Table des matières

Chapitre 1

Introduction

1.1 Les défis actuels de l'agriculture

Avec l'évolution de notre mode de vie, la population mondiale ne cesse de croître. Elle atteindra les 9 milliards d'individus d'ici 2050 d'après les études menées par [Bergerman et al., 2016], [Foley et al., 2011] et [Steensland and Zeigler, 2017]. D'après les travaux présentés dans [Grift et al., 2008], [Bommarco et al., 2013] et [FAO, 2019], l'accroissement de la population a engendré une augmentation de la demande en nourriture, mais aussi de celle en fibre et bio-fuel, que ce soit en termes quantitatifs ou qualitatifs. Pour répondre à ces besoins, la production agricole actuelle [1] a besoin d'être doublée.

Cette problématique ayant été identifiée il y a quelques décennies, la politique agricole menée depuis les années 60 a eu pour objectif d'accroître la productivité [Pretty, 2008]. Les stratégies appliquées ont principalement porté sur : l'extension de 11% des terres cultivables, le doublement du nombre des machines agricoles, l'intensification de l'irrigation, et la multiplication par quatre de l'utilisation des ressources extérieures telles que les fertilisants et pesticides. L'agriculture s'est ainsi transformée en la bio-industrie que nous connaissons aujourd'hui, permettant d'augmenter la production de 145% [FAO, 2019].

Cependant, il s'est avéré que ce mode d'agriculture est nocif pour l'environnement [FAO, 2019], [Steensland and Zeigler, 2017] et [Pretty, 2008]. Convertir davantage d'espaces en terres arables implique la transformation de forêts en champs ou en pâturages, ce qui dérègle l'écosystème. Étendre les systèmes d'irrigation nécessaires pour lutter contre la sécheresse de certaines régions et pour multiplier les saisons de récolte mène à l'appauvrissement des réserves existantes en eau douce. De plus, la qualité des sols et de l'environnement est menacée par la pollution provoquée par l'emploi de fertilisants et pesticides, par l'érosion due aux labours intensifs, par le déséquilibre nutritionnel causé par la monoculture, etc... En parallèle des problèmes environnementaux, le développement de l'agriculture doit relever des défis concernant la main d'œuvre. En effet, de récentes études [Bergerman et al., 2016], [Taylor et al., 2012], [Rye and Scott, 2018] et [Vougioukas, 2018] ont montré que durant la période de récolte, la production agriculture nécessite une main d'œuvre sept fois plus importante que durant les autres saisons. Cette tendance est particulièrement marquée dans les domaines où les fenêtres de récolte sont très courtes comme celles des fruits et légumes. Dans le même temps, la main d'œuvre qualifiée se fait de plus en plus rare suite aux migrations rurales, aux bas salaires proposés [Taylor et al., 2012], [Rye and Scott, 2018] ainsi qu'à la pénibilité

1. Le terme "agriculture" utilisé fait référence à la définition du dictionnaire Merriam-Webster (https://www.merriam-webster.com/dictionary/agriculture)

du travail à effectuer (apparition de Troubles Musculo-Squelettiques (TMS)) [Osborne et al., 2012]. Par conséquent, l'augmentation de la production agricole doit de plus en plus tenir compte des défis que représentent la préservation de l'environnement et l'amélioration des conditions de travail. Des initiatives dans ce sens ont déjà été prises, comme en France par exemple, où le Grenelle de l'environnement prévoit de réduire de 25% le recours aux produits phytosanitaires d'ici 2020 [Ministère de la Transition écologique et solidaire, 2018].

Pour répondre sur le long terme à ces défis, une solution consiste à modifier les pratiques agricoles actuelles. L'agro-écologie représente une des pistes les plus intéressante et prometteuse [Steensland and Zeigler, 2017]. Il s'agit d'une discipline utilisant la bio-diversification dans et entre les secteurs agricoles (production de céréales, légumes, fruits, bois, animaux, poissons, etc ...) pour tirer avantage des interactions entre espèces, variétés et races. Elle permet par exemple d'améliorer les processus de pollinisation, de mieux réguler les nuisibles et les maladies, de préserver les sols et l'eau, d'améliorer la qualité de l'air et de mieux faire face au changement climatique [FAO, 2019]. Cependant, sa mise en œuvre dans le système agricole intensif actuel s'avère un véritable défi. En effet, tout d'abord, la monoculture est utilisée dans la grande majorité des exploitations, ce qui nécessite une restructuration complète du mode de production. Ensuite, cette bio-diversification nécessite des recherches poussées sur les interactions entre les différents éléments de l'écosystème (plantes, animaux, ...) [Gée C, 2015]. Enfin, des développements technologiques doivent être réalisés pour planter, fertiliser, traiter et récolter des champs comportant plusieurs cultures.

Dans ce contexte, l'agriculture de précision est amenée à se développer [Gée C, 2015]. Il s'agit d'un ensemble de techniques et de pratiques limitant l'utilisation d'intrants dans les parcelles agricoles. Elle consiste à utiliser de manière coordonnée les nouvelles technologies numériques (capteurs, drones, robots, agriculture connectée, etc.) pour ajuster la dose d'intrants en quantité, en temps et en lieu dans les exploitations agricoles [INRA, 2019]. Elle permet donc entre autres de gérer de manière précise et individuelle les traitements à apporter à chaque parcelle, ouvrant la voie à la bio-diversification des cultures. Elle limite également l'emploi de pesticides ou fertilisants, favorisant la protection de l'environnement [Gée C, 2015]. Enfin, elle peut répondre aussi aux enjeux sociétaux évoqués plus haut à travers l'automatisation (ou semi-automatisation) de certaines tâches spécifiques, pour augmenter le confort, réduire le risque d'accident et favoriser la création de postes plus qualifiés et mieux rémunérés.

Parmi les techniques les plus prometteuses mises en avant dans l'agriculture de précision, la robotique apparaît comme une solution efficiente [Grift et al., 2008], [Vougioukas, 2018], [Bergerman et al., 2016] et [Blackmore, 2009]. En effet, elle permet par exemple de superviser les besoins d'une exploitation ou les besoins individuels des plantes et animaux, de les analyser et d'y répondre de façon optimale par l'application d'une action locale et précise. En conséquence, l'utilisation de robots effectuant des tâches agricoles permettrait d'augmenter le rendement de la production existante tout en limitant son impact sur la planète et les personnes y travaillant.

1.2 La robotique dans l'agriculture

Dans l'optique de répondre aux problématiques posées précédemment, la robotique agricole suscite l'intérêt des chercheurs depuis de nombreuses années. Le degré d'autonomie et de complexité des solutions proposées dépend fortement de l'environnement dans lequel le robot évolue. Ces environnements sont très variés : certains sont forte-

ment structurés par l'homme (serres, exploitations, etc.) ; d'autres sont façonnés par l'homme mais évoluent de manière naturelle, ce qui en fait des milieux semi-structurés (champs ouverts, vergers, etc.) ; d'autres enfin ne connaissent qu'une faible intervention de l'homme et sont donc peu structurés (forêts). Chacun de ces environnements présente donc des caractéristiques particulières, créant des défis spécifiques et conduisant à des solutions adaptées.

Les premiers environnements cités sont caractérisés par un espace réduit et bien connu, ce qui leur confère une structure forte. Souvent, ces environnements présentent de fortes similitudes avec le contexte industriel. En effet, les tâches à effectuer sont simples et demandent une autonomie réduite mais fiable. De plus, l'espace est restreint et statique. La valeur marchande de la production est forte. Tous ces facteurs expliquent l'instrumentation importante de ce milieu. C'est pourquoi de nombreuses solutions robotisées sont désormais disponibles sur le marché agricole. Par exemple, dans les fermes animalières, des robots permettent d'assurer des tâches répétitives comme l'approvisionnement en fourrage, en eau, etc. Ainsi, les robots Vector [Ruizenaar and Smit, 2017] et Juno [Van Kuilenburg, 2015] commercialisés par l'entreprise Lely, naviguent dans les allées pour fournir automatiquement le fourrage et le repousser si besoin. Le défi est donc la navigation autonome dans un environnement bien maîtrisé et la prise de décision pour l'entretien des animaux. Cependant il est aussi nécessaire de doter le robot de capacités d'action pour réaliser certaines opérations. Ainsi, certains systèmes robotisés effectuent la traite des vaches [van der Lely, 1998], la tonte des moutons [Trevelyan, 1989] ou la récolte des œufs [Cronin et al., 2008]. Enfin, l'instrumentation présente dans ces environnements permet le développement de systèmes de vision par ordinateur dédiés à la surveillance et l'analyse des besoins [Dunn et al., 2003] [Kanjilal et al., 2014] ou de l'état de santé des animaux [Zhu et al., 2010]. Malgré un contexte industriel moins marqué, les serres présentent de fortes similitudes avec les environnements évoqués plus haut en termes de taille, de structure et d'instrumentation. Dans ce milieu particulier, les cultures ont généralement une haute valeur marchande et nécessitent un traitement individuel adapté. De fait, les tâches de navigation et de manipulation requises doivent être exécutées de manière précise et suffisamment robuste par rapport aux conditions environnementales. En outre, l'environnement étant intérieur, les techniques de localisation dépendent de capteurs relatifs (par opposition aux systèmes absolus comme le Global Navigation Satellite System (GNSS)) et de l'instrumentation de l'environnement. Cependant, par souci de coût et de généricité, il y a un souhait de plus en plus marqué de s'affranchir de ces contraintes liées à l'installation de matériel destiné à aider la navigation. Ainsi, [Sammons et al., 2005] développe un robot naviguant dans les serres en autonomie, se servant des tuyaux de chauffage comme rails de guidage le long des rangées et pulvérisant des traitements sur les plants de tomates de façon ciblée. [Ohi et al., 2018] effectue une cartographie de la serre et des plantes durant la navigation. De plus, un bras robotique est installé sur le robot afin de polliniser les plants de mûres, de cassis et de framboises.

A l'opposé de ces milieux très maîtrisés, d'autres environnements ne sont que très peu, voire pas du tout structurés par l'homme. Ils sont de fait plus difficiles à automatiser de par leur haut degré de complexité. C'est notamment le cas des milieux naturels comme les forêts. Dans un tel environnement, plusieurs problèmes se posent. D'abord, son aspect très varié et complexe ne permet pas d'établir une cartographie précise des lieux. Ensuite, le terrain peut être accidenté, compliquant le mouvement du robot et pouvant nécessiter la détection, le choix et le suivi des chemins les mieux adaptés. Enfin, la canopée, les troncs, les branches et les rochers perturbent fortement la réception

des signaux satellites, engendrant des erreurs de localisation pouvant aller jusqu'à 70 mètres selon [Jianyang Zheng et al., 2005]. Naviguer en autonomie complète dans cet environnement apparaît alors extrêmement compliqué et revient à explorer un milieu inconnu. De ce fait, à ce jour, très peu de systèmes autonomes ou semi-autonomes ont été commercialisés pour cet environnement [Bergerman et al., 2016]. Cependant, ce domaine a fait l'objet de quelques travaux de recherches. En termes de navigation, [Rossmann et al., 2009] et [Robmann et al., 2009] s'affranchissent de la technologie GNSS grâce à l'utilisation de scanners laser couplés à des méthodes de filtrage. Il s'agit alors de créer une carte de l'environnement dans lequel le robot peut se situer. D'un point de vue manipulation, la recherche porte actuellement sur la semi-automatisation des grues de levage, afin d'en améliorer le contrôle, d'optimiser les trajectoires et de faciliter la communication entre l'opérateur et la machine [Mettin et al., 2009] et [Hansson and Servin, 2010].

Au confluent de ces deux types de milieux, des environnements naturels sont semi-structurés. Aménagés par l'homme, ils ne sont connus que partiellement. Ces environnements sont en général extérieurs et faiblement (voire pas du tout) instrumentés (champs ouverts, vergers) du fait de la plus faible valeur marchande des cultures au mètre carré, et de l'espace beaucoup plus vaste à couvrir. Ci-après est proposée une brève présentation de chacun de ces environnements. Les champs ouverts couvrent deux milieux distincts, les cultures céréalières et maraîchères. Dans les deux cas, la technologie GNSS est utilisable. Les cultures céréalières qui ont une valeur marchande faible sont caractérisées par un milieu peu structuré, constitué de sols plats, sans obstacle, le rendant facilement navigable. Par conséquent, les contraintes imposées par cet environnement ne sont pas très fortes, d'autant que la technologie GNSS facilite le processus de localisation. C'est pourquoi, des solutions commerciales efficaces comme les tracteurs autonomes sont déjà présentes sur le marché. Ces machines intègrent dans leur stratégie de navigation des méthodes de planification optimale telles que celles développées dans [Taïx et al., 2006], [Hofstee et al., 2009], [Oksanen and Visala, 2009] et [J. Jin and L. Tang, 2010]. Des activités de recherche sont également menées pour modéliser les perturbations dues au terrain et les compenser afin d'améliorer l'exécution de tâches comme le labourage dans [Katupitiya, 2014] et [Eaton et al., 2008]. D'autres encore traitent le sujet de coordination entre robots de façon à développer des flottes de robots autonomes [Emmi, 2014].

Les cultures maraîchères sont quant à elles structurées par la création de rangées de plantations. Les plants ont une valeur marchande moyenne et requièrent des traitements individuels et précis. Il est alors nécessaire de réaliser les tâches de navigation et de manipulation de manière robuste vis-à-vis des perturbations engendrées par l'environnement extérieur. C'est pourquoi certains champs peuvent être légèrement instrumentés (marqueurs de fin de rangées, etc.). L'un des défis consiste notamment à améliorer la navigation autonome afin de s'affranchir de cette contrainte. Des travaux de recherches ont été développés dans ce sens. Par exemple, [Emmi et al., 2019] construit un modèle de l'environnement et améliore la perception du robot désherbeur Oz commercialisé par Naio technologies pour limiter l'instrumentation du champ. Un autre défi lié à ce type de culture est le développement de méthodes robustes pour la manipulation afin d'effectuer diverses tâches comme l'analyse, le traitement, ou encore la récolte. Plusieurs travaux ont été menés pour répondre à cette problématique comme par exemple [Longo and Muscato, 2013] qui propose deux solutions basées sur l'utilisation de deux bras manipulateurs couplés à un système de vision afin de récolter des artichauts.

Enfin, de la même manière que les cultures maraîchères, les vergers sont structurés

par la présence de rangées d'arbres. Ce type d'environnement induit de nombreuses contraintes qu'il est nécessaire de prendre en compte pour réaliser efficacement les tâches de navigation et de manipulation requises pour leur exploitation. Ces contraintes sont détaillées au sein d'une section spécifique puisque cet environnement est au cœur de nos travaux.

1.3 La robotique dans les vergers : Contexte et contraintes

Le sujet de cette thèse s'inscrit dans le cadre du contrôle d'un robot mobile muni de plusieurs bras manipulateurs servant à effectuer des opérations de phénotypages, d'application de traitements ciblés, de taille de branches, d'éclaircies des fruits ou encore de récoltes dans les vergers. Pour mener à bien ces opérations, le robot a besoin d'une part de naviguer en autonomie dans les parcelles et d'autre part d'effectuer des tâches de manipulation afin d'atteindre les cibles définies par le type d'opération en cours (feuilles, branches, fruits, ...). Les problématiques concernant la robotique dans les vergers se situent à mi-chemin entre celles s'appliquant aux champs ouverts et celles relatives à la sylviculture. En effet, les processus à automatiser sont similaires à ceux effectués dans les champs ouverts (i.e. surveiller/analyser, planter/reproduire, traiter, récolter, trier, conditionner, etc.). Bien que les vergers soient des environnements naturels, ceux-ci sont façonnés par les agriculteurs. Comme pour les champs ouverts, certains paramètres peuvent être connus a priori. Par exemple, un verger est structuré en une succession de rangées parallèles ; les arbres peuvent être taillés de façon à obtenir des configurations connues (espalier, haute tige, treillis, etc.) ; ou encore la distance entre les arbres peut être considérée comme régulière et dépendant du type de culture comme le montre la figure 1.1.

(a) Verger hautes tiges (b) Verger en espalier (c) Verger treillis en Y

FIGURE 1.1 – Les vergers et leurs structures

Cependant, les contraintes s'appliquant à la sylviculture se retrouvent aussi dans les vergers. Pour les mêmes raisons, la localisation par GNSS ne peut pas être prise en compte. La situation du robot doit donc être estimée localement en utilisant des capteurs embarqués tels que des caméras, scanners Light Detection And Ranging (LiDAR), etc... De même, l'environnement étant extérieur, des contraintes comme l'ensoleillement peuvent perturber les mesures fournies par les caméras et doivent donc être prises en compte. Les technologies dans ce domaine ont beaucoup évolué ces dernières années permettant une mesure plus robuste en extérieur, d'autant qu'elle se fait sous la canopée. La dynamique d'un verger est aussi une contrainte à ne pas négliger. Celle-ci représente le changement naturel de l'environnement et les modifications dues aux opérations effectuées. Par exemple, d'une année sur l'autre, des arbres peuvent

être coupés, d'autres peuvent être plantés ; au cours des saisons, les feuilles tombent et de nouvelles poussent ; les fruits grossissent, prennent du poids, sont récoltés, ce qui modifie la position des branches et des fruits qui y restent attachés. De plus, la navigation peut être perturbée par la nature du terrain. En effet, celui-ci n'est pas toujours plat ; le passage du robot ou d'autres machines agricoles peut marquer le sol, le rendant ainsi inégal ; la pluie peut aussi engendrer un terrain glissant ou boueux, générant des difficultés de navigation. Enfin, l'environnement peut être partagé avec des opérateurs humains, des animaux ou d'autres robots. Le véhicule doit donc pouvoir travailler en sécurité et s'adapter à des situations imprévues comme des obstacles, fixes ou en mouvement. Il devient alors nécessaire de prendre en compte ces contraintes qui modifient l'environnement du robot au cours du temps.

Ces contraintes ont déjà été considérées dans le cadre de recherches scientifiques. En effet, les approches permettant de s'affranchir des systèmes de localisation absolue comme le GNSS ont fait l'objet de nombreuses publications. Ainsi, par exemple, [Bergerman et al., 2015, Zhang et al., 2014, Bayar et al., 2015] présentent une stratégie de navigation complète basée sur l'utilisation d'un scanner laser et d'encodeurs odométriques. Cependant, elle nécessite une instrumentation de l'environnement avec l'installation préalable de marqueurs en fin de rang et la construction d'une carte. [Sharifi and XiaoQi Chen, 2015] propose quant à lui une technique utilisant une caméra pour déterminer le milieu de la rangée d'arbres de façon à naviguer le long de celle-ci. Néanmoins, la solution proposée repose uniquement sur l'utilisation de caméras, peu robustes aux changements d'illumination. Dans [Subramanian et al., 2006] deux méthodes de détection de rangées de citronniers pour la navigation autonome sont comparées. L'une se base sur les informations reçues par un radar, l'autre sur les données visuelles perçues par une caméra. Ces deux approches ont conduit à de bons résultats, et les auteurs proposent de les associer pour améliorer la robustesse de la navigation et d'intégrer des fonctionnalités d'évitement d'obstacles. Pour s'affranchir du changement éventuel de la structure du champ, [Barawid Jr et al., 2007, Ryo et al., 2004] proposent d'estimer la ligne droite représentant au mieux la rangée d'arbres grâce aux informations extraites d'un scan laser. De même, [Vougioukas, 2007] propose une solution optimale pour suivre un chemin prédéfini tout en évitant d'éventuels obstacles imprévus.

Les contraintes liées à la dynamique de l'environnement sont aussi prises en compte dans l'exécution de tâches de manipulation. En effet, [Rosa et al., 2008] ont développé un système robotique pour faire vibrer la branche souhaitée afin de faire tomber une partie de ses fruits. Ce système est muni de caméras permettant d'analyser la distribution des fruits ainsi récoltés. Il est néanmoins limité car d'une part il n'est pas totalement autonome et d'autre part les fruits peuvent s'abîmer en cas de chocs avec les branches. [Baeten et al., 2008, De-An et al., 2011] montrent la faisabilité d'un système de récolte automatique basé sur un bras manipulateur équipé d'une caméra. Celle-ci permet d'identifier et de localiser les pommes à cueillir. Le robot est ensuite commandé par asservissement visuel pour garder les pommes à l'endroit adéquat dans l'image afin de les saisir. Cependant, des améliorations sont encore à apporter quant à la préhension nécessaire pour cueillir le fruit. De plus, la vitesse et la précision de l'exécution de la tâche face à une cible bloquée par un obstacle ou en mouvement doivent encore être améliorées. Enfin, [Zion et al., 2014] montre que pour être économiquement viable, les solutions développées doivent présenter un meilleur taux de productivité. Ainsi, l'auteur propose de réaliser un système robotique composé de plusieurs bras parallèles afin d'augmenter le taux de ramassage de melons. Néanmoins, la production pourrait être

encore optimisée par la collaboration de plusieurs bras qui pourraient alors partager leur espace de travail.

Ainsi, bien que d'importants progrès aient été effectués, les robots agricoles évoluant dans les vergers ne répondent pas encore aux besoins des agriculteurs. Une des premières raisons est la difficulté à se localiser pour le système robotique. En effet, les vergers sont des environnements naturels avec une canopée plus ou moins fournie et le signal GNSS est donc souvent perturbé, voire absent. De plus, il s'agit d'un environnement dynamique et évoluant au cours des jours. C'est pourquoi, les approches traditionnellement utilisées en robotique mobile, localisation par GNSS et localisation dans une carte métrique, se révèlent peu efficaces pour gérer ce type de situation. En revanche, il paraît intéressant de considérer les stratégies de commande dites référencées capteurs au sens où elles exploitent les données sensorielles issues de capteurs extéroceptifs directement dans la loi de commande. Elles permettent un contrôle plus réactif et sont connues pour être robustes aux différentes sources d'imprécisions (mesures, modèles, environnement, etc.). La seconde raison concerne la productivité encore trop réduite de la plupart des systèmes actuels. Dans ce contexte, un système robotisé équipé de plusieurs bras manipulateurs pourrait permettre de répondre à ce problème. Pour cela, il est nécessaire de développer des stratégies de commande réactive, permettant d'éviter des obstacles tels que les branches et gérant le fait que les bras partagent un même espace de travail.

Cette thèse s'intéresse donc à la commande référencée capteur d'un système mobile multi-bras capable d'une part, de naviguer dans un verger de manière autonome et d'autre part, de positionner les organes terminaux de plusieurs bras manipulateurs partageant un même espace de travail. Nos contributions sont les suivantes :

- Synthèse d'une stratégie de navigation complète comprenant :
  - le développement de lois de commande réactives, basées sur des capteurs extéroceptifs embarqués, pour l'exécution de demi-tours d'une rangée vers une autre ;
  - le développement d'un algorithme de détection des arbres, robuste aux variations de l'environnement, pour réduire l'instrumentation du verger et renforcer l'autonomie du système ;
- Synthèse d'une stratégie de commande référencée vision permettant de contrôler simultanément les mouvements du système multi-bras pour réaliser des tâches de positionnement de manière réactive et sûre. Pour cela, différentes contraintes seront prises en compte : partage de l'espace de travail, limitations physiques en termes de position et de vitesse, collisions avec l'environnement, perte du signal visuel, etc.

Ces deux contributions font l'objet des chapitres 2 et 3 qui sont respectivement consacrés au développement d'une architecture pour la navigation autonome dans les vergers et à la construction d'une stratégie de commande permettant le positionnement d'un système multi-bras partageant l'espace de travail. Les résultats obtenus par simulation et expérimentation ont montré l'intérêt des différentes solutions élaborées. Enfin le chapitre 4 propose un bilan des travaux effectués et dégage les pistes les plus prometteuses pour les travaux futurs.

Chapitre 2

Navigation autonome

2.1 Introduction

Un robot naviguant de façon autonome dans un verger doit être capable de se déplacer dans les rangées d'arbres afin d'y effectuer des tâches d'analyses, de traitements, de récoltes, etc. Pour entrer dans la rangée suivante, il doit aussi pouvoir exécuter un demi-tour. Ces deux actions sont alors répétées jusqu'à la couverture totale de la parcelle. Afin de concevoir un tel système de navigation, il est nécessaire de relever les défis que représente un environnement naturel extérieur, ainsi que de garantir la sécurité du robot et des acteurs présents dans ses alentours (hommes, animaux, ...).

2.1.1 Stratégie générale pour la navigation

Une architecture de navigation autonome peut se décomposer en processus interagissant entre eux, tel que cela est décrit dans les travaux de [Siegwart and Nourbakhsh, 2004], [Latombe, 2012] ou encore [LaValle, 2006]. Il est aussi possible d'utiliser la décomposition proposée dans [Durand-Petiteville, 2012] qui repose sur six processus : perception, modélisation, localisation, planification, action et décision.

1. **Perception** : La perception a pour objectif de donner au robot la capacité d'acquérir des informations nécessaires à la réalisation de la navigation. Ces acquisitions se font à partir de capteurs qui mesurent et extraient les informations jugées intéressantes. Ces capteurs peuvent être classés par catégories. Il en existe deux types : les capteurs proprioceptifs et les capteurs extéroceptifs. Les capteurs proprioceptifs mesurent les propriétés internes du système (encodeurs, gyroscopes, etc). Les capteurs extéroceptifs renseignent le robot sur son environnement extérieur au travers de capteurs tels que des scanners lasers, des caméras, des télémètres ultra-son, des bumpers, etc.

   Il est important de noter que tous les capteurs sont sujets à des erreurs de mesure. Selon le capteur utilisé une probabilité d'incertitude peut être connue.
2. **Modélisation** : Pour mener à bien sa tâche de navigation, un robot peut utiliser un modèle représentant l'environnement. Ce modèle est construit initialement à partir de connaissances a priori, puis enrichi au cours de la navigation grâce aux informations venant de la perception. Deux types de modélisations existent : les cartes métriques et les cartes topologiques.

   Les cartes métriques sont des représentations géométriques continues ou discrètes de l'environnement [Thrun, 1998]. Elles sont définies par rapport à un repère absolu associé. Cette représentation géométrique peut faire l'objet de

transformations, notamment afin de créer l'espace des configurations du robot [1]. Dans cet espace, la géométrie du robot et des obstacles est connue avec une certaine précision, formant alors les espaces libres et occupés de la scène. En cas de modifications de l'environnement, les informations acquises par la perception peuvent servir à la mise à jour de la carte. Cependant, ces données doivent préalablement être exprimées dans le repère de la carte avant d'y être ajoutées.

Les cartes topologiques sont des représentations discrètes de l'environnement. Elles sont définies sous la forme d'un graphe connexe, orienté ou non. Dans cette carte, les nœuds représentent les situations d'un ensemble de caractéristiques de l'environnement. Cet ensemble de caractéristiques est défini par l'utilisateur ainsi que les conditions d'adjacence reliant les nœuds. Les conditions d'adjacence permettent quant à elles, de représenter le passage entre deux situations, par exemple un chemin dans l'espace des configurations ou la visibilité d'un certain nombre d'objets dans la scène. La mise à jour d'une carte topologique n'est nécessaire que lorsque l'ensemble des caractéristiques d'un nœud n'est plus suffisant pour représenter la situation du robot ou que les conditions d'adjacence entre deux nœuds ont changé. De ce fait, sa précision dépend des caractéristiques associées aux nœuds. La carte topologique est donc moins sensible aux variations de l'environnement et ne nécessite, en général, pas de mise à jour régulière pouvant impacter les ressources utilisées par le robot. De plus, il est possible d'attacher des actions à un ou plusieurs nœuds afin d'adapter le comportement du robot en fonction de sa situation dans la carte.

3. **Localisation** : La localisation a pour objectif de connaître la situation du robot dans le modèle de l'environnement défini. Le type de localisation est alors directement lié au type de modélisation utilisé. Ainsi, il en existe deux types : la localisation métrique et la localisation topologique.

   La localisation métrique permet de déterminer la situation du robot par rapport à un repère dans une carte métrique. Cette situation est définie en utilisant les informations venant de la perception. Cependant, l'utilisation seule de capteurs proprioceptifs peut induire d'importantes erreurs. En effet, dans ce cas, l'estimation de la localisation se base sur une intégration afin de calculer la situation du robot. Une erreur de dérive peut alors apparaître au cours du temps. De plus, pour calculer cette situation, un modèle du robot est utilisé. Une erreur sur le modèle impactera alors directement la précision de la localisation. Enfin, un capteur proprioceptif ne prend pas en compte l'environnement extérieur agissant sur le robot. Il est donc incapable de gérer les erreurs engendrées par l'environnement telles que le glissement des roues sur le terrain. Pour faire face à ce type de problème et améliorer la localisation, le couplage de capteurs proprioceptifs avec des capteurs extéroceptifs est une solution couramment utilisée par le biais par exemple d'un filtre de Kalman [Grewal, 2011].

   La localisation topologique a pour objectif de connaître la situation du robot dans le graphe modélisant l'environnement. Cette localisation associe les données issues de la perception, les conditions d'adjacence et la situation précédente dans ce même graphe pour actualiser la situation dans la carte. La localisation topologique est donc moins sensible aux erreurs de mesures que la localisation métrique. Cependant sa précision est directement liée aux paramètres utilisés pour décrire l'environnement.

1. Espace caractérisé par l'état d'un système.

4. **Planification** : La planification consiste à définir un itinéraire à suivre pour le robot afin d'atteindre un objectif. Pour ce faire, de multiples techniques existent. Ces techniques dépendent fortement du type de représentation choisi pour la modélisation de l'environnement. Un bon aperçu des méthodes utilisées en planification est proposé dans [Latombe, 2012] et [LaValle, 2006].
   Ainsi un itinéraire défini dans une carte métrique peut être représenté par un chemin ou une trajectoire dans un espace géométrique. L'espace géométrique peut aussi être transposé dans l'espace des configurations du robot. Ainsi le robot est représenté par un point et un chemin ou une trajectoire peut être calculé dans l'espace des configurations libre jusqu'à atteindre sa configuration finale. Lors d'une modélisation continue de l'environnement, le calcul du chemin ou de la trajectoire peut être effectué grâce à des graphes de visibilité [Turner et al., 2001] ou des diagrammes de Voronoï [Choset and Burdick, 1995]. Lorsque la représentation est discrète, la planification peut être opérée en utilisant des méthodes venant de la théorie des graphes comme l'algorithme A* [Hart et al., 1968] ou l'algorithme de Dijkstra [Dijkstra, 1971]. L'inconvénient de ce type de planification est le coût de calcul qui peut être élevé. Pour y pallier, des techniques de planification probabiliste existent comme *probabilistic roadmap* [Hsu et al., 2006] ou *rapidly exploring random tree* [LaValle et al., 2000]. Il peut arriver que le modèle de l'environnement, construit à partir de connaissances a-priori, ne soit pas complet à l'initialisation. Ainsi, lors de la mise à jour de la carte pour y ajouter de nouveaux éléments, il peut être nécessaire de replanifier un nouvel itinéraire. Une autre méthode propose de déformer l'itinéraire initial afin de prendre en compte la détection d'éventuels obstacles [Quinlan and Khatib, 1993, Keller et al., 2014]. Cependant, lors de changements majeurs dans le modèle alors une replanification globale peut être requise.

   Lorsque la modélisation est une carte topologique, l'itinéraire est typiquement représenté par une suite de situations à atteindre. Ces situations peuvent être définies dans un repère de l'environnement ou dans le repère associé à un capteur (cibles dans l'espace image). Cependant, suivant les paramètres choisis pour représenter l'environnement, certains changements peuvent ne pas être pris en compte dans le graphe, par exemple l'apparition de nouveaux obstacles. Ce cas est alors à prendre en compte dans la définition des actions à appliquer au robot.
5. **Action** :
   De façon à mener à bien des tâches de navigation, il est nécessaire de calculer les mouvements que le robot doit exécuter. Pour ce faire, différents correcteurs doivent être synthétisés. Il en existe deux types, les correcteurs à retour d'état ou les correcteurs à retour de sortie.
   Les correcteurs à retour d'état ont pour objectif d'annuler une erreur entre l'état courant du système et un état de référence correspondant à une situation finale souhaitée. À chaque instant l'état du système doit être connu et sa détermination s'effectue le plus souvent grâce à une localisation métrique.
   Les correcteurs à retour de sortie ont pour objectif d'annuler une erreur définie entre la mesure courante et la mesure de référence. Ces mesures sont acquises dans un repère relatif au robot (généralement le repère du capteur) par rapport à un élément de la scène. Ces données diffèrent suivant les capteurs utilisés. Pour des capteurs de proximité tels que des scanners lasers ou des télémètres ultra-son, les mesures sont définies par des distances. Pour des capteurs tels que des camé-

ras, ces mesures peuvent correspondre à des primitives géométriques comme des points, des segments, des droites, des ellipses ([Francois Chaumette, 2002, Hadj-Abdelkader et al., 2008]), ou encore des moments ([Chaumette, 2004, Tahri and Chaumette, 2005]). Elles peuvent aussi être de nature différente, comme des histogrammes [Bateux and Marchand, 2017] ou encore une mesure de maximum de vraisemblance entre deux images [Dame and Marchand, 2013]. Ces techniques utilisent donc données sensorielles directement dans la loi de commande. La situation du robot dans un repère de la scène n'a ainsi pas besoin d'être connue : Seule la mesure par rapport à la référence est nécessaire. Cependant cela implique que celle-ci puisse être perçue par le capteur à tout instant lors de la navigation.

6. **Décision** : Pendant la tâche de navigation, il peut s'avérer nécessaire de prendre des décisions lors de l'exécution de ces cinq processus. Ces décisions peuvent être de différentes natures et agir sur des instances de tous niveaux tels que le changement de correcteurs, le changement de ses paramètres, la mise à jour du modèle de l'environnement, la replanification de l'itinéraire, etc. Ces décisions sont généralement prises par des algorithmes de supervision prenant en compte les mesures acquises par les capteurs extéroceptifs.

2.1.2 Navigation dans les vergers : Spécificités et solutions

Un robot autonome effectuant des tâches agricoles doit être capable de naviguer sans assistance dans la parcelle à traiter. Les architectures de navigation proposées dans la littérature sont souvent organisées en trois processus : i) perception, ii) modélisation/localisation/planification, et iii) action.

Un des problèmes majeurs de la perception dans un environnement tel que les vergers est l'inefficacité des systèmes GNSS. En effet, ces systèmes sont très utilisés dans les problèmes de navigation en milieux ouverts [Bak and Jakobsen, 2004, Fang et al., 2006, Eaton et al., 2008, Johnson et al., 2009]. Dans le cadre d'une navigation dans les vergers, le robot doit se déplacer sous une canopée. Celle-ci perturbe les signaux venant des satellites impliquant alors une mauvaise précision des mesures GNSS. Le robot ne peut alors exploiter que des données locales telles que fournies par les encodeurs, par exemple. Cependant, l'environnement étant naturel, la détection via les encodeurs seuls n'est pas fiable du fait des perturbations engendrées par l'environnement extérieur sur le robot (glissement des roues sur le sol, etc.). De plus, les connaissances a priori du terrain sont insuffisantes pour se déplacer en toute sécurité sur la base de capteurs proprioceptifs seuls. C'est pourquoi les capteurs extéroceptifs (scanners laser, caméras, etc.) sont très utilisés dans la littérature pour la navigation dans les vergers et notamment pour l'exécution du suivi de rangs. [Bargoti et al., 2015] développent un algorithme permettant de détecter les troncs de pommiers à différentes saisons. Pour cela, un scanner laser 3D fournit un nuage de points géo-référencés de plusieurs rangées successives d'arbres. Puis, une classification sur la taille des pixels dans l'image est opérée pour rejeter les résultats faux-positifs. Cependant, cette détection est exécutée hors ligne et n'est donc pas adaptée à notre problème. Un autre travail [Shalal et al., 2015] propose un algorithme détectant les troncs présents dans les vergers. Cette détection se fait via la fusion de données entre une caméra et un scanner laser 2D. La caméra est utilisée dans ce cas pour trouver des régions d'intérêts, basées sur la couleur et les bordures de troncs. Toutefois dans ces travaux, la détection de troncs requiert une étape de pré-navigation pour acquérir diverses données sur le verger. De plus, leur

localisation est basée sur la mesure effectuée par une coupe laser : la méthode n'est donc pas robuste à la présence d'herbes hautes ou de branches tombantes.

D'autres techniques reposent sur l'estimation des alignements d'arbres pour extraire un chemin à suivre dans le rang. Ainsi, dans les travaux proposés par [Barawid Jr et al., 2007, Hansen et al., 2009, Bradley Hamner et al., 2011, Zhang et al., 2014, Blok et al., 2019], différentes méthodes sont évaluées pour localiser les rangées d'arbres à partir des données reçues par des scanners lasers afin d'en extraire le chemin à suivre par le robot. Des méthodes comme des transformées de Hough, des filtres de Kalman, des Random Sample Consensus (RANSAC) ou encore des filtres à particules ont été proposées et comparées. Cependant à l'instar des méthodes de détection proposées précédemment, les arbres complets sont pris en compte dans leurs estimations. Ainsi, la mesure des branches et des feuilles génèrent des bruits, des erreurs et des incertitudes dans les résultats. Dans l'étude proposée par [Subramanian et al., 2006], deux méthodes pour déterminer le chemin à suivre sont présentées. La première opère une segmentation basée sur une image acquise par une caméra RGB. Cette segmentation a pour but d'extraire les alignements gauche et droite formés par les rangées d'arbres. La deuxième méthode utilise les discontinuités de mesure d'une coupe laser 2D afin d'y estimer les deux lignes représentant les rangées d'arbres. Finalement les résultats montrent que ces deux méthodes sont d'efficacité équivalente et qu'une fusion des deux approches permettrait d'aller vers une navigation plus robuste. [Bayar et al., 2015] proposent de comparer deux solutions de navigation. La première associe une localisation par filtrage particulaire et un correcteur de type "pure pursuit". La deuxième couple une localisation fusionnant les informations d'un scanner laser et de l'odométrie avec un correcteur de suivi de chemin. Cette étude montre que ces deux méthodes parviennent à suivre les rangées d'arbres. Néanmoins, la seconde méthode montre une erreur latérale centrée en zéro. La commande à fournir au robot est donc moins brusque que celle calculée par la première méthode. Dans les travaux effectués par [Blok et al., 2019], un correcteur Proportionnel Intégral Dérivé est utilisé pour suivre le chemin préalablement extrait. Ainsi, dans le contexte de suivi de rangs, les méthodes de commande utilisées sont simples mais offrent de bons résultats pour ce type de tâche. Elles reposent sur l'extraction de lignes droites représentant les rangées d'arbres afin de calculer le chemin que le robot doit suivre. Elles se fient donc à un itinéraire construit à partir d'une localisation métrique. Ce type de localisation nécessite la prise en compte de la nature du terrain, des environnements pentus ou glissants introduisant des imprécisions tant dans la localisation que dans l'acquisition des données. Enfin, un grand nombre de ces solutions ont pour objectif la construction d'une carte métrique comme les travaux présentés dans [Zhang et al., 2014]. Ce type de cartes est difficile à maintenir à jour du fait de la nature fortement dynamique de l'environnement. En cas de changement de l'environnement, une modification du modèle, une replanification et la re-localisation du robot dans ce modèle est nécessaire. Ces actions peuvent être coûteuses en termes de temps de calcul ainsi que de stockage de données. Ces ressources ne sont pas à négliger pour exécuter des processus en temps réel sur des plateformes embarquées.

De nombreuses solutions ont ainsi été proposées pour la navigation à travers les rangées d'arbres. Concernant maintenant l'exécution des demi-tours en fin de rangée, très peu de méthodes abordent ce problème dans la littérature par manque de données mesurées dans cet espace. La plupart des solutions proposées comme [Bergerman et al., 2015, Bayar, 2013, Bochtis and Vougioukas, 2008a] exploitent les informations issues des encodeurs. Or, les erreurs provoquées par le glissement des roues sur le terrain sont particulièrement importantes lors de la navigation dans l'espace de manœuvres. Cet

espace, dans lequel les demi-tours sont exécutés en bout de rangées, est communément nommé la zone de fourrière.

