance limite $|\delta^-|^2$ et la somme des carrés des distances de dérive du robot i et de la caméra. De là, le nombre d'itérations N_{min_i} pour le $i$ème robot peut être exprimé tel que :

$$N_{min_i} = \frac{|\delta^-|^2}{9\tau^2(\sigma_c^2 + \sigma_i^2)}. \tag{3.26}$$

N_{min_i} peut alors être introduite dans (3.24) pour calculer le seuil d'incertitude du $i$ème robot .

Enfin, en ce qui concerne N_{min_0} qui est lié à la caméra, on doit s'assurer que tous les modèles des robots puissent être corrigés malgré les dérives de la caméra. Cela signifie que nous devons nous assurer que le modèle de la caméra est corrigé autant que celle du robot qui présente la plus grande dérive. Par conséquent, N_{min_0} est égale au N_{min_i} qui a la plus petite valeur :

$$N_{min_0} = \min(N_{min_i}). \tag{3.27}$$

N_{min_0} peut alors être introduite dans (3.24) pour calculer le seuil d'incertitude de la caméra.

Nous avons décrit dans cette partie comment calculer les seuils d'incertitude p_{max_j} afin de l'utiliser dans les conditions (3.21). Nous décrivons maintenant comment calculer la valeur de l'horizon de prédiction N_p pour nous assurer que le comportement souhaité puisse toujours être réalisé.

3.5.4 Horizon de prédiction

Pour garantir que les modèles puissent être corrigés malgré les erreurs introduites par les bruits, nous devons nous assurer qu'aucune incertitude ne dépasse son seuil. Ceci

peut être garanti par un calcul approprié de l'horizon de prédiction N_p.

La caméra ne peut pas se déplacer d'une position à une autre de façon immédiate. L'horizon de prédiction doit donc correspondre au nombre d'itérations nécessaire pour que la caméra accomplisse le chemin le plus long, et ce en fonction de sa vitesse :

$$N_p \geqslant \frac{d_{max}}{\tau v_{cmin}} \tag{3.28}$$

où $v_{cmin} = \min(|v_{cx}^-|, |v_{cy}^-|, |v_{cx}^+|, |v_{cy}^+|)$ afin de prendre en compte la plus petite valeur de vitesse de la caméra, et où d_{max} représente le chemin le plus long emprunté par la caméra pour atteindre les robots et l'amer. Sa valeur dépend des configurations initiales de la scène (un exemple est fourni dans la section 3.6.1.3).

Finalement, nous pouvons utiliser l'horizon de prédiction N_p calculé dans (3.28) et les nombres d'itérations minimum déterminés dans (3.26) et (3.27) pour déterminer si le comportement désiré peut être réalisé sans aucune perte d'information, ce qui se produit si :

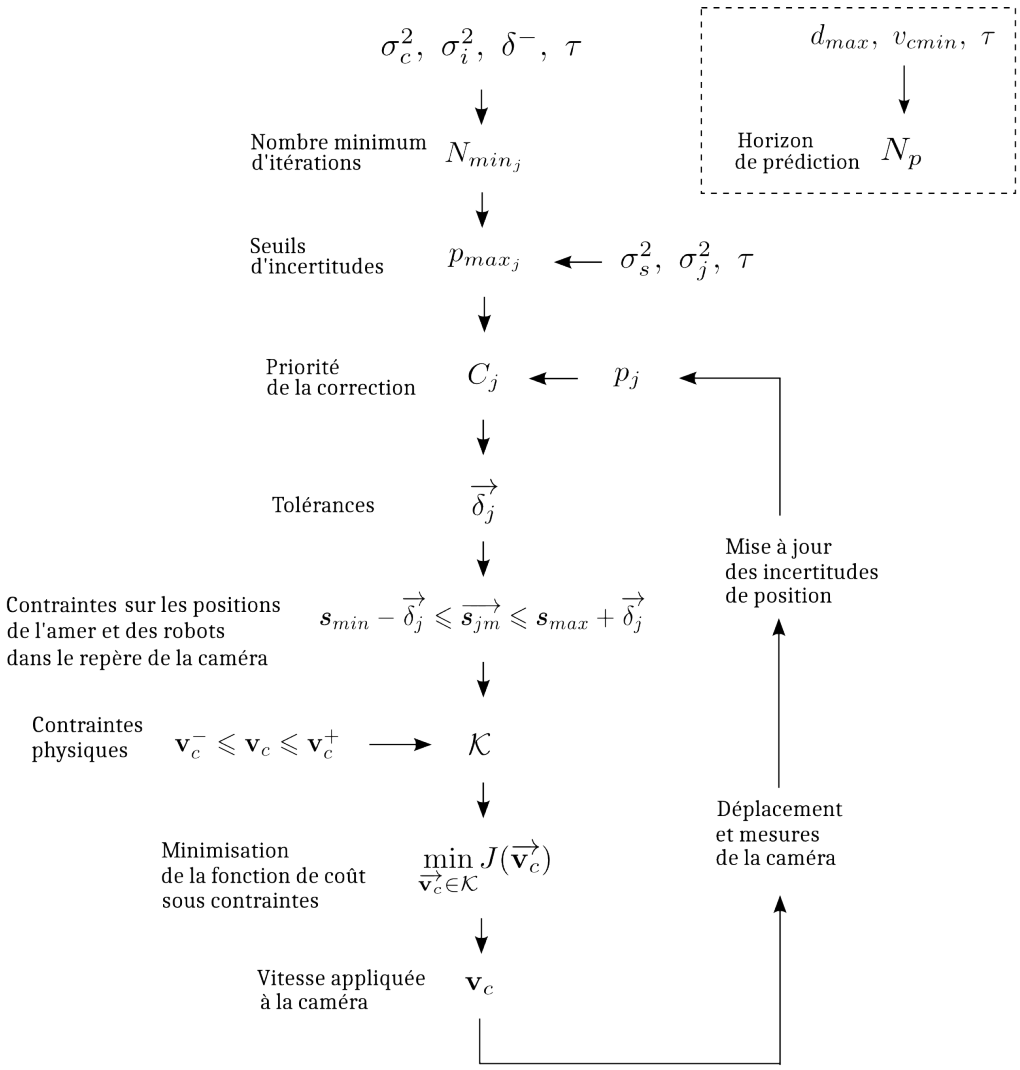
$$N_p \geqslant \min(N_{min_j}). \tag{3.29}$$

En effet, il est nécessaire pour la commande d'anticiper correctement les dépassements des seuils d'incertitude pour que la caméra change de cible à atteindre au bon moment. Si N_p est supérieur ou égal au nombre minimum d'itérations du robot ou de la caméra qui présente la plus forte dérive (et donc le plus petit N_{min}), la caméra est capable de retrouver tous les modèles dans son champ de vue avant que la distance définie par $|\delta^-|$ ne puisse être atteinte par une des dérives.

La stratégie de commande prédictive utilisée pour un système composé d'une caméra embarquée sur un drone et plusieurs robots mobiles a été détaillée dans cette section. Il apparaît que les contraintes définies dans 3.5.3 influencent totalement les mouvements de la caméra, qui peut alors fournir des mesures permettant de corriger les dérives des robots mobiles qui accomplissent leurs suivis de trajectoires. La section suivante propose de visualiser le comportement de la caméra influencé par la commande prédictive au travers de résultats de simulation.

3.5.5 Visualisation du schéma de commande prédictive

Afin de résumer schématiquement la commande définie dans cette section 3.5, nous pouvons visualiser ci-dessous la Figure 3.4 qui décrit comment les différentes variables, introduites dans les parties précédentes, interagissent afin d'assurer la réalisation de la tâche désirée.



Valeurs fixées

σ_c^2 : Variance du bruit sur la dynamique de la caméra

σ_i^2 : Variance du bruit sur la dynamique des robots

σ_s^2 : Variance du bruit de mesure de la caméra

$$\sigma_j^2 = \begin{cases} \sigma_c^2 & \text{si } j = 0 \\ \sigma_i^2 & \text{si } j \neq 0 \end{cases}$$

δ^- : Valeur de la tolérance négative

τ : Temps d'échantillonnage

v_{cmin} : Vitesse de la caméra la moins importante

d_{max} : Distance maximum empruntée par la caméra

FIGURE 3.4 – Intéraction des variables qui interviennent pour la réalisation du schéma de commande prédictive.

3.6 Résultats de simulation

Les résultats de simulation de la méthode exposée ci-dessus sont présentés dans cette section. Sont considérées ici une caméra embarquée sur un drone et deux robots mobiles. Les robots suivent une trajectoire chacun qui décrit un cercle.

3.6.1 Paramètres

On décrit dans un premier temps tous les paramètres du système. Toutes les coordonnées décrites ci-dessous sont exprimées en mètres.

3.6.1.1 Configurations initiales

La pose initiale de la caméra est $\boldsymbol{c}(t_0) = \big(0.7\,,\,0.6\big)$, la pose de l'amer est $\boldsymbol{m} = \big(0.5\,,\,0.5\big)$ et les poses des deux robots sont $\boldsymbol{r}_1(t_0) = \big(0.5\,,\,0.7\big)$ et $\boldsymbol{r}_2(t_0) = \big(1\,,\,0.7\big)$. Nous considérons les modèles des positions comme parfaitement estimés au début de la simulation $\boldsymbol{c}_m(t_0) = \boldsymbol{c}(t_0)$, $\boldsymbol{r}_{1m}(t_0) = \boldsymbol{r}_1(t_0)$ et $\boldsymbol{r}_{2m}(t_0) = \boldsymbol{r}_2(t_0)$. Les incertitudes sont donc initialisées à leurs valeurs minimum à l'initialisation $p_j(t_0) = \sigma_s^2$ comme il a été introduit dans (3.18).

Le champ de vue de la caméra est défini par $\boldsymbol{s}_{min} = \big(-0.08\,,\,-0.08\big)$ et $\boldsymbol{s}_{max} = \big(0.08\,,\,0.08\big)$.

En ce qui concerne les bruits ajoutés aux déplacements de la caméra et des robots, leurs moyennes sont nulles $\mu_c = \mu_1 = \mu_2 = 0$, et leurs écarts types sont $\sigma_c = 0.07$, $\sigma_1 = 0.09$ et $\sigma_2 = 0.08$. Le bruit ajouté aux mesures est défini par $\mu_s = 0$ et $\sigma_s = 0.005$.

Les tolérances introduites dans (3.23) dépendent de la valeur maximum des tolérances $\delta^+ = 2$, et de la valeur minimale $\delta^- = -0.08$ correspondant à la taille du champ de vue depuis le centre de l'image.

Les valeurs minimum et maximum qui peuvent être atteintes par la caméra sont $\mathbf{v}_c^- = -2\text{m/s}$ et $\mathbf{v}_c^+ = +2\text{m/s}$. Pour le contrôle des robots (3.7), le gain est fixé à $\gamma = 1$. Le temps total de simulation est divisé en 375 itérations avec une période d'échantillonnage de $\tau = 0.04\text{s}$ correspondant à une simulation de 15s.

Enfin, la méthode numérique choisie pour minimiser J selon les contraintes $\mathcal{K}$ est disponible dans la fonction *COBYLA* (Constrained Optimization BY Linear Approximation) [Pow98] de la bibliothèque C++ NLOPT [Joh].

3.6.1.2 Trajectoires désirées

On définit ici les trajectoires désirées des robots à chaque itération et sur l'horizon de prédiction :

$$\begin{cases} \overrightarrow{\boldsymbol{r}_1^*} = \mathbf{A}\boldsymbol{r}_1^*(t_k) + \mathbf{BE}\begin{pmatrix} 0.2\cos(t_k) \\ 0.2\sin(t_k) \end{pmatrix} \\ \overrightarrow{\boldsymbol{r}_2^*} = \mathbf{A}\boldsymbol{r}_2^*(t_k) + \mathbf{BE}\begin{pmatrix} -0.2\cos(t_k) \\ -0.2\sin(t_k) \end{pmatrix} \end{cases} . \tag{3.30}$$

avec $\boldsymbol{r}_1^*(t_0) = \boldsymbol{r}_1(t_0)$, $\boldsymbol{r}_2^*(t_0) = \boldsymbol{r}_2(t_0)$,

$$\mathbf{E} = \begin{pmatrix} \cos(\tau) & -\sin(\tau) \\ \sin(\tau) & \cos(\tau) \\ \vdots & \vdots \\ \cos(N\tau) & -\sin(N\tau) \\ \sin(N\tau) & \cos(N\tau) \end{pmatrix} \in \mathbb{R}^{2N_p \times 2} \tag{3.31}$$

et les matrices $\mathbf{A}$ et $\mathbf{B}$ définies dans (3.13). Ces modèles permettent d'obtenir deux trajectoires circulaires répétées indéfiniment par les deux robots. Les rayons de ces cercles sont $R_1 = R_2 = 0.2$ et leurs centres sont $\boldsymbol{C}_1 = \big(0.5\,,\,0.9\big)$ et $\boldsymbol{C}_2 = \big(1\,,\,0.5\big)$.

3.6.1.3 Horizon de prédiction

Pour calculer l'horizon de prédiction, nous déterminons dans un premier temps le chemin le plus long emprunté par la caméra pour corriger tous les modèles. Avec les configurations considérées précédemment et comme il est illustré sur la Figure 3.5, le chemin le plus long est défini tel que :

$$d_{max} = d_m + d_r \tag{3.32}$$

avec d_r la distance la plus grande entre les deux robots sur leurs trajectoires :

$$d_r = R_1 + R_2 + \sqrt{(C_{1_x} - C_{2_x})^2 + (C_{1_y} - C_{2_y})^2} \tag{3.33}$$

et d_m la distance entre l'amer et le robot situé le plus loin de l'amer sur sa trajectoire :

$$d_m = R_2 + \sqrt{(C_{2_x} - m_x)^2 + (C_{2_y} - m_y)^2} \tag{3.34}$$

En combinant (3.28) et (3.32) nous pouvons fixer $N_p = 22$. La condition (3.29) est respectée, ce qui veut dire que la correction des modèles est garantie.

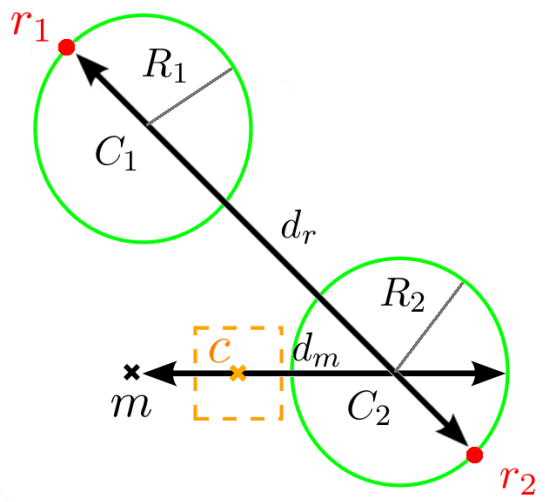


FIGURE 3.5 – Chemin le plus long emprunté par la caméra pour atteindre les robots et l'amer : $d_{max} = d_m + d_r$. Les trajectoires des robots sont des cercles représentées en vert dont les rayons R_1, R_2 ainsi que les positions des centres $\boldsymbol{C}_1$ et $\boldsymbol{C}_2$ sont connus.

Les simulations présentées ci-dessous peuvent être visualisées sous la forme d'une vidéo disponible en suivant l'URL suivant : `https://youtu.be/YjrhjtorqEU`. Celle-ci présente les trajectoires des robots et du drone depuis une vue du dessus et en vue 3D, ainsi que les évolutions des incertitudes au cours du temps.