L'analyse effectuée ci-dessus a permis de souligner les principaux verrous concernant la navigation autonome dans les vergers. Tout d'abord, une des contraintes majeures relevée dans ce milieu est l'imprécision, voire le blocage total des signaux GNSS dû à la canopée. La localisation via des capteurs locaux devient alors nécessaire. De plus, l'environnement étant naturel, diverses difficultés surviennent lors de la navigation. En effet, un tel milieu change au cours du temps en termes d'aspect et de mouvement. Le terrain peut alors être irrégulier et glissant. Ainsi, l'utilisation de capteurs odométriques seuls ne peut amener à une localisation précise et fiable. C'est pourquoi, les solutions proposées se concentrent essentiellement sur l'utilisation de capteurs extéroceptifs locaux (caméras, scanners laser, etc.). De nombreuses études proposent la construction de stratégies basées sur ces capteurs. Cependant, les solutions étudiées reposent sur une localisation, une modélisation et une planification métrique peu robuste aux variations de l'environnement. De plus, divers travaux utilisent des scanners laser ou des caméras afin d'extraire les rangées d'arbres, mais ceux-ci considèrent la mesure des arbres entiers (troncs, branches, feuilles), ce qui génère des imprécisions sur l'estimation des alignements. Enfin, à notre connaissance, très peu d'études se sont concentrées sur le contrôle du demi-tour en fin de rangée par manque de données mesurées.

Approche retenue

Nous portons notre attention dans ce chapitre à la résolution des différents problèmes relevés lors de l'analyse de la littérature proposé ci-dessus. Pour effectuer une navigation robuste, il nous semble important de nous attacher à la détection et la localisation précise des troncs d'arbres tout au long de la navigation. Cette localisation devrait permettre le calcul d'un chemin à suivre plus précis lors du suivi de rangées et l'utilisation du dernier tronc d'arbre comme référence lors de l'exécution du demi-tour dans la zone de fourrière. Il a été décidé pour cela, d'équiper le robot avec quatre systèmes de stéréovision, lui offrant ainsi un champ de vision supérieur à 180 degrés. De plus, afin de faire face à l'environnement naturel et dynamique que représente un verger, nous proposons de développer une stratégie de commande référencée capteurs. Pour cela, cette stratégie repose sur deux correcteurs réactifs : un premier, effectuant un suivi de ligne, permet au robot de naviguer au travers des rangées, tandis que le second, permettant de réaliser des spirales, est utilisé pour réaliser les demi-tours entre les rangées. Enfin, le verger sera modélisé par un graphe, dont les nœuds correspondent à la division de l'espace physique en fonction du correcteur devant être utilisé. Cette carte topologique permet la prise en compte uniquement de paramètres stables au cours du temps comme la présence de troncs d'arbres.

Dans les sections suivantes, nous présentons d'abord le robot utilisé pour naviguer dans les vergers, avant de décrire le type de modélisation choisie pour représenter l'environnement d'intérêt. Puis, nous nous focalisons sur la partie perception et décrivons l'algorithme qui nous permet de détecter et localiser les troncs d'arbres à partir de nuages de points fournis par les capteurs embarqués. La partie action décrit ensuite les correcteurs réactifs synthétisé pour effectuer le suivi des rangées ainsi que les demi-tours. Enfin, l'architecture complète est résumée et validée par une navigation longue distance.

2.2 Description du robot

Dans ce chapitre, le robot considéré est un *Toro workman MDE* disponible à l'université de Californie Davis (cf. figure 2.1a). Ce véhicule possède une direction avant de type *Ackermann* et évolue grâce à deux moteurs électriques alimentés par des batteries de 48V. L'un de ces moteurs est utilisé pour la propulsion, l'autre pour la direction. Ce véhicule a été modifié par l'entreprise *Sensible Machine* pour y intégrer un système de transmission par chaîne avec des réducteurs afin de l'adapter pour la navigation dans les vergers. Ainsi, deux modes de fonctionnement ont été mis en place : un mode manuel et un mode automatique. Le mode manuel permet au robot de se déplacer avec une vitesse linéaire maximale de 8 km/h. En mode automatique, la communication entre le véhicule et l'ordinateur embarqué est réalisée à travers une liaison TCP/IP. Les dimensions de ce robot sont de 3.2 m de longueur, 1.6 m de largeur et 1.95 m de hauteur, lui permettant d'entrer dans les rangées de vergers commerciaux pour la récolte de pommes, de poires, de pêches, etc. Deux catégories de capteurs sont présentes sur le robot. La première concerne les capteurs proprioceptifs. Ici, un encodeur est fixé sur l'axe arrière pour mesurer la rotation de ses roues motrices. Un autre encodeur est placé sur l'arbre de direction pour mesurer l'orientation des roues directrices. La deuxième catégorie couvre les capteurs extéroceptifs. Quatre caméras ZED stéréo commercialisées par *Stereolabs* sont ainsi installées à l'avant et sur les côtés du robot comme illustré sur la figure 2.1b. Ces caméras passives haute résolution offrent un champ de vision de 90° chacune et permettent une acquisition de profondeur jusqu'à 20m. Enfin, un système RTK-GPS *Trimble* associée avec une centrale inertielle (IMU) de *Vectornav* est installé de façon à pouvoir disposer d'une localisation précise en environnement ouvert. Ce système est utilisé uniquement pour les phases de validation des correcteurs car inutilisable dans les vergers.

(a) Toro workman MDE (b) Positionnement des caméras stéréo

FIGURE 2.1 – Présentation de la plateforme robotique

2.3 Modélisation

Cette partie s'attache à détailler l'approche retenue pour modéliser l'environnement de navigation. Lorsque l'on traite de la navigation dans un verger commercial, plusieurs points spécifiques sont à prendre en compte. Tout d'abord, un verger commercial étant modelé par la main de l'homme, il s'agit d'un environnement dont la structure générale est connue (nombres de rangées, longueur, largeur,...). En effet, comme illustré dans

la figure 2.2, il est possible de considérer que les arbres sont plantés de façon à former des rangées droites et parallèles dont les extrémités s'ouvrent sur un espace vide dédié aux manœuvres (zone de fourrière). De plus, les largeurs de rangées et l'espacement entre deux arbres consécutifs du même rang sont approximativement similaires d'une rangée à l'autre. Il existe donc un certain nombre de paramètres qui sont constants et peuvent être connus à l'avance. Mais, un verger est aussi un environnement hautement dynamique. En effet, il est impossible de connaître à l'avance la position des branches, des feuilles ou des fruits. Et pour cause, tout au long de l'année les feuilles poussent puis tombent, les fruits se développent et déforment les branches, qui elles mêmes sont taillées pour entretenir les arbres, ...

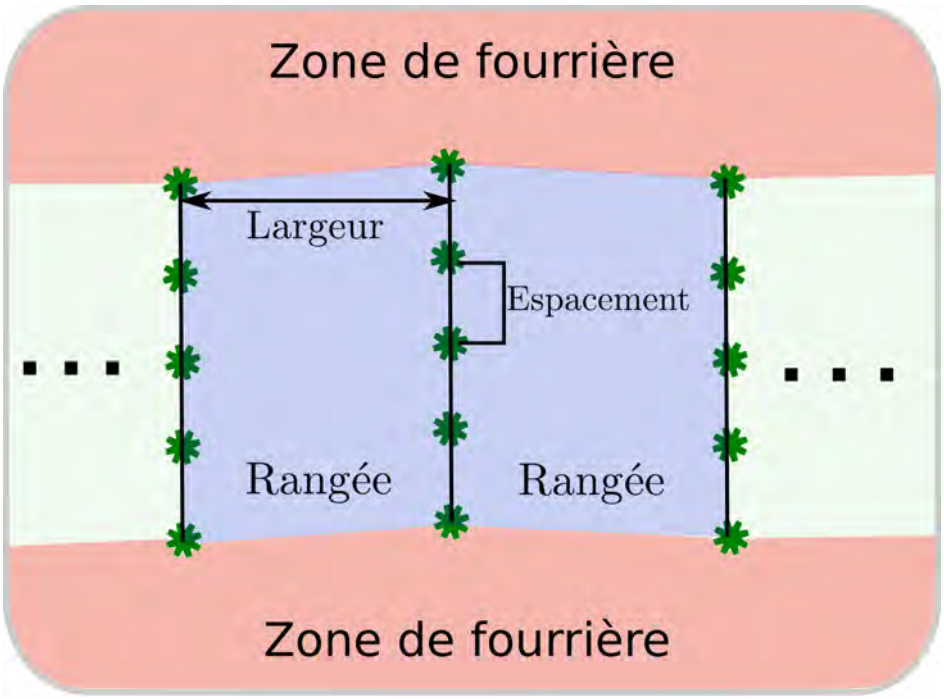


FIGURE 2.2 – Connaissances a priori sur les vergers

A la vue de cette description, les cartes métriques généralement utilisées ne paraissent pas proposer une solution adaptée pour modéliser efficacement l'environnement de navigation. En effet, il semble difficile de tenir à jour une telle carte connaissant la nature dynamique de l'environnement. Par exemple, en période de récolte, la position des branches enregistrée peut ne plus être pertinente d'une semaine sur l'autre. Si la cueillette se fait en plusieurs passages afin de récolter les fruits au meilleur moment, la position des branches doit être mise à jour à chaque fois qu'un fruit est détaché et que celle-ci se redresse libérée du poids. Un autre élément allant à l'encontre des cartes métriques est leur trop grande richesse. En effet, pour naviguer dans un verger, il n'est pas nécessaire de connaître à l'avance la position de tous les arbres, branches, fruits, ... Pour traverser une rangée, il semble suffisant d'être capable de déterminer l'espace libre et forcer le robot à se situer en son centre. De même pour passer d'une rangée à l'autre, il est suffisant de mesurer la position des arbres environnants et de détecter le centre de la prochaine rangée. Dans ce travail, il est donc proposé de modéliser le verger en utilisant une carte topologique. Celle-ci doit contenir les informations nécessaires au bon déroulement de la navigation, tout en étant le moins sensible possible aux évolutions de l'environnement. Elle consiste en un graphe orienté dont les nœuds représentent soit les entrées hautes (notée H_x pour la rangée x) ou basses (notée B_x

pour la rangée x) des rangées, soit un point dans la zone de fourrière se situant entre deux rangées (notée H_{xy} pour un espace haut entre les rangées x et y). Un nœud N_1 est connecté par un arc à un nœud N_2 s'il existe une manœuvre permettant de naviguer le robot de l'espace représenté par N_1 à l'espace représenté par N_2. La manoeuvre reliant les deux espaces est attachée à l'arc connectant les deux nœuds correspondants. Il est à noter que l'espace représenté par un nœud n'est pas complètement défini dans l'espace métrique. De plus, ces espaces peuvent ne pas partager de frontière physique malgré l'existence d'une connection dans le graphe.

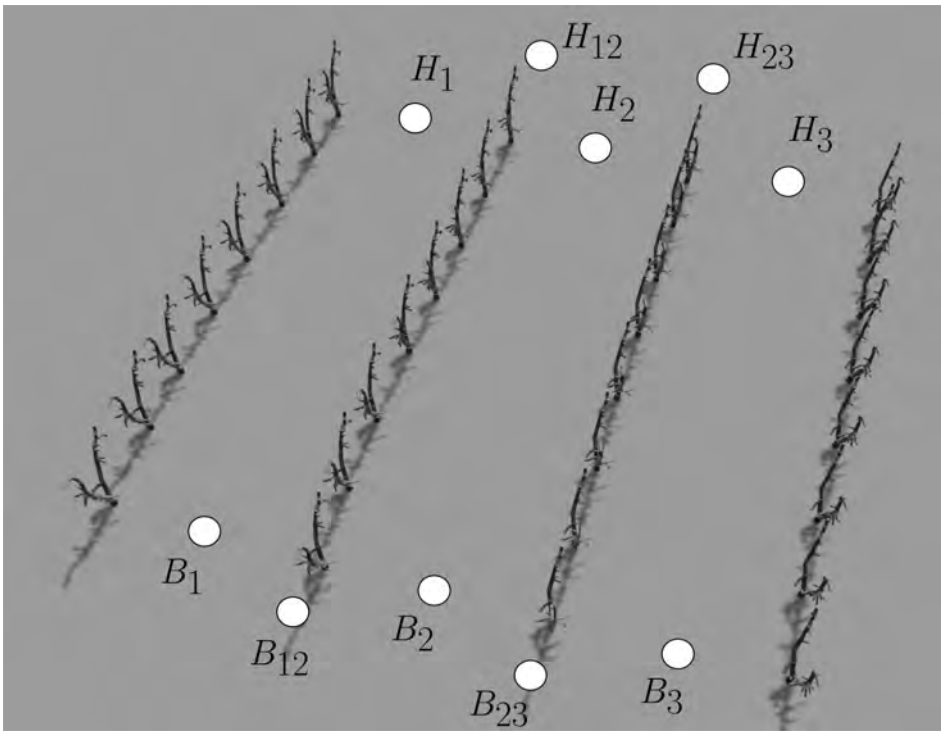


FIGURE 2.3 – Exemple d'un verger composé de trois rangées

Afin d'illustrer nos propos, nous avons construit la carte topologique 2.4 modélisant le verger présenté dans la figure 2.3. Les nœuds H_x et B_x représentent les points d'entrée-sortie des trois rangées, tandis que les nœuds H_{xy} et B_{xy} représentent des points de la zone fourrière à mi-parcours entre chaque rangée. Finalement chaque arc contient l'action permettant de naviguer d'un nœud à l'autre. Par exemple, pour atteindre B_1 à partir de H_1, le robot doit effectuer un suivi de rangée (SR), tandis que pour naviguer de H_{12} vers H_2, il doit réaliser un suivi de spirale (SS). Ainsi, à partir de la carte topologique, pour une tâche donnée et pour des nœuds de départ et d'arrivée donnés, il est possible de calculer la séquence de commandes qui doit être utilisée par le robot pour mener à bien la navigation. Par exemple, en partant du nœud B_1, le robot doit parcourir toutes les rangées pour récolter les fruits. Dans ce cas, la séquence de navigation est la suivante : $(B_1) - SR - (H_1) - SS - (H_{12}) - SS - (H_2) - SR - (B_2) - SS - (B_{23}) - SS - (B_3) - SR - (H_3)$. Dans un autre exemple, à partir du nœud B_1 le robot doit se rendre dans la rangée 3 afin de réaliser une inspection de cette dernière. Ici, la séquence de commande est : $(B_1) - SS(B_{12}) - DR - (B_{23}) - SS - (B_3) - SR - (H_3)$. Finalement, la structure du verger est utilisée pour détecter les instants pour lesquels le robot doit passer d'une action à une autre. Par exemple, il est nécessaire de détecter les entrées-sorties des rangées pour déterminer quand le robot arrive aux nœuds B_x ou H_x. Il faut aussi être capable de détecter lorsqu'une rangée d'arbres se situe à la

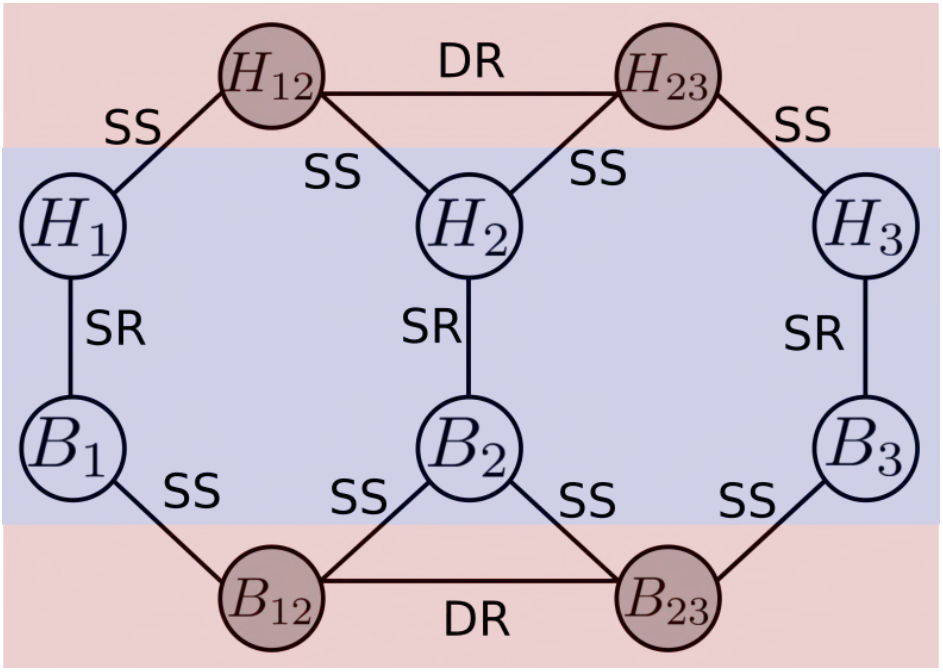


FIGURE 2.4 – Exemple de carte topologique avec les actions Suivit de Rangée (SR), Suivi de Spirale (SS) et Dead Reckoning (DR)

perpendiculaire du robot lorsque celui-ci arrive aux nœuds B_{xy} ou H_{xy}. Ainsi, afin de mener à bien la navigation, le processus de perception doit être capable de détecter les arbres aux alentours du robot.

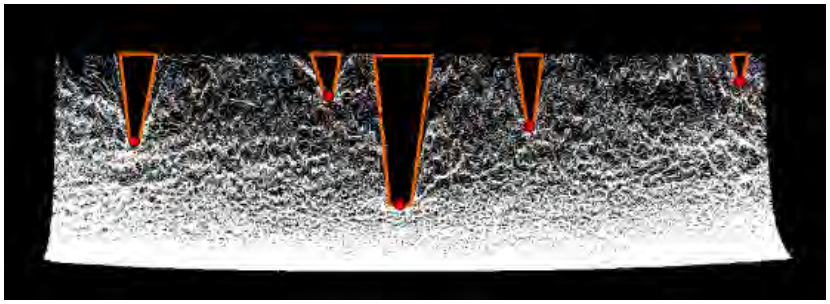
2.4 Perception

Le système de navigation présenté dans ces travaux repose sur la connaissance de la position des arbres au voisinage du robot. Il est donc nécessaire de les détecter puis de les localiser. Des techniques similaires ont déjà été développées comme cela a été présenté dans la section 2.1. La majorité de ces travaux requiert une pré-navigation et a pour objectif de construire une carte métrique de l'environnement. De plus, le calcul de l'espace de navigation ainsi que de la ligne centrale devant être suivi pour traverser une rangée, sont effectués en tenant compte de l'intégralité des arbres (troncs, branches, feuilles et fruits). La répartition non uniforme des branches dans la rangée mène généralement à des résultats décorrélés de la situation physique de la rangée. Afin de calculer avec une plus grande précision la position d'un robot dans une rangée, il apparaît intéressant de se focaliser sur la détection des troncs, dont la position n'évolue pas au cours du temps.

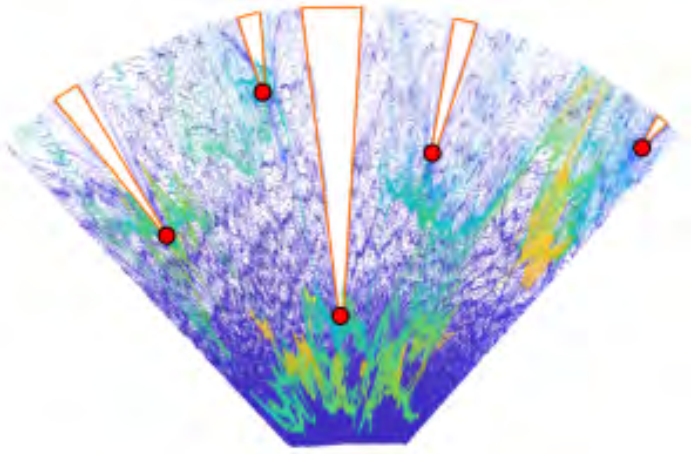
Il est donc proposé dans cette section de détecter et localiser les troncs d'arbres à partir des données provenant de capteurs extéroceptifs tels que des scanners laser ou des caméras 3D. Les capteurs utilisés ici sont des caméras stéréoscopiques. Elles représentent une solution peu coûteuse, permettant ainsi d'en utiliser plusieurs pour agrandir le champ de vue du robot. La méthode décrite dans cette section est réalisée à partir de nuages de points et peut donc être utilisée aussi bien avec des informations venant de caméras 3D que de scanners lasers. Elle repose sur l'utilisation des espaces vides (ou ombres) dans le nuage de points induits par la présence des arbres. En effet, lors de la détection d'un objet comme un arbre, la caméra ne peut calculer de points

3D derrière l'objet en question. La présence d'un espace vide peut alors être interprétée comme l'ombre de l'arbre dans le nuage de points. Considérons la figure 2.5 qui illustre différentes étapes de détection des troncs.

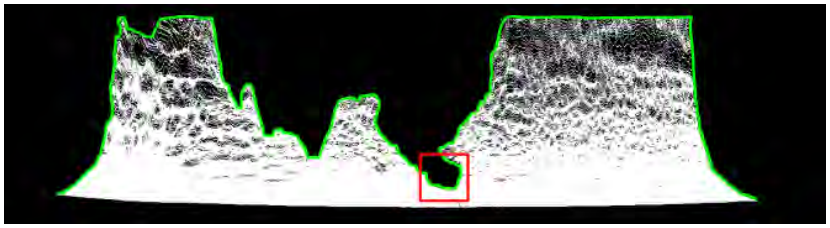
(a) Nuage de points 3D



(b) Exemple de carte polaire - Limites orange : Concavités - Cercles rouge : Troncs



(c) Nuage de points vue du haut



(d) Exemple de carte polaire - Ligne verte : Bordure du nuage de points

FIGURE 2.5 – Étapes de traitements

La figure 2.5a représente l'acquisition d'un nuage de points dans un verger. La vue de dessus de ce nuage de points est affichée dans la figure 2.5c. Les triangles oranges montrent les espaces vides se situant derrière les arbres, eux-mêmes représentés par les cercles rouges. L'algorithme présenté ici a pour objectif de détecter ces espaces vides

formés par les arbres (triangles oranges) et leurs origines (cercles rouges) dans le nuage de points. Dans un premier temps, les points 3D appartenant au sol sont extraits du nuage de points initial. Puis, ces points sont transformés pour créer une carte 2D exprimée en coordonnées polaires comme montré dans la figure 2.5b. Les points résultants de cette transformation forment des concavités représentant l'ombre des troncs d'arbres. Finalement, le fond de ces concavités, qui correspond à la localisation des troncs, est calculé. Ainsi, la deuxième partie de cette méthode consiste à trouver ces concavités et les points correspondant aux troncs. La figure 2.5b pourrait laisser penser que la bordure du nuage de points pourrait être représentée par une fonction régulière. De cette manière, déterminer le fond de ces concavités reviendrait à calculer son minimum à l'aide d'un algorithme de descente de gradient. Toutefois, comme il est illustré dans la figure 2.5d, dans laquelle la ligne verte représente la bordure du nuage de points, la présence de branches et les irrégularités dues à la forme des troncs dans le champ de vue de la caméra empêchent la représentation de la bordure par une fonction de valeur unique. C'est pourquoi, un nouvel algorithme de détection des concavités a été proposé. Il est basé sur des cartes 2D de différentes résolutions et exploitant l'orientation vers le haut de ces concavités. De plus, cet algorithme ne filtre pas les points formant la carte, ce qui permet d'éviter sa distorsion et donc une imprécision dans l'estimation de la position des troncs. L'utilisation de ces différentes résolutions permet d'atténuer l'effet des contours non-réguliers dus à la présence des bruits introduits par les capteurs. Enfin, une carte de densité est créée afin de filtrer les résultats obtenus par la détection du fond des concavités pour éviter les résultats faux positifs éventuels. Ainsi, grâce à la combinaison de l'extraction du sol et de la détection de concavités, une méthode efficace et robuste permettant la localisation des troncs à partir de nuages de points est mise en place. Le pipeline de cette méthode est présenté dans la figure 2.6 dont chaque fonction est décrite par la suite.

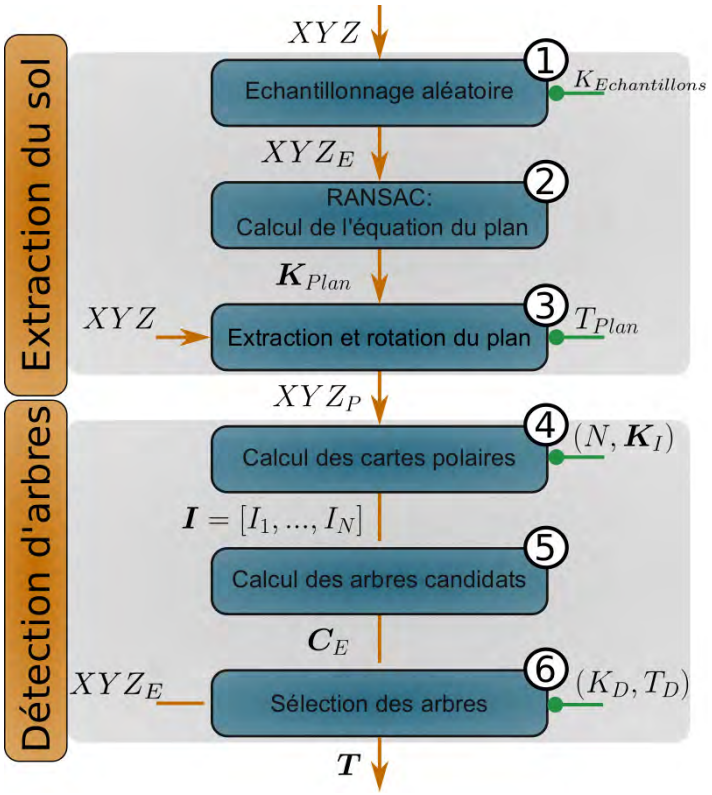


FIGURE 2.6 – Pipeline pour détecter des troncs d'arbres à partir d'un nuage de points

2.4.1 Extraction du sol

La première partie de cette méthode consiste à extraire les points 3D correspondant au sol à partir du nuage de points acquis (noté XYZ) sur la figure 2.6. Pour cela, trois étapes sont mises en œuvre. Premièrement, le processus devant fonctionner en temps réel, le nuage de points est échantillonné aléatoirement (étape 1) de façon à réduire le nombre de points à traiter par la suite. Le nuage de points échantillonné résultant est nommé XYZ_E (figure 2.7a) et sa taille a été réduite à une dimension définie par un scalaire $K_{Echantillons}$. Puis, le paramètre K_{Plan} qui caractérise le plan du sol est extrait grâce une régression par moindres carrés utilisant l'algorithme RANSAC (étape 2). Avec l'équation du plan, le nuage de points original est alors filtré pour ne garder que les points appartenant au plan défini par K_{Plan}. Les points se trouvant à une distance euclidienne de ce plan plus petite que celle définie par le paramètre T_{Plan} sont aussi conservées (étape 3). Finalement, une rotation de ce nuage de points est appliquée afin d'aligner le plan formé par le sol sur l'axe de la caméra. Le nuage de points résultant est alors nommé XYZ_p (figure 2.7b).

(a) Nuage de points XYZ_E

(b) Nuage de points XYZ_p

FIGURE 2.7 – Extraction du sol

2.4.2 Détection des arbres

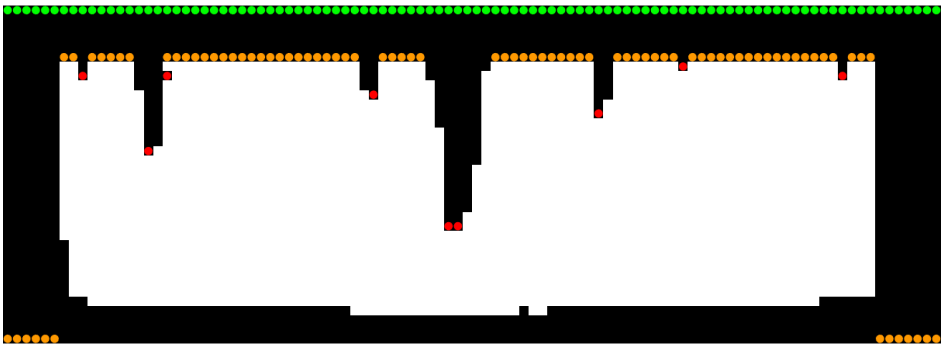
Une fois le plan représentant le sol extrait, il reste à détecter les points d'inflexion présents dans les concavités. Pour cela, le nuage de points représentant le sol est transformé en coordonnées polaires puis projeté sur N plans 2D de différentes résolutions (étape 4). Ces N images sont alors stockées dans un vecteur $I = [I_1, \ldots, I_N]$, dans lequel I_1 et I_N sont respectivement les images de plus basse et de plus haute résolutions. La création de ces différentes images permet de s'affranchir du bruit inhérent au processus de mesure ainsi que de réduire le temps de calcul nécessaire. La transformation opérée

pour changer de résolution est décrite par le paramètre $K_I = [\mu_1, \theta_1, \ldots, \mu_N, \theta_N]$, dans lequel μ_i et θ_i, sont respectivement la longueur et l'angle représenté par chaque pixel avec $i \in [1, \ldots, N]$. Comme le montrent les figures 2.8a et 2.8b, les images obtenues présentent désormais des concavités orientées verticalement. L'étape suivante consiste à détecter ces concavités et plus particulièrement les points d'inflexion se situant au fond de celles-ci (étape 5). Pour cela, l'algorithme de calcul des arbres candidats place des jetons sur chaque pixel du bord supérieur de l'image I_1 tel que présenté dans la figure 2.8a. Ces jetons sont alors lâchés et se déplacent vers le bas de l'image I_1 comme illustré dans la figure 2.8c. À chaque exploration d'un nouveau pixel, la ligne du jeton est évaluée pour y détecter les bords environnants (point orange). Trois cas sont alors possibles :

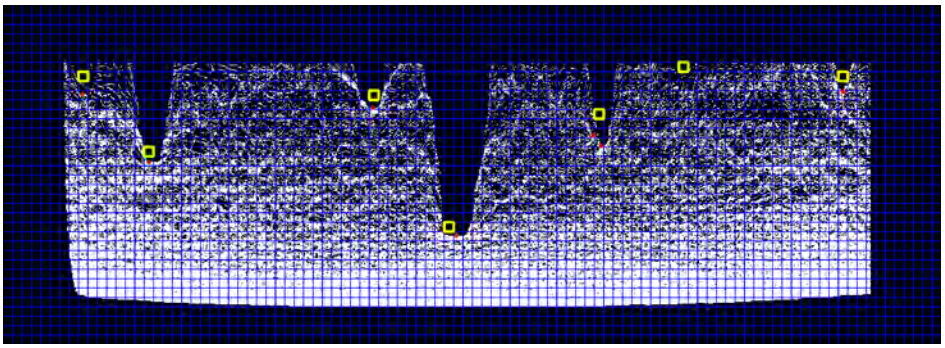
1. Les bords gauche et droit sont les limites de l'image, alors le jeton est autorisé à continuer son déplacement vers la prochaine ligne (jetons blancs), si ce déplacement est impossible, il est éjecté.
2. L'un des bords est une limite de l'image, l'autre est la forme du sol, alors ce jeton est éjecté.
3. Les deux bords sont détectés comme faisant partie du sol, alors une concavité est détectée (jetons violets). Lorsqu'un jeton est évalué comme étant dans une concavité alors celui-ci est positionné de façon aléatoire entre les deux bords de sa ligne (jetons cyans). Cette opération est répétée tant que le jeton peut se déplacer vers le bas. Dans le cas contraire, la position du jeton est considérée comme le fond de la concavité et est enregistrée (jetons rouges).

Par la suite, l'image de résolution $i + 1$ est considérée. Les jetons sont alors lâchés depuis leurs positions enregistrées de l'image précédente comme le présente la figure 2.8b. Ainsi, l'exploration aléatoire de la concavité continue pour atteindre plus précisément la position du point d'inflexion réel. Cette opération est répétée dans toutes les résolutions d'images jusqu'à obtenir le résultat dans l'image I_N. Les positions finales des jetons dans l'image I_N sont alors sauvegardées dans un vecteur C_E. Ce vecteur C_E est le résultat de l'étape de calcul des arbres candidats (étape 5). Il peut arriver que, dû à un sol irrégulier, de l'herbe haute ou encore des équipements laissés sur place, des ombres qui ne représentent pas des arbres apparaissent. Afin de réduire les résultats faux-positifs, un autre processus de sélection est ajouté (étape 6). Ce processus a pour objectif de vérifier la présence d'un tronc, de branches et de feuilles au-dessus de chacun des arbres candidats (C_E). Cette sélection se résume alors à vérifier si le nombre de points est assez important au-dessus du candidat considéré. Pour cela, une carte discrète M_D avec une résolution similaire à I_N est construite comme le montre la figure 2.9a. A chaque case de cette carte est attaché le nombre de points XYZ_S présents dans cette zone. Finalement, la valeur des cases comprises dans un carré de dimension K_D et centré sur la position du candidat sont additionnés de façon à calculer un score d'occupation S_D. Si S_D est plus grand qu'un seuil T_D prédéfini, alors le point candidat est ajouté au vecteur T comprenant toutes les positions de troncs d'arbres détectés et validés par l'algorithme (figure 2.9b).

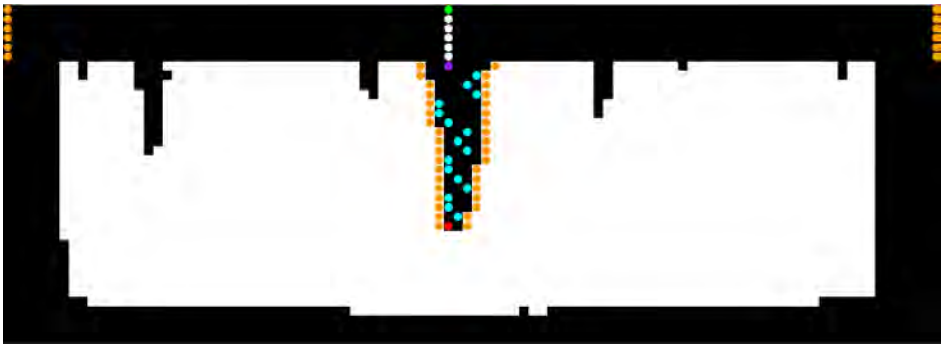
Remarque 1 *L'exploration des concavités se fait de façon aléatoire afin de gérer au mieux le bruit présent dans la forme acquise du sol. En effet, ce bruit peut faire apparaître de petites concavités à l'intérieur de celles désirées. Dans ce cas, une approche aléatoire permet de diminuer le nombre de jetons bloqués dans celles-ci et ainsi minimiser l'impact du bruit dans le processus d'exploration par un filtrage du nombre de jetons.*



(a) Détection de concavités à basse résolution I_1

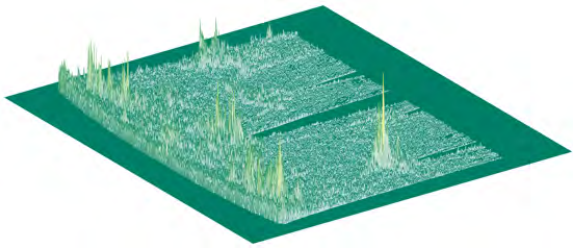


(b) Détection de concavités à haute résolution I_N

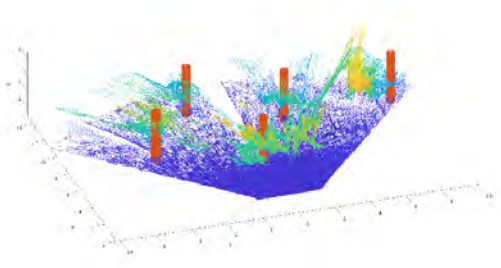


(c) Lâché d'un jeton à basse résolution I_1

FIGURE 2.8 – Exemple de la méthode de lâché de jetons



(a) Carte de densité M_D



(b) Arbres sélectionnés

FIGURE 2.9 – Résultats intermédiaire du pipeline

2.4.3 Expérimentation

Le processus de localisation des troncs d'arbres décrit précédemment a été expérimenté sur le robot, muni de caméras ZED, paramétrées à une résolution de 720p.

Pour ces expérimentations, l'ordinateur utilisé est équipé d'un processeur Intel Core i7-6700H, de 16GB de RAM et d'une carte graphique NVIDIA GeForce GTX 960M. Chaque caméra fournit un nuage composé de 777600 points acquis sur une distance de 15m. De façon à évaluer les performances de cette méthode, des données provenant de trois différents vergers ont été collectées (cf. figure 2.10), leurs caractéristiques étant visibles dans le tableau 2.1. Ainsi, la première collecte de données a été effectuée dans un verger composé de pruniers durant l'hiver (Données n°1) et durant le printemps (Données n°2). Une seconde campagne expérimentale a ensuite été réalisée au printemps dans des vergers composés de noyers (Données n°3) et d'amandiers (Données n°4).

(a) Verger n°1 en hiver (données n°1)

(b) Verger n°1 au printemps (données n°2)

(c) Verger n°2 au printemps (données n°3)

(d) Verger n°3 au printemps (données n°4)

FIGURE 2.10 – Vergers

Données n°	Saison	Type	Largeur (m)	Espacement (m)	Diamètre des troncs (m)
1	Hiver	Prunes	5	3	0.5
2	Printemps	Prunes	5	3	0.5
3	Printemps	Noix	4	3.5	0.35
4	Printemps	Amandes	6	2.5	0.35

TABLE 2.1 – Description des vergers - Largeur, espacement et diamètre moyen exprimé en mètres

L'algorithme de localisation des arbres a été implémenté en utilisant les langages de programmation C++ et CUDA ainsi que les librairies openCV et PCL dans l'environnement ROS. L'algorithme a été initialisé avec les paramètres affichés dans le tableau 2.2.

Dans le tableau 2.3, les performances du système de détection sont données pour

$K_{Echantillons}$	T_{Plan}	N	K_I	K_D	T_D
350000	0.25	2	$[0.5m, 1°, 0.05m, 0.1°]$	11	200

TABLE 2.2 – Paramètres de simulation

les quatre vergers décrits plus haut. Ces résultats montrent le nombre d'arbres détectés ainsi que le pourcentage de précision et de rappel de la classification obtenue.

	0-5 m	5-10 m	10+ m	Total
Données n°1				
Nombre d'arbres	38	67	49	154
Précision	100%	98.5%	64.7%	90.2%
Rappel	97.3%	91%	44.8%	77.9%
Données n°2				
Nombre d'arbres	198	213	N/A	411
Précision	97.5 %	98.7%	N/A	98%
Rappel	98.9%	72.4%	N/A	85.1%
Données n°3				
Nombre d'arbres	89	101	N/A	190
Précision	97.7%	97.6%	N/A	97.7%
Rappel	96.6%	83.1%	N/A	89.4%
Données n°4				
Nombre d'arbres	107	60	N/A	167
Précision	97.1%	98.2%	N/A	97.5%
Rappel	96.2%	90%	N/A	91.5%

TABLE 2.3 – Performances de détection des arbres - Les résultats sont donnés pour 0 à 5 m (0-5), 5 à 10 m (m) et à plus de 10 m

Les résultats sont classés en trois catégories de distance : les arbres situés à moins de 5m, entre 5 et 10m et à plus de 10m. Après la collecte de données faite en hiver, les caméras ont été orientées davantage vers le sol pour permettre une meilleure extraction de l'équation du plan (K_Plan). Les données acquises au printemps ne permettent donc plus de détecter les arbres à plus de 10m. Les mesures effectuées par les caméras ZED sont de bonne qualité jusqu'à 5m. Ainsi l'algorithme de détection obtient de bons résultats dans cette plage de distance. En effet, tous les ensembles de données montrent des résultats de précision supérieurs à 97% avec un taux de rappel de plus de 96%. Entre 5 et 10m, la précision reste bonne avec une précision de 97%. Cependant le taux de rappel chute de manière notable dans l'ensemble des données n°2 et n°3 dans lesquelles sa valeur tombe respectivement à 72.4% et 83.1%. Deux paramètres peuvent expliquer cette chute. Premièrement, plus les points à acquérir sont loin, plus le bruit de mesure est grand et la précision de la position obtenue pour les points est faible. Deuxièmement, au printemps spécifiquement, le champ de vue des caméras peut être bloqué par les feuilles, empêchant ainsi l'acquisition de données correspondant aux arbres plus éloignés. Cependant, les résultats obtenus jusqu'à une distance de 10m sont suffisamment satisfaisants pour permettre une navigation dans des vergers commerciaux.

Les résultats présentés dans la figure 2.11 illustrent la détection opérée dans l'image I_2. Les carrés jaunes représentent les points candidats obtenus à la fin du traitement

de l'image basse résolution I_1, tandis que les carrés verts sont les arbres sélectionnés à la fin du traitement. Enfin, les carrés rouges représentent les candidats sélectionnés dans l'image haute résolution I_2 mais rejetés par la carte de densité. La première image (figure 2.11a) représente la détection des arbres avec une donnée acquise par la caméra positionnée à l'avant gauche du robot naviguant dans le verger n°1 au printemps. Dans ce résultat, trois arbres ont été détectés par l'algorithme (carrés verts) et deux ont été rejetés par le critère de la carte de densité (carrés rouges). Ce cas est intéressant car il montre la nécessité d'une exploration aléatoire des concavités. En effet, les deux arbres de gauche font partie d'une même large concavité comportant quatre points d'inflexion. Dans le cas où l'exploration par la méthode du lâché de jetons se ferait de manière déterministe (par exemple, le jeton se replacerait toujours au milieu de la concavité), alors un seul point d'inflexion serait détecté. Dans les deux exemples suivants présentés dans les figures 2.11b et 2.11c, les résultats montrés sont ceux obtenus par les caméras à l'avant et sur le côté gauche du robot naviguant dans le verger n°2. Ici aussi, tous les arbres du champ de vue sont détectés par l'algorithme mis en place et les résultats faux-positifs filtrés par la carte de densité. Il est important d'expliquer ici que l'arbre dans la zone 0-5m est un arbre taillé en V. Ainsi la caméra est capable de percevoir le sol entre les deux branches secondaires de l'arbre, créant alors un espace non vide derrière le tronc principal. Cet exemple explique parfaitement la raison du développement d'un nouvel algorithme de détection de concavité, au lieu de mettre en place une méthode basée sur une fonction telle que la descente de gradient. Finalement, l'image montrée sur la figure 2.11d présente le résultat du traitement de la caméra avant droite dans le verger n°3. Dans cette image, seuls deux des trois arbres présents sont détectés à la fin du traitement, le dernier n'ayant pas obtenu un score suffisant lors de l'application du critère de la carte de densité.

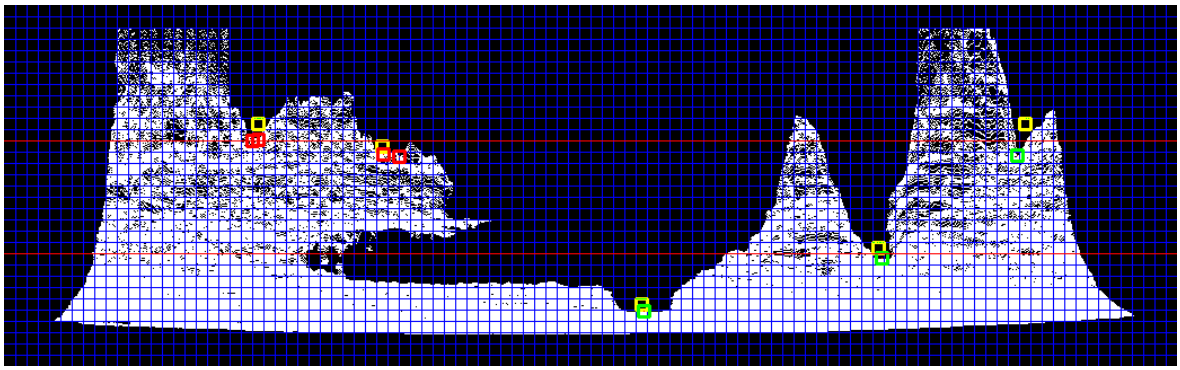
Les derniers résultats présentés dans la figure 2.12 sont un exemple d'utilisation de l'algorithme de détection des troncs d'arbre pour la navigation du robot dans un verger. Dans ces deux images, les nuages de points venant des quatre caméras installées sont affichés simultanément et localisés dans le repère du robot. Dans la figure 2.12a, les nuages de points ont été acquis lors du déplacement du robot dans une rangée du verger n°1. De façon à calculer les lignes représentant la rangée, la position des troncs sélectionnés ainsi qu'un score correspondant au nombre de jetons arrivant à la même position est couplé à un algorithme RANSAC cherchant deux lignes parallèles. Un exemple de demi-tour dans la zone de fourrière est illustré dans la figure 2.12b. Dans ce cas le dernier arbre (servant de référence lors du demi-tour) est obtenu en sélectionnant l'arbre dans T avec le plus haut score.

Dans le tableau 2.4, les temps requis par la plate-forme pour traiter les différentes étapes de l'algorithme sont présentés. Le temps total moyen pour détecter les arbres dans un nuage de points est de 10ms avec des pics allant jusqu'à 15ms. Il peut être utile de préciser qu'une intégration précédente sans CUDA requérait un temps moyen de calcul de 150ms.

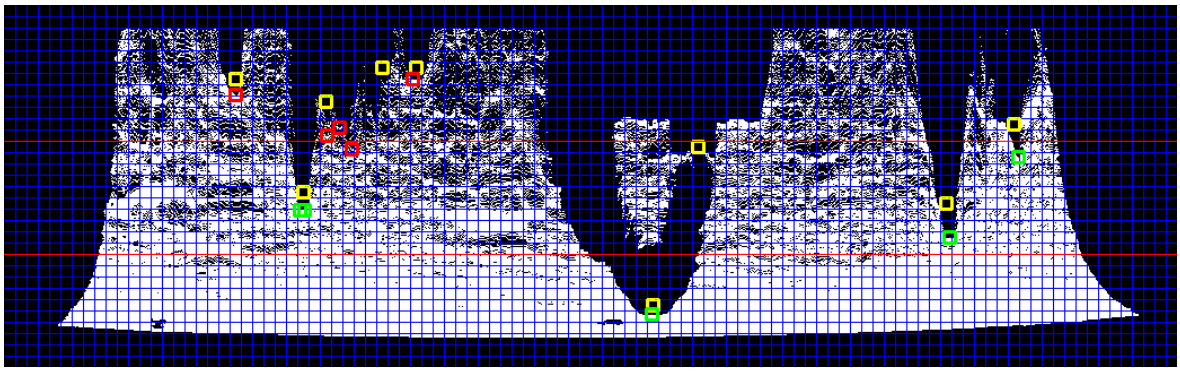
1 : Échantillonnage	2.5 ms	2 : Ransac	4.7 ms
3 : Extraction et rotation du plan	1.8 ms	4 : Images polaires	1 ms
5 : Détection des concavités	1 ms	6 : Sélection des arbres	0.1 ms

TABLE 2.4 – Temps de traitement moyen sur 100 nuages de points (ms)

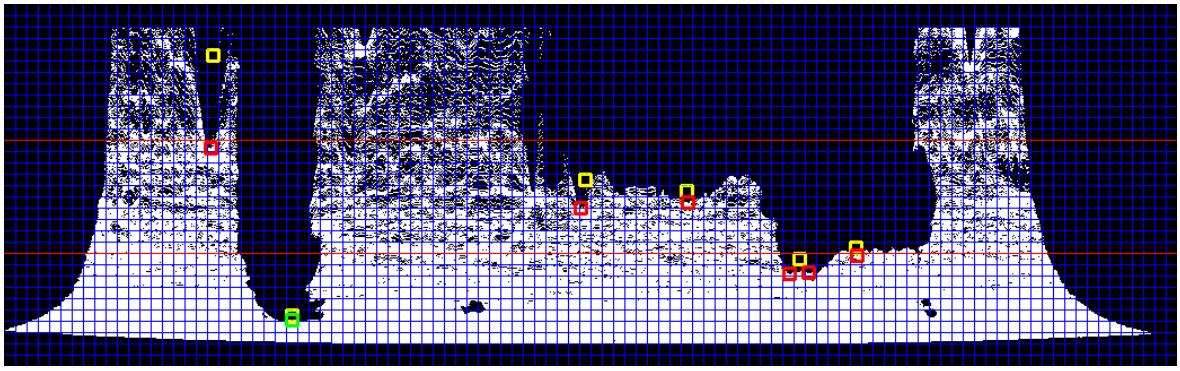
Une liste sommaire des performances de différents travaux est présentée ci-dessous,



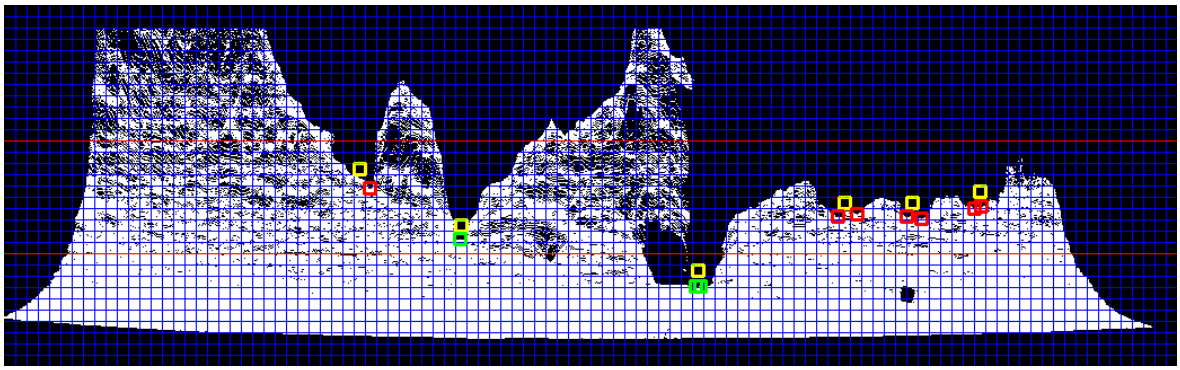
(a) Caméra avant gauche, données n°2



(b) Caméra avant gauche, données n°3



(c) Caméra coté gauche, données n°3



(d) Caméra avant droit, données n°4

FIGURE 2.11 – Exemple de détection des arbres dans l'image I_2 - Carrés jaunes : Candidats sélectionné dans l'image I_1 - Carrés verts : Candidats sélectionnés par l'algorithme - Carrés rouges : Candidats rejetés par la carte de densité - Lignes rouges : limites de 5m et 10m

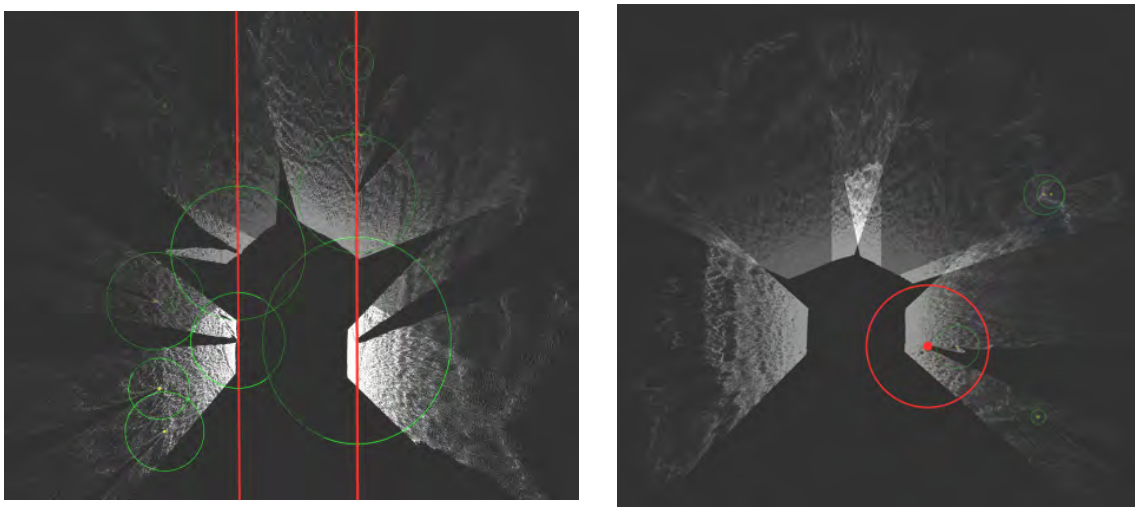
bien qu'aucune indication quant à la distance entre le capteur et l'arbre détecté ne soit fournie. La méthode présentée dans [He et al., 2011] est utilisée pour détecter des pommiers. Elle conduit à un taux de reconnaissance correcte de 91.7% grâce à une caméra monoculaire. Dans [Juman et al., 2016], les auteurs présentent un algorithme permettant de détecter des palmiers à huile avec une caméra stéréo. Ils affichent un taux de détection de 97.8% avec un taux de faux-positifs de 15.2%. Une détection de plaqueminiers est également développée dans [Shalal et al., 2015], grâce à la combinaison d'une caméra avec un scanner laser. Cette méthode possède un taux précision de 96.77% à 98.94% et un taux de rappel de 93.75% à 96.88%. Dans [Chen et al., 2018], des orangers sont détectés utilisant des caméras et des capteurs ultrasons. L'algorithme proposé présente un taux de précision de 95.49% ainsi qu'un taux de rappel de 92.14%. Enfin, [Liu et al., 2018] présente une détection de citronniers utilisant une caméra et obtenant un taux de vrai-positif de 90.8% et un taux de faux-positif de 95.49%.

Ainsi la détection de troncs d'arbres proposée dans cette section montre des performances au moins égales à celles d'autres techniques issues de la littérature. Cependant, il est assez compliqué de comparer les performances obtenues de par la différence des capteurs, des plate-formes de calcul, et de la nature de l'environnement dans lequel évolue le système.

Conclusion

La méthode proposée dans cette section a montré son efficacité pour détecter et localiser les arbres dans l'espace capteur à partir d'un nuage de points. Cette méthode se base sur l'exécution de deux processus. Le premier consiste à détecter et extraire le sol du nuage de points acquis, pour finalement le réorienter suivant l'axe du capteur. Le second, récupère le nuage de points ainsi transformé pour y détecter les concavités et localiser leurs points d'inflexion respectifs. Le pipeline proposé a été implémenté sur le robot utilisant Robot Operating System (ROS) et validé avec des acquisitions effectuées dans différents vergers. Les résultats ont montré de bons résultats tant en terme de détection qu'en terme de temps de calcul, prouvant ainsi la possibilité de son exécution en temps réel.

Ces données seront par la suite utilisées par le processus localisation afin d'actualiser



(a) Cas de suivi de rangées

(b) Cas de demi-tour vers la droite

FIGURE 2.12 – Exemple de détection d'arbres intégrée au robot

la position du robot dans la carte topologique et donner les informations nécessaires aux processus action et décision.

2.5 Action

Dans cette section, la stratégie de commande permettant à un robot de naviguer dans un verger est présentée. Elle repose sur plusieurs correcteurs réactifs, chacun d'eux effectuant une des deux tâches élémentaires d'une navigation dans des vergers. La première permet au robot de naviguer au travers des rangées, tandis que la seconde est utilisée pour effectuer des demi-tours menant d'une rangée à une autre. Un correcteur et une consigne de référence sont sélectionnés par le processus décision en fonction de la localisation du robot dans la carte topologique. De plus, les mesures nécessaires au bon fonctionnement des correcteurs sont fournies par le processus localisation.

Le premier correcteur réalise un suivi de ligne droite tandis que le second permet un suivi de spirale. Cette dernière est construite à partir d'un modèle de spirale défini dans [Boyadzhiev, 1999] et qui a déjà été utilisé dans des applications d'évitement d'obstacles pour des robots aériens [Mcfadyen et al., 2014] et terrestres [Futterlieb et al., 2014, Leca et al., 2019]. Une première partie est consacrée à la modélisation cinématique de la spirale puis à celle du robot. Ensuite, plusieurs lois de commande sont synthétisées, analysées et testées pour effectuer les tâches requises par le suivi de rangs et la réalisation de demi-tours.

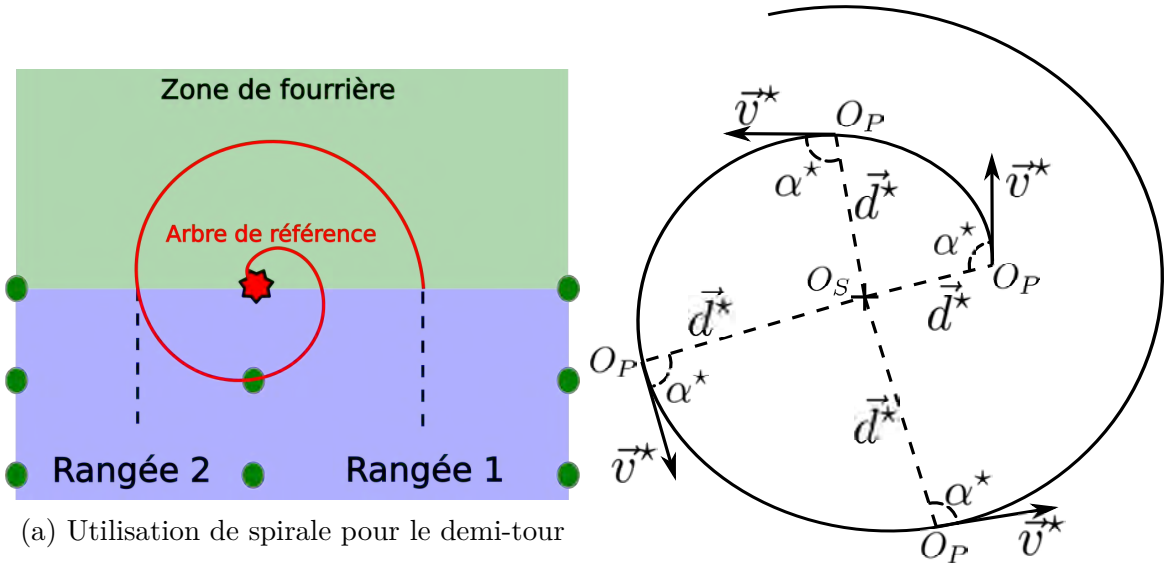
2.5.1 Modélisation

Modélisation de la spirale

Pour permettre au robot de passer d'une rangée à une autre à l'aide d'une loi de commande utilisant des données extéroceptives, il est proposé d'effectuer un suivi de spirale. Celle-ci est centrée sur le dernier arbre se situant entre les deux rangées d'intérêt, et définit un chemin dans la zone de fourrière, allant de la sortie de la rangée courante vers l'entrée de la suivante tel qu'illustré dans la figure 2.13a. Pour ce type d'environnement, le choix de la spirale semble le plus approprié. En effet, dans un verger, les rangées d'arbres peuvent être parfois de largeur inégale. De plus, le robot peut sortir du rang précédent avec une dérive et ne pas être correctement centré par rapport au milieu de la rangée. Dans ce contexte, une spirale centrée sur le dernier tronc d'arbre offre une certaine flexibilité pour s'adapter à ces variations.

Il existe une multitude de modèles de spirales en mathématiques. La spirale la plus commune en planification est la clothoïde qui est définie par un rayon de courbure constant. Cependant dans le cas d'une clothoïde, la position du centre de la spirale varie dans le repère relatif du robot. C'est pourquoi nous nous intéressons à un autre type de spirale, la conchospirale qui a la propriété de garder le centre de la spirale dans une position fixe, avec un rayon de courbure variant au cours du temps. Le contrôleur effectuant les demi-tours en fin de rangées repose sur ce modèle.

La conchospirale a été utilisée dans [Boyadzhiev, 1999] pour représenter le vol d'insectes autour d'une source lumineuse. Ce modèle est illustré dans la figure 2.13b et est défini par un point fixe O_s comme étant le centre de la spirale et par un point mobile O_p. Ce point O_p possède une vitesse linéaire $\vec{v}^*$. Le vecteur $\vec{d}^*$ représente la distance entre les points O_s et O_p. De plus, un angle α^* est défini entre les vecteurs $\vec{d}^*$ et $\vec{v}^*$. En



(a) Utilisation de spirale pour le demi-tour

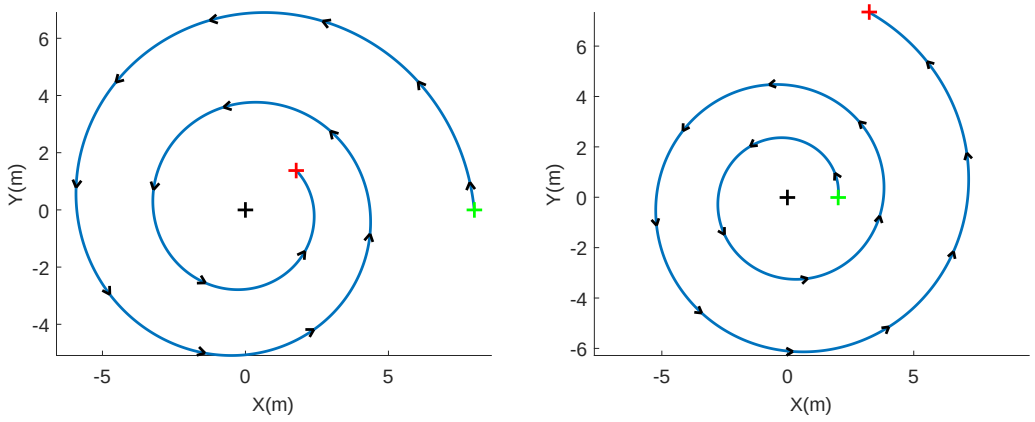
(b) Modèle de conchospirale

FIGURE 2.13 – Utilisation de la spirale

imposant v^* et α^* constantes, le point O_p décrit une spirale autour du point O_s suivant le modèle cinématique suivant (2.1) :

$$\dot{d}^*(t) = -v^*(t)\cos(\alpha^*(t)) \tag{2.1}$$

Le type de spirale est alors défini par la valeur de l'angle α^* (cf. figure 2.14). En effet, si cet angle est situé dans l'intervalle $[0,\ \pi]$, la spirale est parcourue dans le sens anti-horaire. Si au contraire, $\alpha^* \in [-\pi,\ 0]$ alors le sens de rotation est horaire. Enfin, si l'angle $\alpha^* \in [-\frac{\pi}{2},\ \frac{\pi}{2}]$, la spirale est entrante, tandis que si $\alpha^* \in [\frac{\pi}{2},\ \pi] \cup [-\pi,\ -\frac{\pi}{2}]$ elle est sortante. En revanche, si $\alpha^* = \frac{\pi}{2}$ alors la spirale est un cercle.



(a) Spirale entrante

(b) Spirale sortante

FIGURE 2.14 – Paramétrisation de la spirale

Considérons que O_P représente un robot et O_S un point d'intérêt tel que le dernier arbre d'une rangée. L'angle α^* représente alors l'angle entre la direction du robot et le vecteur reliant le centre du robot à l'arbre. De plus, α^* peut être directement mesuré par un capteur embarqué tel qu'une caméra ou un laser. Cette unique mesure permet de savoir si le robot s'approche, s'éloigne ou reste à une distance constante de l'arbre.

Ce sont ces aspects qui ont motivé notre décision d'utiliser une spirale pour réaliser les demi-tours.

Modélisation du robot

La modélisation du robot est une étape indispensable à la conception des correcteurs à venir. Dans un premier temps les repères nécessaires sont introduits dans la figure 2.15. Tout d'abord un repère global $F_w = (O_w,\ \vec{x}_w,\ \vec{y}_w,\ \vec{z}_w)^T$ est défini ainsi qu'un repère attaché au robot $F_r = (O_r,\ \vec{x}_r,\ \vec{y}_r,\ \vec{z}_r)^T$.

Un robot tel que le Toro Workman possède une direction de type *Ackermann* qui se distingue par la différence de rayon de courbure opérée par les roues directrices lorsque le véhicule tourne comme cela est illustré dans la figure 2.15a. Cependant, ce modèle peut être approximé par un modèle bicyclette, comme expliqué dans [Siegwart and Nourbakhsh, 2004] et illustré dans la figure 2.15b. Dans ce modèle, on ôte la dimension correspondant à la largeur du véhicule, ne considérant qu'une roue directrice à l'avant et une roue motrice à l'arrière. Le vecteur d'état correspondant à ce nouveau modèle de véhicule peut alors s'écrire $\chi(t) = [x(t)\ y(t)\ \theta(t)\ \gamma(t)]^T$ avec $x(t)$, $y(t)$ et $\theta(t)$ étant la position et l'orientation du robot dans le repère global et $\gamma(t)$ l'orientation de la roue directrice exprimée dans le repère du robot. Le vecteur de commande du robot est donné par : $u(t) = [v(t)\ \gamma(t)]^T$ avec $v(t)$ la vitesse linéaire du robot et son vecteur d'état réduit à : $[x(t)\ y(t)\ \theta(t)\ \gamma(t)]^T$.

Connaissant le lien reliant l'orientation de la roue directrice $\gamma(t)$ et la vitesse angulaire $\theta(t)$ (cf. équation (2.2)), il est possible d'utiliser un modèle de robot différentiel (figure 2.15c). De cette façon, le vecteur de commande est redéfini comme étant $u(t) = [v(t)\ \omega(t)]^T$ avec $\omega(t)$ la vitesse angulaire du robot.

$$\gamma(t) = \arctan\left(\frac{L\omega(t)}{v(t)}\right) \tag{2.2}$$

avec L, un scalaire représentant la distance entre les essieux avant et arrière du robot.

Le nouveau modèle cinématique correspondant au modèle différentiel est donné par :

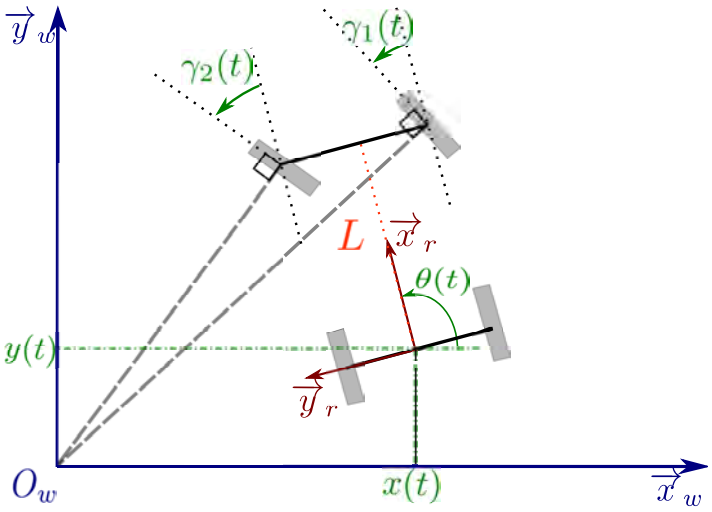
$$\begin{bmatrix}\dot{x}(t)\\ \dot{y}(t)\\ \dot{\theta}(t)\end{bmatrix} = \begin{bmatrix}\cos(\alpha(t)) & 0\\ \sin(\alpha(t)) & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}v(t)\\ \omega(t)\end{bmatrix} \tag{2.3}$$

Le modèle différentiel sera utilisé pour synthétiser les correcteurs nécessaires à la navigation. Cependant, l'adaptation de ce modèle vers le modèle bicyclette sera requis pour une intégration sur le robot réel.

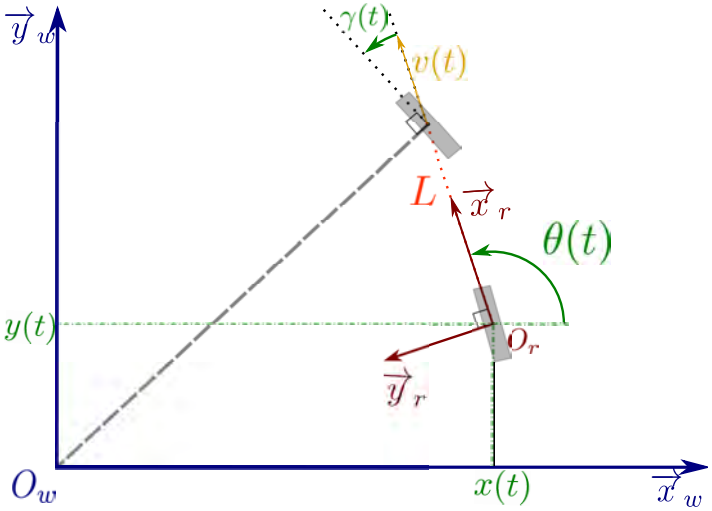
Pour répondre à la problématique d'un suivi de spirale, il est intéressant d'exprimer la pose du robot en coordonnées polaires. Ainsi, comme indiqué dans la figure 2.15c, le vecteur d'état peut être défini comme $\zeta(t) = [d(t), \alpha(t), \beta(t)]^T$ avec $d(t)$ la norme du vecteur définie entre les points O_w et O_r, $\alpha(t)$ l'angle de l'axe $\vec{x}_r$ du robot vers le vecteur $\vec{d}$ et finalement $\beta(t)$ l'angle allant de l'axe $\vec{x}_w$ du repère de référence vers le vecteur $\vec{d}$. Dans ce cas, nous utiliserons les équations suivantes [Boyadzhiev, 1999] :

$$\alpha(t) = \pi - \theta(t) + \beta(t) \tag{2.4}$$

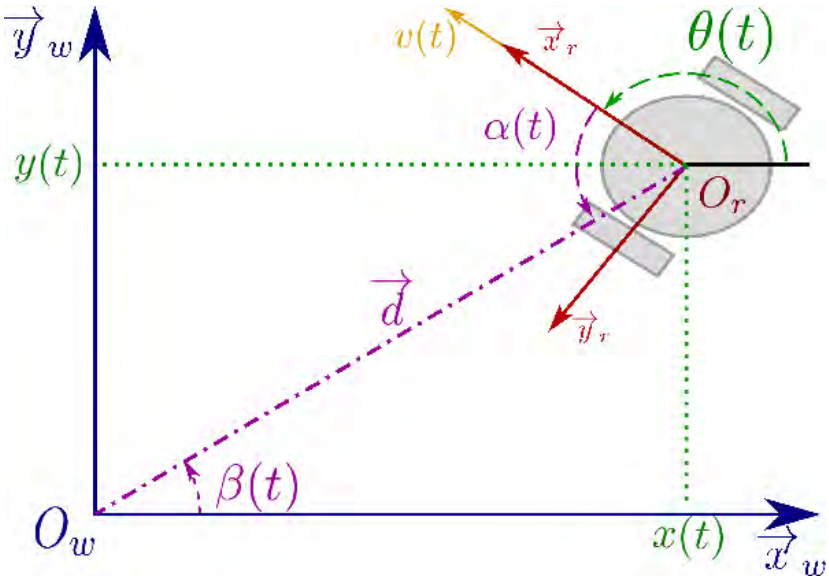
$$\dot{\alpha}(t) = -\dot{\theta}(t) + \dot{\beta}(t) = -\omega(t) + \frac{v(t)}{d(t)}\sin(\alpha(t)) \tag{2.5}$$



(a) Modèle de véhicule à direction de type Ackermann



(b) Modèle bicyclette



(c) Modèle du robot différentiel

FIGURE 2.15 – Modèles du robot

$$\dot{d}(t) = -v\cos(\alpha(t)) \tag{2.6}$$

2.5.2 Synthèse de lois de commande référencés capteur pour le parcours d'un verger

L'objectif de cette partie est la conception des différents correcteurs permettant la réalisation des tâches requises pour la navigation (suivi de rangée et demi-tours).

Synthèse d'une loi de commande pour l'exécution d'un suivi de rangée

Un premier correcteur a été construit pour parcourir les rangées. Nous supposons ici que plusieurs troncs sont localisés par le processus de perception, que les paramètres des droites formant les rangées à droite et à gauche (Δ_L, Δ_R) du robot ont été estimés et que la ligne centrale (Δ_M) a été calculée (cf figure 2.16). L'objectif du correcteur est alors d'asservir le robot de manière à suivre Δ_M.

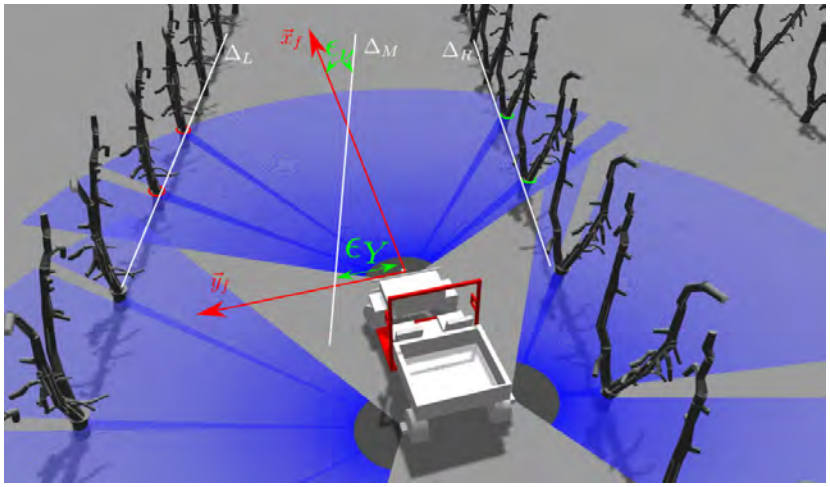


FIGURE 2.16 – Simulation de navigation à travers les rangées

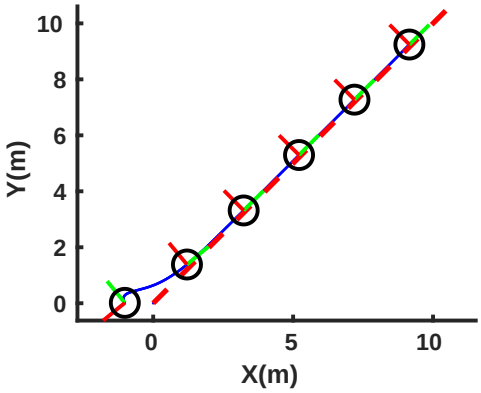
La situation du robot par rapport à Δ_M est définie par deux erreurs. La première, $\epsilon_Y(t)$ concerne la distance latérale entre le robot et la ligne. La deuxième, $\epsilon_\nu(t)$ est l'angle de déviation représentant l'orientation du robot par rapport à la ligne. Un suivi de ligne correspondant à la régulation à zero de ces deux erreurs, il est proposé d'utiliser le correcteur suivant :

$$\omega(t) = \lambda_\nu \epsilon_\nu(t) + \lambda_Y \epsilon_Y(t) \tag{2.7}$$

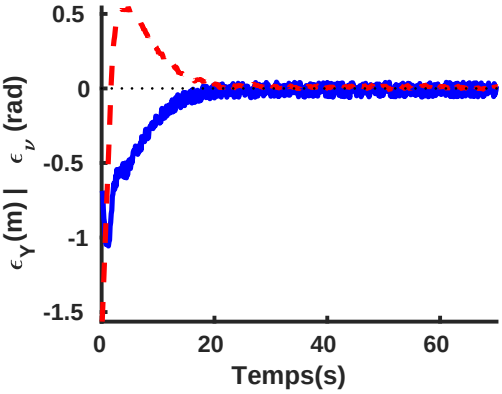
où λ_ν et λ_Y sont deux scalaires strictement positifs réglant la vitesse de convergence du correcteur.

Résultats :

Les performances de ce correcteur ont été évaluées en simulation grâce au logiciel *Matlab* [MATLAB, 2010]. Les résultats obtenus sont présentés dans la figure 2.17.



(a) Trajectoire du robot - Ligne continue bleue : Position du robot - Ligne tiretée rouge : Ligne de référence



(b) Erreurs - Ligne continue bleue : $\epsilon_Y(t)$ - Ligne tiretée rouge : $\epsilon_\nu(t)$

FIGURE 2.17 – Suivi de ligne

Dans cette simulation le robot est initialisé à $\chi(t) = [-1\ 0\ \frac{3\pi}{4}]^T$. La ligne de référence est une droite d'angle $\pi/4$ par rapport au repère global qui passe par la position $[0\ 0]^T$. La vitesse linéaire $v(t)$ du robot est fixée à $0.2m.s^{-1}$ et une saturation de sa vitesse angulaire est imposée avec $\omega_{min/max} = \pm 1 rad.s^{-1}$. Le correcteur est réglé par les gains $\lambda_\nu = 1$ et $\lambda_Y = 1$. Enfin, pour évaluer la loi de commande dans un contexte réaliste, du bruit a été ajouté à l'estimation de la position du robot par rapport à la ligne ($\epsilon_Y(t)$ et $\epsilon_\nu(t)$). Ce bruit a été fixé à 0.1m pour la position et 1° pour l'orientation. Le premier résultat présenté par la figure 2.17a illustre la trajectoire parcourue par le robot (ligne continue bleue) pour suivre la ligne de référence (ligne tiretée rouge). Nous pouvons remarquer que bien que le robot soit initialisé avec une mauvaise position et orientation par rapport à la ligne, il converge rapidement vers la spirale souhaitée puis la suit précisément. La figure 2.17b montre l'évolution des erreurs $\epsilon_Y(t)$ et $\epsilon_\nu(t)$ tout au long de la simulation. Nous pouvons observer ici que les deux erreurs s'annulent bien au régime permanent, malgré le bruit introduit dans l'estimation des erreurs.

Ce premier correcteur devant permettre de suivre les rangées d'arbres a montré son efficacité malgré le bruit intégré sur les mesures. Ce correcteur sera donc utilisé lorsque le robot sera localisé dans un nœud correspondant à une rangée dans la carte topologique.

Synthèse de lois de commande pour la réalisation de demi-tours

Dans cette partie, nous nous intéressons à la synthèse des contrôleurs permettant de réaliser le passage d'une rangée à une autre, autrement dit les demi-tours. Pour cela, nous nous appuyons sur le modèle de spirale décrit précédemment. Il est supposé que la position du dernier tronc de la rangée est donnée par le processus de perception et donc que l'angle α et la distance d sont connus. De plus, la forme de spirale que le robot doit suivre, c'est-à-dire α^* et d^*, sont eux aussi connus et fournis par le processus décision. Dans cette section, nous présentons premièrement un correcteur où la tâche

à réaliser est simplement exprimée en fonction de α^*, puis nous nous attachons à la synthèse de deux autres correcteurs réalisant la minimisation d'une erreur dépendant de α^* et d^*.

Correcteur minimisant une erreur fonction de α :

Il a été montré lors de la modélisation de la spirale que le seul paramètre définissant le type d'une spirale (entrante ou sortante, sens horaire ou anti-horaire) est le paramètre α^*. Il semble alors intéressant de synthétiser un correcteur faisant converger l'angle $\alpha(t)$ vers l'angle α^*. Dans ce cas, nous imposons une vitesse linéaire constante au robot et à la spirale, c'est-à-dire $v(t) = v = v^* \neq 0$. De plus, nous définissons l'erreur à minimiser comme :

$$e_\alpha(t) = \alpha(t) - \alpha^* \tag{2.8}$$

L'évolution de cette erreur est ensuite calculée à l'aide des équations 2.8 et 2.5 :

$$\dot{e}_\alpha(t) = \dot{\alpha}(t) = -\omega(t) + \frac{v}{d(t)} \sin(\alpha(t)) \tag{2.9}$$

En imposant une décroissance exponentielle à l'erreur, $\dot{e}_\alpha(t) = -\lambda_\alpha e_\alpha(t)$, il vient alors la loi de commande suivante :

$$\omega(t) = \lambda_\alpha e_\alpha(t) + \frac{v}{d(t)} \sin(\alpha(t)) \tag{2.10}$$

avec λ_α un gain strictement positif, réglant la vitesse de convergence de l'erreur vers zéro. De façon à étudier la stabilité de cette loi de commande, la fonction de Lyapunov suivante est proposée :

$$V_B(x_B(t)) = \frac{x_B^2(t)}{2} \tag{2.11}$$

avec $x_B(t) = e_\alpha(t)$. Le point d'équilibre est alors défini pour $x_{BE} = 0$. Ceci correspond au robot suivant le chemin de la spirale caractérisé par α^*. De plus, la fonction $V_B(x_B(t))$ est définie strictement positive sauf au point d'équilibre où $x_B(t) = x_{BE}$ où $V_B(x_B(t)) = 0$. Pour étudier l'évolution de cette fonction, sa dérivée en fonction du temps est calculée dans l'équation 2.12.