3.6.1.4 Horizon de prédiction faible

Nous considérons dans un premier temps notre stratégie de contrôle avec un horizon de prédiction très faible $N_p = 1$. Dans ce cas, le temps de calcul requis est approximativement de 1 ms pour chaque appel de la fonction *COBYLA*. La Figure 3.6 représente l'évolution des vitesses appliquées à la caméra durant la simulation. On peut voir que les vitesses ne dépassent jamais les limites $\mathbf{v}_c^-$ et $\mathbf{v}_c^+$ grâce à la contrainte (3.17). Les Figures 3.7a et 3.7b montrent l'évolution des primitives visuelles réelles s_x, s_y et désirées s_x^* et s_y^* de chaque robot dans le repère de la caméra. La Figure 3.7c représente l'évolution des coordonnées de l'amer dans le repère de la caméra. Le champ de vue de la caméra est représenté en orange tandis que l'évolution des tolérances est en violet (pour le premier robot), en cyan (pour le second robot) et en noir (pour l'amer). Nous pouvons ainsi visualiser quand les modèles ont été corrigés, c'est-à-dire lorsque les courbes rouges et bleues se trouvent entre les deux lignes oranges au même moment. Finalement, la Figure 3.7d montre l'évolution des incertitudes dans le temps ainsi que les seuils d'incertitude correspondant à chaque élément.

Il peut être observé à $t = 0$ que les incertitudes ne dépassent pas leurs seuils. Cependant, celle liée à la caméra est plus petite que les autres sur l'horizon de prédiction. La condition (3.21) est satisfaite pour la caméra, δ^- est donc attribué à la tolérance

de l'amer tandis que δ^+ est attribué à la tolérance des robots. On peut également voir sur la Figure 3.7c que $s_{ax}(t_0)$ et $s_{ay}(t_0)$ ne se trouvent pas dans l'intervalle décrit par $s_{min} - \delta_0$ et $s_{max} + \delta_0$. La caméra doit par conséquent se déplacer vers l'amer pour satisfaire la contrainte (3.22). Le modèle de la caméra est ainsi corrigé par l'observation de l'amer. De la même façon, la caméra doit se déplacer vers le premier robot à $t = t_1$, quand l'incertitude propagée sur l'horizon de prédiction $p_1(t_{k+N_p})$ atteint p_{max1}. On peut voir que p_1 dépasse p_{max1} durant le temps pris par la caméra pour se déplacer vers le premier robot. La caméra est cependant toujours en état de retrouver le robot dans son champ de vue. En effet, la somme des distances des dérives du premier robot et de la caméra n'est pas assez importante pour dépasser la distance décrite par $|\delta^-|$. Cette distance est cependant dépassée pour le deuxième robot à $t = t_2$. Dans cette situation, la valeur minimum de tolérance δ^- est attribuée à la tolérance du deuxième robot, alors que δ^+ est attribuée aux tolérances du premier robot et de l'amer jusqu'à la fin de la simulation. Les robots ne sont alors plus assistés par les observations de la caméra et divergent de leurs trajectoires désirées. Ainsi, les courbes bleues et rouges divergent des courbes vertes sur les Figures 3.7a et 3.7b. On peut finalement observé sur la Figure 3.6 la conséquence de la perte de l'amer et des robots sur les vitesses appliquées à la caméra qui se limitent à lui permettre de suivre la trajectoire du modèle du deuxième robot. On peut visualiser cette simulation dans la première partie de la vidéo `https://youtu.be/YjrhjtorqEU`. On peut y observer le moment où le robot $\boldsymbol{r}_2$ ne peut plus être retrouvé par la caméra dans son champ de vue à partir de $t = 14$s. Celle-ci cherche alors à satisfaire la contrainte liée à la position du modèle $\boldsymbol{r}_{2m}$ dans son champ de vue et reste focalisée sur celui-ci, puisque l'incertitude de position de $\boldsymbol{r}_2$ n'est plus mis à jour, et le modèle garde donc la priorité de la correction.

La solution consiste à calculer correctement l'horizon de prédiction suivant la méthode décrite dans la section 3.5.4 et détaillée dans la section 3.6.1.3.

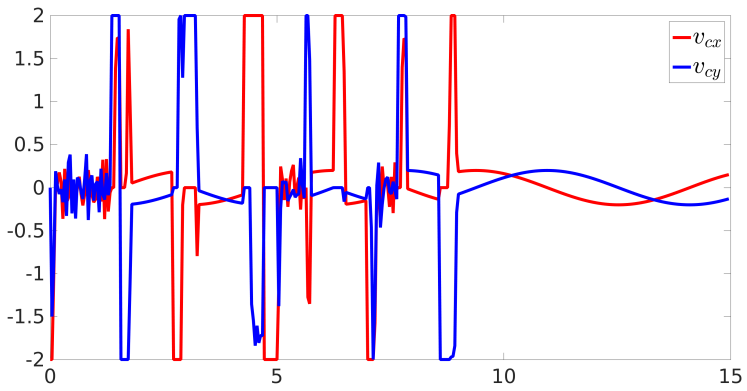
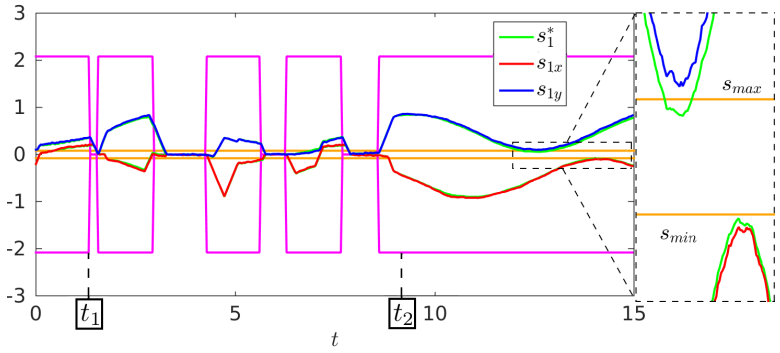
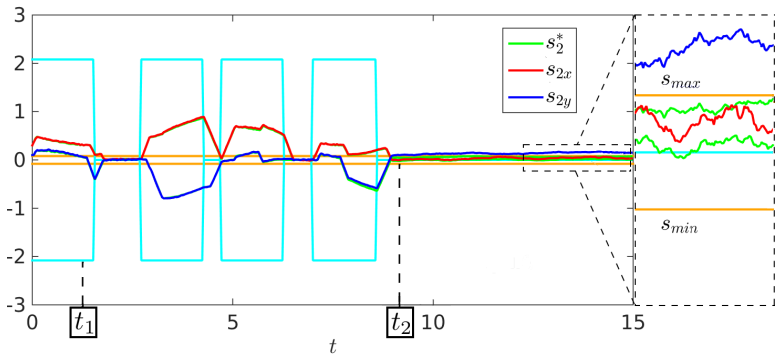


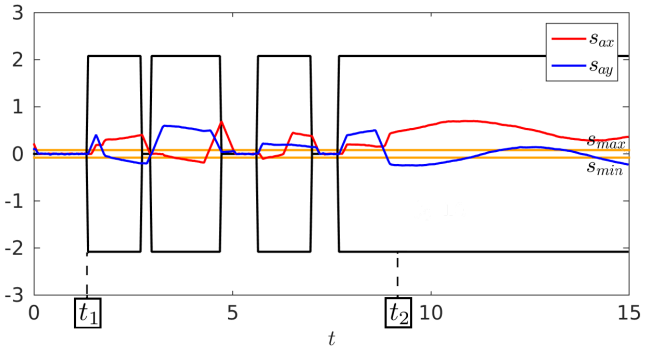
FIGURE 3.6 – Petit horizon de prédiction : Évolution des vitesses appliquées à la caméra.



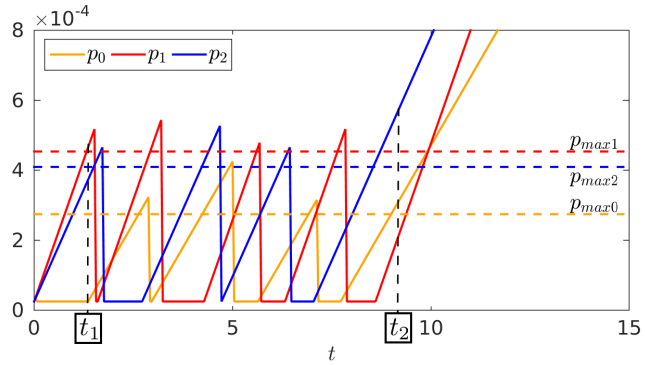
(a)



(b)



(c)



(d)

FIGURE 3.4 - Petit horizon de prédiction : (a) et (b) montrent les coordonnées des robots dans le plan image de la caméra, avec $\boldsymbol{s}_{min} - \delta_i$ et $\boldsymbol{s}_{max} + \delta_i$ représentés en violet et en cyan. (c) montre les coordonnées réelles de l'amer dans la plan image de la caméra, avec $\boldsymbol{s}_{min} - \delta_0$ et $\boldsymbol{s}_{max} + \delta_0$ représentés en noir. (d) montre les incertitudes et leurs seuils. À $t = t_2$, la caméra n'est pas capable d'observer le second robot dans son champ de vue à cause du petit horizon de prédiction qui ne permet pas d'anticiper les erreurs liées aux modèles.