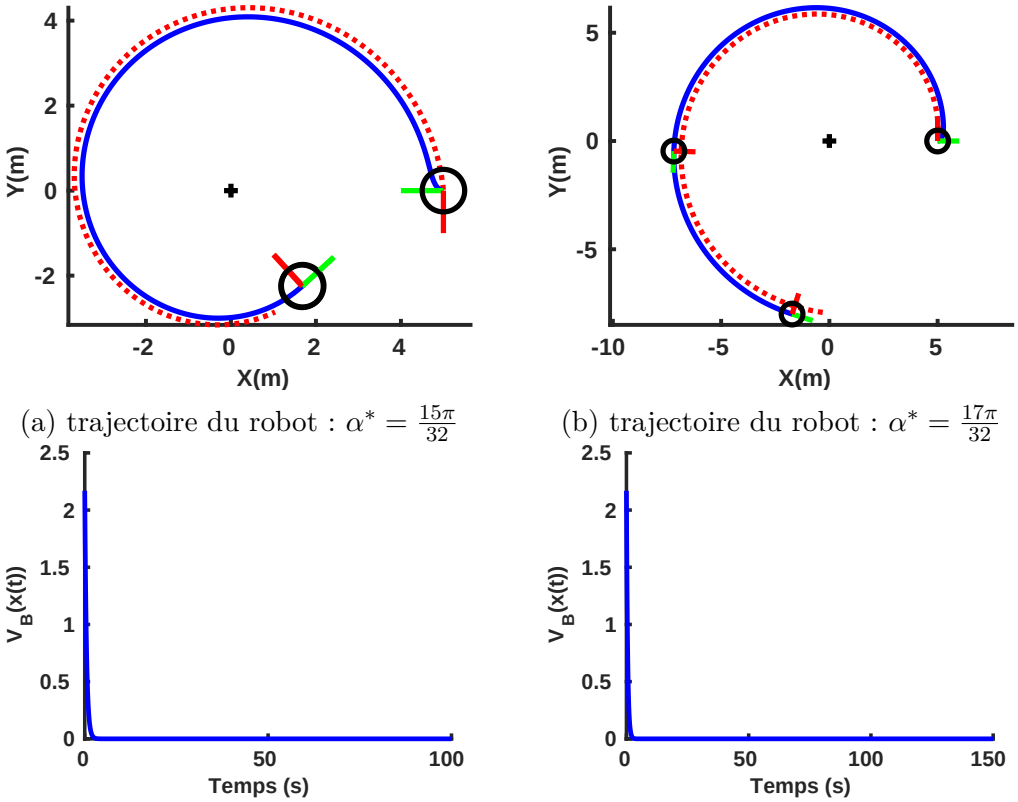
$$\begin{aligned} \dot{V}_B(x_B(t)) &= \dot{x}_B(t) x_B(t) \\ &= \dot{\alpha}(t)(\alpha(t) - \alpha^*) \\ &= [-\omega(t) + \frac{v}{d(t)} \sin(\alpha(t))][\alpha(t) - \alpha^*] \\ &= -\lambda_\alpha e_\alpha(t)^2 \end{aligned} \tag{2.12}$$

Nous obtenons alors $\dot{V}_B(x_B(t)) < 0$ lorsque $x_B(t) \neq x_{BE}$ si $\lambda_\alpha > 0$ et $\dot{V}_B(x_B(t)) = 0$ lorsque $x_B(t) = x_{BE}(t)$ et $\lambda_\alpha > 0$. Le système rebouclé avec la loi de commande précédente est donc globalement asymptotiquement stable.

Résultats :

Dans un premier temps cette loi de commande a été validée en simulation grâce au logiciel Matlab©. Chacune de ces simulations est réalisée avec une période d'échantillonnage de $T_s = 0.1$s. De plus, le centre des spirales est défini aux coordonnées $[0, 0]^T$ dans le repère absolu F_w et la vitesse linéaire $v(t) = v = 0.2\ m.s^{-1}$. La première série

de simulations (figure 2.18) a pour objectif d'analyser le comportement du correcteur présenté dans l'équation (2.10) obtenu pour un gain $\lambda_\alpha = 1$. Cette loi de commande est appliquée à un robot différentiel pour qu'il suive différentes spirales avec différents états initiaux du robot.



(a) trajectoire du robot : $\alpha^* = \frac{15\pi}{32}$ (b) trajectoire du robot : $\alpha^* = \frac{17\pi}{32}$

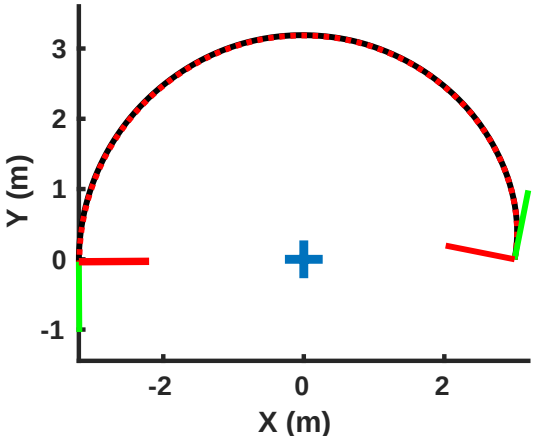
(c) $V_B(x_B(t))|\alpha^* = \frac{15\pi}{32}$ (d) $V_B(x_B(t))|\alpha^* = \frac{17\pi}{32}$

FIGURE 2.18 – Suivi de spirale - Croix noire : centre de la spirale - Ligne bleue continue : Trajectoire du robot - Ligne rouge pointillée : Spirale de référence

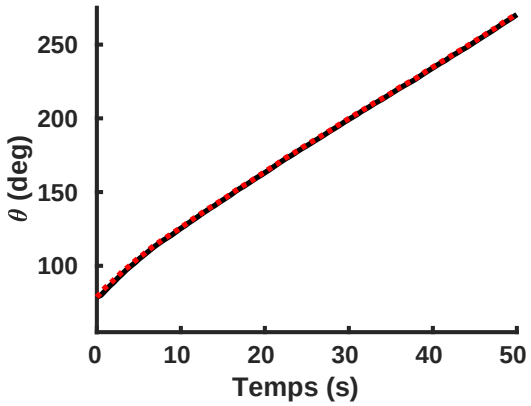
La première simulation (figure 2.18a) vise à suivre une spirale entrante définie par $\alpha^* = \frac{15\pi}{32}$. L'état initial du robot est défini dans le repère cartésien de la spirale et réglé à $\chi(0) = [5, 0, \pi]^T$. La seconde simulation (figure 2.18b) suit une spirale sortante définie par $\alpha^* = \frac{17\pi}{32}$ avec un état initial du robot imposé à $\chi(0) = [5, 0, 0]^T$. Dans les figures 2.18a et 2.18b la spirale de référence est représentée par les lignes en pointillés rouges et la trajectoire du robot par les lignes continues bleues. De plus, les segments verts et rouges qui représentent respectivement les vecteurs du repère robot $\vec{x}_r$ et $\vec{y}_r$ sont affichés de façon à illustrer régulièrement l'état du robot au cours de la simulation. Ces résultats montrent que, malgré la mauvaise orientation initiale du robot, celui-ci converge vers l'orientation désirée définie par l'angle α^* puis suit le type de spirale souhaitée en gardant $\alpha(t) = \alpha^*$. Le comportement décrit est confirmé par l'analyse des figures 2.18c et 2.18d qui représentent l'évolution des fonctions de Lyapunov $V_B(x_B(t))$. En effet, cette analyse montre que la fonction $x_B^2(t) = (\alpha(t) - \alpha^*)^2$ décroît jusqu'à zéro, puis reste stable sur cette valeur. Ce résultat valide l'analyse de stabilité effectuée. Cependant on remarque que la loi de commande définie par l'équation (2.10) ne permet pas au robot de suivre une spirale spécifique. En effet, dans ce cas, la spirale suivie est définie par les paramètres α^* et $d^*(t)$ obtenus lorsque l'orientation $\alpha(t)$ a effectivement convergé vers l'orientation de la spirale α^* (*i.e.* lorsque la fonction $V_B(x_B(t)) = 0$).

Le robot utilisé étant un véhicule à direction Ackermann, la phase d'expérimentation

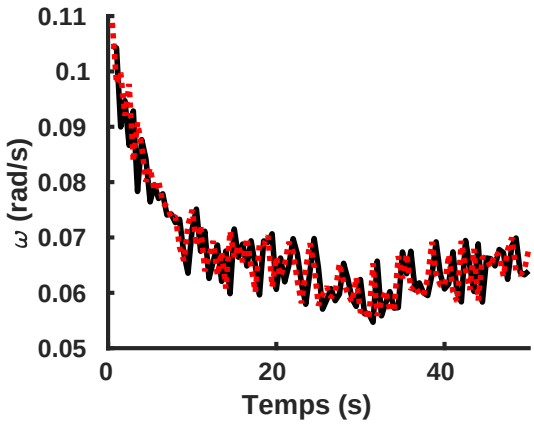
sur le robot réel nécessite de transformer le résultat de la loi de commande représentée par la vitesse angulaire du robot en angle d'orientation de ses roues directrices. Il s'agit donc de tourner de la même façon en utilisant la vitesse angulaire du robot ou l'angle d'orientation de la roue directrice. Cependant, l'angle d'orientation désiré de la roue doit être atteint instantanément afin d'obtenir la même vitesse angulaire du robot. Autrement dit, la vitesse d'orientation de la roue $\dot{\gamma}(t)$ doit être de valeur infinie. Imposer une vitesse infinie est bien sûr impossible en pratique. Le modèle bicyclette et le modèle différentiel ont donc été utilisés pour suivre la même spirale afin d'évaluer l'impact de cette erreur de modèle. Le premier robot simulé est différentiel tandis que le second robot est représenté grâce au modèle bicyclette comportant une vitesse d'orientation de la roue directrice $\dot{\gamma}(t)$ finie. La spirale à suivre est définie par $\alpha^* = \pi/2$ et son centre se situe aux coordonnées $(0, 0)$. Les deux robots sont asservis par la loi de commande fournie par l'équation (2.10), réglée avec les paramètres $\lambda_\alpha = 0.2$ et $v = 0.2\ m/s$. Finalement, de façon à obtenir des résultats plus proches de la réalité, un bruit Gaussien aléatoire de 5% est ajouté aux vitesses linéaire et angulaire. Les résultats obtenus sont visibles dans la figure 2.19.



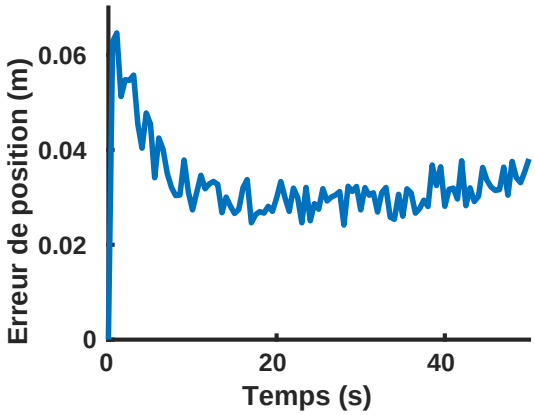
(a) Trajectoire du robot



(b) Orientation du robot $\theta(t)$



(c) Vitesse angulaire $\omega(t)$



(d) Erreur en distance entre la position des deux robots

FIGURE 2.19 – Comparaison des modèles - ligne noire continue : Modèle bicyclette - ligne rouge pointillée : Modèle différentiel

Les figures 2.19a et 2.19d montrent que malgré l'erreur commise sur modèle bicyclette, les trajectoires sont fortement similaires. De plus, les orientations respectives de chaque robot présentées dans la figure 2.19b ne divergent pas l'une de l'autre. Les vitesses angulaires affichées dans la figure 2.19c semblent globalement identiques mais

un léger retard peut être cependant observé. D'après ces résultats, il est possible de convertir la vitesse angulaire du robot $\omega(t)$ en angle d'orientation de la roue directrice $\gamma(t)$ présent dans le modèle bicyclette, même si la vitesse angulaire d'orientation de la roue $\dot{\gamma}(t)$ n'est pas définie comme infinie.

La commande étant à présent adaptée au type de véhicule utilisé, le correcteur est implémenté dans une architecture créée avec le middleware ROS. Ces expérimentations sont réalisées pour montrer le comportement de la loi de commande sur un robot réel et dans un milieu extérieur ouvert (figure 2.20). La position et l'orientation du robot sont acquises via le RTK-GPS et l'IMU installés sur le robot. Pour ces tests, la période d'échantillonnage du correcteur est réglée à 100 ms et son gain $\lambda_\alpha = 0.3$. Dans les figures 2.20a et 2.20b sont représentées les différentes spirales effectuées par le robot. Comme précédemment, les situations du robot au cours de l'expérimentation sont affichées régulièrement. Ces situations sont représentées par des vecteurs de couleur verte et rouge et définissent respectivement les axes $\vec{x_r}$ et $\vec{y_r}$ du repère robot. Enfin, la localisation initiale du robot est marquée par une croix verte et se situe aux coordonnées $(0, 0)$. La localisation finale quant à elle est marquée par une croix rouge.

Le premier ensemble d'expérimentations est un suivi de spirale entrante définie par $\alpha^* = \frac{15\pi}{32}$. Le centre de la spirale est défini au coordonnées $(0, 5)$ dans le repère initial du robot. Comme le montre la figure 2.20a, le robot exécute bien une spirale entrante. Cependant le robot, au bout d'un certain temps, n'est plus capable de suivre la spirale de référence car celui-ci atteint la saturation de sa vitesse angulaire. En effet, la figure 2.20c montre que la valeur d'angle d'orientation des roues souhaitée $\gamma^*(t)$ dépasse sa valeur de butée ($\gamma_{MIN-MAX}$). C'est pourquoi, dans la figure 2.20e, l'angle $\alpha(t)$ converge vers α^* puis diverge lorsque le robot ne peut plus suivre la spirale définie. Une autre expérimentation a été conduite de façon à suivre une spirale sortante caractérisée par $\alpha^* = \frac{17\pi}{32}$. Le centre de cette spirale est défini aux coordonnées $(0, 3)$ dans le repère initial du robot. Cette fois, comme le montrent les résultats présentés par les figures 2.20b, 2.20d et 2.20f, la valeur maximale d'orientation des roues n'est pas atteinte. Ainsi, il peut être observé que le robot exécute bien le suivi de spirale désiré.

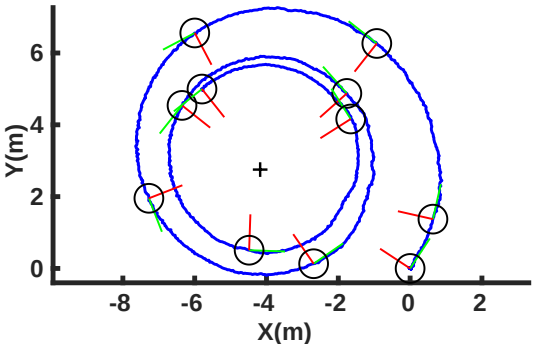
Ainsi, cette loi de commande permet de garantir que l'angle $\alpha(t)$ converge vers l'angle α^*. Il est donc possible de contrôler le robot en augmentant, diminuant ou encore en gardant constante la distance d du centre de la spirale. Cependant, il est à noter que le robot ne suit pas de spirale spécifique mais adopte le comportement du type de spirale défini par l'angle α^* imposé. En effet, la prise en compte de ce seul paramètre caractérise une infinité de spirales. C'est pourquoi, nous avons développé un nouveau correcteur permettant de contrôler à la fois l'angle $\alpha(t)$ mais aussi la distance $d(t)$.

Suivi de spirale : Correcteur $\alpha - d$:

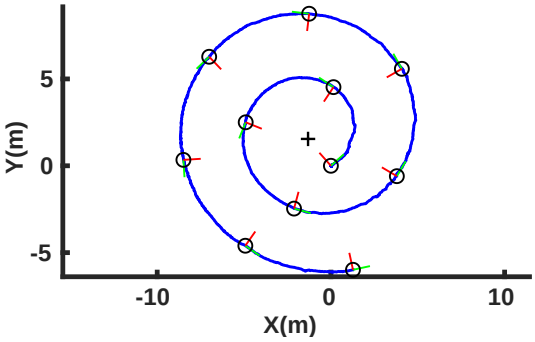
Dans un contexte de demi-tours dans les vergers, il peut être nécessaire de contrôler le robot pour lui permettre de converger vers une spirale spécifique. Celle-ci est caractérisée par ses paramètres α^* et $d^*(0)$. Comme précédemment la vitesse du robot est définie comme constante $v(t) = v \neq 0$. L'erreur du robot est donc reformulée pour prendre en compte l'erreur en distance entre le robot et la spirale :

$$e_S(t) = \alpha(t) - \alpha_S(t) = \alpha(t) - \alpha^* - \alpha_D(t)\varepsilon(t) \tag{2.13}$$

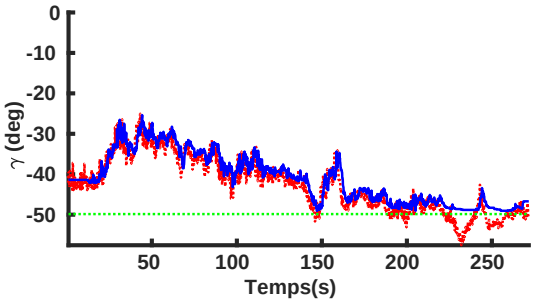
L'angle de référence constant α^* du précédent correcteur est remplacé par l'angle $\alpha_S(t) = \alpha^* + \alpha_D\varepsilon(t)$ variable en fonction de la distance entre le robot et la spirale



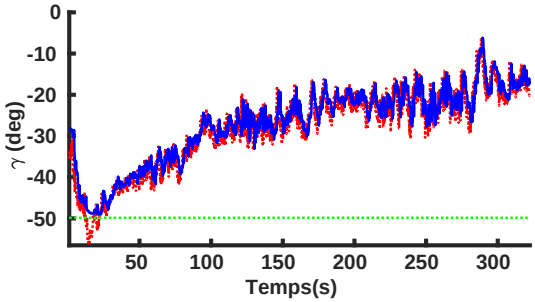
(a) Trajectoire du robot avec $\alpha^* = \frac{15\pi}{32}$ - Ligne bleue continue : Position du robot - Croix noire : Centre de la spirale - Croix verte : Position initiale - Croix rouge : Position finale



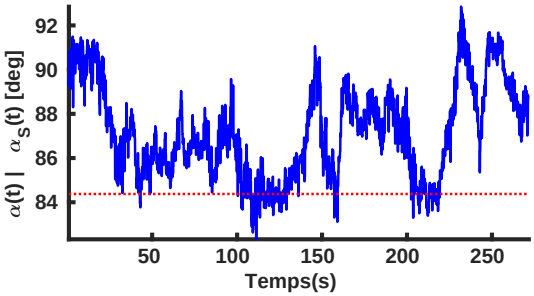
(b) Trajectoire du robot ($\alpha^* = \frac{17\pi}{32}$) - Ligne bleue continue : Position du robot - Croix noire : Centre de la spirale - Croix verte : Position initiale - Croix rouge : Position finale



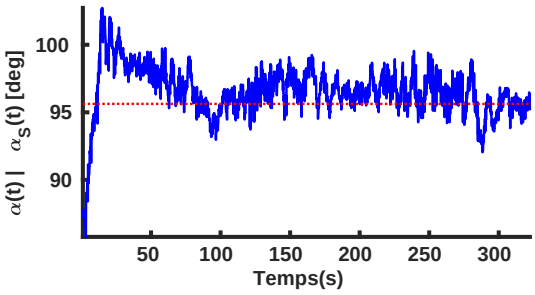
(c) Ligne bleue continue : $\gamma(t)$ - Ligne rouge pointillée : $\gamma^*(t)$ - Ligne verte pointillée : γ_{MIN}



(d) Ligne bleue continue : $\gamma(t)$ - Ligne rouge pointillée : $\gamma^*(t)$ - Ligne verte pointillée : γ_{MIN}



(e) Ligne bleue continue : $\alpha(t)$ - Ligne rouge pointillée : α^*



(f) Ligne bleue continue : $\alpha(t)$ - Ligne rouge pointillée : α^*

FIGURE 2.20 – Suivi de spirale dans un cas réel

de référence. Ainsi, l'annulation de l'erreur définie dans l'équation (2.13) mène à deux comportements successifs du robot. Lorsque $d(t) \neq d^*(t)$, le robot a besoin dans un premier temps de se rapprocher vers la spirale pour converger vers $d(t) = d^*(t)$. Pour ce faire, l'angle de référence $\alpha_S(t)$ a été modifié par l'introduction d'un angle $\alpha_D \varepsilon(t)$. Ce nouvel angle α_D est l'angle maximal pouvant être soustrait ou ajouté à α^* pour faire converger le robot vers la spirale sans que celui-ci ne change de sens de rotation par rapport au centre de la spirale. Ensuite le paramètre $\varepsilon(t)$ doit être défini comme égal à 1 lorsque $||d(t) - d^*(t)|| \geq ||d(0) - d^*(0)||$ et égal à 0 lorsque $d(t) = d^*(t)$. Ainsi, lorsque le robot est loin de la spirale à suivre, celui-ci converge en utilisant le plus grand angle possible, *i.e.* $\alpha_S(t) = 0$ ou $\alpha_S(t) = \pm\pi$ suivant les conditions initiales du robot. De plus, lorsque le robot atteint $d(t) = d^*(t)$ alors l'angle désiré $\alpha_S(t) = \alpha^*$ correspond

alors à l'angle désiré défini pour la loi de commande précédente. De façon à obtenir le comportement souhaité, la variable $\varepsilon(t)$ et l'angle α_D sont proposés comme suit :

$$\varepsilon(t) = \text{signe}(d^*(t) - d(t))\min\left(\left\|\frac{d^*(t) - d(t)}{|d^*(0) - d(0)|}\right\|, 1\right) \tag{2.14}$$

et

$$\alpha_D = \begin{cases} \text{signe}(\alpha^*) * \pi - \alpha^* & \text{si } d^*(0) > d(0) \\ \alpha^* & \text{si} d^*(0) < d(0) \end{cases} \tag{2.15}$$

La variable $\varepsilon(t)$ est définie comme l'erreur normalisée entre $d^*(t)$ et $d(t)$. Sa valeur appartient au domaine $[-1, 1]$. En effet lorsque $d^*(0) > d(0)$ alors $\varepsilon(t) \in [0, 1]$ et lorsque $d^*(0) < d(0)$ alors $\varepsilon(t) \in [-1, 0]$. Ainsi, $d(t) \neq d^*(t)$ implique $\alpha_S(t) \neq \alpha^*$ et $d(t) = d^*(t)$ implique $\alpha_S = \alpha^*$. De plus, d'après l'équation (2.15), lorsque $\alpha^* \in [0, \pi]$ alors $\alpha_S \in [0, \pi]$ et lorsque $\alpha^* \in [0, -\pi]$ alors $\alpha_S \in [0, -\pi]$. L'équation (2.14) et (2.15) garantissent que le terme $\alpha_D\varepsilon(t)$ lors de la convergence vers $d^*(t)$ ne forcera pas le robot à changer de sens de rotation.

La nouvelle erreur définie est dérivée par rapport au temps afin d'exprimer sa dynamique.

$$\begin{aligned} \dot{e}_s(t) &= \dot{\alpha}(t) - \alpha_D\dot{\varepsilon}(t) \\ &= -\omega(t) + \frac{v}{d(t)}\sin(\alpha(t)) - \alpha_D\dot{\varepsilon}(t) \end{aligned} \tag{2.16}$$

Afin d'annuler $e_S(t)$, une décroissance exponentielle est imposée :

$$\omega(t) = \lambda_S e_S(t) + \frac{v(t)}{d(t)}\sin(\alpha(t)) - \alpha_D\dot{\varepsilon}(t) \tag{2.17}$$

Où λ_S un scalaire positif permettant de fixer la vitesse de convergence de $e_S(t)$ vers zéro. Ainsi la loi de commande conçue dans l'équation (2.17) permet au robot de suivre tout type de spirales.

De façon à analyser la stabilité du système rebouclé, la fonction de Lyapunov suivante est suggérée :

$$V_S(x_S(t)) = \frac{x_{S1}^2(t) + x_{S2}^2(t)}{2} \tag{2.18}$$

Avec

$$x_S(t) = \begin{bmatrix} x_{S1}(t) \\ x_{S2}(t) \end{bmatrix} = \begin{bmatrix} \alpha(t) - \alpha^* - \alpha_D\varepsilon(t) \\ d(t) - d^*(t) \end{bmatrix}$$

L'état d'équilibre est donné par $x_{SE} = [x_{S1}(t), x_{S2}(t)]^T = [0, 0]^T$ qui correspond à la situation du robot lorsqu'il suit la spirale pré-établie, *i.e.* $\alpha(t) = \alpha^*$ et $d(t) = d^*(t)$. De plus, la fonction $V_S(x_S(t))$ est définie positive pour tout $x_S(t) \neq x_{SE}$ et $V_S(x_S(t)) = 0$ pour $x_S(t) = x_{SE}$. De façon à étudier l'évolution de cette fonction, sa dérivée en fonction du temps est calculée.

$$\begin{aligned} \dot{V}_S(x_S(t)) &= \dot{x}_{S1}(t)x_{S1}(t) + \dot{x}_{S2}(t)x_{S2}(t) \\ &= \lambda_S e_S^2(t) + [\dot{d}(t) - \dot{d}^*(t)][d(t) - d^*(t)] \\ &= \lambda_S e_S^2(t) - v[\cos(\alpha(t)) - \cos(\alpha^*)][d(t) - d^*(t)] \end{aligned} \tag{2.19}$$

L'analyse effectuée à travers l'équation (2.19) montre que $\dot{V}_S(x_S(t)) = 0$ pour $x_S(t) = x_{SE}$. Cependant, cette fonction ne permet pas de prouver que $\dot{V}_S(x_S(t)) < 0$ pour tout

$x_S(t) \neq x_{SE}$. En effet, même si le premier terme $\lambda_S e_S^2(t)$ est strictement négatif pour tout $x_S(t) \neq x_{SE}$, le second terme $v[\cos(\alpha(t)) - \cos(\alpha^*)][d(t) - d^*(t)]$ quant à lui ne l'est pas toujours. Suivant les conditions initiales le signe de $\cos(\alpha(t)) - \cos(\alpha^*)$ peut être différent du signe de $d(t) - d^*(t)$. Par ailleurs, l'orientation initiale du robot et l'utilisation d'un modèle de robot non-holonome avec une vitesse linéaire constante v peut s'opposer à la décroissance de $||d(t) - d^*(t)||$. En effet, lors du début de la navigation le robot peut s'éloigner du centre de la spirale le temps de se réorienter vers α_S. Ainsi, cette fonction de Lyapunov ne permet pas de prouver que cette loi de commande est globalement asymptotiquement stable. Par contre, il peut être montré que :

$$\begin{aligned} &\text{Si } d(t) > d^*(t) : \cos(\alpha(t)) - \cos(\alpha^*) > 0 \text{ pour } \alpha(t) > \alpha^* \\ &\text{Si } d(t) < d^*(t) : \cos(\alpha(t)) - \cos(\alpha^*) < 0 \text{ pour } \alpha(t) < \alpha^* \end{aligned} \tag{2.20}$$

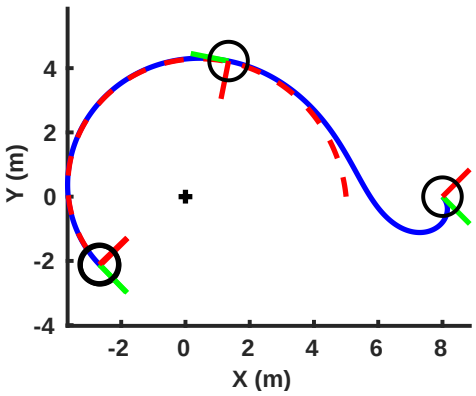
L'équation (2.20) montre alors que la loi de commande est localement asymptotiquement stable lorsque $\alpha(t)$ dépasse α^*. Les résultats obtenus ne satisfont pas entièrement nos espérances, cependant ils suffisent dans le cadre de notre projet. La preuve d'une stabilité asymptotique et globale pourrait éventuellement être trouvée par la création d'une fonction de Lyapunov plus complexe, impliquant le rayon de braquage maximum du robot. Une autre solution pourrait être la relaxation de la contrainte sur la distance $d(t)$ lors de la phase de réorientation grâce au correcteur défini dans l'équation (2.10). Puis de changer de correcteur lorsque le robot a convergé vers α^* pour s'asservir sur la spirale définie par α^* et $d^*(t)$ avec le correcteur de l'équation (2.17).

Résultats :

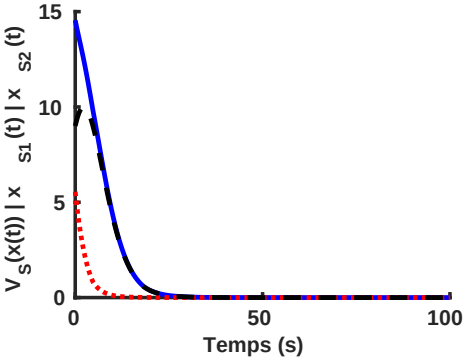
Les résultats qui suivent illustrent les performances et les limites du correcteur décrit par l'équation (2.17). De la même façon que pour le correcteur précédent, une première partie évalue les résultats effectués en simulation puis une seconde présente ceux obtenus via la mise en place du correcteur sur le robot réel. Toutes les expérimentations sont réalisées avec une période d'échantillonnage $T_s = 100$ms.

Dans le premier test présenté dans la figure 2.21, la spirale de référence est définie par $\alpha^* = \frac{15\pi}{32}$ et initialisée à une distance $d^*(0) = 5$m du centre de la spirale localisé en $(0, 0)$. L'état du robot au début de la tâche est fixé à $\chi(0) = [8, 0, -\frac{\pi}{4}]^T$. La vitesse linéaire du robot et de la spirale sont imposées constantes à $v = 0.2$m/s et le gain de la loi de commande est réglé à $\lambda_S = 0.2$.

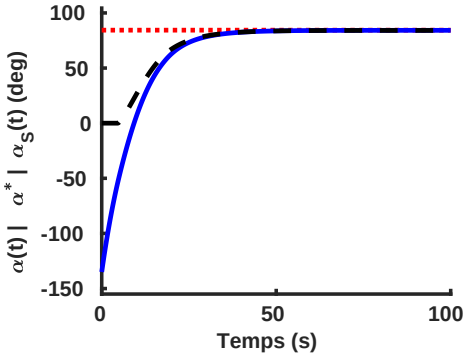
La figure 2.21a montre que le robot suit avec succès la spirale prédéfinie en gérant simultanément la convergence de son orientation $\alpha(t)$ vers celle de la spirale α^* (figure 2.21c) ainsi que la convergence de sa distance $d(t)$ vers $d^*(t)$. La figure 2.21b illustre l'évolution de la fonction de Lyapunov $V_S(x_S(t))$ décrite dans l'équation (2.18). L'évolution de $x_{S1}(t) = e_S(t)$ montre que celle-ci décroît rapidement jusqu'à converger vers zéro, tandis que $x_{S2}(t) = d(t) - d^*(t)$ croit légèrement avant de converger vers zéro et ne s'annule qu'après la stabilisation de $x_{S1}(t)$ à zéro. Cet effet est dû à la mauvaise orientation initiale du robot non-holonome. En conséquence, ce dernier a besoin de s'orienter dans un premier temps vers la spirale de référence de façon à annuler l'erreur $e_S(t)$. Durant le début de cette phase d'orientation, le robot s'écarte de la spirale augmentant alors $x_{S2}(t)$. Une fois la réorientation partiellement effectuée, le robot converge vers la spirale et la suit progressivement. Il peut aussi être remarqué que la fonction $V_S(x_S(t))$ décroît continuellement jusqu'à se stabiliser à zéro, ce qui se produit lorsque le robot a atteint et suit la spirale. Dans ce cas, l'augmentation de x_{S2} n'est pas assez



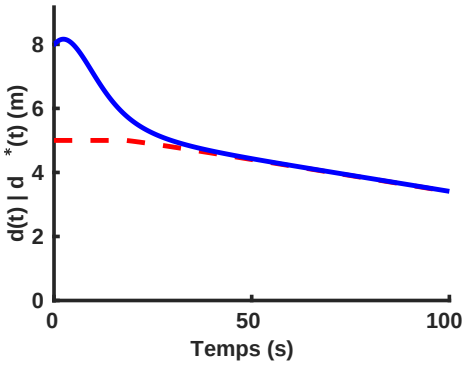
(a) Trajectoire du robot - Ligne bleue continue : Position du robot - Ligne pointillée rouge : Spirale de référence



(b) Fonction de Lyapunov - Ligne bleue continue : $V_S(x_s(t))$ - Ligne rouge pointillée : x_{S1} - Ligne tiretée noires : x_{S2}



(c) Mesure de l'angle $\alpha(t)$ - Ligne bleue continue : $\alpha(t)$ - Ligne rouge pointillée : α^* - Ligne tiretée noire : $\alpha_S(t)$



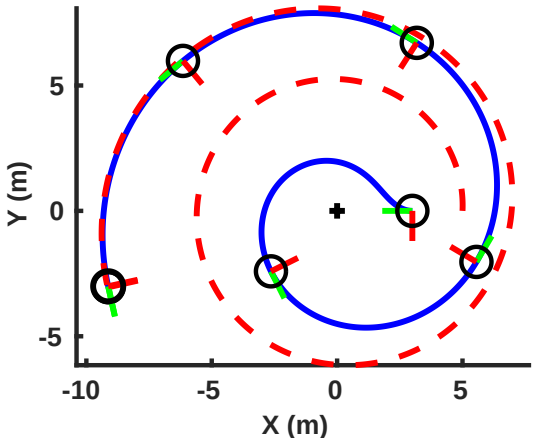
(d) Mesure de la distance $d(t)$ - Ligne bleue continue : $d(t)$ - Ligne rouge pointillée : $d^*(t)$

FIGURE 2.21 – Suivi de spirale utilisant le correcteur α-d avec $\alpha^* = \frac{15\pi}{32}$

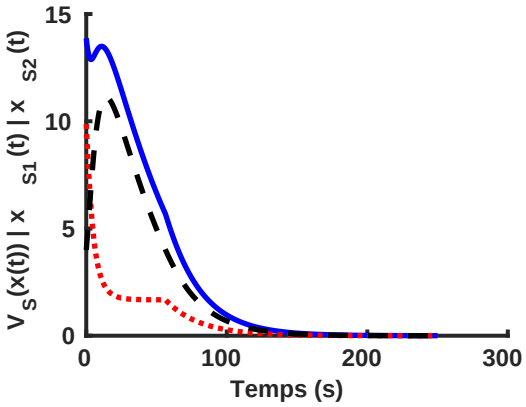
rapide par rapport à convergence de $x_{S1}(t)$ pour empêcher la fonction $V_S(x_S(t))$ de cesser de décroître de manière monotone. Ainsi, les résultats présentés sont en accord avec l'analyse de Lyapunov présentée dans l'équation (2.18). L'utilisation de la fonction $V_S(x_s(t)) = \frac{x_{S1}^2(t)}{2}$ n'aurait pas été suffisante pour garantir le suivi de la spirale par le robot. En effet, la variable $x_{S1}(t)$ est nulle avant qu'il ait pu atteindre la spirale souhaitée. De plus, seule la convergence de l'orientation permet à $x_{S2}(t)$ et $V_S(x_S(t))$ de décroître à leur tour. Néanmoins, malgré la mauvaise orientation initiale du robot, celui-ci est capable d'accomplir la tâche souhaitée.

Le second ensemble de simulations présenté dans la figure 2.22 a pour objectif le suivi d'une spirale sortante initialisée à $\alpha^* = \frac{17\pi}{32}$ avec une distance $d(0) = 5$m dont le centre se situe aux coordonnées $(0, 0)$. Le robot est quant à lui initialisé à $\chi(0) = [3, 0, \pi]^T$. Le gain de la loi de commande est réglé à $\lambda_S = 0.02$.

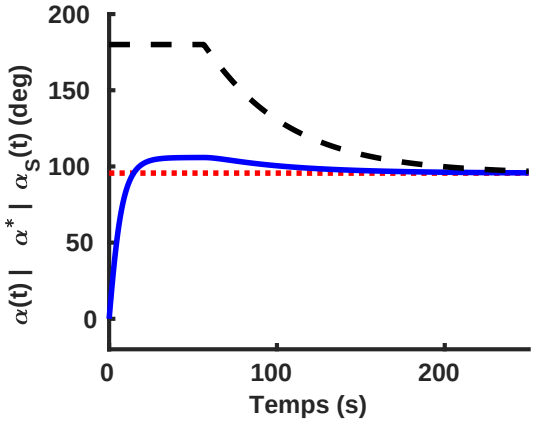
Comme observé dans la figure 2.22a, le robot atteint la spirale de référence, malgré un temps de convergence assez long. De la même manière que la simulation précédente, la mauvaise orientation initiale du robot nécessite une étape de réorientation pendant laquelle $x_{S2}(t)$ doit croître comme le montrent les figures 2.22d et 2.22b. De plus, afin d'annuler $x_{S2}(t)$, $\alpha(t)$ doit dépasser la valeur de α^* comme illustré dans la figure 2.22c. Cet effet est dû au terme $\alpha_D * \varepsilon(t)$ de l'équation (2.13) qui compense l'erreur en distance par l'ajout d'un angle afin de converger vers la spirale de référence. Cependant, cet ajout



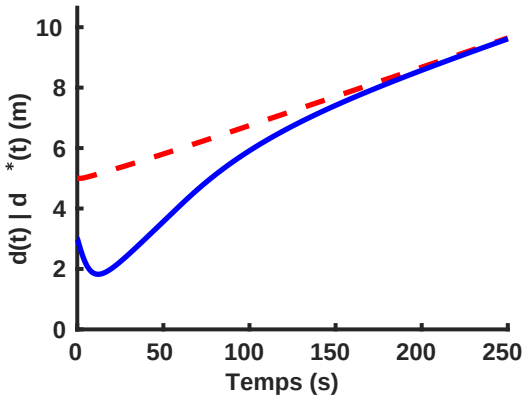
(a) Trajectoire du robot - Ligne bleue continue : Position du robot - Ligne pointillée rouge : Spirale de référence



(b) Fonction de Lyapunov - Ligne bleue continue : $V_B(t)$ - Ligne rouge pointillée : x_{S1} - Ligne noire tiretée : x_{S2}



(c) Mesure de l'angle $\alpha(t)$ - Ligne bleue continue : $\alpha(t)$ - Ligne rouge pointillée : α^* - Ligne noire tiretée : $\alpha_S(t)$



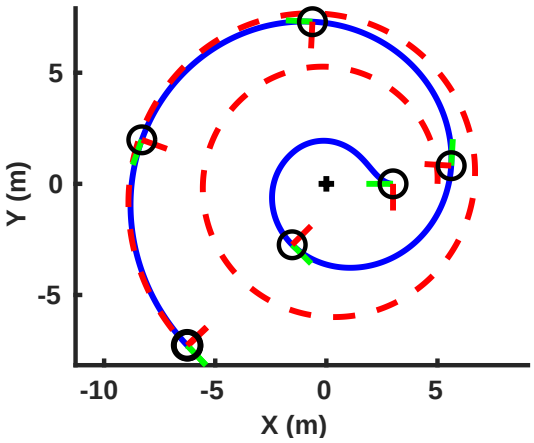
(d) Mesure de la distance $d(t)$ - Ligne bleue continue : $d(t)$ - Ligne rouge pointillée : $d^*(t)$

FIGURE 2.22 – Suivi de spirale utilisant le correcteur α-d avec $\alpha^* = \frac{17\pi}{32}$

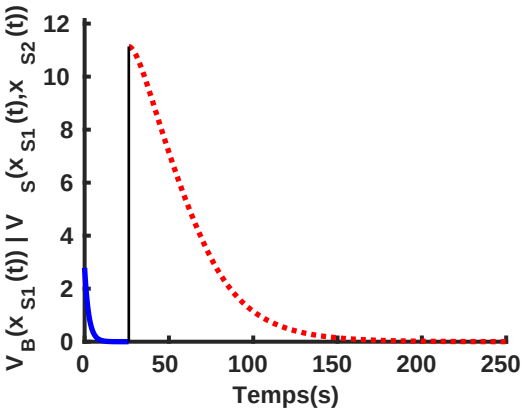
ne mène pas à l'accroissement de x_{S1} durant l'exécution de la tâche (figure 2.22b). Dans notre cas, la réduction du gain $\lambda_S = 0.2$ par rapport à la première simulation induit un accroissement de $V_S(x_S(t))$ au début de la simulation. En effet, la variation de $x_{S2}(t)$ causée par la phase de réorientation du robot n'est plus compensée par la forte décroissance de $x_{S1}(t)$. La décroissance de $V_S(x_S(t))$ est alors seulement garantie lorsque la phase de réorientation a été correctement réalisée. Cette simulation montre un cas pour lequel la fonction de Lyapunov utilisée n'assure pas la stabilité globale du système. En contrepartie, on peut remarquer que le robot achève avec succès la tâche de convergence et de suivi de spirale.

Pour pallier à ces problèmes de stabilité asymptotique globale, il a été proposé d'utiliser successivement les deux lois de commande présentées jusqu'ici. L'analyse du comportement du robot utilisant cette solution est présenté dans la figure 2.23.

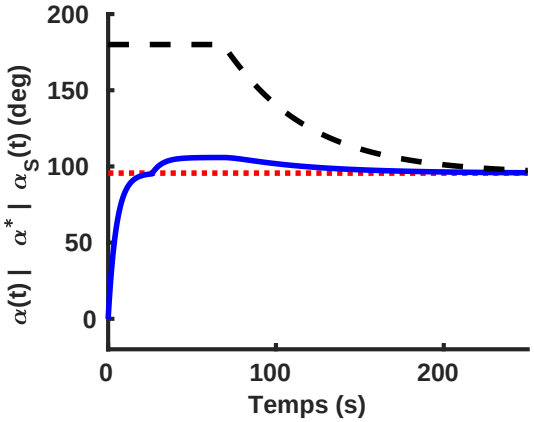
Initialement le robot est alors asservi par le premier correcteur et ainsi ne suit pas de spirale spécifique mais s'oriente vers α^*. Dès lors que $\alpha(t) = \alpha^*$, le robot est commandé avec le second correcteur et suit la spirale spécifique définie par α^* et $d(t)$. Dans ce cas, la spirale à suivre et l'état initial du robot sont identiques à la simulation précédente (i.e., $\alpha^* = \frac{17\pi}{32}$, $d(0) = 5$ m, $\chi(0) = [3, 0, \pi]^T$). Le premier correcteur est alors utilisé afin d'atteindre l'état du robot qui garantira la convergence du second



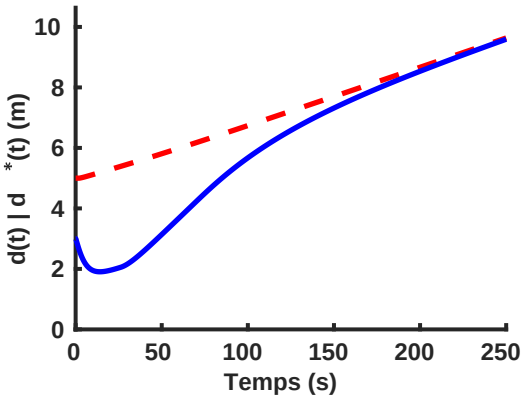
(a) Trajectoire du robot - Ligne bleue continue : Position du robot - Ligne rouge pointillée : Spirale de référence



(b) Fonction de Lyapunov - Ligne bleue continue : $V_S(x_S(t))$ - Ligne rouge pointillée : $V_S(x_S(t))$



(c) Mesure de l'angle $\alpha(t)$ - Ligne bleue continue : $\alpha(t)$ - Ligne rouge pointillée : α^* - Ligne noire tiretée : $\alpha_S(t)$



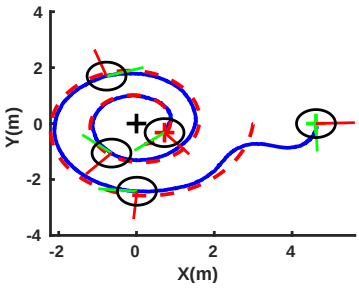
(d) Mesure de la distance $d(t)$ - Ligne bleue continue : $d(t)$ - Ligne rouge pointillée : $d^*(t)$

FIGURE 2.23 – Suivi de spirale utilisant les deux correcteurs α-d avec $\alpha^* = \frac{15\pi}{32}$

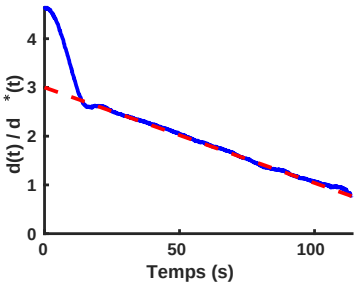
correcteur. En effet, dans la figure 2.23b, les deux fonctions de Lyapunov $V_B(t)$ et $V_S(t)$ sont décroissantes. De plus, l'évolution de la trajectoire observée dans la figure 2.23a ainsi que celle des paramètres $\alpha(t)$ et $d(t)$ montrées sur les figures 2.23c et 2.23d sont similaires aux résultats observés dans la simulation précédente. Cette approche permet de garantir la convergence mais ne change pas réellement les performances de l'asservissement.

Les tests suivants montrent l'effet de cette loi de commande (équation (2.17)) sur le robot réel naviguant dans un environnement extérieur. Deux cas de suivi de spirales sont présentés dans la figure 2.24. La première ligne de résultats illustre le cas d'une spirale entrante de centre $(0, 0)$ et initialisée à $d^*(t) = d^* = 4$ m et $\alpha^* = -\frac{15\pi}{32}$ rad. Le robot est quant à lui positionné à une distance $d(0) = 5$ m. La seconde ligne représente le cas d'une spirale sortante elle aussi de centre $(0, 0)$ et initialisé avec $d^* = 3$ m et $\alpha^* = \frac{17\pi}{32}$. Ici, le robot est positionné à une distance $d(0) = 1.73$ m.

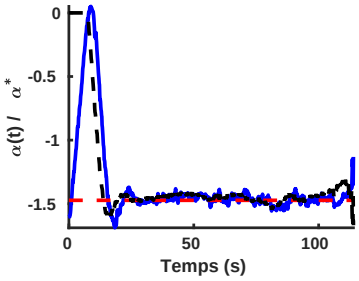
Il peut être remarqué en observant les figures 2.24d et 2.24a que dans les deux cas le robot parvient à converger vers la spirale puis à la suivre. De plus, les figures 2.24b et 2.24e montrent que le robot atteint avec succès la distance souhaitée d^* et réussit à la maintenir. Les figures 2.24c et 2.24f, illustrent comment l'évolution de $\alpha(t)$ et $\alpha_S(t)$ est gérée afin de réaliser ces tâches. Dans ces deux différentes expérimentations, la valeur



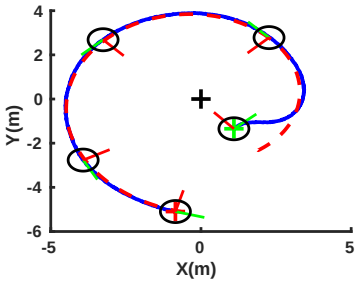
(a) Trajectoire du robot - Ligne bleue continue : Position du robot - Ligne rouge pointillée : Spirale de référence



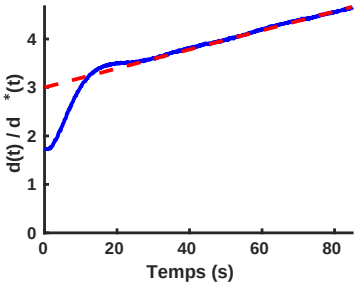
(b) Mesure de la distance $d(t)$ - Ligne bleue continue : $d(t)$ - Ligne rouge pointillée : $d^*(t)$



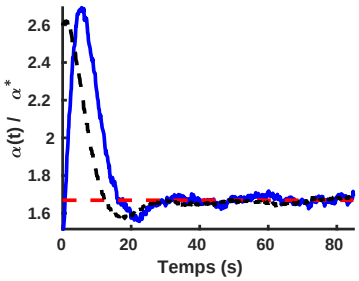
(c) Mesure de l'angle $\alpha(t)$ - Ligne bleue continue : $\alpha(t)$ - Ligne rouge pointillée : α^* - Ligne noire tiretée : $\alpha_S(t)$



(d) Trajectoire du robot - Ligne bleue continue : Position du robot - Ligne rouge pointillée : Spirale de référence



(e) Mesure de la distance $d(t)$ - Ligne bleue continue : $d(t)$ - Ligne rouge pointillée : $d^*(t)$



(f) Mesure de l'angle $\alpha(t)$ - Ligne bleue continue : $d(t)$ - Ligne rouge pointillée : $d^*(t)$

FIGURE 2.24 – Expérimentations de suivi de spirale

de $\alpha_S(t)$ est complètement différente car celle-ci doit d'abord s'orienter vers la spirale avant de converger vers α^* et doit rester sur la spirale désirée.

La conception d'une loi de commande permettant de suivre une spirale spécifique a donc été mise en place. Cette commande asservit le robot grâce aux paramètres α et $d(t)$ directement fournis par les capteurs embarqués. L'analyse de la stabilité a néanmoins montré que le système rebouclé n'est que localement asymptotiquement stable et ce résultat a été illustré. Cependant, en pratique il a été observé que, même si l'erreur en distance augmente pendant la phase de réorientation, le robot parvient quand même à converger puis à rester sur la spirale désirée. Cependant, pour résoudre ces problèmes de stabilité, une dernière loi de commande inspirée par l'automatique classique a été synthétisée afin de suivre une spirale spécifique.

Correcteur de suivi de spirale par linéarisation exacte entrée/sortie :

Malgré des résultats de simulation et d'expérimentation satisfaisants, il a semblé intéressant d'explorer une autre approche plus classique afin de synthétiser un correcteur de suivi de spirale. L'objectif était d'obtenir des performances similaires tout en obtenant une preuve de stabilité plus générale.

Nous rappelons que le suivi d'une spirale particulière équivaut à faire converger la distance $d(t)$ et l'angle $\alpha(t)$ vers $d^*(t)$ et α^*. Pour ce faire, nous définissons en premier lieu les erreurs suivantes :

$$\begin{cases} e_d(t) = d(t) - d^*(t) \\ e_\alpha(t) = \alpha(t) - \alpha^* \end{cases} \tag{2.21}$$

L'objectif est maintenant de concevoir un correcteur faisant converger asymptotiquement vers zéro les erreurs définies par le système de l'équation (2.21). Pour des systèmes non-linéaires, comme c'est ici le cas, une approche classique consiste à récrire le système afin d'obtenir une linéarisation exacte entrée/sortie [Isidori, 2013]. Dans un premier temps, nous rappelons que comme précédemment, nous imposons $v(t) = v \neq 0$. Ainsi, le robot est contrôlé uniquement avec sa vitesse angulaire $\omega(t)$. A présent, nous remodelons le système afin de le mettre sous la forme adéquate. La première étape consiste à calculer la dynamique des erreurs (2.21) qui peut s'exprimer sous la forme :

$$\begin{cases} \dot{e}_d(t) = v[\cos(\alpha^*) - \cos(\alpha(t))] \\ \dot{e}_\alpha(t) = -\omega(t) + \dfrac{v}{d(t)} \sin(\alpha(t)) \end{cases} \tag{2.22}$$

En combinant les équations (2.21) et (2.22), nous obtenons :

$$\begin{cases} \dot{e}_d(t) = v[\cos(\alpha^*) - \cos(e_\alpha(t) + \alpha^*)] \\ \dot{e}_\alpha(t) = -\omega(t) + \dfrac{v}{e_d(t) + d^*(t)} \sin(e_\alpha(t) + \alpha^*) \end{cases} \tag{2.23}$$

Afin d'obtenir un modèle linéaire, il est proposé de définir un nouveau vecteur d'état z comme le montre l'équation (2.24) L'idée principale de cette méthode est de linéariser le système d'erreur de l'équation (2.23). Pour cela la transformation suivante est appliquée :

$$z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} z_1 \\ \dot{z}_1 \end{bmatrix} = \begin{bmatrix} e_d(t) \\ v[\cos(\alpha^*) - \cos(e_\alpha(t) + \alpha^*)] \end{bmatrix} = T(e) \tag{2.24}$$

Ce nouvel état du système est relié au vecteur d'erreur $e = [e_d, e_\alpha]^T$ à l'aide du difféomorphisme T. Il est important ici de remarquer que $T(0) = 0$ appartient au domaine D défini par :

$$D = \left\{ (e_d, e_\alpha) \in \Re^2 | e_d \in \Re, e_\alpha \in]-\alpha^*, -\alpha^* + \pi[\right\} \tag{2.25}$$

L'équation dynamique de ce nouveau système est de la forme :

$$\begin{cases} \dot{z}_1 = z_2 \\ \dot{z}_2 = v \sin(e_\alpha(t) + \alpha^*) \dot{e}_\alpha(t) \end{cases} \tag{2.26}$$

Cette dernière équation peut être combinée à l'équation (2.23), afin de faire apparaître la variable de commande $\omega(t)$.

$$\begin{cases} \dot{z}_1 = z_2 \\ \dot{z}_2 = v \sin(e_\alpha(t) + \alpha^*)(-\omega(t) + \dfrac{v}{e_d(t) + d^*(t)} \sin(e_\alpha(t) + \alpha^*)) \end{cases} \tag{2.27}$$

Le second terme de la deuxième ligne étant non-linéaire, il est proposé de le substituer par $\omega_2(t)$, une nouvelle loi de commande, afin de réaliser une linéarisation exacte, telle que :

$$\begin{cases} \dot{z}_1 = z_2 \\ \dot{z}_2 = \omega_2(t) \end{cases} \tag{2.28}$$

Notons que pour $\omega_2(t)$ donnée, il est possible de calculer $\omega(t)$ à l'aide de la relation suivante :

$$\omega(t) = \frac{1}{v\sin(e_\alpha(t)+\alpha^*)}\omega_2(t) + \frac{v}{e_d(t)+d^*(t)}\sin(e_d(t)+\alpha^*(t)) \tag{2.29}$$

Désormais, le système décrit dans l'équation (2.28) est linéarisé. Il est donc possible de mettre en place un retour d'état linéaire classique :

$$\omega_2(t) = -\lambda_1 z_1(t) - \lambda_2 z_2(t) \tag{2.30}$$

avec λ_1, λ_2 deux scalaires positifs stabilisant le système (2.28). Ainsi, le théorème suivants est proposé :

Théorème 1 *Considérant deux scalaires positifs λ_1 et λ_2, le système d'erreurs présenté dans l'équation (2.22) rebouclé avec la loi de commande définie dans les équations (2.29), le système linéarisé dans l'équation (2.28), la loi de commande du système linéarisé de l'équation (2.30) et avec $z = T(e)$ dans l'équation (2.24) alors le système en boucle fermée est localement asymptotiquement stable.*

Preuve 1 *Il est suffisant de remarquer que z converge asymptotiquement vers zéro et que $z = T(e)$ définit un difféomorphisme avec $T(0) = 0$.*

Remarque 2 *λ_1 et λ_2 sont deux gains positifs permettant le réglage de la vitesse de convergence du système de la boucle fermée.*

Remarque 3 *Le terme $\sin(e_\alpha(t)+\alpha^*)$ présent dans le dénominateur de l'équation (2.29) fait apparaître des singularités lorsque $\alpha(t) = k\pi$ avec $k \in \mathbb{N}$. Cela peut se produire lorsque le robot fait face à l'arbre de référence ou le laisse dans son dos. Bien que cela soit un problème théorique majeur, les situations menant à la singularité sont quasi inaccessibles dans le cadre notre application. En effet, pour que le robot ait le temps et l'espace pour s'orienter face (ou dos) à l'arbre afin de converger vers la spirale désirée, il est nécessaire que la distance entre la position courante du robot et la spirale soit très grande. Dans la quasi totalité des cas, une légère ré-orientation du robot suffit à atteindre la spirale visée, comme cela sera montré dans les exemples de simulation.*

Remarque 4 *Afin de définir l'état de notre système en coordonnées polaires, il est nécessaire d'utiliser un vecteur de d'état de dimension trois, comme par exemple $[d(t), \alpha(t), \beta(t)]$. Or, le correcteur présenté ici ne permet d'asservir que deux des trois états. Autrement dit, le degré relatif de la sortie e_d choisie pour la linéarisation est deux, tandis que la dimension de l'état du système est de trois. Un des états n'est donc pas commandé et il existe une dynamique des zéros. Cette dernière est définie par :*

$$\dot{e}_\beta(t) = \dot{\beta}(t) - \dot{\beta}^*(t) = v(-\frac{\sin(\alpha(t))}{d(t)} + \frac{\sin(\alpha^*)}{d^*}) \tag{2.31}$$

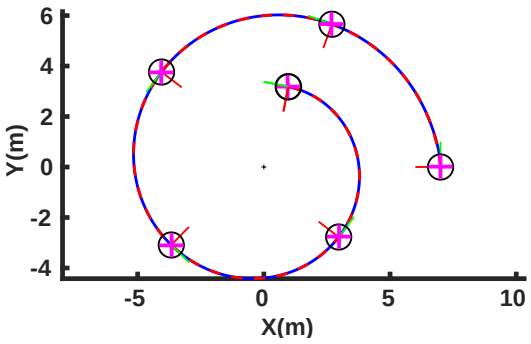
D'après [Isidori, 2013], il est trivial de remarquer que la dynamique des zéros définie par $\dot{e}_\beta(t) = 0$ est uniquement stable. De ce fait, en régime permanent, une erreur statique entre $\beta(t)$ et sa valeur désirée $\beta^*(t)$ est attendue. Il sera montré dans les résultats obtenus en simulation que ce retard est de petite amplitude dans notre cadre d'utilisation, et ne présente pas d'obstacle majeur à la réalisation de la tâche.

Résultats :

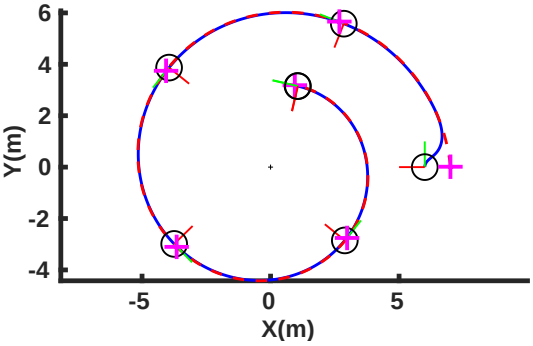
Ce correcteur a été testé en simulation. Les résultats sont illustrés dans la figure 2.25. Ceux-ci montrent l'évolution des paramètres du système avec les gains $\lambda_1 = 0.1$ et $\lambda_2 = 0.5$. La première colonne (*i.e.* Figures 2.25a, 2.25c, 2.25e et 2.25g) présente le comportement du robot lorsqu'il démarre sur la spirale de référence. Pour cela, le robot est initialisé à $\chi(0) = [7, 0, \pi/2]^T$ et a pour objectif de suivre une spirale entrante définie par $\alpha^* = \frac{15\pi}{32}$ et $d^*(0) = 7$m. La seconde colonne (*i.e.* Figures 2.25b, 2.25d, 2.25f et 2.25h) présente le comportement du robot lorsqu'il ne démarre pas sur la spirale. Ainsi, le robot est initialisé avec $\chi(0) = [6, 0, \pi/2]^T$ et suit la même spirale que celle définie précédemment. La figure 2.25a montre que l'exécution du suivi de spirale lorsque le robot se situe sur celle-ci à l'initialisation est bien accomplie. Dans la figure 2.25b, le robot converge dans un premier temps vers la spirale puis la suit. Cependant, la trajectoire n'est pas précisément sur la spirale de référence. Comme indiqué dans la remarque 4, l'erreur observée est due à l'erreur statique sur $\beta(t)$ dans le régime permanent. En effet, dans les figures 2.25c et 2.25d montrant l'évolution de $\beta(t)$ et β^*, il est constaté un léger décalage lorsque le robot ne commence pas sur la spirale. Cependant, cette erreur statique sur $\beta(t)$ ne pénalise pas les performances de l'asservissement du point de vue de $e_\alpha(t)$ et $e_d(t)$ qui convergent tous les deux vers zéro (figures 2.25e et 2.25f). Enfin, les figures 2.25g et 2.25h affichent l'évolution de la commande $\omega(t)$ utilisée pour suivre la spirale de référence. Dans les deux cas, la commande apparaît appropriée pour une utilisation sur un robot réel.

Bien que le correcteur proposé ne garantisse lui aussi qu'une stabilité locale, il permet au robot de converger et de suivre une spirale précise définie par ses paramètres α^* et $d^*(t)$ et est capable de la suivre. De plus, une erreur statique sur le paramètre $\beta(t)$ est observée. Cette erreur peut s'expliquer par la phase de convergence du robot vers la spirale. En effet, le robot et la spirale ayant la même vitesse linéaire, le temps pris par le robot robot pour atteindre la spirale peut être vue comme un retard du robot sur la spirale. Cet effet est assez minime mais peut aussi être constaté avec le correcteur $\alpha - d$. Enfin, les singularités relevées pour $\alpha(t) = k\pi$ peuvent être considérées comme négligeables dans notre problème car ces valeurs correspondent à une configuration où le robot fait face ou est dos à l'arbre de référence. En pratique, au vu des dimensions d'un verger, cette configuration n'est pas atteignable.

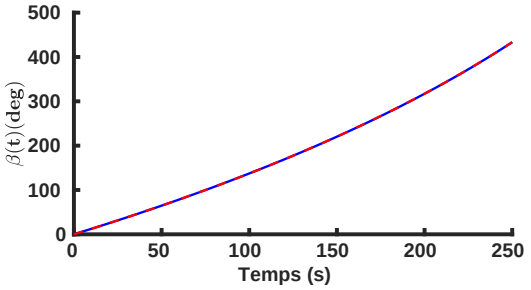
Jusqu'ici trois lois de commande permettant un suivi de spirale ont été conçues. Malgré les quelques limites que celles-ci présentent, elles sont capables d'accomplir avec succès leur tâche. Ces correcteurs vont ainsi pouvoir être utilisés pour la réalisation de demi-tours dans les vergers en considérant la position du dernier tronc d'arbre de la rangée comme centre de spirale. Cependant, lors de la navigation dans un verger, l'espacement entre deux rangées peut parfois être trop petit pour la réalisation d'un demi-tour simple. Dans la partie suivante, nous verrons comment faire face à ce problème.



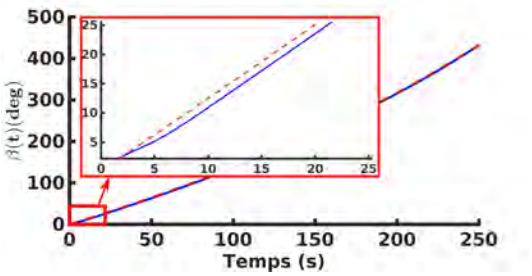
(a) Trajectoire du robot - Ligne bleue continue : Position du robot - Ligne rouge tiretée : Spirale de référence - Croix rose : Position de référence $[d^*(t), \beta^*(t)]^T$



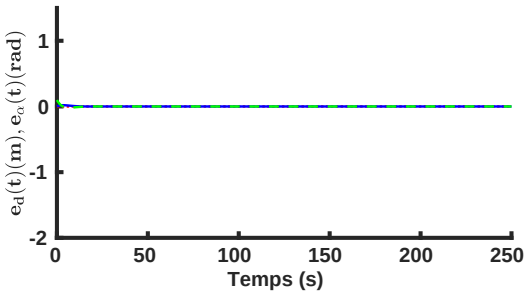
(b) Trajectoire du robot - Ligne bleue continue : Position du robot - Ligne rouge tiretée : Spirale de référence - Croix rose : Position de référence $[d^*(t), \beta^*(t)]^T$



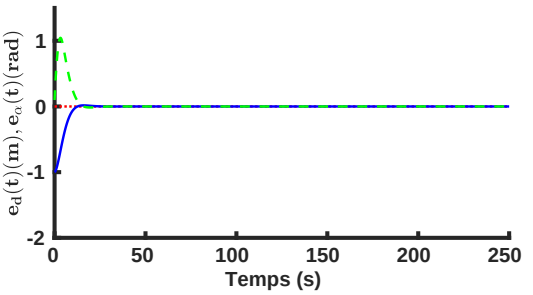
(c) Mesure de l'angle $\beta(t)$ - Ligne bleue continue : $\beta(t)$ - Ligne rouge tiretée : $\beta^*(t)$



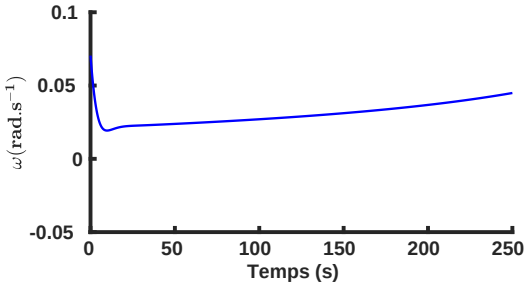
(d) Mesure de l'angle $\beta(t)$ - Ligne bleue continue : $\beta(t)$ - Ligne rouge tiretée : $\beta^*(t)$



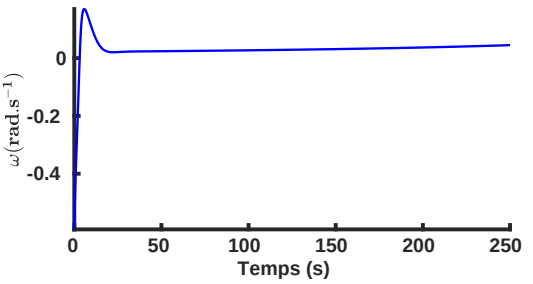
(e) Erreur en distance - Ligne bleue continue : $e_d(t)$ - Ligne verte tiretée : $e_\alpha(t)$ - Ligne rouge pointillée : Zéro



(f) Erreur en distance - Ligne bleue continue : $e_d(t)$ - Ligne verte tiretée : $e_\alpha(t)$ - Ligne rouge pointillée : Zéro



(g) Vitesse angulaire - Ligne bleue continue : $\omega(t)$



(h) Vitesse angulaire - Ligne bleue continue : $\omega(t)$

FIGURE 2.25 – Suivi de spirale utilisant le correcteur entrée/état pour le suivi de spirale avec $\alpha^* = \frac{15\pi}{32}$

Synthèse de loi de commande pour l'exécution de demi-tours en Ω

Dans certains cas, la configuration du champ peut être trop petite pour effectuer des demi-tours suivant de simples trajectoires en spirale comme dans les études effectuées

dans [Bochtis and Vougioukas, 2008b], [Cariou et al., 2010], ou encore dans [Zhang et al., 2014]. En effet, en fonction du robot utilisé, son rayon de braquage peut être trop petit pour atteindre la rangée suivante. Pour cela il est nécessaire de définir des trajectoires plus complexes qui permettront au robot de réaliser sa tâche de demi-tour. Cette partie porte sur la synthèse d'une stratégie de commande permettant l'accomplissement d'un demi-tour en forme de Ω. Dans un premier temps, la modélisation d'un tel demi-tour est présentée, puis un nouveau correcteur par retour de sortie sera créé pour répondre aux besoins requis par l'exécution de ce nouveau type de demi-tour.

Modélisation du demi-tour en Ω :

Tout d'abord, la modélisation d'un demi-tour en forme de Ω est illustrée dans la figure 2.26. Celui-ci est formé de deux spirales S_1 et S_2. La première, notée S_1, part de la position en sortie de rangée du robot p_i et passe par la position désirée dans la nouvelle rangée. Typiquement, si cette spirale impose au robot un rayon de braquage trop petit alors le simple demi-tour n'est pas possible. Dans ce cas, une spirale plus large S_2 est définie. Cette spirale doit pouvoir être suivie par le robot, *i.e.,* sa courbure doit être compatible avec le rayon de braquage du robot. Par la suite, le demi-tour est divisé en trois étapes :

1. Le déplacement de la position initiale p_i sur la spirale S_1 à la position souhaitée sur la spirale S_2 (φ_1)
2. Le suivi de la spirale jusqu'à S_2 (φ_2)
3. Le déplacement de la spirale S_2 jusqu'à la position p_f sur la spirale S_1 (φ_3)

Lors de toutes ces étapes le rayon de braquage du robot doit être assez grand pour être capable de suivre la référence. A chaque phase φ_1, φ_2 et φ_3 correspond une distance angulaire β_{φ_1}, β_{φ_2} et β_{φ_3} avec $\beta_{\varphi_1} \geq 0$, $\beta_{\varphi_2} \geq 0$ et $\beta_{\varphi_3} \geq 0$. La contrainte suivante peut alors être déduite :

$$\beta_{\varphi_1} + \beta_{\varphi_3} \leq \beta_{\varphi} \tag{2.32}$$

Ainsi, les étapes φ_1 et φ_3 doivent permettre au robot de se déplacer de la spirale S_1 (qui est impossible à suivre) vers une spirale S_2 (dont le rayon de courbure est assez grand) et inversement, tout en respectant la contrainte définie dans l'équation (2.32).

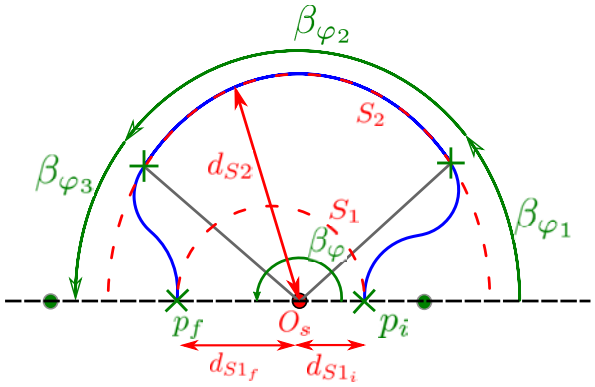


FIGURE 2.26 – Modèle de demi-tour en Ω

Par conséquent, les lois de commande construites précédemment peuvent être utilisées pour la réalisation de l'étape φ_2 qui consiste à suivre la spirale S_2. Une nouvelle loi de commande doit cependant être synthétisée pour piloter robot pendant l'exécution

des étapes φ_1 et φ_3.

Synthèse d'un correcteur pour l'exécution de changement de spirales :

Un nouveau correcteur permettant d'effectuer les changements de spirale lors des étapes φ_1 et φ_3 doit être conçu. Pour ce faire, il est proposé d'utiliser la méthode de linéarisation exacte entrée/sortie utilisée précédemment. Par conséquent, un profil en distance $d_\beta(\beta(t))$, garantissant au robot d'atteindre la spirale souhaitée pour une valeur angulaire $\beta(t_f)$ donnée a donc été défini. Dans un premier temps, nous présenterons ce profil, puis nous nous attacherons à la synthèse du correcteur.

Afin de créer le profil en distance, un angle normalisé $\bar{\beta}(t)$ avec $0 \leq \bar{\beta}(t) \leq 1$ est d'abord défini :

$$\bar{\beta}(t) = \frac{\beta(t) - \beta(t_0)}{\beta(t_f) - \beta(t_0)} \tag{2.33}$$

Avec t_0 et t_f l'instant initial et final. $\beta(t)$ est déterminé à partir de la mesure des encodeurs ou de la mise en place d'une odométrie visuelle. Ainsi $\beta(t_0)$ et $\beta(t_f)$ représentent l'angle initial et final. De plus, $\bar{d}_\beta(\bar{\beta}(t))$ est définie comme la distance normalisée à suivre, avec $0 \leq \bar{d}_\beta(\bar{\beta}(t)) \leq 1$. Pour garantir la continuité du chemin, les orientations initiale et finale du robot sont imposées comme tangentes aux spirales. Le profil en distance $\bar{d}_\beta(\bar{\beta}(t))$ normalisé proposé s'écrit :

$$\bar{d}_\beta(\bar{\beta}(t)) = k_0 \tan^{-1}(k_1(\bar{\beta}(t) + k_2)) + k_3 \tag{2.34}$$

k_0, k_1, k_2 et k_3 sont des scalaires définissant les paramètres de la fonction normalisée. De plus, le choix de ces paramètres doit garantir la faisabilité du suivi en fonction du rayon de braquage maximum du robot. Enfin, en utilisant les équations (2.33) et (2.34) le profil en distance à suivre s'écrit :

$$d_\beta(\beta(t)) = d(t_0) + \bar{d}_\beta(\bar{\beta}(t))[d(t_f) - d(t_0)] \tag{2.35}$$

Un exemple de profil de distance est donné par la figure 2.27 avec $k_0 = 0.365$, $k_1 = 10$, $k_2 = 0.5$, $k_3 = 0.5$, $\beta_0 = 0$rad, $\beta(t_f) = \pi/4$rad, $d(t_0) = 5$m, and $d(t_f) = 10$m.

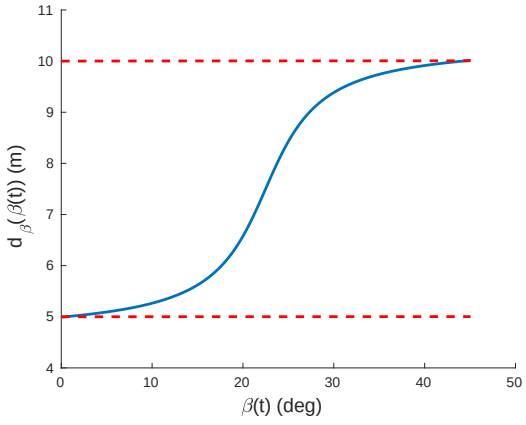


FIGURE 2.27 – Exemple de profil en distance : Lignes tiretées rouges : d_{S1} & d_{S2} || Ligne plein bleue : $d_\beta(\beta(t))$.

Afin que le robot suive le profil défini, il est nécessaire de faire converger la distance du robot vers sa nouvelle distance de référence $d_\beta(\beta(t))$. Le problème se réécrit alors à travers une nouvelle définition de l'erreur en distance telle que :

$$\begin{aligned} e_{d_\beta}(t) &= d(t) - d_\beta(\beta(t)) \\ &= d(t) - [d(\bar{\beta}(t_0)) + \bar{d}_\beta(\bar{\beta}(t))d_{gap}] \end{aligned} \tag{2.36}$$

Avec $d_{gap} = d(t_f) - d(t_0)$. La loi de commande développée doit alors faire converger cette erreur vers zéro. Pour assurer cette convergence, la méthode de linéarisation exacte entrée/sortie est utilisée. Par conséquent la nouvelle transformation s'écrit :

$$z_s = \begin{bmatrix} z_{1s} \\ z_{2s} \end{bmatrix} = \begin{bmatrix} e_{d_\beta}(t) \\ -v\cos(e_\alpha(t) + \alpha^* + d'_\beta(\beta(t))\frac{\sin(e_\alpha(t)+\alpha^*)}{d(t)} \end{bmatrix} = T_s(e_{d_\beta}, e_\alpha) \tag{2.37}$$

Où $d'_\beta(\beta(t))$ représente la dérivée de d_β en fonction de β. Il peut être prouvé que la transformation T_s ainsi définie est un difféomorphisme. En appliquant cette transformation au système il vient :

$$\dot{e}_{d_\beta}(t) = -v\cos(\alpha(t)) - k_0 k_1 \frac{\dot{\bar{\beta}}(t) d_{gap}}{k_1^2(\bar{\beta}(t) + k_2)^2 + 1} \tag{2.38}$$

Puis, le calcul de la dérivée de z_{2s} le long des trajectoires du système original donne :

$$\dot{z}_{2s} = \ddot{e}_{d_\beta}(t) = v\dot{\alpha}(t)\sin(\alpha(t)) - \frac{k_0 k_1 \ddot{\bar{\beta}}(t) d_{gap}}{k_1^2(\bar{\beta}(t) + k_2)^2 + 1} + \frac{2k_0 k_1^3 \dot{\bar{\beta}}^2(t) d_{gap}(\bar{\beta}(t) + k_2)}{(k_1^2(\bar{\beta}(t) + k_2)^2 + 1)^2} \tag{2.39}$$

Grâce à l'équation (2.33), $\dot{\bar{\beta}}(t)$ et $\ddot{\bar{\beta}}(t)$s'exprime comme suit :

$$\dot{\bar{\beta}}(t) = \frac{v\sin(\alpha(t))}{d(t)\beta_{gap}} \tag{2.40}$$

$$\ddot{\bar{\beta}}(t) = -\frac{v\sin(\alpha(t))\dot{d}(t)}{d^2(t)\beta_{gap}} + \frac{v\dot{\alpha}(t)\cos(\alpha(t))}{d(t)\beta_{gap}} \tag{2.41}$$

Avec $\beta_{gap} = \beta(t_f) - \beta(t_0)$. Le système (2.37) peut alors être réécrit comme suit :

$$\begin{cases} \dot{z}_{1s} = z_{2s} \\ \dot{z}_{2s} = f(\alpha(t), \beta(t), d(t)) + g(\alpha(t), \beta(t), d(t))\omega(t) \end{cases} \tag{2.42}$$

Où les fonctions f et g sont définies par les équations (2.39), (2.40), (2.41). La loi de commande est alors donnée par :

$$\omega(t) = \frac{1}{g(\alpha, \beta, d)}(f(\alpha, \beta, d) + \omega_3(t)) \tag{2.43}$$

Avec $\omega_3(t)$, le nouveau contrôle du système linéaire obtenu par la double intégration de $e_{d_\beta}(t)$:

$$\begin{cases} \dot{z}_{1s} = z_{2s} \\ \dot{z}_{2s} = \omega_3(t) \end{cases} \tag{2.44}$$

La loi de commande suivante est alors proposée pour $\omega_3(t)$:

$$\omega_3(t) = -\lambda_{1s}\dot{e}_{d_\beta}(t) - \lambda_{2s}e_{d_\beta}(t) \tag{2.45}$$

Où λ_{1s} et λ_{2s} sont deux scalaires positifs qui assurent la stabilité en boucle fermée du système décrit dans l'équation (2.42), avec la loi de commande proposée dans les équations (2.43) et (2.45). Ainsi les théorèmes suivants sont proposés :

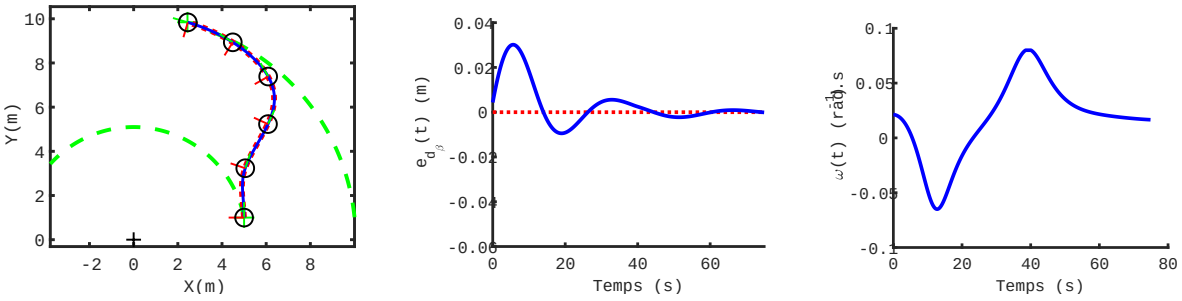
Théorème 2 *Considérant deux scalaires positifs λ_{1s} et λ_{2s} ; Le système d'erreurs défini dans l'équation (2.42) en boucle fermée ; La loi de commande proposée dans les équations (2.43) et (2.45) ; La transformation $z_s = T_s(\chi(t))$ dans l'équation (2.37) ; Le système en boucle fermée est alors localement asymptotiquement stable.*

Remarque 5 *Les lois de commande définies dans les équations (2.29) et (2.43), présentent les mêmes singularités que celles définies dans la remarque 3 (i.e. lorsque $\alpha(t) = k\pi$ avec $k \in \mathbb{N}$).*

Remarque 6 *Tel que noté dans la remarque 4, le système en boucle fermée présente aussi une dynamique des zéros stable.*

Résultats :

Cette loi de commande est à présent testée en simulation afin d'évaluer ses performances. La simulation illustrée par la figure 2.28 a été réalisée avec une période d'échantillonnage $T_s = 100$ms , un centre de spirales défini à $(0, 0)$ et une vitesse linéaire $v = 0.15m.s^{-1}$. De plus, de la même manière que précédemment, deux vecteurs vert et rouge représentent respectivement les axes $\vec{x}_r$ et $\vec{y}_r$ du repère robot. Cette loi de commande permet donc de changer de spirale de référence et ainsi compléter les phases φ_1 et φ_3 du demi-tour. La loi de commande est réglée avec les gains $\lambda_1 s = 0.05$ et $\lambda_2 s = 0.1$ et le profil en distance $d_\beta(\beta(t))$ est défini par $k_0 = 0.365$, $k_1 = 10$, $k_2 = -0.5$, $k_3 = 0.5$, $\beta(t0) = 0$ et $\beta(tf) = 3\pi/8$. Le robot est initialisé à $[5, 0, \pi/2]^T$ et doit atteindre la spirale S_2 donnée par $d^* = 10$m et $\alpha^* = \pi/2$ rad.