3.6.2 Grand horizon de prédiction

On applique maintenant notre méthode avec un horizon de prédiction fixé à $N_p = 22$ suivant le calcul présenté dans la section 3.6.1.3. Le temps de calcul requis pour la minimisation de J est par conséquent plus grand que dans le cas précédent. Cependant pour être en mesure de respecter la période d'échantillonnage τ, et pour obtenir ainsi un système pouvant fonctionner en temps réel, nous avons limité le temps de la résolution de la fonction de coût (3.14) à 40 ms. Ceci est possible avec la bibliothèque NLOPT mais peut introduire une perte de précision dans le calcul de la solution, qui n'est pas assez significative ici pour observer un problème conséquent.

Comme on peut le voir sur les Figures 3.9a, 3.9b, 3.9c, cet horizon de prédiction permet à la caméra de changer de cible plus rapidement que dans la simulation précédente. En effet, en anticipant les dépassements de seuil depuis un horizon de prédiction approprié, la fréquence de modification des valeurs des tolérances entre δ^- et δ^+ augmente de manière significative pour les deux robots et l'amer. Ceci permet un plus grand nombre de corrections réalisées par la caméra via (3.9) pour satisfaire les contraintes (3.22) qui évoluent dans le temps. On peut ajouter à cela les vitesses appliquées à la caméra représentées sur la Figure 3.8 qui lui permettent de changer de cible plus régulièrement que dans le cas où $N_p = 1$.

On peut également observer sur la Figure 3.9d que les incertitudes ne dépassent jamais leurs seuils. La somme des distances des dérives de la caméra et des robots n'est donc jamais assez grande pour dépasser la distance décrite par $|\delta^-|$. On est assuré ainsi de ne jamais perdre les positions réelles lors des déplacements de la caméra. La caméra peut toujours retrouver les robots dans son champ de vue, ce qui permet le succès des suivis de trajectoire des robots durant la simulation, et ce grâce au choix de l'horizon de prédiction approprié. On peut visualiser cette simulation dans la seconde partie de la vidéo `https://youtu.be/YjrhjtorqEU`.

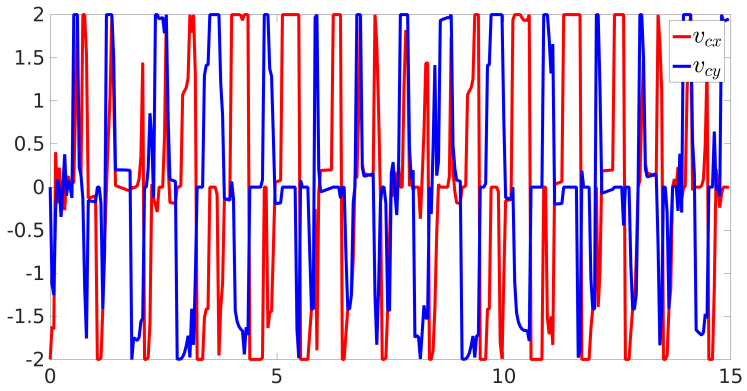
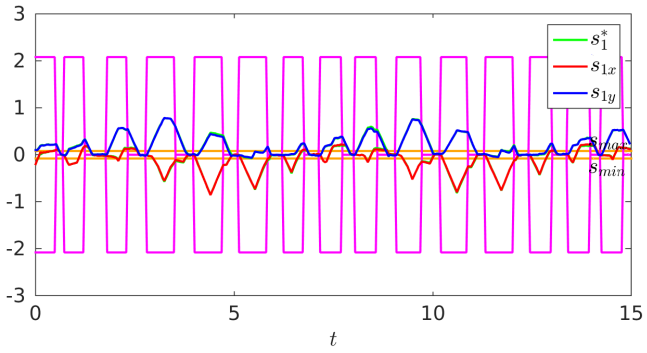
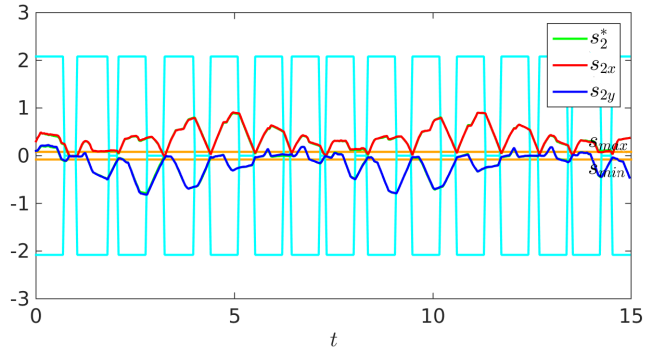


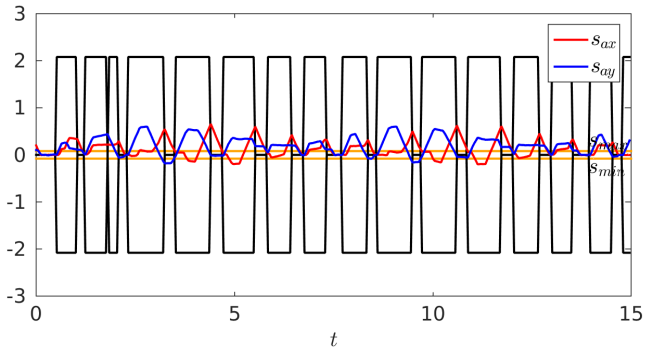
FIGURE 3.8 – Grand horizon de prédiction : Évolution des vitesses appliquées à la caméra.



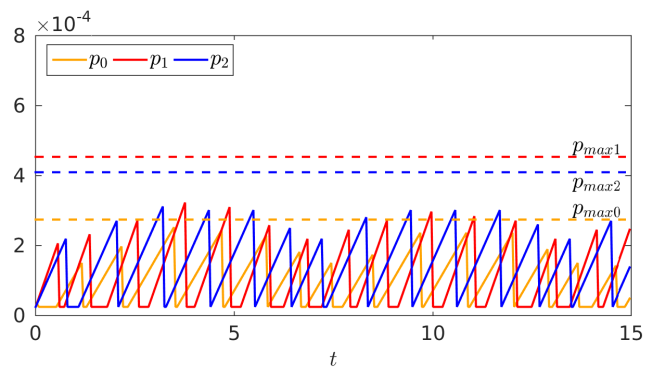
(a)



(b)



(c)



(d)

FIGURE 3.5 - Grand horizon de prédiction : Comparé au cas où $N_p = 1$, on constate une augmentation de la fréquence de modification des valeurs de tolérance, et ce grâce à un horizon de prédiction plus grand. Un plus grand nombre de correction des modèles est ainsi réalisé par la caméra, ce qui lui permet d'assister indéfiniment le suivi de trajectoire des robots.

3.7 Conclusion

Nous avons introduit une nouvelle méthode de contrôle de la caméra pour accomplir une succession de tâches d'asservissement visuel en suivant le principe de la perception active. Cette méthode est basée sur la commande prédictive pour anticiper les erreurs de modèles sur un horizon de prédiction futur et, à l'aide des contraintes, forcer la caméra à se déplacer pour corriger ces erreurs. Cette méthode a été utilisée pour assister plusieurs robots, sujets à des bruits, à suivre des trajectoires. Avec les résultats de simulation reportés, nous avons montré l'efficacité de notre méthode et l'importance de l'horizon de prédiction dans la configuration des paramètres de la commande.