(a) Trajectoire du robot - Ligne bleue continue : Position du robot - Ligne rouge pointillée : Trajectoire de référence - Lignes verte tiretée : Spirales S_1 et S_2

(b) Erreur en distance - Ligne bleue continue : $e_{d_\beta}(t)$ - Ligne rouge pointillée : Zéro

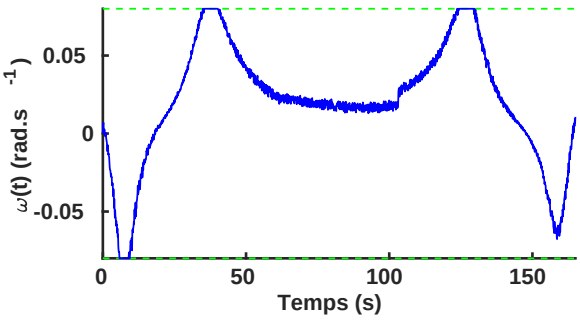
(c) Vitesse angulaire du robot - Ligne bleue continue : $\omega(t)$

FIGURE 2.28 – Suivi de profil en distance

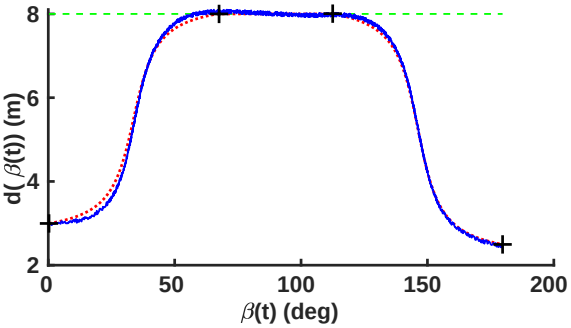
Comme il peut être remarqué dans la figure 2.28a, le robot atteint avec succès la spirale S_2 au point défini par $[d^*, \beta(tf)]$. De plus, le profil en distance a bien été suivi (figure 2.28b) avec une commande satisfaisante (figure 2.28c).

Le dernier ensemble de simulations représente l'association de ces deux correcteurs afin d'effectuer un demi-tour en forme de Ω dans le contexte d'une navigation dans la zone de fourrière d'un verger (figure 2.29). Premièrement, la loi de commande de suivi de rangée est appliquée tant que le robot est localisé dans le rang. Puis le demi-tour en Ω est activé jusqu'à entrer dans la rangée suivante. Enfin, l'asservissement de suivi de rangée est réactivé. Pour cette simulation, une limite sur $\omega(t)$ est imposée avec $\omega_{MIN-MAX} = \pm 0.08$rad/s, ce qui correspond à la vitesse angulaire maximum du robot réel. Afin de réaliser le demi-tour, les deux correcteurs sont requis. Ceux-ci sont réglés avec leurs gains respectifs $\lambda_1 = 0.01$ et $\lambda_2 = 0.05$ pour le correcteur de suivi de spirale, ainsi que $\lambda_{1s} = 0.01$ et $\lambda_{2s} = 0.05$ pour le correcteur de changement de spirale. Un profil en distance est créé pour ce dernier avec $k_0 = 0.365$, $k_1 = 10$, $k_2 = -0.5$, $k_3 = 0.5$. Le robot commence le demi-tour à $\chi(0) = [3, 0, \pi/2]^T$ et a pour objectif d'atteindre la spirale S_2 définie par $d_{S2} = 8$ m et $\alpha^*_{S2} = \pi/2$ rad avec un angle de référence de $\beta(\varphi_1) = 3\pi/8$rad. Finalement, le dernier changement de spirale a pour but de revenir sur la spirale S_1. Ce dernier point est défini par $d_{s1_f} = 2.5$m et un angle de référence $\beta(\varphi_3) = 3\pi/2$rad. De façon à évaluer la sensibilité des lois de commande, un bruit Gaussien centré a été ajouté aux mesures $d(t)$ et $\alpha(t)$, avec respectivement des amplitudes de 0.05m et 0.0175 rad. La trajectoire du robot est affichée dans la figure 2.29c. Le robot effectue avec succès le demi-tour en forme de Ω afin d'entrer dans la rangée suivante en alternant les deux correcteurs présentés. En effet, d'après la figure 2.29b le profil en distance ainsi que le suivi de spirale sont bien exécutés par le robot. De plus, l'erreur statique en régime permanent relevée précédemment n'a pas d'incidence sur l'exécution du demi-tour. En effet, le profil en distance pour l'étape φ_1 est conçu pour permettre au suivi de spirale de démarrer lorsque le robot est sur celle-ci. Ainsi, cet effet, n'a que très peu d'impact sur la navigation. Il est aussi important de remarquer que le bruit ajouté sur les mesures n'affecte pas le comportement du robot. Enfin, un réglage approprié de paramètres définissant les profils en distance ainsi que les spirales à suivre permettent d'exécuter un demi-tour en forme de Ω tout en gérant la contrainte liée au rayon de braquage maximum du robot.

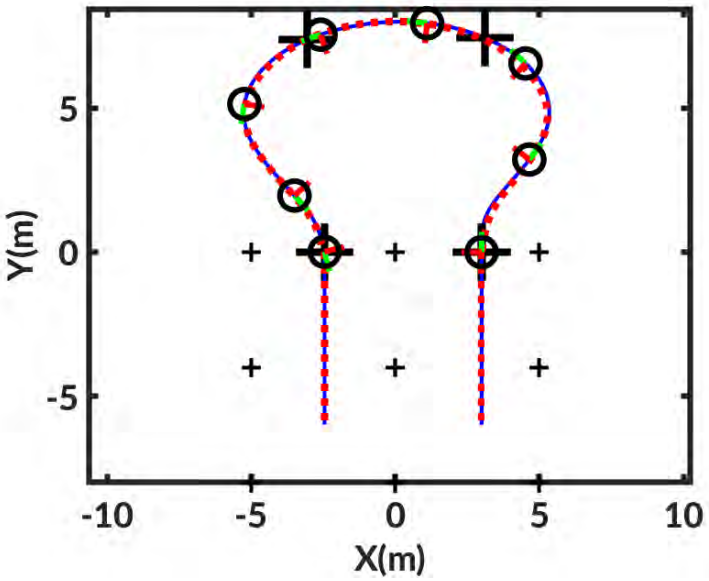
À présent, tous les correcteurs référencés capteur permettant d'exécuter des demi-tours en Ω ont été présentés et testés. Le premier permet ainsi de suivre une spirale spécifique et sera activé pour la navigation de l'étape φ_2. Le deuxième permet le changement d'une spirale à l'autre et sera activé pour la navigation lors des étapes φ_1 et φ_3.



(a) Vitesse angulaire du robot - Ligne bleue continue : $\omega(t)$



(b) Profil en distance - Ligne bleue continue : $d(t)$ - Ligne rouge pointillée : $d(\beta(t))$



(c) Trajectoire du robot - Ligne bleue continue : Position du robot - Ligne rouge tiretée : Trajectoire de référence - Croix noires : Changement de correcteur

FIGURE 2.29 – Demi-tour en Ω

Synthèse d'un demi-tour en queue d'aronde

Pour réduire au maximum l'espace utilisé dans la zone de fourrière, nous nous intéressons à présent à l'exécution d'un demi-tour en queue d'aronde. Ce type de demi-tour permet d'entrer dans la rangée suivante lorsque les dimensions de la zone de fourrière et le rayon de braquage minimum ne permettent pas l'exécution de demi-tour en spirale ou en Ω. Le travail présenté ici fait partie d'une première approche pour répondre au problème de contrôle pour l'exécution du demi-tour en queue d'aronde.

Modélisation du demi-tour en queue d'aronde :

De la même manière que lors de la modélisation du demi-tour en Ω, trois étapes notées φ_1, φ_2 et φ_3 sont nécessaires pour cette manœuvre. Tout d'abord, un modèle du demi-tour est illustré dans la figure 2.30.

Cette manœuvre consiste à utiliser deux spirales, S_1 et S_2, ainsi qu'une ligne droite L_{arr} à parcourir en marche arrière. La première spirale (S_1) est utilisée durant la phase φ_1. Elle est définie comme une spirale sortante, *i.e.*, $\alpha^* > \pi/2$, partant de la

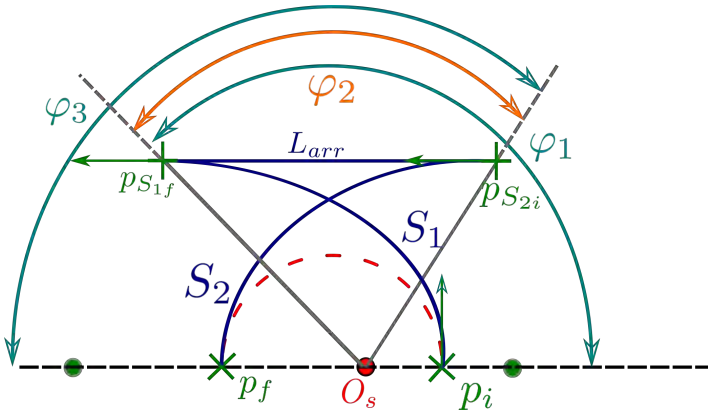


FIGURE 2.30 – Modélisation du demi-tour en queue d'aronde

position initiale du robot p_i et terminant à la position finale $p_{S_{1f}}$. Ensuite, le robot doit parcourir une ligne droite en marche arrière pendant la phase φ_2. Cette ligne démarre de la position $p_{S_{1f}}$ et se termine en S_2. Enfin, la deuxième spirale (S_2) est utilisée pour la phase φ_3. Elle est caractérisée par une spirale entrante, partant de la position $p_{S_{2i}}$ et arrivant à la position désiré dans la rangée suivante p_f.

Il nous faut à présent calculer les points de départ et d'arrivée pour chaque étape. Dans notre cas, quatre points sont à établir : p_i, p_f, $p_{S_{1f}}$ et $p_{S_{2i}}$. Pour cela, nous utilisons les conditions suivantes :

1. Le point p_i est défini comme la sortie de la rangée. En pratique les derniers arbres de chaque côté de la plate-forme se situent plus ou moins à 90° dans le repère robot. Cela se traduit par la détection des deux derniers arbres de la rangée courante par les caméras latérales. De plus, l'orientation de cette rangée permet de définir l'angle θ_s qui sera utilisé comme référence : $\theta_s = 0$.
2. Le point p_f est caractérisé par la position souhaitée du robot pour entrer dans la rangée suivante. La valeur de l'angle θ_s en ce point dépend des paramètres de la spirale S_2.
3. Le point $p_{S_{1f}}$ est positionné sur la spirale S_1 lorsque la valeur de l'angle θ_S atteint $\frac{\pi}{2}$. Il est aussi important de vérifier que la position de ce point se situe dans les limites imposées par les dimensions de la zone de fourrière.
4. Le point $p_{S_{2i}}$ appartient à la ligne droite définie par $p_{S_{1f}}$ et à la spirale S_2 passant par p_f. La valeur de l'angle θ_s en ce point est également de $\frac{\pi}{2}$. De la même manière que précédemment, il est nécessaire de s'assurer que ce point reste donc les limites de la zone de fourrière

Ainsi, l'exécution de la manœuvre de demi-tour en queue d'aronde est opérée par l'activation successive des correcteurs associés aux étapes φ_1, φ_2 et φ_3.

Résultats :

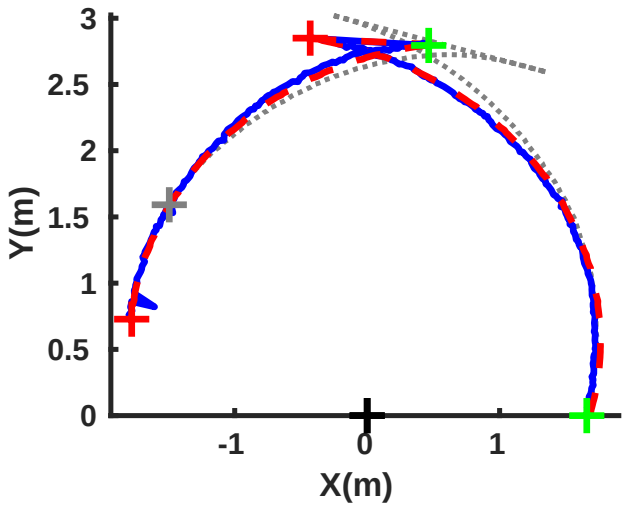
Afin d'illustrer l'approche proposée, la manœuvre a été réalisée hors du contexte agricole, dans un espace ouvert. Cette expérimentation a été réalisée sur l'une des plateforme du LAAS, le robot Aircobot (figure 2.31). Ce robot est de type skid steering drive et a été conçu pour la navigation en extérieur. De plus, il est muni de scanners laser à l'avant et à l'arrière, ce qui lui procure une perception sur 360°. Dans cette expérimentation, la détection de bandes réfléchissantes fixées sur un poteau par les scanners

FIGURE 2.31 – Robot Aircobot utilisé pour cette expérimentation

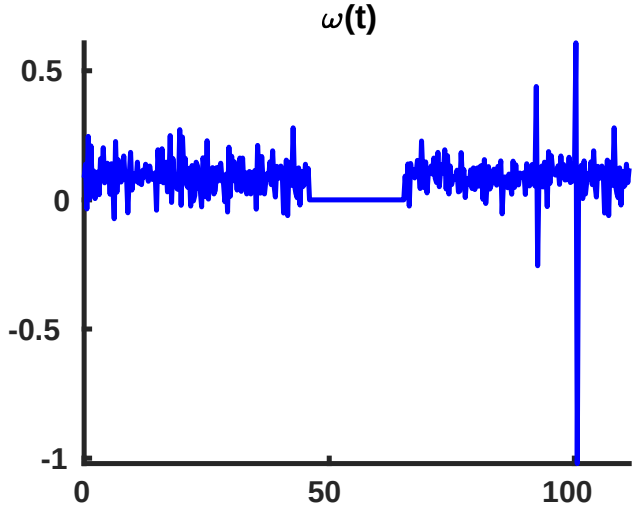
laser permet ici de simuler la localisation du dernier arbre. Ainsi, les paramètres mesurés $\alpha(t)$ et $d(t)$ seront directement utilisés dans les lois de commande. Les spirales des étapes φ_1 et φ_3 sont suivies grâce au correcteur α-d (partie 2.5.2), tandis que la marche arrière de l'étape φ_2 est accomplie sur la seule base de l'odométrie. La première spirale est déterminée par la position initiale du robot $d^*(0) = 1.65$m et $\alpha^* = \frac{19\pi}{32}$rad. La seconde spirale est caractérisée par $d^*(0) = 2.85$m et $\alpha^* = \frac{13\pi}{32}$rad. Cette dernière définit une trajectoire passant par le repère p_f positionné aux coordonnées $(-2.9, 0.2)$ par rapport au centre de la spirale. Le résultat de cette expérimentation est illustré dans la figure 2.32.

La figure 2.32a montre les trajectoires durant l'exécution du demi-tour. La ligne tiretée rouge représente la trajectoire de référence. La ligne continue bleue montre la trajectoire parcourue par le robot basée sur la mesure des scanners laser. Enfin, la ligne pointillée grise illustre la trajectoire exprimé à partir de l'odométrie. Ce résultat nous montre que la stratégie de commande utilisée permet de réaliser efficacement ce type de demi-tour. En effet, la trajectoire réalisée par le robot (ligne continue bleue) est proche de la trajectoire souhaitée (ligne tiretée rouge). Le tracé de la trajectoire relevée par la mesure de l'odométrie (ligne pointillée grise) permet d'illustrer les erreurs accumulées par ce type de capteur et donc son inefficacité pour commander le robot lors de l'exécution de ce type de demi-tour. Il peut être noté que la mesure odométrique induit une grande erreur entre la position finale et celle souhaitée. Nous pouvons de plus remarquer que le repère p_f est positionné légèrement avant l'entrée de la rangée. Ceci est effectué de façon à laisser le temps au robot de se réorienter en fonction de l'alignement de la rangée.

La figure 2.32b affiche la commande de vitesse angulaire envoyée au robot lors de l'exécution de la manœuvre. Nous pouvons remarquer que les erreurs introduites par la mesure génèrent des perturbateurs sur la commande. Cependant, celles-ci n'impactent pas la bonne réalisation du demi-tour.



(a) Trajectoire du robot - Ligne continue bleue : Trajectoire du robot (scanners laser) - Ligne rouge tiretée : Trajectoire de référence - Ligne pointillée grise : Trajectoire du robot (odométrie) - Croix vertes : Activation du correcteur α-d - Croix rouges : Désactivation du correcteur α-d - Croix grise : Position finale du robot (odométrie) - Croix noire : Centre des spirale



(b) Mesure de la vitesse angulaire du robot $\omega(t)$

FIGURE 2.32 – Expérimentation du demi-tour en queue d'aronde

L'expérimentation présentée ici a été effectuée avec le correcteur α-d. Le correcteur basé sur la linéarisation exacte (partie 2.5.2) a aussi montré son efficacité pour suivre une spirale et aurait tout aussi bien été en mesure d'exécuter cette manœuvre. De plus, cette expérimentation a mis en évidence la nécessité de ne pas se fier à une navigation uniquement basée sur l'odométrie. Bien que la phase φ_2 soit relativement courte, il aurait été intéressant de développer un correcteur de suivi de ligne basé sur la position des arbres aux alentours, afin de ne pas se reposer uniquement sur la mesure odométrique lors de la réalisation de cette étape. Enfin, une amélioration des conditions de basculement entre les correcteurs permettrait d'augmenter la généricité et donc le degré d'autonomie du système.

2.5.3 Conclusion

Dans cette partie, plusieurs lois de commande permettant la navigation dans un verger ont été présentées et testées. Ainsi un correcteur a été développé pour mener à bien le suivi de rangée, puis trois correcteurs ont été synthétisés pour permettre le suivi de spirale afin d'accomplir les demi-tours en fin de rangée. Enfin, des stratégies permettant l'exécution de demi-tours plus complexes ont été présentées et ont conduit au développement d'un correcteur permettant le changement de spirales. Ces lois de commandes seront activées par le processus décision en fonction de la localisation du robot dans la carte topologique. Les données fournies par les capteurs sont ainsi directement utilisées dans les lois de commande, ce qui permet d'être moins sensible aux variations de l'environnement.

2.6 Architecture générale

2.6.1 Interactions entre processus

L'objectif de cette section est de résumer le fonctionnement de l'architecture choisie. Ainsi les processus décrits dans ce chapitre ainsi que leurs interactions sont repris et expliqués brièvement. Tout d'abord, un schéma de l'architecture proposée est illustrée dans la figure 2.33. Les processus marqués par un fond bleu sont réalisés durant une étape de pre-navigation. En effet, dans un premier temps, le processus modélisation a pour objectif de construire la carte topologique. Dans notre cas, cette carte est construite à partir des connaissances a priori du terrain. Ensuite, la partie planification définit un chemin à suivre dans cette carte topologique. Ce chemin est construit grâce à la connaissance de l'état initial du robot ainsi qu'à son objectif final. Dès lors que ces deux processus sont terminés, le chemin topologique est calculé et la navigation peut alors commencer. Lors de la navigation, le processus perception acquiert les données venant des capteurs et les traite. Dans l'architecture choisie, les données venant des caméras sont traitées pour en extraire la position des troncs d'arbres ainsi que détecter les début et fin de rangées, qui sont envoyées au processus localisation. Ce processus peut alors déterminer où se trouve le robot dans le chemin construit précédemment par le processus planification. Ainsi, le processus localisation est capable d'indiquer au processus décision le nœud de la carte topologique qui est actif. Les informations nécessaires à la commande (ligne à suivre, arbre le plus proche, etc) sont fournis au processus action par le processus perception. Le processus décision quant à lui choisit le correcteur à appliquer, les paramètres à utiliser, ainsi que les consignes à atteindre en fonction du nœud actif. Enfin, le processus action applique les lois de commande construites pour contrôler le robot afin de réaliser la tâche de navigation.

2.6.2 Application de l'architecture

La simulation d'une navigation longue distance est alors mise en place pour évaluer l'efficacité de l'architecture proposée. Cette simulation est implémentée sous ROS avec l'environnement de simulation 3D *Gazebo*. La simulation du robot peut être observée dans la figure 2.34. La figure 2.34a illustre le suivi d'une rangée d'arbres tandis que la figure 2.34b montre la réalisation d'un demi-tour dans la zone de fourrière. Dans cette simulation trois capteurs laser 2D sont utilisés pour simuler la localisation des troncs venant du processus perception. Ces figures font apparaître les données de distance

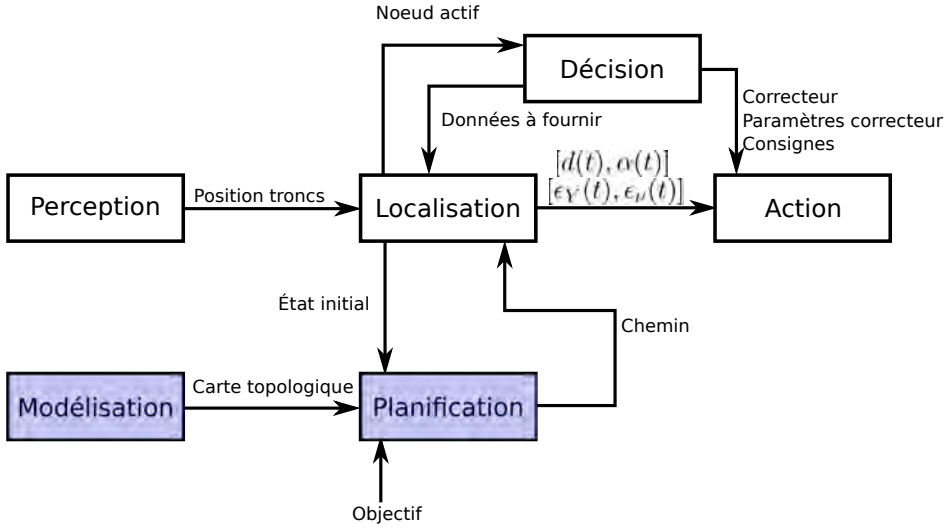
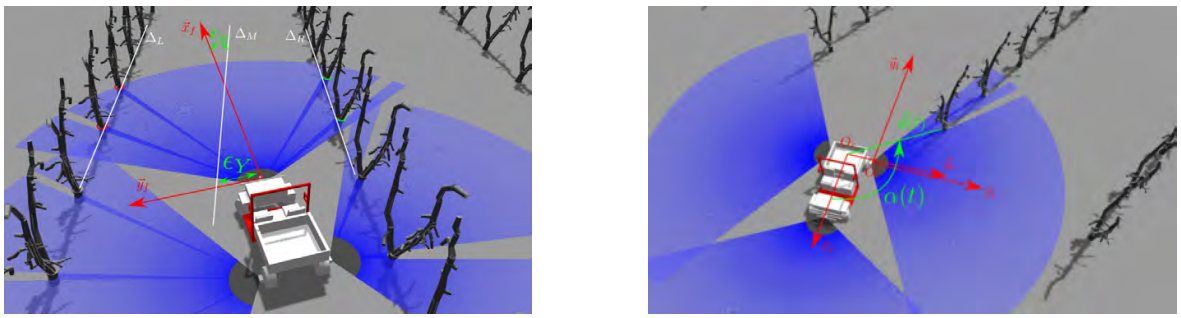


FIGURE 2.33 – Schéma d'interaction entre les processus, appliqué au problème posé. Processus bleu : Actif uniquement à l'initialisation

latérale et d'angle de déviation pour le suivi de rang ainsi que la distance au dernier tronc et son angle respectif dans l'espace caméra pour le demi-tour. Le modèle 3D des arbres utilisé provient de mesures effectuées lors d'un autre projet [Vougioukas et al., 2016].



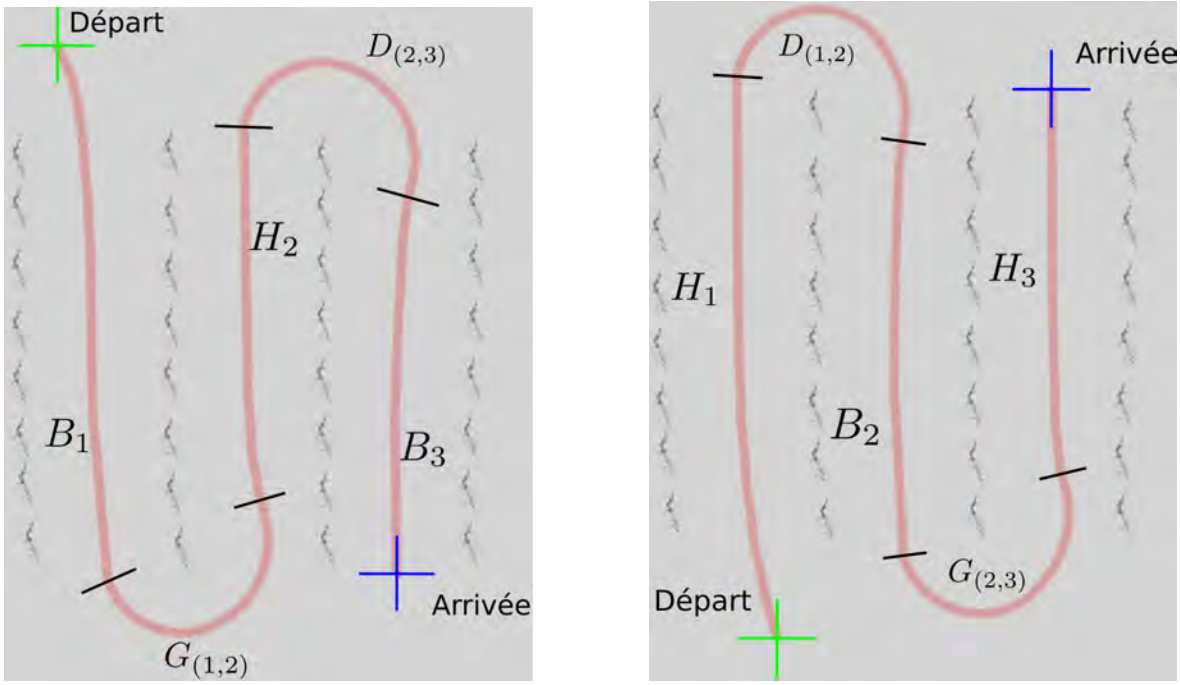
(a) Simulation du robot dans la rangée

(b) Simulation du robot dans la zone de fourrière

FIGURE 2.34 – Simulation du robot dans l'environnement *Gazebo*

La figure 2.35 montre les résultats de navigation dans le simulateur. Les croix vertes correspondent aux localisations de départ du robot tandis que les croix bleues matérialisent les points d'arrivée. Les nœuds actifs sont marqués le long du parcours et les traits noirs représentent les changements de localisation dans la carte topologique et donc les changements de correcteurs. Les figures 2.35a et 2.35b illustrent le parcours du robot. Le premier montre le résultat obtenu lorsque l'on alterne le correcteur de suivi de rangée et le correcteur α. Pour la seconde, ce dernier est remplacé le correcteur α-d.

Dans les deux cas, on remarque que les résultats sont similaires et que la navigation se déroule sans encombre, puisque le robot parcourt consécutivement toutes les rangées jusqu'à sa situation finale.



(a) Navigation utilisant le correcteur α pour le demi-tour

(b) Navigation utilisant le correcteur $\alpha - d$ pour le demi-tour

FIGURE 2.35 – Simulation dans l'environnement *Gazebo*

2.7 Conclusion

Ce chapitre a proposé une architecture complète dédiée à la navigation dans les vergers. Cette architecture est organisée autour de six processus : perception, modélisation, localisation, décision, action, planification. Elle a été conçue de manière à lever les principaux verrous que présente la navigation dans un environnement naturel fortement évolutif tel qu'un verger :

- la difficulté d'obtenir une localisation métrique précise (présence de la canopée qui bloque le signal GNSS ; terrain inégal ou glissant susceptible de perturber l'odométrie, etc.)
- la difficulté de détecter précisément les rangées et de les suivre, du fait de l'aspect variable des arbres, des conditions d'illumination variées, etc.
- la difficulté de réaliser un demi-tour du fait de la faible présence de données sensorielles dans la zone de fourrière
- la difficulté de conserver une carte métrique à jour de l'environnement du fait de sa haute variabilité
- le besoin d'algorithmes rapides et performants, facilement embarquables

Nous avons plus particulièrement contribué sur les processus 'perception' et 'action'. Nous avons ainsi proposé un algorithme rapide permettant de détecter les arbres de manière précise et fiable. Il est basé sur la détection des concavités dans une image de profondeur grâce à un système adéquat de tombé de jetons. Les résultats obtenus expérimentalement ont montré l'efficacité de cet algorithme pour différents vergers à différentes saisons. Cet algorithme est une pièce essentielle de notre stratégie car il fournit les données nécessaires à la réalisation d'un vrai demi-tour référencé capteur, donc plus précis que les approches classiquement utilisées dans la littérature. Nous avons donc également synthétisé un ensemble de correcteurs permettant de suivre une spirale

centrée sur le dernier arbre détecté. Nous les avons validés pour partie en simulation et pour partie en expérimentation. Les résultats obtenus sont convaincants et montrent l'intérêt de l'approche proposée. Une première carte topologique a aussi été réalisée dans l'optique de la réalisation de longs déplacements. Nous nous intéressons maintenant à la commande du système multi-bras pour réaliser les tâches de positionnement nécessaires au ramassage des fruits.

Chapitre 3

Contrôle simultané des mouvements d'un système multi-bras

3.1 Introduction

Dans le chapitre précédent, une architecture complète permettant la navigation autonome d'un robot dans un verger a été développée. La base-mobile utilisée dans la partie précédente peut permettre d'embarquer des bras robotiques afin de réaliser des tâches agricoles telles que : l'analyse des plantes ([Schor et al., 2016]), l'application localisée de traitements ([Oberti et al., 2016, Blasco et al., 2002]), la taille ([Rosa et al., 2008, Morris, 2007]), ou encore le ramassage des fruits ([Van Henten et al., 2003, Zhao et al., 2016a, Hayashi et al., 2010, Ceres et al., 1998, Reed et al., 2001, Hayashi et al., 2002, Foglia and Reina, 2006]). Malgré les avancées, les études présentées dans [Grift et al., 2008] et [Bechar and Vigneault, 2017] montrent que le rendement obtenu avec un manipulateur mobile reste trop faible pour répondre aux besoins des agriculteurs. Une première solution consiste à équiper la base mobile de plusieurs bras manipulateurs, chacun d'eux travaillant dans un espace délimité. Par exemple, [Zhao et al., 2016b] développent un système utilisant deux bras robotiques afin de récolter les tomates dans une serre. Dans [Zion et al., 2014], un robot muni de plusieurs bras permet de ramasser des melons. Cependant, l'étude menée dans [Vougioukas et al., 2016] dans le cadre des vergers, montre que la répartition des fruits dans les arbres n'est pas uniforme. Ainsi, afin de minimiser le temps total nécessaire pour récolter les fruits d'un arbre, il semble nécessaire d'attribuer plusieurs bras à certaines zones. Cela signifie que plusieurs manipulateurs doivent partager un même espace de travail. Il existe divers travaux allant dans ce sens, tels que ceux présentés dans [Todt et al., 2000] qui proposent une analyse et une classification de plusieurs méthodes permettant la coordination de robots partageant un même espace. La solution présentée dans [Rodriguez-Angeles and Nijmeijer, 2001] permet de coordonner deux bras manipulateurs à travers une architecture maître/esclave. L'asservissement en position des bras est alors effectué grâce à des observateurs non-linéaires estimant les vitesses respectives appliquées sur chaque bras. Le travail de [Merchan-Cruz and Morris, 2006] propose d'éviter des obstacles statiques à l'aide d'une approche à base de logique floue. Toutefois, la tâche est restreinte à l'évitement du deuxième bras qui reste statique, ce qui ne répond pas totalement à notre problème. Deux bras manipulateurs sont coordonnés dans [Fleurmond, 2016] grâce à des stratégies de commande basées sur des asservissements visuels pour mener à bien une tâche co-manipulation [1]. Une approche présentée dans [Jasour and Farrokhi, 2009]

1. Tâche durant laquelle plusieurs entités coopèrent pour manipuler une même charge.

pour asservir un bras manipulateur redondant semble intéressante. En effet, ce travail exploite une méthode de commande basée sur des modèles prédictifs non-linéaires (ou Non-linear Model Predictive Control (NMPC)) pour prédire les vecteurs d'état des bras robotiques sur un horizon de temps fixe, afin de calculer un vecteur de commande optimal à appliquer au robot. Cette méthode facilite la prise en compte de contraintes lors de la recherche de solutions optimales.

Les solutions citées précédemment permettent d'obtenir des résultats probants dans les contextes pour lesquelles elles ont été développées, mais ne répondent pas complètement aux contraintes spécifiques de la cueillette de fruits dans un verger. Ainsi, comme pour la partie navigation, les contraintes sont liées à la nature et à la dynamique de l'environnement. Par exemple, au cours des saisons : les fruits poussent et font plier les branches sous leurs poids, modifiant ainsi leurs positions ; lors de la récolte, la cueillette d'un fruit modifie la configuration de la branche. De plus, les branches peuvent être considérées comme des obstacles inconnus a priori devant être évités. Les feuilles peuvent occulter la vision du fruit à récolter. Ainsi, pour positionner l'organe terminal du bras robotique afin de saisir un fruit, il est nécessaire d'utiliser une méthode réactive tenant compte d'un certain nombre de contraintes. Dans ce contexte, les méthodes reposant sur une modélisation et une localisation métriques permettent de prendre en compte un certain nombre de contraintes, ne semblent pas adaptées pour gérer un environnement changeant. A l'opposé, des solutions purement réactives sont idéales pour évoluer dans un cadre dynamique, mais peu adaptées pour la prise en compte de contraintes. Afin de répondre à tous les défis mentionnés plus haut, il est donc proposé dans ce projet de coupler une stratégie basée sur le contrôle réactif par asservissement visuel [Chaumette and Hutchinson, 2006], aux méthodes de commande prédictive, permettant la prise en compte de contraintes. L'association de ces deux techniques est parfois dénommée asservissement visuel prédictif (Visual Predictive Control VPC), dénomination que nous conserverons dans ce travail [Allibert et al., 2010].

Ce chapitre est dédié à la présentation d'un asservissement visuel prédictif pour réaliser le contrôle simultané de plusieurs bras partageant un même espace de travail. Pour ce faire, les prochaines sections sont consacrées aux présentations des méthodes d'asservissement visuel, de la commande prédictive et finalement de l'asservissement visuel prédictif. Un asservissement visuel prédictif est ensuite synthétisé pour l'application considérée. Différentes étapes ont été menées : modélisation du système, présentation des différents modèles de prédiction des indices visuels, formulation de la fonction de coût et mise en équation des différentes contraintes. Le chapitre se conclut par une section dédiée à des simulations illustrant le travail présenté et mettant en avant les performances de la méthode proposée.

3.2 Asservissement visuel

La tâche principale à accomplir consiste à atteindre des situations désirées dépendant de la tâche agricole à effectuer (*i.e.* situation d'un fruit à récolter, d'une branche à couper, d'une zone à traiter, etc). Dans un environnement extérieur, les caméras sont très utilisées pour la richesse des informations qu'elles sont capables de fournir au système (*i.e.* couleur, texture, forme, etc). De nombreux travaux en robotique agricole exploitent ce type de capteur [Henten et al., 2002], [Foglia and Reina, 2006] ou encore [Oberti et al., 2016]. La section précédente a relevé l'importance de développer des solutions réactives pour prendre en compte la nature dynamique de l'environnement.

Dans ce travail, les bras sont commandés par des asservissements de type retour de sortie sur la base des informations visuelles. Les méthodes d'asservissement visuel sont généralement classifiées en deux grandes catégories [Chaumette, 1990].

- L'asservissement visuel 3D (ou Position Based Visual Servoing (PBVS)) permet de contrôler une caméra via l'annulation de l'erreur entre la pose actuelle de la caméra et une pose désirée. La pose courante de la caméra est estimée à partir d'informations visuelles mesurées dans l'image (généralement des indices visuels appartenant à un objet). Cette estimation nécessite la connaissance de certains paramètres comme les dimensions de l'objet ou la distance entre la caméra et l'objet. Une solution permettant l'estimation de cette dernière est l'utilisation d'une caméra 3D, d'une caméra déportée ou encore de méthodes d'estimation comme décrit dans [François Chaumette and Seth Huchinson, 2007]. Finalement, les trajectoires sont calculées dans l'espace opérationnel, sans tenir compte de l'évolution des indices visuels. Il peut donc arriver que ces derniers soient perdus, menant à un échec de la tâche. De plus, le processus d'estimation de la position de la caméra rend la commande plus sensible aux erreurs de mesures et de modélisation.
- L'asservissement visuel 2D (ou Image Based Visual Servoing (IBVS)) consiste à contrôler les déplacements d'une caméra via l'annulation d'un vecteur erreur exprimé dans l'espace image [Chaumette and Hutchinson, 2006]. A cet effet, les informations visuelles sont directement utilisées dans la boucle de commande et peuvent être composées des coordonnées de points, segments, droites, ellipses, ou encore moments ([Chaumette, 2004]). Contrairement à l'asservissement visuel 3D, l'avantage de cette technique est sa robustesse par rapport aux bruits de mesure ainsi qu'aux erreurs de modélisation. Cependant, dans sa version la plus simple, la convergence des indices visuels s'effectuant dans l'espace image, les mouvements du robot peuvent alors paraître contre-intuitifs dans l'espace de travail. Ce type d'asservissement peut ainsi amener le robot en singularité ou encore en butée, ce qui peut conduire à l'échec de la tâche lors de grands déplacements ([Chaumette, 1998], [Kermorgant, 2011]).

Dans ce travail, nous nous focalisons sur l'asservissement visuel 2D (IBVS). Ce choix est motivé par la réactivité et la robustesse face aux erreurs de modélisation. Au delà de l'approche classique reposant sur la décroissante exponentielle de l'erreur via la matrice d'interaction [Chaumette and Hutchinson, 2006], de nombreux travaux ont fait évoluer cette technique afin de modifier les trajectoires de la caméra dans l'espace de travail ou de gérer des problèmes tels que la perte des indices visuels, la collision de la caméra avec un obstacle, ... Parmi eux, les travaux présentés dans [Francois Chaumette, 2002], [Chaumette, 2003] étudient l'utilisation d'indices visuels plus avancés pour obtenir des trajectoires de la caméra paraissant plus appropriées. Ces travaux ont permis d'obtenir des résultats pertinents mais ne permettent pas de définir une trajectoire garantissant la non-collision avec un obstacle par exemple. Un autre groupe de travaux propose de coupler l'asservissement visuel avec d'autres correcteurs élémentaires à activer ou désactiver en fonction de la situation courante (obstacles, occultations, butées articulaires, ...) [Pissard-Gibollet, 1993, Cadenat, 1999, Folio, 2007, Durand-Petiteville, 2012]. Cette approche peut nécessiter de développer de nombreuses lois de commande *ad-hoc*, et la définition des transitions entre elles n'est pas toujours triviale. Une autre approche s'appuie sur la redondance du robot afin de déformer la trajectoire pour satisfaire les contraintes [Marchand et al., 1996, Chaumette and Marchand, 2001]. Cependant, cette technique n'est possible que si la tâche principale

n'utilise pas tout les degrés de liberté du robot [Mansard, 2006], ce qui peut être le cas pour des systèmes possédant un grand nombre de degrés de liberté, tels que les robots humanoïdes. Finalement, une dernière approche consiste à associer planification et asservissement visuel [Mezouar and Chaumette, 2002, Kazemi et al., 2010]. Pour cela, il est généralement nécessaire de connaître *a priori* l'environnement pour inclure dans la planification la position des obstacles, par exemple. Le travail proposé ici se rapproche de cette dernière philosophie. En effet, la commande prédictive peut être vue comme une planification à chaque itération, où les contraintes sont ajustées en fonction des données nouvellement acquises. C'est dans cet optique que nous nous intéressons donc maintenant à la commande prédictive non-linéaire.

3.3 Commande par modèle prédictif non-linéaire

NMPC est une méthode de commande de systèmes non-linéaires basée sur la prédiction des futurs états du système et de l'optimisation de la trajectoire prédite. En robotique, cette méthode est régulièrement utilisée pour résoudre des problèmes de planification [Liu et al., 2017, Shim et al., 2012], de calcul de trajectoire pour des voitures autonomes [Abbas et al., 2017] ou encore pour le contrôle d'un bras robotique installé sur un satellite [Rybus et al., 2017].

La commande prédictive se compose de quatre étapes [Lars and Pannek, 2016]. Premièrement, il est nécessaire de prédire à partir de modèles dynamiques, les états du système pour une commande donnée et sur un horizon de temps défini. Notons que la précision des modèles dynamiques influe très fortement sur la qualité des prédictions. Dans un second temps, les états prédits sont utilisés pour définir une fonction de coût à minimiser. L'annulation de cette fonction de coût doit correspondre à la réalisation de la tâche. En plus de la fonction de coût, il est aussi nécessaire de mettre en équation les contraintes devant être prises en compte lors de la minimisation de la tâche. Ces contraintes peuvent être écrites sous forme d'égalité ou d'inégalité, linéaire ou non-linéaire. Finalement, un solveur est utilisé pour calculer la séquence de commandes permettant de minimiser la fonction de coût tout en respectant les contraintes données. Il existe un large choix de solveurs disponibles : SQP, point-intérieur, Lagrangien augmenté, ... [Chong and Żak, 2001]. Le choix de l'algorithme se fait en fonction de sa capacité à travailler avec des systèmes linéaires ou non-linéaires, et des contraintes sous forme d'égalité ou d'inégalité, linéaire ou non-linéaire. Dans ce travail il est décidé d'utiliser l'algorithme SQP qui permet de travailler avec un système non-linéaire et des contraintes sous forme d'inégalités non-linéaires.

Dans le cadre d'un contrôle par NMPC, le solveur calcule à chaque itération la séquence de commandes optimales. Cependant, il est très courant que seule la première commande soit appliquée et que le processus de minimisation soit relancé à chaque itération. Ce type d'approche est donc peu adapté à des systèmes nécessitant une courte période d'échantillonnage. Des travaux sont ainsi menés dans le but de réduire le temps de calcul comme ceux proposés dans [Mansard et al., 2018] qui pré-planifie hors ligne les configurations de départ du NMPC pour guider le solveur vers la solution optimale. De plus, les plateformes de calculs sont de plus en plus performantes et le développement du calcul parallèle offre de nouvelles possibilités en terme de puissance de calcul. Le coût d'une telle méthode est donc de moins en moins un problème.

L'asservissement visuel et le schéma de contrôle NMPC étant présentés, nous nous proposons de nous concentrer sur l'asservissement prédictif. Cette approche doit permettre de synthétiser un correcteur réactif tenant compte d'un certain nombre de

contraintes.

3.4 Asservissement Visuel Prédictif

Comme expliqué précédemment, la commande prédictive nécessite d'utiliser un modèle permettant de calculer les futurs états du système robotique. Dans le cas de l'asservissement visuel prédictif, il s'agit de prédire les coordonnées des indices visuels connaissant une séquence de vecteurs de commande. Pour cela, d'après la présentation faite dans [Allibert et al., 2010], les modèles utilisés peuvent être classés en deux catégories : les modèles locaux et globaux. Les premiers reposent sur la matrice d'interaction [Chaumette and Hutchinson, 2006] tandis que les seconds utilisent la pose de la caméra. Dans les deux cas, ces modèles sont construits en considérant une caméra volante, c'est-à-dire que le système portant la caméra n'est pas pris en compte. Finalement, aucun de ces modèles ne propose de solution analytique à la prédiction des indices visuels. Ainsi, les prédictions sont obtenues en utilisant un modèle linéaire, approximation du premier ordre, ou avec des schémas d'intégration plus avancés, par exemple Runge-Kutta.

La majorité des travaux existants peuvent être classés dans la première catégorie, c'est-à-dire reposant sur des modèles locaux. Par exemple, dans [Assa and Janabi-Sharifi, 2014], les auteurs utilisent une approximation du premier ordre pour contrôler une caméra montée sur un bras robotique. Des approches similaires sont proposées dans [Heshmati-alamdari et al., 2014] pour une caméra volante et dans [Ke et al., 2017] pour un robot mobile. Dans [Wang et al., 2012] et [Hajiloo et al., 2016], le schéma d'asservissement visuel pour une caméra montée sur un bras robotique est représenté par un système linéaire à paramètres variants. L'objectif de ces travaux est de montrer la faisabilité de l'asservissement visuel prédictif sous à des contraintes sur le vecteur commande et sur le champ de vue de la caméra. Les résultats montrent la pertinence de l'approche, mais sont obtenus pour de petits écarts entre la position initiale et celle désirée, minimisant ainsi l'impact des erreurs de prédiction. Dans un contexte totalement différent, le travail présenté dans [Mcfadyen et al., 2013] propose de contrôler une plate-forme volante tout en évitant les autres engins aux alentours. Dans ce cas, la distance entre les véhicules est tellement grande qu'une mesure de la profondeur n'est pas nécessaire et une simple approximation est suffisante.

Malgré l'intérêt porté à l'asservissement visuel prédictif, il n'existe pas à notre connaissance de travaux proposant une étude approfondie de cette méthode. Par exemple, il n'existe pas de comparaisons entre les deux types de modèles, ni d'évaluation de l'impact de l'utilisation de la profondeur réelle ou approximée. De plus, la majorité des travaux se focalise sur la réalisation de l'asservissement visuel sans y inclure de contraintes externes. Finalement, ces travaux se concentrent sur des caméras embarquées sur des bras robotiques, impliquant de petits déplacements.

Dans ce chapitre, il est tout d'abord proposé d'évaluer la précision des différents schémas de prédiction avec la profondeur réelle ou approximée. Cette évaluation se place dans notre contexte de caméras embarquées sur des bras manipulateurs. Le schéma de prédiction global est donc modifié en y incluant le système portant la caméra. Il devient alors possible de prédire les configurations futures du bras robotique à l'aide d'un schéma d'intégration numérique de premier ordre sans pour autant faire d'approximation. Dans un second temps, le schéma de contrôle VPC est détaillé, ainsi que les différentes contraintes qui sont prises en compte pour l'application proposée. A ces contraintes, nous ajoutons une contrainte terminale permettant d'évaluer la faisabilité

de la tâche et de garantir la convergence vers l'objectif souhaité. Finalement, plusieurs simulations seront présentées pour illustrer nos propos et mettre en avant l'intérêt de cette approche.

3.5 Modélisation

3.5.1 Modélisation du robot

Cette partie s'attache à présenter le modèle du système robotique considéré (voir figure 3.1). Dans le contexte de notre étude nous considérons un robot composé de plusieurs bras manipulateurs opérant dans un espace de travail commun. Ainsi, le système est constitué de n_a bras identiques, comportant m_q articulations rotoïdes chacun. Tous les organes terminaux de ce système sont pourvus d'une caméra.

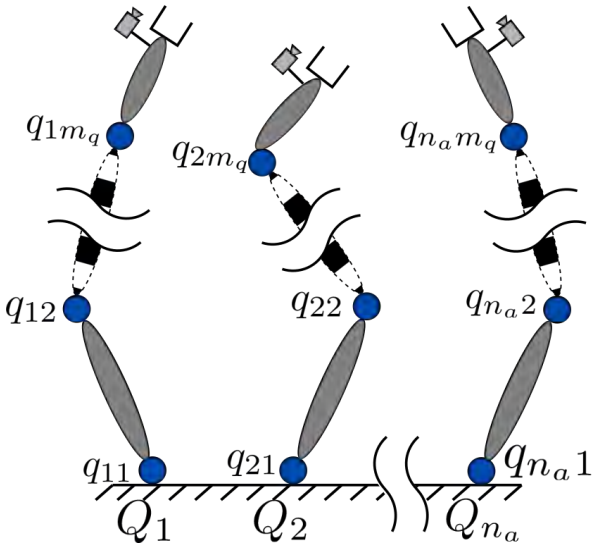


FIGURE 3.1 – Schéma illustrant le système multi-bras considéré

Un repère global commun à tous les bras est fixé à la base du robot et défini comme $F_b = (O_b, \mathbf{x_b}, \mathbf{y_b}, \mathbf{z_b})$. Chaque caméra i de centre optique O_{ci} est pourvue d'un repère propre $F_{ci} = (O_{ci}, \mathbf{x_{ci}}, \mathbf{y_{ci}}, \mathbf{z_{ci}})$ avec $i \in [1, \ldots, n_a]$. De plus, nous dénotons les valeurs angulaires de chaque articulation ainsi que leurs vitesses angulaires q_{ij} et $\dot{q}_{ij}$ avec $j \in [1, \ldots, m_q]$. De cette manière, le vecteur de configuration de chaque bras est décrit par $Q_i = [q_{i1}, \ldots, q_{im_q}]^T$ et le vecteur de commande associé par $\dot{Q}_i = [\dot{q}_{i1}, \ldots, \dot{q}_{im_q}]^T$. Par extension, le vecteur des configurations du système robotique à bras multiples est écrit $Q = [Q_1^T, \ldots, Q_{n_a}^T]^T$ et le vecteur de commande $\dot{Q} = [\dot{Q}_1^T, \ldots, \dot{Q}_{n_a}^T]^T$.

3.5.2 Modélisation des caméras

Toutes les caméras utilisées sont représentées grâce à un modèle sténopé (ou Pinhole Model) comme présenté sur la figure 3.2. Ce modèle est construit sur l'hypothèse que tous les rayons passent par un seul point appelé centre optique. Dans notre cas le centre optique se trouve être l'origine du repère caméra F_{ci}. De façon à calculer la transformation entre le repère de la caméra F_{ci} et le plan image $F_{I_i} = (O_{I_i}, \mathbf{X_{I_i}}, \mathbf{Y_{I_i}})$, une matrice de projection $H_{I_i/ci}$ est utilisée.

$$\begin{bmatrix} X_{I_i} \\ Y_{I_i} \\ z_{ci} \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} f/z_{ci} & 0 & 0 & 0 \\ 0 & f/z_{ci} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{H_{I_i/ci}} \begin{bmatrix} x_{ci} \\ y_{ci} \\ z_{ci} \\ 1 \end{bmatrix} \tag{3.1}$$

avec f la distance focale de la caméra. Cette équation permet d'exprimer un point 3D de coordonnées (x_{ci}, y_{ci}, z_{ci}) du repère F_{ci} dans le plan image de la caméra. Elle réalise donc la projection perspective de ce point 3D en un point 2D de coordonnées (X_{I_i}, Y_{I_i}) du repère F_{I_i} qui est lié à ce plan image.

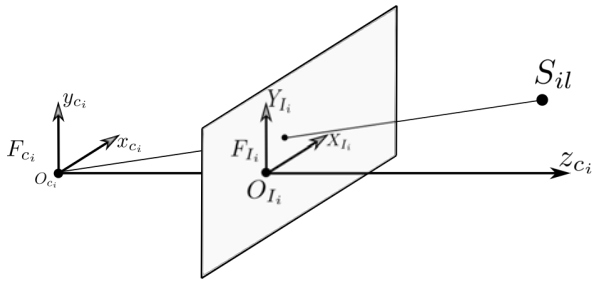


FIGURE 3.2 – Modèle sténopé d'une caméra

Dans notre cas d'étude, les bras sont commandés pour permettre à la caméra d'atteindre une pose définie dans l'image. Pour définir cette pose, les projections sur le plan image de points caractéristiques d'une cible sont utilisés comme indices visuels. Pour cela, nous considérons n_a cibles formées de quatre points. Pour chaque couple caméra/cible, le vecteur des indices visuels alors noté $S_{il} = [X_{il}, Y_{il}]^T$ avec $i \in [1, \ldots, n_a]$ et $l \in [1, 2, 3, 4]$. Ainsi la cible associée à chaque bras est définie par $S_i = [S_{i1}^T, S_{i2}^T, S_{i3}^T, S_{i4}^T]^T$. Comme précédemment, un vecteur représentant les coordonnées de toutes les cibles est construit tel que $S = [S_1^T, \ldots, S_{n_a}^T]^T$. Suivant la même démarche, un vecteur contenant les coordonnées des indices visuels désirés est donné par $S_i^{*T} = [S_{i1}^{*T}, S_{i2}^{*T}, S_{i3}^{*T}, S_{i4}^{*T}]^T$.

3.5.3 Modèles de prédiction

Lors de l'exécution d'un VPC, les coordonnées des indices visuels doivent être prédites de façon à calculer un vecteur de commande optimal permettant la réalisation de la tâche souhaitée. Dans [Allibert et al., 2010], deux méthodes de prédiction sont proposées pour une *free flying camera*, c'est-à-dire une caméra possédant six degrés de liberté (DDL) et n'étant attachée à aucun système robotique. Dans cette partie, nous présentons ces deux approches étendues au cas d'une caméra fixée sur l'organe terminal d'un bras robotique à six DDL.

Modèle global

Cette première méthode consiste généralement à calculer la pose de la caméra à partir du torseur cinématique de cette dernière [Allibert et al., 2010]. Cette approche fait appel à des schémas d'intégration numérique, notamment pour calculer l'orientation. Ici, il est proposé d'utiliser le modèle géométrique du robot ainsi que la matrice de projection décrite dans l'équation (3.1). L'idée consiste alors premièrement à prédire la future configuration du bras connaissant le vecteur de commande appliqué. Dans

un second temps, les indices visuels (2D) sont ramenés dans le repère caméra courant (3D), puis dans le repère caméra prédit (3D) et enfin re-projeté sur le plan image prédit (2D).

Dans un premier temps, nous considérons le système fonctionnant à temps discret avec une période d'échantillonnage T_s, telle que $t_{k+1} = t_k + T_s$. Par la suite nous utiliserons k en lieu et place de t_k pour alléger les notations. La configuration prédite du $i^{\text{ème}}$ bras $Q_i(k+1)$ après l'application de la commande $\dot{Q}_i(k)$ est donnée par :

$$Q_i(k+1) = Q_i(k) + \dot{Q}_i(k)T_s \tag{3.2}$$

Ainsi, connaissant les configurations aux instants k et $k+1$, la construction des matrices de passages homogènes permettant le passage des repères F_b et F_{ci} à ces instants sont calculés et permettant alors la transformation des coordonnées d'un point comme suit :

$$\bar{O}_{F_b}(.) = H_{b/ci}(.)\bar{O}_{F_{ci}}(.) \tag{3.3}$$

où $\bar{O}$ représente le vecteur de coordonnées homogène d'un point dans le repère cartésien de la base F_b ou de la caméra F_{ci}, et $H_{b/ci}$ la matrice de passage homogène permettant la transformation d'un point du repère F_{ci} vers le repère F_b. La notation (.) permet la généralisation de l'équation à tous les instants k. Enfin, par l'association des équations (3.1) et (3.3), les coordonnées des indices visuels mesurés à l'instant k sont alors prédits dans l'espace image de l'instant $k+1$ de la manière suivante :

$$S_{il}(k+1) = H_{Ii/ci}(k+1)H^{-1}_{b/ci}(k+1)H_{b/ci}(k)H^{-1}_{Ii/ci}(k)S_{il}(k) \tag{3.4}$$

Ainsi, lorsque le modèle global est utilisé, la valeur de la profondeur est utilisée deux fois : pour amener les indices visuels dans le repère caméra courant et pour les re-projeter sur le plan image prédit. Dans le cas du modèle pinhole, il est courant de considérer la seconde étape comme peu sensible aux variations sur la profondeur. Inversement, la projection dans le repère caméra à partir du plan image peut s'avérer très sensible aux erreurs sur la valeur de la profondeur. Ainsi, il est possible de penser qu'un modèle global ne fournira des prédictions raisonnablement proches de la valeur réelle que lorsque la profondeur est connue précisément ou que le VPC a lieu dans un domaine où il y a peu de variations sur la profondeur et que la valeur souhaitée est utilisée comme approximation.

Modèle local

La prédiction par modèle local se réfère à l'exploitation de l'équation différentielle représentant la dérivée des coordonnées des indices visuels $\dot{S}_{il}$ par rapport au vecteur de commande $\dot{Q}_i$. Pour cela, la jacobienne du bras robotique J_i, faisant le lien entre le torseur cinématique de la caméra T_{ci} et le vecteur de commande, est requise (équation (3.5)), de même que la matrice d'interaction L_{il} liant $\dot{S}_{il}$ à T_{ci} (équation (3.6)).

$$T_{ci} = J_i\dot{Q}_i \tag{3.5}$$

$$\dot{S}_{il} = L_{il}T_{ci} \tag{3.6}$$

Pour des points, L_{il} est définie par [Chaumette and Hutchinson, 2006] :

$$L_{il} = \begin{bmatrix} -1/z_l & 0 & X_l/z_l & X_lY_l & -(1+X_l^2) & Y_l \\ 0 & -1/z_l & Y_l/z_l & 1+Y_l^2 & -X_lY_l & X_l \end{bmatrix} \tag{3.7}$$

Grâce à la combinaison des équations (3.5) et (3.6), l'évolution des indices visuels est donnée comme suit :

$$\dot{S}_{il} = L_{il} J_i \dot{Q}_i \tag{3.8}$$

L'objectif est d'obtenir la prédiction des indices visuels à partir de cette dernière équation. Une première approche, qui est généralement celle retenue, consiste en l'utilisation d'une approximation linéaire du modèle, telle que :

$$S_{il}(k+1) = S_{il}(k) + L_{il} J_i \dot{Q}_i(k) T_s \tag{3.9}$$

La seconde solution repose sur l'utilisation de méthodes d'intégration numérique plus avancées, requérant davantage de puissance de calcul mais offrant des résultats de prédiction plus précis. Dans ce travail, cette seconde solution est réalisée à l'aide de la méthode de Runge-Kutta classique.

Utilisation des modèles de prédictions

Afin d'utiliser ces modèles sur des systèmes expérimentaux, certains paramètres sont à prendre en compte. En effet, les deux modèles proposés ci-dessus requièrent les valeurs de la profondeur z_{ci} de chaque point formant les cibles (équations (3.1) et (3.6)). Cette valeur peut être soit mesurée dans le cas d'utilisation de caméras 3D, ou bien estimée dans le cas de caméras monoculaires [Durand-Petiteville, 2012]. Une méthode très utilisée en asservissement visuel 2D consiste à fixer cette valeur comme étant une constante représentant la valeur de profondeur initiale ou désirée des indices visuels comme indiqué dans [Chaumette and Hutchinson, 2006].

Suivant la méthode sélectionnée, le résultat de l'approximation sera plus ou moins précis. Dans la section résultats (3.7) nous proposons une évaluation de ces différentes solutions appliquée à notre cas d'application.

3.6 Synthèse d'un asservissement visuel prédictif

Cette partie a pour objectif de synthétiser une loi de commande permettant aux bras de positionner les caméras tout en respectant les contraintes nécessaires au fonctionnement. Dans un premier temps, nous présentons la fonction de coût à minimiser pour atteindre les poses désirées, puis dans un deuxième temps, les contraintes à respecter sont définies.

3.6.1 Présentation du problème

Le correcteur que nous proposons est la combinaison d'un correcteur NMPC et d'un asservissement IBVS. Ainsi, la commande par NMPC permet le calcul d'une séquence de commande optimale $\bar{\dot{Q}}(.)$ qui minimise une fonction de coût J_{N_p} sur un horizon de temps N_p donné et prenant en compte les contraintes définies dans $C(\bar{\dot{Q}}(.))$. La commande par IBVS, quant à elle, permet d'accomplir la tâche principale définie comme une erreur dans l'espace image et permettant de positionner les caméras. Par conséquent, la fonction de coût que le NMPC doit minimiser est décrite comme une tâche classique de l'IBVS. Cette fonction de coût est alors définie comme la somme des erreurs quadratiques entre les coordonnées des indices visuels prédits $\hat{S}(.)$ sur l'horizon

de prédiction N_p et les coordonnées des indices visuels désirés S^*. L'asservissement visuel prédictif peut alors s'écrire comme le problème d'optimisation suivant :

$$\bar{\dot{Q}}(.) = \min_{\dot{Q}(.)} \Big(J_{N_P}(S(k), \dot{Q}(.)) \Big) \tag{3.10}$$

avec

$$J_{N_P}(S(k), \dot{Q}(.)) = \sum_{p=k+1}^{k+N_p-1} [\hat{S}(p) - S^*]^T [\hat{S}(p) - S^*] \tag{3.11}$$

sous la contrainte que

$$\hat{S}(.) = (3.4) \text{ ou } (3.9) \tag{3.12a}$$

$$\hat{S}(k) = S(k) \tag{3.12b}$$

$$C(\bar{\dot{Q}}(.)) \leq 0 \tag{3.12c}$$

Comme indiqué dans la section 3.5, nous utilisons le modèle global ou local pour prédire les coordonnées des indices visuels (3.12a). Pour calculer les prédictions, la dernière mesure des indices visuels est utilisée (équation (3.12b)). Enfin, les contraintes à prendre en compte lors de la minimisation de la fonction de coût sont rassemblées dans l'équation (3.12c).

La résolution de ce problème d'optimisation sous contrainte mène alors à l'obtention du vecteur de commande optimale $\bar{\dot{Q}}(.)$. Dans notre application, seul le premier vecteur de commande $\bar{\dot{Q}}(1)$ est appliqué au système. À l'itération suivante, le problème d'optimisation est relancé et une nouvelle séquence de commande optimale est calculée. Cette boucle est répétée tant que la fonction de coût (équation (3.11)) n'a pas atteint un minimum.

3.6.2 Définition des contraintes

Tel qu'il a été présenté précédemment, la minimisation du problème peut intégrer des contraintes restreignant les solutions possibles. Dans cette partie, nous présentons d'abord les contraintes liées au système robotique comme la géométrie du robot, la vitesse des articulations ou encore le champ de vue des caméras. Nous définirons ensuite un ensemble de contraintes garantissant la non-collision des bras entre eux (lors du partage de l'espace de travail) et avec des éléments extérieurs.

Contraintes du système

Lorsque l'on travaille avec un système robotique, la vitesse de déplacement des articulations ne peut dépasser un certain seuil dû aux limites mécaniques et électriques du système. Ces limites hautes et basses sont représentées ici par deux vecteurs $\dot{Q}_{min}$ et $\dot{Q}_{max}$, de longueur $N_{\dot{Q}}$, avec $N_{\dot{Q}} = n_a m_q N_p$. Ces deux vecteurs sont constitués des limites de vitesse hautes et basses de chaque articulation du système le long de l'horizon de prédiction. En conséquence, afin de garantir la non-saturation des actionneurs, il est nécessaire que le vecteur de commande $\dot{Q}$ appartienne au domaine $\kappa = [\dot{Q}_{min}, \dot{Q}_{max}]$. Nous écrivons alors la contrainte de vitesse des articulations comme suit :

$$\begin{bmatrix} \dot{Q} - \dot{Q}_{max} \\ \dot{Q}_{min} - \dot{Q} \end{bmatrix} \leq 0 \tag{3.13}$$

De manière similaire, les articulations ont un débattement limité. C'est pourquoi, nous définissons des seuils hauts et bas sur leurs valeurs angulaires. Ces seuils sont représentés grâce à deux vecteurs de longueur $N_Q = n_a m_q N_p$ et sont notés Q_{min} et Q_{max}. Ainsi, les contraintes de valeurs angulaires du robot sont données par :

$$\begin{bmatrix} Q - Q_{max} \\ Q_{min} - Q \end{bmatrix} \leq 0 \tag{3.14}$$

Enfin, le contrôle étant réalisé à travers les mesures effectuées par les caméras, les indices visuels doivent nécessairement rester dans leur champ de vue. Pour cela, une contrainte de visibilité est introduite. Elle est définie par la dimension de l'image. Cette contrainte assure alors que les indices visuels restent dans le domaine représenté par deux vecteurs S_{min} et S_{max} de dimension $N_S = 8 n_a N_p$. La contrainte s'écrit comme suit :

$$\begin{bmatrix} S - S_{max} \\ S_{min} - S \end{bmatrix} \leq 0 \tag{3.15}$$

où S_{min} et S_{max} représentent les coordonnées minimales et maximales que peuvent avoir les indices visuels.

Partage de l'espace de travail

Lorsque plusieurs robots manipulateurs partagent un même un espace de travail, il est indispensable d'assurer la non-collision entre les bras. Pour cela, nous utilisons une contrainte basée sur la distance minimale $\hat{d}_{i,j|i',j'}$ entre le $j^{\text{ème}}$ corps du $i^{\text{ème}}$ bras et le $j'^{\text{ème}}$ corps du $i'^{\text{ème}}$ bras. Cette distance minimale entre deux corps est définie comme étant la norme de l'unique segment perpendiculaire aux deux corps, comme illustré dans la figure 3.3. Ainsi, il est possible de définir la contrainte d'évitement d'auto-collision de la manière suivante :

$$\left[D_{min} - \hat{D}\right] \leq 0 \tag{3.16}$$

où D_{min} est un vecteur contenant la valeur de la distance minimale autorisée entre deux corps et $\hat{D}$ représente un vecteur concaténant le résultat des plus courtes distances évaluées entre chaque corps, *i.e.* $\hat{D} = [\hat{d}_{1,1|2,1}, \ldots, \hat{d}_{n_a-1,m_q|n_a,m_q}]^T$. Le calcul des distances entre chaque corps peut influer significativement sur le temps de calcul nécessaire. Il paraît alors important d'effectuer un travail hors ligne pour n'évaluer que les corps susceptibles d'entrer en collision. En effet, suivant la géométrie du robot et ses contraintes mécaniques, certaines articulations n'ont aucun risque de collision avec d'autres. Ainsi, le nombre de liaisons à tester et par conséquent le coût de calcul requis pour cette opération peut être grandement réduit.

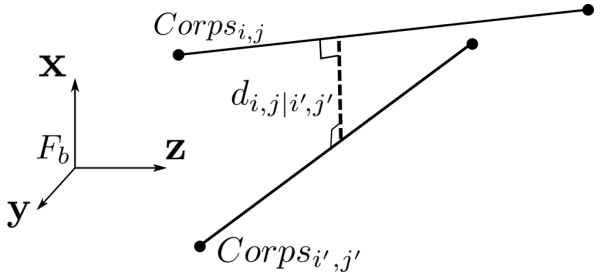


FIGURE 3.3 – Calcul de la plus courte distance $\hat{d}_{i,j|i',j'}$ entre deux corps $C_{i,j}$ et $C_{i',j'}$

Évitement de collision avec un élément extérieur

Lors de l'exécution de la tâche, des obstacles peuvent se situer sur la trajectoire des bras et doivent être évités. Ici, nous considérons des obstacles statiques, représentés par des sphères et dont la position dans l'espace de travail est connue. De la même manière que la contrainte précédente, nous évaluons donc la distance minimale $\hat{d}_{obs}$ entre les corps constituant les bras et le centre de l'obstacle. De plus, nous fixons une distance minimale $d_{obs_{min}}$ à respecter pour éviter la collision. Ainsi, la contrainte d'évitement de collision à un obstacle est définie comme étant :

$$\left[D_{obs_{min}} - \hat{D}_{obs}\right] \leq 0 \tag{3.17}$$

où $\hat{D}_{obs}$ est le vecteur contenant les distances mesurées entre chaque corps et l'obstacle et $D_{obs_{min}}$ un vecteur regroupant les distances de sécurité à respecter. Ces deux vecteurs sont de dimensions maximum $N_{coll} = n_a * m_q$ mais peuvent être réduits en fonction de la géométrie du robot.

Contrainte terminale

Lorsqu'un schéma de contrôle NMPC est utilisé, il est nécessaire de s'assurer que la séquence de commandes calculée permet d'atteindre la pose désirée. Étant donné que le NMPC minimise la distance entre une trajectoire prédite et une trajectoire donnée, il n'est pas garanti que l'ultime état prédit correspond à l'état désiré. De plus, il est possible que l'horizon de prédiction soit trop court et les contraintes trop restrictives pour que l'objectif puisse être atteint. Ainsi, pour assurer la convergence de la loi de commande, il est nécessaire de vérifier que l'ultime état prédit corresponde à l'état désiré, ou autrement dit que le problème de commande optimale est réalisable [Lars and Pannek, 2016]. Pour cela, une contrainte supplémentaire, appelée contrainte terminale, est définie sous la forme d'une erreur entre la prédiction des indices visuels obtenue à la fin de l'horizon de prédiction $\hat{S}_{il}(k+N_p-1)$ et les indices visuels désirés S^*_{il}. Dans le cas où le solveur ne peut trouver une séquence de commandes $\bar{Q}(.)$ qui respecte cette contrainte, alors le problème est considéré comme non-réalisable et aucune garantie ne peut être donnée quant à la convergence du système. Cette contrainte s'écrit sous la forme suivante :

$$\sigma(k+N_p-1) - \varepsilon_S \leq 0 \tag{3.18}$$

Où $\sigma(k+N_p-1)$ est un vecteur de taille $4 * N_a$ dont les éléments sont définis par les erreurs suivantes :

$$\|\hat{S}_{il}(k+N_p-1) - S^*_{il}\|,$$

l et i désignant respectivement les indices de la cible et du bras considéré. ε_S est également un vecteur de dimension $4 * N_a$ et représente la marge d'erreur autorisée pour les indices visuels autours de la valeur souhaitée :

$$\varepsilon_S = [\varepsilon \dots \varepsilon]^T$$

Deux remarques peuvent être faites concernant l'utilisation de la contrainte terminale :

1. Au cours de l'étude bibliographique faite sur la commande VPC, la contrainte terminale n'est jamais apparue, malgré son utilité et une mise en œuvre aisée. Cela peut notamment s'expliquer par le fait que seuls de petits déplacements ont été effectués et que les problèmes de faisabilité ne sont pas apparus. De plus, les auteurs préfèrent avoir recours à une pondération de l'ultime prédiction fonction de la distance entre celle-ci et l'état désiré. Cette technique permet de guider le solveur vers une solution optimale mais contrairement à la contrainte terminale, elle ne garantit pas la convergence du schéma NMPC.
2. L'utilisation d'une contrainte terminale n'est pas une garantie absolue de la réalisation de la tâche : il est aussi primordial de prédire les futurs états avec précision. En effet, dans le cas où les prédictions sont fortement erronées, alors le système réel ne convergera pas vers l'état désiré.

Toutes les contraintes étant à présent définies, elles peuvent être intégrées dans la fonction non-linéaire $C(\bar{\dot{Q}}(.))$ qui devient :

$$C(\bar{\dot{Q}}) \leq 0 \Rightarrow \begin{bmatrix} \dot{Q} - \dot{Q}_{max} \\ \dot{Q}_{min} - \dot{Q} \\ Q - Q_{max} \\ Q_{min} - Q \\ S - S_{max} \\ S_{min} - S \\ D_{min} - \hat{D} \\ D_{obs_{min}} - \hat{D}_{obs} \\ \sigma(k + N_p - 1) - \varepsilon_S \end{bmatrix} \leq 0 \tag{3.19}$$

Finalement, la commande optimale calculée par le solveur devra appartenir au domaine décrit par $C(\bar{\dot{Q}}(.))$.

3.7 Résultats

Dans cette section, il est proposé d'illustrer à l'aide de simulations, les performances du schéma VPC précédemment présenté. Pour cela, nous simulons un système robotique grâce au logiciel *Matlab* [MATLAB, 2010] et l'outil *Robotics Toolbox* développé dans [Corke, 2017]. Le système représenté est un robot muni de deux bras manipulateurs symétriques, chacun comportant sept articulations. Dans cette simulation, seuls les bras manipulateurs seront modélisés.

Nous avons relevé dans la partie 3.5.3 que plusieurs paramètres peuvent influencer la précision des modèles de prédiction. Par conséquent, une première partie comparera la précision de ces modèles pour différentes valeurs des paramètres lors de l'exécution d'une tâche de manipulation simple. Puis, avec le modèle de prédiction choisi, plusieurs simulations plus poussées montreront les résultats du correcteur synthétisé pour à différents cas d'étude.

Dans toutes les simulations présentées, la tâche consiste à faire converger chaque caméra vers une situation souhaitée. Pour cela, une cible est définie pour chaque caméra. Ces cibles se composent de quatre points 3D dans l'espace opérationnel notés $\bar{O}_{il} = [x_{i1},\ y_{i1},\ z_{i1}]^T$. Ils sont initialisés dans le repère désiré de la caméra avec un vecteur de coordonnées $\bar{O}_{F_{ci}} = [\bar{O}_{i1}, \bar{O}_{i2}, \bar{O}_{i3}, \bar{O}_{i4}]^T$. Chacun de ces points est ensuite transposé dans l'espace image des caméras respectives et défini grâce au vecteur $S_i^* = [{S_{i1}^*}^T,\ {S_{i2}^*}^T,\ {S_{i3}^*}^T,\ {S_{i4}^*}^T]^T$. Finalement, l'horizon de prédiction est configuré à

$N_p = 5$. Concernant les contraintes à respecter, la vitesse angulaire de chaque bras doit appartenir au domaine défini par $\kappa = [-0.2\ 0.2]\ rad.s^{-1}$ lors de la première prédiction, puis est augmenté à $\kappa = [-5\ 5]\ rad.s^{-1}$ pour les prédictions suivantes. Cette extension du domaine κ est imposée pour aider le système à vérifier la contrainte terminale sachant que l'horizon de prédiction est limité et que seul le premier vecteur de commande ($\bar{\dot{Q}}(1)$) sera appliqué au système. Dans ces simulations, l'espace image utilisé est normalisé (*i.e.* les paramètres intrinsèques spécifiques à la caméra ne sont pas utilisés). La taille des images normalisées est alors définie arbitrairement avec $S_{min} = [-4\ \ -4]^T$ et $S_{max} = [4\ 4]^T$. Les limites angulaires des articulations sont définies comme illustré dans le tableau 3.1. Enfin, les contraintes permettant l'évitement de collisions entre les bras sont réglées avec $d_{min} = 0.1m$, et pour l'évitement d'obstacle à $d_{obs_{min}} = 0.05m$. Pour chacune des simulations présentées, la configuration initiale des bras est donnée par $q_{init} = [0\ \frac{\pi}{2}\ 0\ 0\ 0\ 0\ 0\ 0\ \frac{\pi}{2}\ 0\ 0\ 0\ 0\ 0]^T$. Finalement, la période d'échantillonnage est fixée à $T_s = 50$ ms.

m_q	$n_a = 1$		$n_a = 2$	
	min (rad)	max (rad)	min (rad)	max (rad)
1	-2.27	0.7	-0.7	2.27
2	-0.5	1.4	-0.5	1.4
3	-3.9	0.77	-0.77	3.9
4	0	2.32	0	2.32
5	inf	inf	inf	inf
6	0	2.27	0	2.27
7	inf	inf	inf	inf

TABLE 3.1 – Valeurs des butées articulaires du robot

3.7.1 Comparaison des modèles de prédictions

Comme nous l'avons mentionné précédemment, les différents modèles de prédiction et leurs paramètres définis dans la partie 3.5.3 vont être comparés pour choisir par la suite celui qui correspond le mieux à notre attente. Nous rappelons que :

1. Deux modèles ont été définis (global et local).
2. Les paramètres de profondeur des indices visuels peuvent être considérés comme connus (z réel) ou fixés (z désiré).
3. Lors de l'utilisation du modèle local, nous avons retenu deux méthodes d'intégration (Euler ou Runge-Kutta).

Par conséquent, l'association de ces différents paramètres mène au test de six configurations différentes. La configuration prenant en compte le modèle global avec la connaissance de la profondeur des indices visuels dans l'image (z réel) est utilisée comme référence ici. En effet, dans cette configuration les erreurs dues à l'estimation de la profondeur et à la phase d'intégration sont nulles.

La méthode de comparaison est exécutée en deux étapes. La première consiste à activer le correcteur NMPC utilisant la configuration de référence (global et z réel) pour accomplir un mouvement simple. La deuxième étape exécute tous les modèles de prédiction utilisant les vecteurs de commande calculés lors de la première étape. Les résultats de positions des indices visuels prédits et ceux calculés grâce au modèle de référence sont alors comparés.

Pour la première étape, nous initialisons la simulation afin d'atteindre une pose de la caméra correspondant à une configuration des bras $q^* = [-\frac{\pi}{8}\ \frac{\pi}{2}\ 0\ 0\ 0\ 0\ 0\ 0\ \frac{\pi}{2}\ 0\ 0\ 0\ 0\ 0]^T$. Le résultat de la simulation est visible sur la figure 3.4 et montre que la situation finale de l'organe terminal (repère rouge) a bien atteint sa situation désirée (repère noir). De plus, la figure 3.5 présente l'évolution des indices indices visuels dans les images gauche (figure 3.5a) et droite (figure 3.5b). Ce résultat montre que la tâche IBVS a bien été exécutée dans l'espace image car les situations finales (croix rouges) et désirée (croix noires) sont confondues.

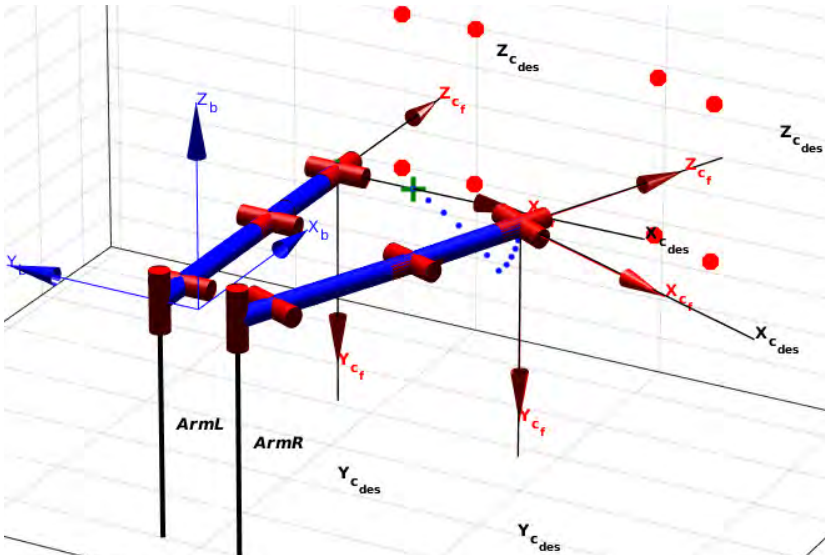
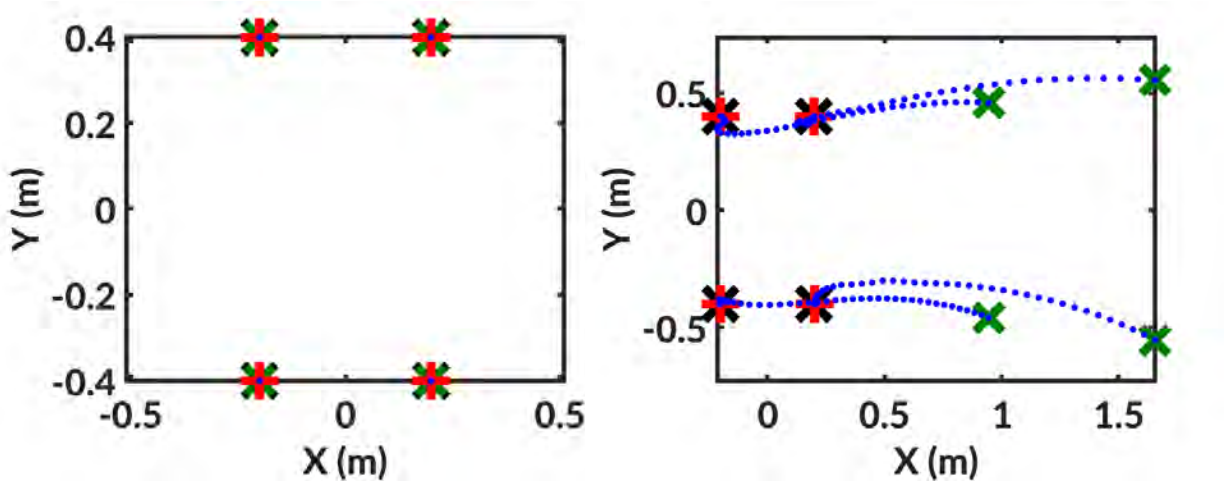


FIGURE 3.4 – Vue 3D - Croix verte : Position initiale de l'organe terminal - Repères rouges : situation finale - Repères noirs : Situation désirée - Points bleus : Trajectoire de l'organe terminal - Points rouges : Cibles



(a) Image gauche (b) Image droite

FIGURE 3.5 – Évolution des indices visuels dans les images - Croix vertes : Position initiale - Croix noires : Position désirée (S^*) - Croix rouges : Position finale - Points bleues : Trajectoire des indices visuels

La simulation utilisant la configuration de référence des modèles de prédiction a été accomplie avec succès. La deuxième étape consistant à comparer les autres modèles est alors exécutée. Le résultat présenté dans la figure 3.6 montre l'évolution de

l'erreur quadratique entre l'état de référence (modèle global avec profondeur réelle) et l'état prédit pour l'ultime itération de l'horizon N_p. La figure 3.7 montre la somme des erreurs quadratiques sur toute la simulation. Ainsi, les différentes configurations de modèle testées ici sont le modèle global (ligne continue), le modèle local avec une intégration effectuée par la méthode d'Euler (ligne tiretée), le modèle local utilisant une intégration numérique Runge Kutta (ligne pointillée). Ceux-ci sont associés à la connaissance de la profondeur des indices visuels z qui peut être la valeur réelle (ligne bleue) ou la valeur désirée (ligne noire). Nous remarquons ici que le modèle global et le modèle local utilisant Runge Kutta donnent de bien meilleurs résultats que ceux obtenus via le modèle local utilisant Euler. En effet, ce dernier introduit de grandes erreurs d'estimation, ce qui pourrait empêcher le système de converger vers la configuration désirée. L'utilisation du z réel mène en général à de meilleures prédictions. Cependant, dans un contexte réel, cette valeur devrait être estimée ou mesurée par des caméras 3D ou autres. Ainsi, nous considérons ici le cas le plus réaliste, où la valeur de profondeur est approximée par une valeur constante z^*. Dans ce cas, le modèle local avec Runge Kutta est le plus précis. C'est pourquoi, nous utiliserons ce modèle pour les simulations à venir.