Les travaux présentés dans ce chapitre sont intimement liés aux travaux présentées dans le chapitre précédent dans le sens où nous proposons une méthode d'asservissement visuel qui, au lieu de les éviter, tolère les pertes d'informations visuelles. La différence majeur ici est que l'on provoque de manière volontaire ces pertes d'informations en fonction de leurs degrés d'importance pour la réalisation de la tâche désirée.

La méthode proposée fait intervenir des priorités sur les tâches de correction selon des critères imposées par l'application. Ici, les suivis de trajectoire nous imposent de gérer la priorité selon les incertitudes de positions. Mais d'autres critères peuvent intervenir. Par exemple, dans une application où les robots ont pour objectif d'atteindre une position désirée constante dans le temps, il est possible de prendre en compte l'erreur entre la pose courante des robots et leurs poses désirées pour attribuer une plus grande priorité de correction au robot qui se trouve être le plus proche de sa position désirée. Ces critères peuvent changer d'une application à une autre mais l'idée d'avoir des distances tolérées entre la pose d'un modèle et le champ de vue de la caméra peut rester inchangé et permet d'envisager un grand nombre d'application différentes faisant intervenir une caméra et plusieurs éléments à observer lorsque c'est nécessaire.

Conclusion

L'étude présentée dans ce document s'est concentrée sur la vision en robotique, et plus particulièrement sur l'asservissement visuel qui a pour objectif de permettre à un robot de se déplacer de manière autonome en utilisant des primitives visuelles extraites d'une image capturée par une caméra. Nous avons vu que cette loi de commande est encore confrontée à des limites, notamment lorsque les primitives visuelles sont perdues, mais qu'il est possible d'utiliser les caractéristiques de la commande prédictive basée modèle pour apporter des solutions efficaces à ces problématiques.

Contributions

L'idée qui est prédominante dans ce document est que les modèles des coordonnées des primitives visuelles peuvent être utilisées à la place de leurs mesures réelles dans une commande d'asservissement visuel.

Après avoir présenté deux méthodes de prédiction qui exploitent soit l'image soit la pose d'un objet dans le repère de la caméra, nous avons proposé une méthode inédite permettant de corriger à la fois les modèles des coordonnées des primitives visuelles dans l'image et à la fois le modèle de la pose de l'objet dans le repère de la caméra. L'intérêt de cette méthode apparaît lorsque les mesures des primitives visuelles sont indisponibles et que les méthodes de prédiction doivent être utilisées pour permettre à une tâche d'asservissement visuel de se poursuivre correctement malgré l'absence de mesures. Cette méthode de correction a été comparée à celle utilisée de manière classique dans l'asservissement visuel, qui a comme principe de ne corriger que les modèles des coordonnées des primitives visuelles dans l'image sans apporter de modification au modèle de la pose. Nous avons démontré que les méthodes de prédiction utilisées après de simples corrections dans l'image sont incapables de remplacer les mesures manquantes lors d'un asservissement visuel soumis à des occultations, alors que celles utilisées après les corrections dans l'image et de la pose permettent de se substituer de manière quasiment parfaite aux mesures de la caméra.

Une méthode inédite d'asservissement visuel basée sur la commande prédictive a également été présentée. L'intérêt de son utilisation a été démontré par le biais d'une application ou cette méthode a été exploitée pour assister plusieurs robots à effectuer des tâches de suivi de trajectoire à l'aide des informations visuelles collectées réguliè-

rement par une caméra embarquée sur un drone. Cette méthode consiste à prendre en compte à la fois la prédiction des modèles des primitives visuelles dans l'image, mais également l'évolution des incertitudes entre les modèles et les coordonnées réelles des robots. L'idée majeure de cette méthode est d'utiliser les contraintes caractérisant la commande prédictive pour forcer la caméra à se déplacer d'un point à un autre, le but étant de minimiser les erreurs qui peuvent intervenir entre les positions réelles des robots et du drone et leurs modèles respectifs.

Perspectives

Plusieurs axes d'améliorations ainsi que de domaines d'applications peuvent être envisagés à la suite des travaux présentés dans ce document.

La prédiction des modèles des coordonnées des primitives visuelles, qui permet d'accomplir une tâche d'asservissement visuel malgré la présence d'occultations, est une solution qui permet d'envisager des applications dans des environnements non maîtrisés tel que des espaces accessibles au public ou à d'autres robots autonomes. On peut imaginer par exemple une tâche simple de positionnement d'un drone embarquant une caméra orientée vers le bas par rapport à une cible tracée au sol. En adoptant les méthodes de prédiction lorsque les primitives visuelles sont perdues dans l'image, il est possible de gérer la présence de personnes ou de véhicules qui passeraient sur la cible.
En complément des expérimentations proposées dans la section 2.5, d'autres types de primitives visuelles que de simples points peuvent être soumis aux mêmes problématiques. Les méthodes de prédiction et de correction décrites ici peuvent ainsi être appliquées à des droites, des segments ou encore des ellipses. De plus, il est serait intéressant de soumettre volontairement les déplacements du robot à des bruits de commande et les mesures de la caméra à des conditions plus difficiles, afin de confronter les méthodes proposées à des cas plus problématiques que ceux qui ont étés mis en œuvre.
Finalement, d'autres méthodes de prédiction peuvent être également envisagées pour fournir des résultats qui concordent plus précisemment aux trajectoires réelles des primitives visuelles dans l'image.

La méthode permettant de forcer une caméra à observer un point plutôt qu'un autre lorsque plusieurs tâches sont accomplies simultanément ouvre des possibilités importantes, notamment dans le domaine de la robotique humanoïde. En effet, ces robots qui ont la particularité d'emprunter un grand nombre de caractéristiques humaines disposent de caméras dont les champs de vue sont limités. De ce fait, leur vue ne peut se focaliser que sur des parcelles de leurs environnements, il est donc indispensable d'une part de déterminer quelles informations sont importantes à collecter, et d'autre part de forcer leurs têtes et leurs yeux à se déplacer vers ces informations. Par exemple on peut envisager le scénario suivant : le robot Romeo représenté sur la Figure 3.10 doit saisir plusieurs éléments placés de part et d'autre d'une table puis les placer sur un plateau. En appliquant notre méthode, Romeo serait capable de déplacer ses deux mains

vers deux éléments en même temps et de concentrer son regard vers la main qui est la plus proche de sa position désirée afin que celle-ci saisisse de manière précise l'objet vers lequel elle se dirige. L'autre main qui n'est pas observée peut ainsi continuer son mouvement sans qu'aucune mesure de la caméra ne soit disponible. La commande n'est alors plus basée sur l'évolution des incertitudes des positions, comme c'est le cas présenté dans le chapitre 3, mais sur l'évolution des erreurs entre les poses courantes et désirées des mains.

FIGURE 3.10 – Le robot humanoïde Romeo.

Une application réelle mettant en jeux le système présenté dans le chapitre 3 peut être également envisagée. Plusieurs problématiques doivent ainsi être prises en compte, notamment en ce qui concerne la dynamique complexe du drone qui ne peut pas être considéré en réalité comme de simples déplacements en 2D. De plus, le temps de calcul qui est nécessaire pour la minimisation de la fonction de coût (3.14) est une donnée très importante pour l'intégration de la méthode dans une situation réelle. En ce sens, le choix de l'horizon de prédiction peut s'en retrouver impacté fortement, puisque plus sa valeur est grande plus le temps de calcul requis est conséquent. C'est pourquoi, un travail important doit être apporté au développement d'un algorithme de résolution pour le problème d'optimisation proposé, afin qu'il fournisse une solution qui soit suffisamment satisfaisante pour pouvoir réaliser l'application souhaitée, tout en étant assez rapide pour pouvoir être intégré sur un drone réel. Une autre des problématiques soulevées par l'application réelle est la méthode de statistique qui doit être développée pour calculer le plus précisemment les dérives des robots et du drone, et ainsi obtenir une évolution des incertitudes associée aux véritables erreurs entre les positions réelles et modélisées

de chaque élément intervenant dans le système.

On peut enfin envisager de combiner efficacement la méthode de correction des modèles des primitives visuelles avec la stratégie de commande prédictive permettant de réaliser plusieurs tâches successives. En effet, on a vu que lorsqu'un robot n'est pas observé par la caméra, une incertitude apparaît entre sa position réelle et le modèle correspondant. Dans une application plus complexe telle que celle présentée ci-dessus mettant en œuvre le robot Romeo, une incertitude va créer une erreur entre la pose réelle de la main qui n'est pas observée et son modèle. Lorsqu'une nouvelle observation de cette main est disponible, il devient essentiel d'appliquer la méthode de correction dans l'image et de la pose en vue d'une nouvelle perte de primitives visuelles, lorsque la caméra se focalisera sur l'autre main et que la prédiction sera utilisée.

Bibliographie

[AAT08] R. Amari, M. Alamir, and P. Tona. Unified MPC strategy for idle-speed control, vehicle start-up and gearing applied to an automated manual transmission. *IFAC Proceedings Volumes*, 41(2) :7079–7085, 2008.

[ACC10] G. Allibert, E. Courtial, and F. Chaumette. Predictive control for constrained image-based visual servoing. *IEEE Transaction on Robotics*, 26(5) :933–939, 2010.

[ACT08] G. Allibert, E. Courtial, and Y. Tour. Real-time visual predictive controller for image-based trajectory tracking of mobile robot. In *17th IFAC World Congr.*, Seoul, Korea, 2008.

[AFP13] D. J. Agravante, Chaumette F., and J. Pags. Visual servoing for the REEM humanoid robot's upper body. In *2013 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5253–5258. IEEE, 2013.

[AH93] S. Atiya and G. D. Hager. Real-time vision-based robot localization. *IEEE Transactions on Robotics and Automation*, 9(6) :785–800, 1993.

[AJS14] A. Assa and F. Janabi-Sharifi. Robust model predictive control for visual servoing. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2715–2720. IEEE, 2014.

[Baj88] R. Bajcsy. Active perception. *Proceedings of the IEEE*, 76(8) :996–1005, 1988.

[Ber56] E. C. Berkeley. *Small Robots : Report*. Berkeley Enterprises, 1956.

[BLS89] J. G. Balchen, D. Ljungquist, and S. Strand. State space model predictive control of a multi stage electro-metallurgical process. *Modeling, Identification and Control*, 10(1) :35, 1989.

[CAL96] A. Casals, J. Amat, and E. Laporte. Automatic guidance of an assistant robot in laparoscopic surgery. 1996.

[Can86] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6) :679–698, 1986.

[CDW+14] N. Cazy, C. Dune, P.-B. Wieber, P. Robuffo Giordano, and F. Chaumette. Pose error correction for visual features prediction. In *IROS*

2014 - IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, Chicago, United States, 2014.

[CH06] F. Chaumette and S. Hutchinson. Visual servo control, part I : Basic approaches. *IEEE Robotics and Automation Magazine*, 13(4) :82–90, 2006.

[CHPV04] G. Chesi, K. Hashimoto, D. Prattichizzo, and A. Vicino. Keeping features in the field of view in eye-in-hand visual servoing : a switching approach. *IEEE Transactions on Robotics*, 20(5) :908–914, 2004.

[CLR79] R. Carrera, D. Loiseau, and O. Roux. Androiden. Die Automaten von Jaquet-Droz. *Lausanne : Scriptar*, 1979.

[CR80] C. R. Cutler and B. C. Ramaker. Dynamic matrix control - a computer control algorithm. *Automatic Control Conference*, 1980.

[CWRGC15] N. Cazy, P.-B. Wieber, P. Robuffo Giordano, and F. Chaumette. Visual servoing when visual information is missing : Experimental comparison of visual feature prediction schemes. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6031–6036. IEEE, 2015.

[DC99] T. Drummond and R. Cipolla. Real-time tracking of complex structures for visual servoing. In *International Workshop on Vision Algorithms*, pages 69–84. Springer, 1999.

[DD92] D. F. DeMenthon and L. S. Davis. Model-based object pose in 25 lines of code. In *European Conference on Computer Vision*, pages 335–343. Springer, 1992.

[DD12] A. Delaborde and L. Devillers. Impact du comportement social d'un robot sur les émotions de l'utilisateur : une expérience perceptive. *Journées d'étude de la parole (JEP 2012)*, 2012.

[DG99] F. Dornaika and C. Garcia. Pose estimation using point and line correspondences. *Real-Time Imaging*, 5(3) :215–230, 1999.

[DHM+11] C. Dune, A. Herdt, E. Marchand, O. Stasse, P.-B. Wieber, and E. Yoshida. Vision based control for humanoid robots. In *IROS Workshop on Visual Control of Mobile Robots (ViCoMoR)*, pages 19–26, 2011.

[DM93] S. De Ma. Conics-based stereo, motion estimation, and pose determination. *International Journal of Computer Vision*, 10(1) :7–25, 1993.

[DMA+11] M. A. Diftler, J. S. Mehling, M. E. Abdallah, N. A. Radford, L. B. Bridgwater, A. M. Sanders, R. S. Askew, D. M. Linn, J. D. Yamokoski, F. A. Permenter, et al. Robonaut 2-the first humanoid robot in space. In *2011 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2178–2183. IEEE, 2011.

[DRLR89] M. Dhome, M. Richetin, J.-T. Laprest, and G. Rives. Determination of the attitude of 3d objects from a single perspective view. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(12) :1265–1278, 1989.

[Dru87] M. Drumheller. Mobile robot localization using sonar. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (2) :325–332, 1987.

[DSW11] D. Dimitrov, A. Sherikov, and P.-B. Wieber. A sparse model predictive control formulation for walking motion generation. In *IROS 2011 - IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2292–2299, San Francisco, United States, 2011. IEEE.

[dV38] J. de Vaucanson. *Le mécanisme du flûteur automate : présenté à Messieurs de l'Académie royale des Sciences : avec la description d'un canard artificiel et aussi celle d'une autre figure également merveilleuse, jouant du tambourin et de la flûte*. 1738.

[DYL93] M. Dhome, A. Yassine, and J.-M. Lavest. Determination of the pose of an articulated object from a single perspective view. In *BMVC*, pages 1–10, 1993.

[FBA+07] P. Falcone, F. Borrelli, J. Asgari, H. E. Tseng, and D. Hrovat. Predictive active steering control for autonomous vehicle systems. *IEEE Transactions on Control Systems Technology*, 15(3) :566–580, 2007.

[FC05] D. Folio and V. Cadenat. A controller to avoid both occlusions and obstacles during a vision-based navigation task in a cluttered environment. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 3898–3903. IEEE, 2005.

[FC08] D. Folio and V. Cadenat. Dealing with visual features loss during a vision-based task for a mobile robot. *Int. Journal of Optomechatronics*, 2(3) :185–204, 2008.

[FP02] D. A. Forsyth and J. Ponce. *Computer vision : a modern approach*. Prentice Hall Professional Technical Reference, 2002.

[FRS07] N. Franceschini, F. Ruffier, and J. Serres. A bio-inspired flying robot sheds light on insect piloting abilities. *Current Biology*, 17(4) :329–335, 2007.

[FS03] P. Favaro and S. Soatto. Seeing beyond occlusions (and other marvels of a finite lens aperture). In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–579. IEEE, 2003.

[GAMASPV05] N. García-Aracil, E. Malis, R. Aracil-Santonja, and C. Pérez-Vidal. Continuous visual servoing despite the changes of visibility in image features. *IEEE Transactions on Robotics*, 21(6) :1214–1220, 2005.

[GFH88] P. Grosdidier, B. Froisy, and M. Hammann. The idcom-m controller. In *Proceedings of the 1988 IFAC Workshop on Model Based Process Control*, pages 31–36, 1988.

[GGdM+04] R. Ginhoux, J. Gangloff, M. de Mathelin, L. Soler, M. M. A. Sanchez, and J. Marescaux. Beating heart tracking in robotic surgery using 500

hz visual servoing, model predictive control and an adaptive observer. In *2004 IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 274–279. IEEE, 2004.

[GGdM+05] R. Ginhoux, J. Gangloff, M. de Mathelin, L. Soler, M. M. A. Sanchez, and J. Marescaux. Active filtering of physiological motion in robotized surgery using predictive control. *IEEE Transactions on Robotics*, 21(1) :67–79, 2005.

[GH06] N. R. Gans and S. Hutchinson. Stable visual servoing through hybrid switched-system control. 23(3) :530–540, 2006.

[GHM08] N. Guenard, T. Hamel, and R. Mahony. A practical visual servo control for an unmanned aerial vehicle. *IEEE Transactions on Robotics*, 24(2) :331–340, 2008.

[GM86] C. E. Garcia and A. M. Morshedi. Quadratic programming solution of dynamic matrix control (qdmc). *Chemical Engineering Communications*, 46 :73–87, 1986.

[GSH+15] M. Garcia, O. Stasse, J.-B. Hayet, C. Dune, C. Esteves, and J.-P. Laumond. Vision-guided motion primitives for humanoid reactive walking : Decoupled versus coupled approaches. *The International Journal of Robotics Research*, 34(4-5) :402–419, 2015.

[GVPG03] Pierre F Gabriel, Jacques G Verly, Justus H Piater, and André Genon. The state of the art in multiple object tracking under occlusion in video sequences. In *Advanced Concepts for Intelligent Vision Systems*, pages 166–173. Citeseer, 2003.

[HAEK+14] S. Heshmati-Alamdari, A. Eqtami, G. C. Karras, D. V. Dimarogonas, and K. J. Kyriakopoulos. A self-triggered visual servoing model predictive control scheme for under-actuated underwater robotic vehicles. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3826–3831. IEEE, 2014.

[HAKE+14] S. Heshmati-Alamdari, G. K. Karavas, A. Eqtami, M. Drossakis, and K. J. Kyriakopoulos. Robustness analysis of model predictive control for constrained image-based visual servoing. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4469–4474. IEEE, 2014.

[HDE98] R. Horaud, F. Dornaika, and B. Espiau. Visually guided object grasping. *Robotics and Automation, IEEE Transactions on*, 14(4) :525–532, 1998.

[Hir93] S. Hirose. Biologically inspired robot. *Oxford University Press*, 1993.

[HJL+89] R. M. Haralick, H. Joo, C.-N. Lee, X. Zhuang, V. G. Vaidya, and M. B. Kim. Pose estimation from corresponding point data. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(6) :1426–1446, 1989.

[HNJ07] A. A. Hafez, A. K. Nelakanti, and C. V. Jawahar. Path planning approach to visual servoing with feature visibility constraints : a convex optimization based solution. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1981–1986. IEEE, 2007.

[Hon07] The honda humanoid robot asimo, technical information. In *Honda Motor Co., Ltd., Public Relations Division*, 2007.

[HZ03] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[IS09] H. Iwata and S. Sugano. Design of human symbiotic robot twendy-one. In *2009 IEEE International Conference on Robotics and Automation (ICRA)*, pages 580–586. IEEE, 2009.

[Ish04] T. Ishida. Development of a small biped entertainment robot qrio. In *International symposium on micro-nano mechatronics and human science*, pages 23–28, 2004.

[JLP+11] D. Janiszek, B. Laetitia, D. Pellier, J. Mauclair, and G.-L. Baron. De l'usage de nao (robot humanoïde) dans l'apprentissage de l'informatique. In *Sciences et technologies de l'information et de la communication en milieu éducatif : Analyse de pratiques et enjeux didactiques.*, pages 231–239. Athènes : New Technologies Editions, 2011.

[Joh] S. G. Johnson. The nlopt nonlinear-optimization package, http ://ab-initio.mit.edu/nlopt.

[JYS04] J. D. Jackson, A. J. Yezzi, and S. Soatto. Tracking deformable moving objects under severe occlusions. In *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, volume 3, pages 2990–2995. IEEE, 2004.

[KAK+97] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa. Robocup : The robot world cup initiative. In *Proceedings of the first International Conference on Autonomous Agents*, pages 340–347. ACM, 1997.

[KDPT+15] J. Koenemann, A. Del Prete, Y. Tassa, E. Todorov, O. Stasse, M. Bennewitz, and N. Mansard. Whole-body model-predictive control applied to the hrp-2 humanoid. In *IROS 2015 - IEEE-RSJ International Conference on Intelligent Robots and Systems*, pages 3346–3351. IEEE, 2015.

[KDR+98] D. Khadraoui, C. Debain, R. Rouveure, P. Martinet, P. Bonton, and J. Gallice. Vision-based control in driving assistance of agricultural vehicles. *The International Journal of Robotics Research*, 17(10) :1040–1054, 1998.

[KGM13] M. Kazemi, K. K. Gupta, and M. Mehrandezh. Randomized kinodynamic planning for robust visual servoing. *IEEE Transactions on Robotics*, 29(5) :1197–1211, 2013.

[KKK+02] K. Kaneko, F. Kanehiro, S. Kajita, K. Yokoyama, K. Akachi, T. Kawasaki, S. Ota, and T. Isozumi. Design of prototype humanoid robotics

platform for hrp. In *2002 IEEE International Conference on Robotics and Automation (ICRA)*, volume 3, pages 2431–2436. IEEE, 2002.

[KOK+74] I. Kato, S. Ohteru, H. Kobayashi, K. Shirai, and A. Uchiyama. Information-power machine with senses and limbs. In *On Theory and Practice of Robots and Manipulators*, pages 11–24. Springer, 1974.

[KST+07] S. Kim, M. Spenko, S. Trujillo, B. Heyneman, V. Mattoli, and M. R. Cutkosky. Whole body adhesion : hierarchical, directional and distributed control of adhesive forces for a climbing robot. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 1268–1273, 2007.

[Law97] J. M. Law. Puppets of nostalgia : The life, death, and rebirth of the japanese awaji ningyo tradition. *Princeton University Press*, 1997.

[LBC11] C. Lazar, A. Burlacu, and C. Copot. Predictive control architecture for visual servoing of robot manipulators. In *IFAC World Congress*, pages 9464–9469, Milano (Italy), 2011.

[LLTC01] J.-F. Lots, D. M. Lane, E. Trucco, and F. Chaumette. A 2d visual servoing for underwater vehicle station keeping. In *2001 IEEE International Conference on Robotics and Automation (ICRA)*, volume 3, pages 2767–2772. IEEE, 2001.

[Low87] D. G. Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial intelligence*, 31(3) :355–395, 1987.

[Mac96] J. W. MacArthur. Rmpct : A new robust approach to multivariable predictive control for the process industries. In *The 1996 Control Systems Conference*, pages 53–60, 1996.

[MAH+97] R. K. Mehra, J. N. Amin, K. J. Hedrick, C. Osorio, and S. Gopalasamy. Active suspension using preview information and model predictive control. In *Control Applications, 1997., Proceedings of the 1997 IEEE International Conference on*, pages 860–865. IEEE, 1997.

[MB88] P. Marquis and J. P. Broustail. Smoc, a bridge between state space and model predictive controllers : application to the automation of a hydrotreating unit. In *Proceedings of the IFAC workshop on model based process control*, volume 82, pages 37–43, 1988.

[MC02a] E. Marchand and F. Chaumette. Virtual visual servoing : a framework for real-time augmented reality. In G. Drettakis and H.-P. Seidel, editors, *EUROGRAPHICS'02 Conf. Proceeding*, volume 21(3) of *Computer Graphics Forum*, pages 289–298, Saarebrücken, Germany, 2002.

[MC02b] Y. Mezouar and F. Chaumette. Path planning for robust image-based control. *IEEE Trans. on Robotics and Automation*, 18(4) :534–549, 2002.

[MCB99] E. Malis, F. Chaumette, and S. Boudet. 2 1/2 D Visual Servoing. *IEEE Transactions on Robotics and Automation*, 15(2) :238–250, 1999.

[MCM13] A. A. Moughlbay, E. Cervera, and P. Martinet. Real-time model based visual servoing tasks on a humanoid robot. In *Intelligent Autonomous Systems 12*, pages 321–333. Springer, 2013.

[MH80] D. Marr and E. Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London B : Biological Sciences*, 207(1167) :187–217, 1980.

[MH98] E. Marchand and G. D. Hager. Dynamic sensor planning in visual servoing. In *1998 IEEE International Conference on Robotics and Automation (ICRA)*, volume 3, pages 1988–1993. IEEE, 1998.

[MKC08] R. Mebarki, A. Krupa, and F. Chaumette. Image moments-based ultrasound visual servoing. In *2008 IEEE International Conference on Robotics and Automation (ICRA)*, pages 113–119. IEEE, 2008.

[MLS+00] G. Morel, T. Liebezeit, J. Szewczyk, S. Boudet, and J. Pot. Explicit incorporation of 2d constraints in vision based control of robot manipulators. In *The Sixth International Symposium on Experimental Robotics VI*, pages 99–108. Springer-Verlag, 2000.

[MR93] H. Michel and P. Rives. Singularities in the determination of the situation of a robot effector from the perspective view of 3 points. *Rapports de recherche- INRIA*, 1993.

[MSC05] E. Marchand, F. Spindler, and F. Chaumette. ViSP for visual servoing : a generic software platform with a wide class of robot control skills. *IEEE Robotics and Automation Magazine*, 12(4) :40–52, 2005.

[MT00] P. Martinet and C. Thibaud. Automatic guided vehicles : robust controller design in image space. *Autonomous Robots*, 8(1) :25–42, 2000.

[MYF06] T. Murao, T. Yamada, and M. Fujita. Predictive visual feedback control with eye-in-hand system via stabilizing receding horizon approach. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 1758–1763. IEEE, 2006.

[Oud09] P.-Y. Oudeyer. Sur les interactions entre la robotique et les sciences de l'esprit et du comportement. *Informatique et Sciences Cognitives : influences ou confluences ?*, 2009.

[Pen55] R. Penrose. A generalized inverse for matrices. In *Mathematical proceedings of the Cambridge philosophical society*, volume 51, pages 406–413. Cambridge Univ Press, 1955.

[Pow98] M. J. D. Powell. Direct search algorithms for optimization calculations. *Acta numerica*, 7 :287–336, 1998.

[Pro53] F. Proschan. Confidence and tolerance intervals for the normal distribution. *Journal of the American Statistical Association*, 48(263) :550–564, 1953.

[Roba] Aldebaran Robotics. Romeo. `http://www.projetromeo.com`. Accessed : 2016.

[Robb] Softbank Robotics. Pepper. `https://www.ald.softbankrobotics.com/fr/cool-robots/pepper`. Accessed : 2016.

[Ros33] T. Ross. Machines that think. *Health*, 243 :248, 1933.

[Ros06] M. Rosheim. Leonardo's lost robots. *Springer Berlin Heidelberg*, 2006.

[RRTP78] J. Richalet, A. Rault, J. L. Testud, and J. Papon. Model predictive heuristic control : Applications to industrial processes. *Automatica*, 14(5) :413–428, 1978.

[SG12] T. Shen and Chesi G. Visual servoing path planning for cameras obeying the unified model. *Advanced Robotics*, 26(8–9) :843–860, 2012.

[SPDC06] M. Sauvée, P. Poignet, E. Dombre, and E. Courtial. Image based visual servoing through nonlinear model predictive control. In *International Conference on Decision and Control*, San Diego, CA, USA, 2006.

[SS04] J. F. Seara and G. Schmidt. Intelligent gaze control for vision-guided humanoid walking : methodological aspects. *Robotics and Autonomous Systems*, 48(4) :231–248, 2004.

[SS09] N. Sharkey and A. Sharkey. Electro-mechanical robots before the computer. *Proceedings of the Institution of Mechanical Engineers, Part C : Journal of Mechanical Engineering Science*, 223(1) :235–241, 2009.

[TBF05] S. Thrun, W. Burgard, and D. Fox. *Probabilistic robotics*. MIT press, 2005.

[TDCBK09] S. Trimboli, S. Di Cairano, A. Bemporad, and I. V. Kolmanovsky. Model predictive control for automotive time-delay processes : An application to air-to-fuel ratio control. *IFAC Proceedings Volumes*, 42(14) :90–95, 2009.

[TIYK85] A. Takanishi, M. Ishida, Y. Yamazaki, and I. Kato. Realization of dynamic walking by the biped walking robot wl-10rd. In *Japan Industrial Robot Assoc*, 1985.

[TK01] G. Taylor and L. Kleeman. Flexible self-calibrated visual servoing for a humanoid robot. In *Australian Conf. on Robotics and Automation (ACRA2001)*, pages 79–84, 2001.

[TPS+02] A. Tewari, J. Peabody, R. Sarle, G. Balakrishnan, A. Hemal, A. Shrivastava, and M. Menon. Technique of da vinci robot-assisted anatomic radical prostatectomy. *Urology*, 60(4) :569–572, 2002.

[VBW+09] N. Vahrenkamp, C. Böge, K. Welke, T. Asfour, J. Walter, and R. Dillmann. Visual servoing for dual arm motions on a humanoid robot. In *2009 9th IEEE-RAS International Conference on Humanoid Robots*, pages 208–214. IEEE, 2009.

[VHHP+07] T. Van Herpe, N. Haverbeke, B. Pluymers, G. Van den Berghe, and B. De Moor. The application of model predictive control to normalize glycemia of critically ill patients. In *European Control Conference (ECC)*, pages 3116–3123. IEEE, 2007.

[VWA+08] N. Vahrenkamp, S. Wieland, P. Azad, D. Gonzalez, T. Asfour, and R. Dillmann. Visual servoing for humanoid grasping and manipulation tasks. In *Humanoids 2008-8th IEEE-RAS International Conference on Humanoid Robots*, pages 406–412. IEEE, 2008.

[Wal50] W. G. Walter. mitation of life. 1950.

[WH05] A. Wills and W. P. Heath. Application of barrier function based model predictive control to an edible oil refining process. *Journal of Process Control*, 15(2) :183–200, 2005.

[YZT10] X. Yuan, C.-X. Zhao, and Z.-M. Tang. Lidar scan-matching for mobile robot localization. *Information Technology Journal*, 9(1) :27–33, 2010.

Table des figures

Résumé

On rencontre aujourd'hui la vision par ordinateur employée pour la réalisation de nombreuses applications de la robotique moderne. L'un des axes de recherche actuel qui tend à améliorer ces systèmes est centré sur la commande. L'objectif est de proposer des schémas de commande originaux permettant de lier efficacement les informations mesurées par les capteurs de vision aux actions que l'on souhaite réaliser avec les robots. C'est dans cet aspect que s'inscrit ce document en apportant de nouvelles méthodes à la commande robotique classique faisant intervenir la vision, l'asservissement visuel.
Le cas de pertes d'informations visuelles pendant la réalisation d'une tâche d'asservissement visuel est étudié. Dans ce sens, deux méthodes de prédiction qui permettent à la tâche d'être réalisée malgré ces pertes sont présentées. Puis une méthode inédite de correction est proposée. Celle-ci permet d'obtenir de meilleurs résultats de prédiction qu'une méthode classique, comme le démontrent des résultats obtenus en simulation et en condition réelle.
Enfin, dans le contexte de la réalisation de plusieurs tâches d'asservissement visuel successives, une nouvelle méthode est présentée. Celle-ci exploite les caractéristiques d'un schéma de commande utilisé depuis quelques dizaines d'années dans l'industrie et la recherche, la commande prédictive basée modèle. Des résultats obtenus en simulation proposent de visualiser les effets de cette méthode sur le comportement d'un drone qui embarque une caméra.

Abstract

The computer vision is used for the achievement of many applications of modern robotics. One of the current research topics that aims to improve these systems is focused on command. The objective consists to propose original control schemes to effectively link the information measured by the vision sensor to the actions that are to be achieved with the robots. This document is part of this look by bringing new methods to classical robotic control involving vision, the visual servoing.
The case of visual information losses during the achievement of a visual servoing task is studied. In this sense, two prediction methods that allow the task to be achieved despite these losses are presented. Then a new method of correction is proposed. This provides better prediction results than a conventional method, as shown by the results obtained in simulation and in real conditions.
Finally, in the context of the achievement of several successive visual servoing tasks, a new method is presented. This exploits the characteristics of a control scheme used for several decades in industry and research, model based predictive control. The results obtained in simulation propose to see the effect of this method on the behavior of a drone that embeds a camera.