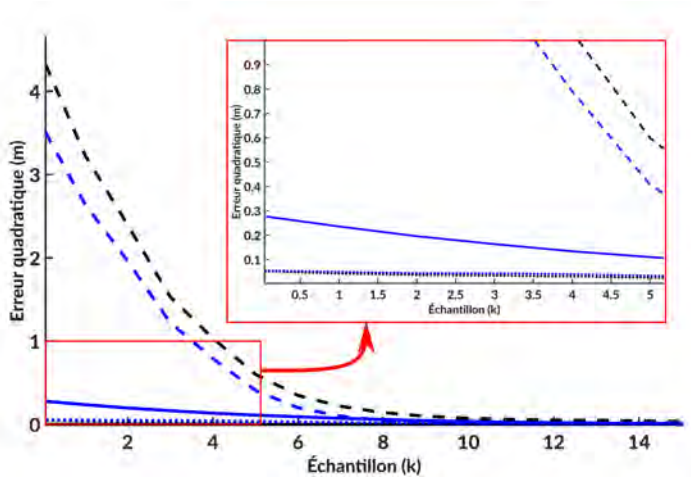


FIGURE 3.6 – Erreur quadratique des prédictions à $N_p = 5$ au cours de la simulation - Lignes continues : Modèle global - Lignes tiretées : Modèle local Euler - Lignes pointillées : Modèle local Runge Kutta - Lignes noires : z réel - Lignes bleues : z désiré

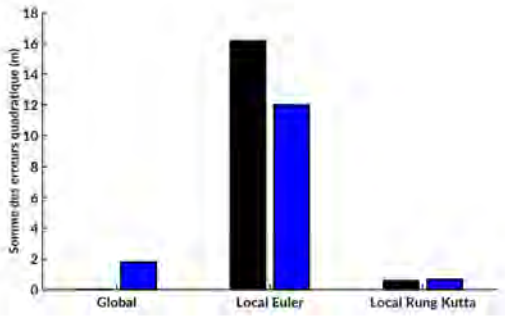
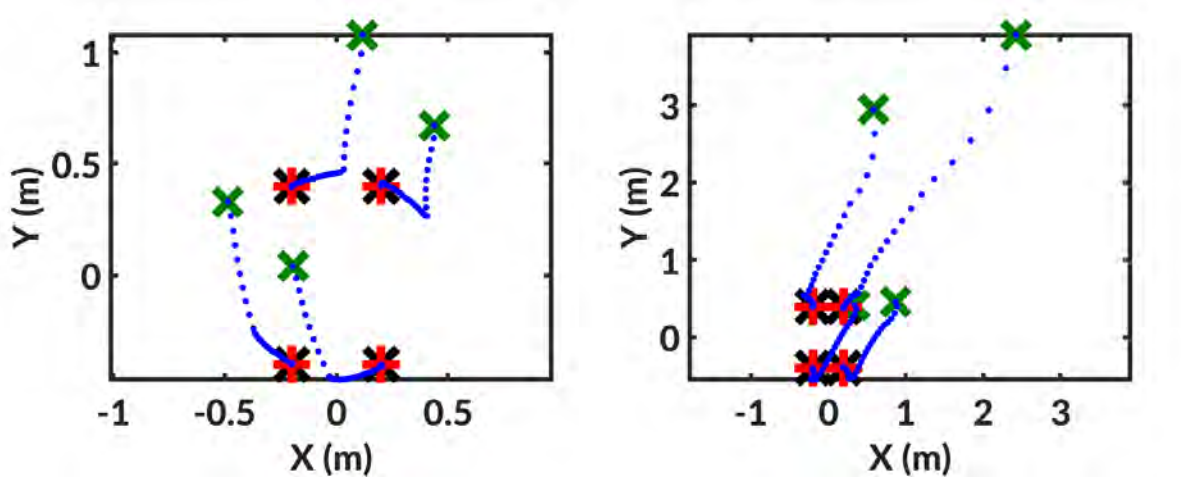


FIGURE 3.7 – Somme des erreurs quadratique des prédictions pour le modèle global, local Euler, local Runge Kutta - Noire : z réel - bleu : z désiré

Mouvement complexe sans contrainte d'évitement de collisions

L'objectif de cette simulation est d'atteindre la position de la caméra définie par la configuration désirée des bras avec $q^* = [-\frac{\pi}{16}\ \frac{3\pi}{8}\ 0\ \frac{\pi}{4}\ 0\ \frac{\pi}{8}\ 0\ \frac{\pi}{16}\ \frac{\pi}{2}\ \frac{\pi}{4}\ \frac{\pi}{8}\ 0\ 0\ 0]^T$ tout en respectant les contraintes énoncées dans la partie 3.6.2. Ici, la prédiction des indices visuels s'effectue grâce au modèle local avec le schéma d'intégration Runge Kutta et une valeur de profondeur des indices visuels fixée à z^*. La figure 3.8 montre que la fonction de coût a bien convergé vers son minimum réalisant la tâche dans l'espace image tandis que la figure 3.9 montre que la réalisation de la tâche a aussi mené à la convergence des organes terminaux vers leurs situations souhaitées dans l'espace opérationnel.



(a) Image gauche (b) Image droite

FIGURE 3.8 – Évolution des indices visuels dans les images - Croix vertes : Position initiale - Croix noires : Position désirée (S^*) - Croix rouges : Position finale - Points bleues : Trajectoire des indices visuels

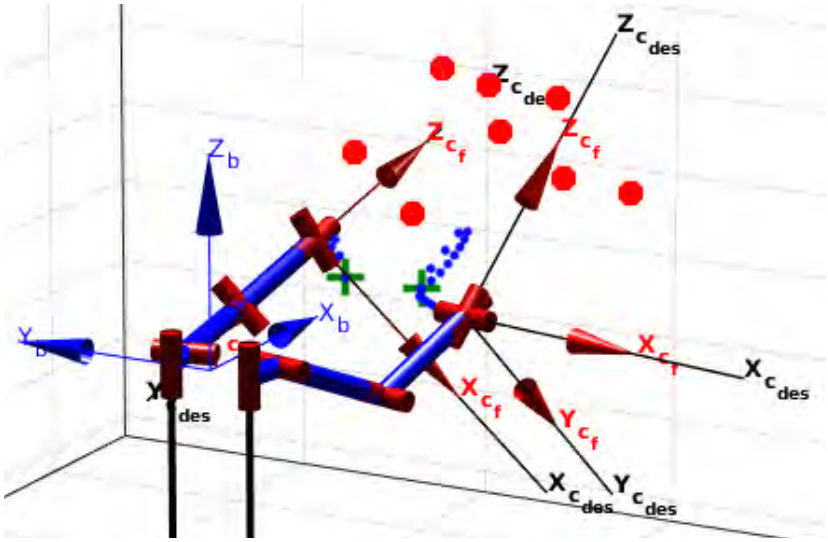
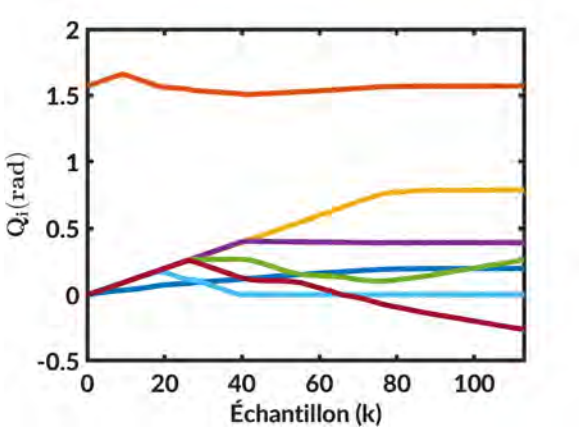


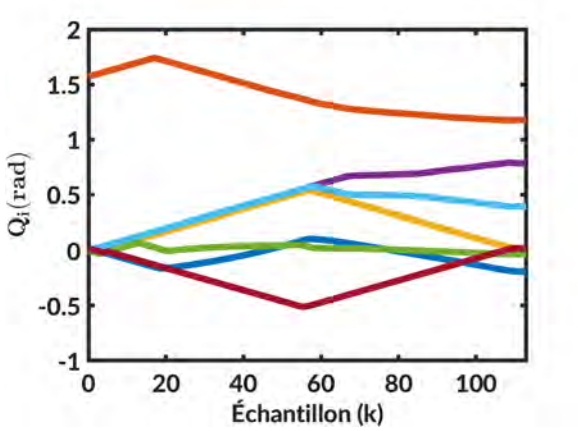
FIGURE 3.9 – Vue 3D - Croix verte : Position initiale de l'organe terminal - Repères rouges : situation finale - Repères noirs : Situation désirée - Points bleus : Trajectoire de l'organe terminal - Points rouges : Cibles

Bien que toutes les contraintes soient actives durant la simulation, la trajectoire des indices visuels n'a pas atteint les limites de l'image. De plus, l'évitement de collision

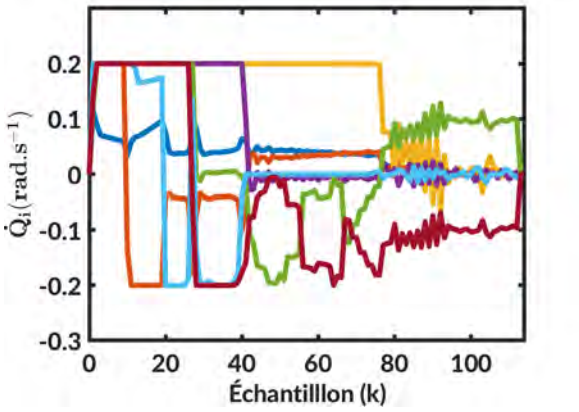
entre les bras n'a pas été nécessaire. En outre, la figure 3.10 montre que les contraintes sur les débattements et vitesses articulaires ont été respectées. En effet, les figures 3.10a et 3.10b représentent l'évolution des configurations des bras gauche et droit tout au long de la simulation. Nous pouvons observer que les articulations peuvent atteindre les limites définies dans le tableau 3.1 mais jamais ne sortent de leurs domaine respectifs. De même, les figures 3.10c et 3.10d représentent l'évolution du vecteur de commande appliqué respectivement aux bras gauche et droit. Nous remarquons ici que les vitesses de chaque articulation restent dans le domaine défini par $\kappa = [-0.2\ 0.2]$.



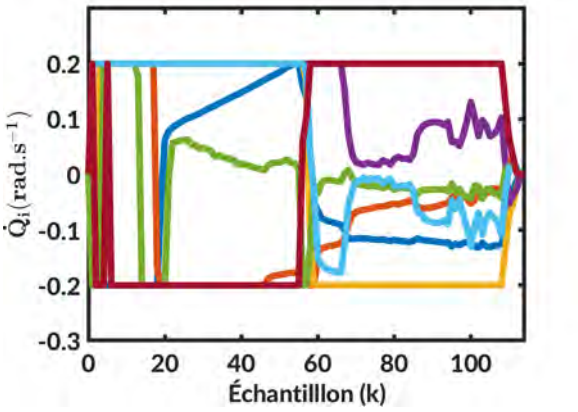
(a) Vecteur de configuration du bras gauche



(b) Vecteur de configuration du bras droit



(c) Vecteur de commande du bras gauche



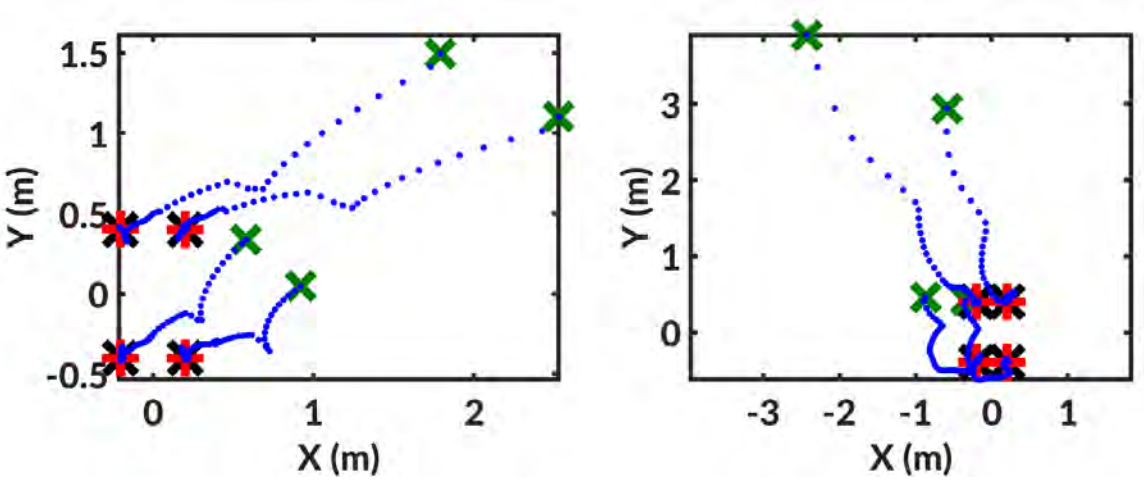
(d) Vecteur de commande du bras droit

FIGURE 3.10 – Vecteurs de configuration et de commande des bras gauche et droit

Mouvement complexe avec contrainte d'évitement de collisions entre les bras

Cette simulation a pour objectif de forcer l'activation de la contrainte de non collision entre les bras. Pour cela, la situation souhaitée des organes terminaux a été définie par la configuration des bras suivante : $q^* = [\frac{\pi}{16}\ \frac{3\pi}{8}\ 0\ \frac{\pi}{4}\ 0\ \frac{\pi}{8}\ 0\ -\frac{\pi}{16}\ \frac{\pi}{2}\ \frac{\pi}{4}\ \frac{\pi}{8}\ 0\ 0\ 0]^T$. Les résultats de la figure 3.11 présentent l'évolution des indices visuels durant la simulation. Ces résultats montrent que la tâche IBVS est à nouveau réalisée avec succès. De même, grâce à la figure 3.12, nous constatons que la réalisation de la tâche dans l'espace image a permis d'atteindre la situation souhaitée des organes terminaux.

De plus, la figure 3.13 illustre le respect de la contrainte imposant une certaine distance entre chaque corps du robot. Nous remarquons ici que bien que certains corps



(a) Image gauche (b) Image droite

FIGURE 3.11 – Évolution des indices visuels dans les images - Croix vertes : Position initiale - Croix noires : Position désirée (S^*) - Croix rouges : Position finale - Points bleues : Trajectoire des indices visuels

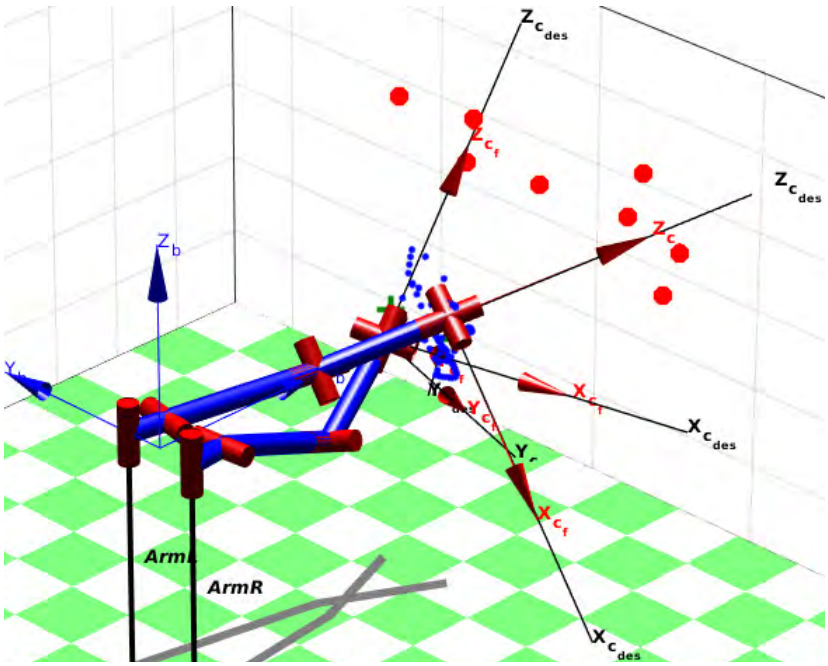


FIGURE 3.12 – Vue 3D - Croix verte : Position initiale de l'organe terminal - Repères rouges : situation finale - Repères noirs : Situation désirée - Points bleus : Trajectoire de l'organe terminal - Points rouges : Cibles

atteignent la limite autorisée, le système ne dépasse jamais la limite fixée à $D_{min} = 0.1$ m. De plus, il peut être noté que le nombre de distances évaluées est ici de six. Ceci est dû au travail préalable effectué pour optimiser le temps de calcul. En effet, dans notre cas, les valeurs de butées articulaires ne permettent pas la collision entre deux corps appartenant au même bras. Ainsi les distances à évaluer correspondent uniquement à des corps de deux bras différents. De plus, les bras sont composés de plusieurs liaisons concourantes. De ce fait, la longueur du corps entre deux de ces liaisons est nulle. Ainsi, le respect de la distance de non-collision de ce corps est validé par le respect de cette même contrainte sur le corps suivant ou précédent. Nous pouvons aussi

constater sur la figure 3.14 que les articulations du bras droit n'atteignent jamais les limites de butées imposées représentées par des lignes rouges tiretées (les articulations affichées sur les figures 3.14e et 3.14g ne possèdent pas de butées articulaires). Enfin, de manière similaire, l'évolution du vecteur de commande est présentée dans la figure 3.15 et montre les saturations du système imposées par les contraintes de vitesse sur les articulations, mais ne dépassent jamais les limites données. Le système a trouvé une solution permettant de satisfaire les contraintes tout en minimisant la fonction de coût qui est, dans notre cas, la réalisation de la tâche IBVS.

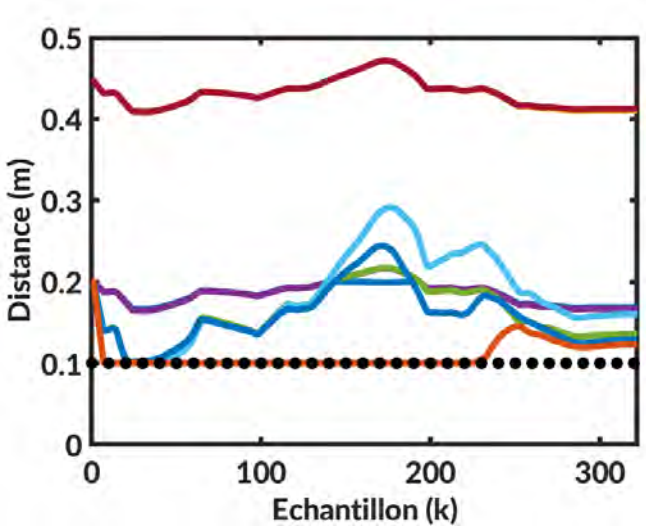
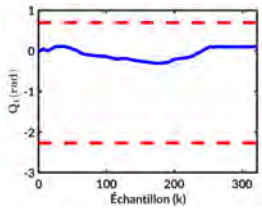
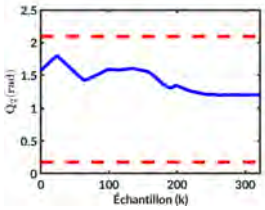


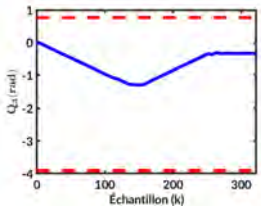
FIGURE 3.13 – Distance de collision d - Lignes continues : Mesure de distance entre les corps, Ligne pointillée : Distance minimale autorisée



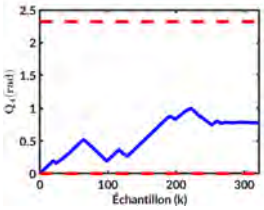
(a) Valeur angulaire de l'articulation q_{11}



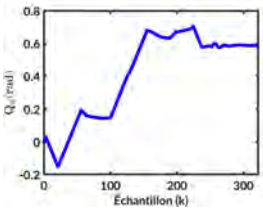
(b) Valeur angulaire de l'articulation q_{12}



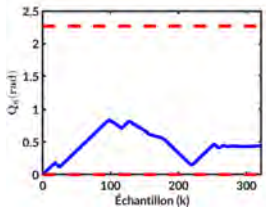
(c) Valeur angulaire de l'articulation q_{13}



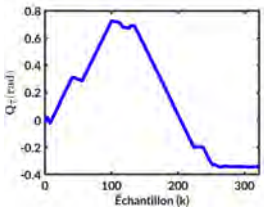
(d) Valeur angulaire de l'articulation q_{14}



(e) Valeur angulaire de l'articulation q_{15}

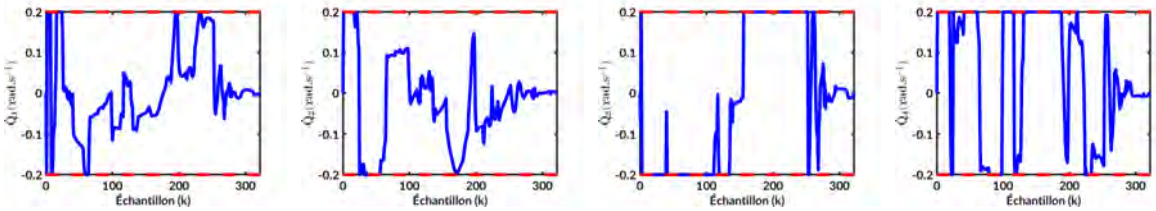


(f) Valeur angulaire de l'articulation q_{16}

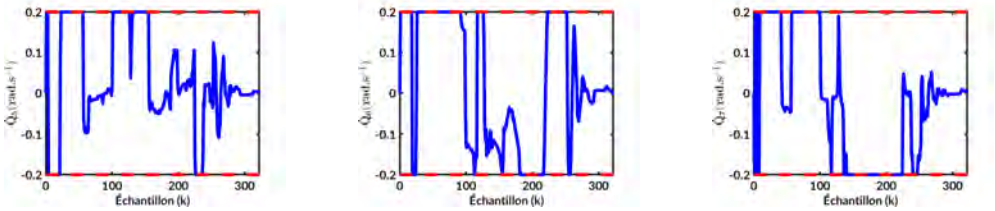


(g) Valeur angulaire de l'articulation q_{17}

FIGURE 3.14 – Configuration du bras droit durant la simulation - Ligne continue bleue : Valeur angulaire de l'articulation - Lignes rouges tiretées : Valeurs de butées



(a) Vitesse angulaire $\dot{q}_{11}$ (b) Vitesse angulaire $\dot{q}_{12}$ (c) Vitesse angulaire $\dot{q}_{13}$ (d) Vitesse angulaire $\dot{q}_{14}$



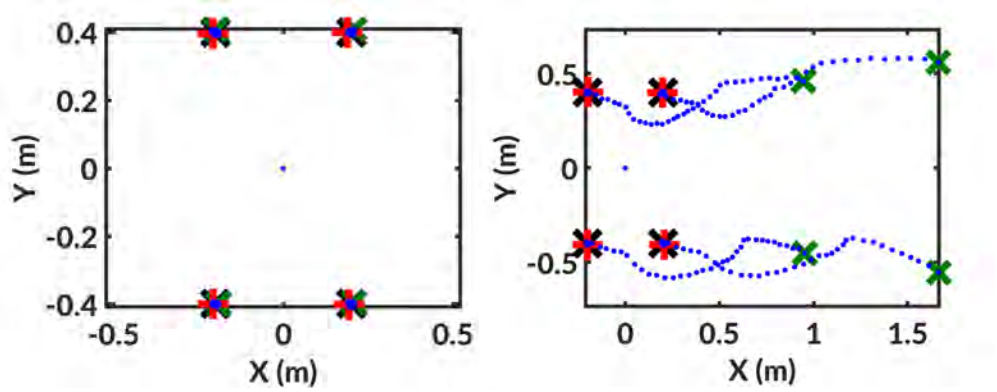
(e) Vitesse angulaire $\dot{q}_{15}$ (f) Vitesse angulaire $\dot{q}_{16}$ (g) Vitesse angulaire $\dot{q}_{17}$

FIGURE 3.15 – Vecteur de commande associé au bras droit durant la simulation - Ligne continue bleue : Vitesse angulaire de l'articulation - Lignes rouges pointillées : Valeurs maximum

Mouvement complexe avec contrainte d'évitement de collisions avec un obstacle extérieur

Cette simulation s'attache à évaluer la capacité du système à respecter la contrainte d'évitement de collision avec un obstacle extérieur fixe. Pour cela, un mouvement relativement simple est exécuté, ayant pour objectif d'atteindre la situation souhaitée des organes terminaux représentée par la configuration $q^* = [-\frac{\pi}{16}\ \frac{3\pi}{8}\ 0\ \frac{\pi}{4}\ 0\ \frac{\pi}{8}\ 0\ \frac{\pi}{16}\ \frac{\pi}{2}\ \frac{\pi}{4}\ \frac{\pi}{8}\ 0\ 0\ 0]^T$. De plus un obstacle est positionné dans le repère global, avec les coordonnées $O_{Obs|_{F_b}} = [0.79, -0.27, -0.08]$. Enfin, une distance minimale à respecter prenant en compte la taille de l'obstacle et la géométrie des corps composant le robot est initialisée à 0.1m.

La figure 3.16 illustre la trajectoire des indices visuels dans le plan image des caméras droite et gauche. Ce résultat montre la convergence des indices visuels vers leurs positions désirées et la minimisation du critère.



(a) Image gauche (b) Image droite

FIGURE 3.16 – Évolution des indices visuels dans les images - Croix vertes : Position initiale - Croix noires : Position désirée (S^*) - Croix rouges : Position finale - Points bleues : Trajectoire des indices visuels

Le résultat du mouvement dans l'espace opérationnel du robot est affiché dans la figure 3.17. Cette figure montre la convergence de la situation des organes terminaux vers leurs situations respectives souhaitées sans collision avec l'obstacle. En effet, nous pouvons constater que la trajectoire des organes terminaux (points bleus) évite avec succès l'obstacle extérieur que nous avons imposé.

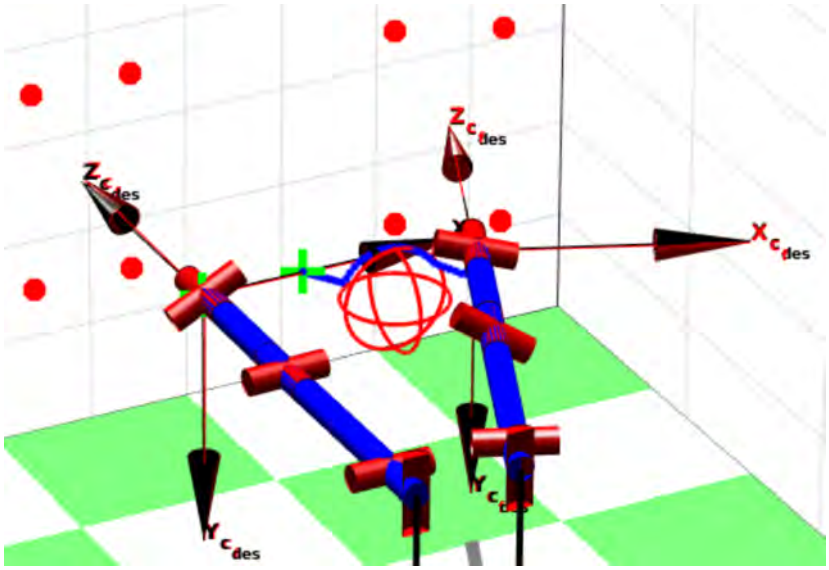


FIGURE 3.17 – Vue 3D - Croix verte : Position initiale de l'organe terminal - Repères rouges : situation finale - Repères noirs : Situation désirée - Points bleus : Trajectoire de l'organe terminal - Points rouges : Cibles

Pour une meilleure visualisation du respect de cette contrainte, la figure 3.18 illustre les distances minimales mesurées entre chaque corps du système et l'obstacle à éviter (lignes continues). De plus, la distance minimale à respecter est représentée par la ligne pointillée noire. Ainsi, nous pouvons observer ici que le système se rapproche de la valeur de distance limite mais ne rentre jamais en collision avec l'obstacle.

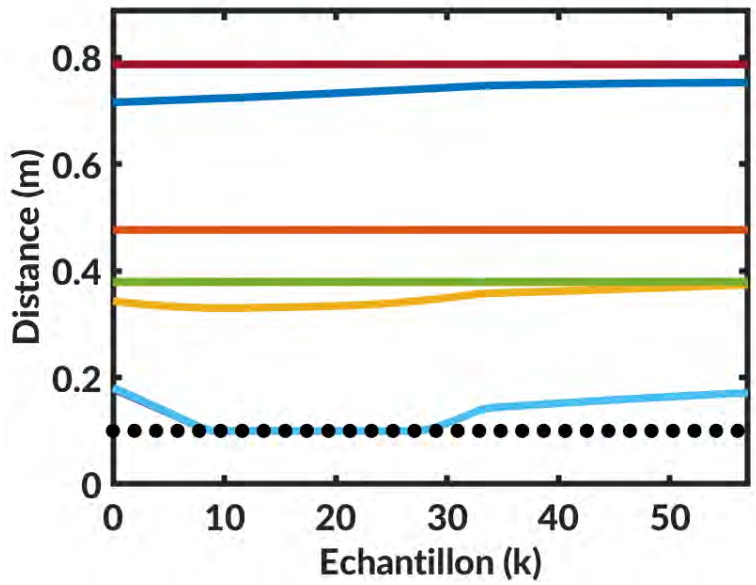
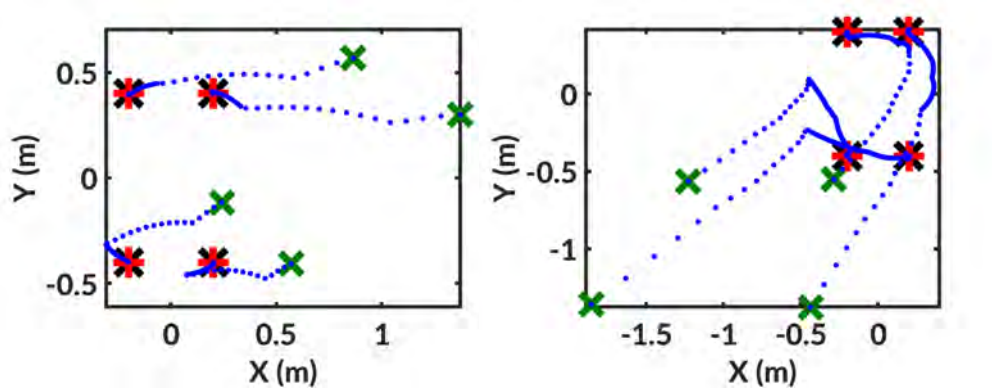


FIGURE 3.18 – Mesure de distance entre les corps des bras et l'obstacle extérieur

Mouvement complexe avec contrainte d'évitement de collisions entre les bras et avec un obstacle extérieur

Cette simulation a pour objectif de montrer la réaction du système face à un risque de collision multiple. En effet, dans cette simulation la trajectoire doit éviter une collision entre les bras et avec un obstacle extérieur. Pour cette simulation le système est initialisé avec la configuration $q_{init} = [-\frac{\pi}{8}\ \frac{\pi}{2}\ 0\ 0\ 0\ 0\ 0\ \frac{\pi}{16}\ \frac{\pi}{2}\ \frac{\pi}{4}\ \frac{\pi}{8}\ 0\ 0\ 0]^T$ correspondant à la configuration finale de la simulation 3.7.1. La configuration désirée est représentée par la configuration $q^* = [\frac{\pi}{20}\ \frac{\pi}{2}\ -\frac{\pi}{2}\ \frac{\pi}{6}\ \pi\ \frac{\pi}{4}\ 0\ -\frac{\pi}{20}\ \frac{\pi}{2}\ \frac{\pi}{2}\ \frac{\pi}{6}\ 0\ 0\ 0]$. L'obstacle extérieur est placé aux coordonnées $\bar{O}_{Obs|F_b} = [0.79, -0.27, -0.08]$ et la distance minimum à respecter est initialisée à 0.1m. Dans la figure 3.19, la convergence des indices visuels dans le plan image des caméras gauche (3.19a) et droite (3.19b) est illustrée.



(a) Image gauche (b) Image droite

FIGURE 3.19 – Évolution des indices visuels dans les images - Croix vertes : Position initiale - Croix noires : Position désirée (S^*) - Croix rouges : Position finale - Points bleues : Trajectoire des indices visuels

La figure 3.20 montre la configuration finale des bras ayant atteint leurs objectifs (respectivement les repères rouges et noirs). De plus, le respect des contraintes d'évitement de collisions est illustré dans cette figure, puisque les trajectoires des organes terminaux (points bleus) contournent l'obstacle rencontré.

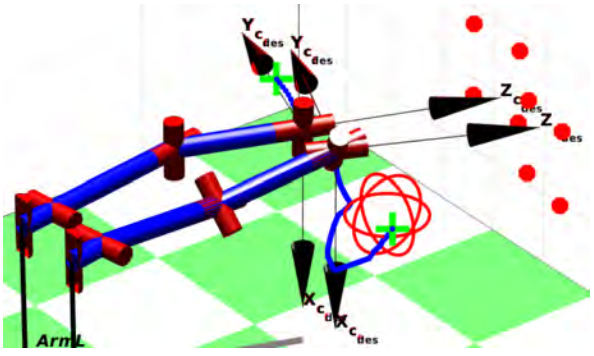
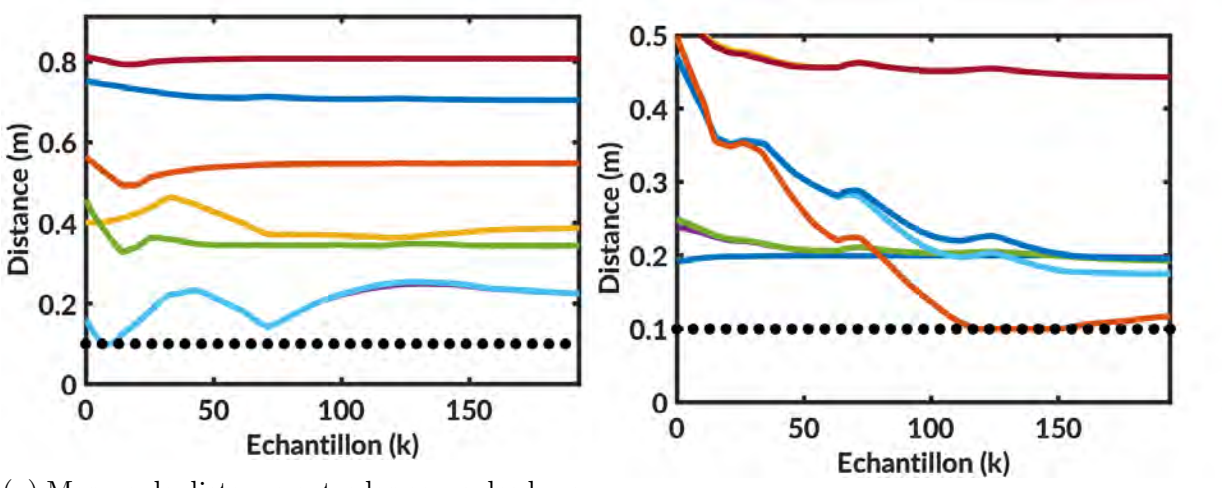


FIGURE 3.20 – Vue 3D - Croix verte : Position initiale de l'organe terminal - Repères rouges : situation finale - Repères noirs : Situation désirée - Points bleus : Trajectoire de l'organe terminal - Points rouges : Cibles

Finalement, la figure 3.21 nous donne un meilleur aperçu du respect des distances imposées au système pour éviter les différentes collisions. Dans les deux résultats, les

lignes pleines de couleur représentent les distances minimales mesurées tandis que les lignes pointillées noires illustrent la distance à respecter. Nous pouvons noter ici que le système ne dépasse jamais la distance limite imposée, ce qui permet d'effectuer une trajectoire en toute sécurité.



(a) Mesure de distance entre les corps des bras et l'obstacle extérieur

(b) Mesure de distance entre les corps des bras

FIGURE 3.21 – Mesure de distances pour l'évitement de collisions

3.8 Conclusion

Ce chapitre a abordé la problématique du contrôle de systèmes robotiques multi-bras partageant un même espace de travail susceptible d'être dynamique. La solution présentée repose sur les techniques d'asservissement visuel 2D couplées avec des techniques de contrôle prédictif. Plus précisément, nous avons défini notre problème sous la forme d'un problème de commande prédictive non linéaire (NMPC) dans lequel la fonction de coût à minimiser est une tâche d'asservissement visuel IBVS à réaliser. De plus, un problème de NMPC permet la prise en compte des contraintes à satisfaire durant la réalisation de la tâche. Ainsi, les contraintes de notre problème de manipulation ont été définies pour prendre en compte les butées et vitesses articulaires des bras, la perte des indices visuels de l'image, les collisions entre les bras et/ou avec un obstacle extérieur. Finalement, cette méthode a été testée en simulation, tout d'abord par l'évaluation des différents modèles de prédiction puis par l'analyse de cas d'utilisation. Cette première phase de simulation a montré que le modèle local utilisant un schéma d'intégration Runge-Kutta donnait les résultats les plus précis en terme de prédiction des indices visuels dans l'image. Puis, l'application d'une commande simple ne violant pas les contraintes de collisions ou de perte des indices visuels a permis une première validation de la stratégie retenue. Enfin, une dernière simulation forçant les bras à partager l'espace de travail encombré par un obstacle a prouvé l'efficacité de cette approche dans des contextes plus exigeants et plus proches de la réalité.

Chapitre 4

Conclusion

Cette thèse s'est donc intéressée à la commande référencée capteur d'un système mobile multi-bras capable d'une part, de naviguer dans un verger de manière autonome et d'autre part, de positionner les organes terminaux de plusieurs bras manipulateurs partageant un même espace de travail. Il s'agissait donc de répondre à deux difficultés majeures : la première concerne la localisation métrique qui s'avère délicate dans un environnement très dynamique et dans lequel le signal GNSS est souvent perturbé ; la seconde porte sur la productivité encore trop réduite pour répondre aux besoins des agriculteurs. Dans ce contexte, nous avons contribué à deux niveaux : tout d'abord, celui de la navigation en développant une stratégie référencée capteur (donc indépendante du GNSS) qui présente l'avantage d'être peu sensible aux imprécisions dues à l'environnement évolutif. Cette stratégie, décrite dans le chapitre 2, repose sur deux éléments fondamentaux qui constituent nos deux contributions sur ce thème :

- le développement d'un algorithme de détection des arbres, robuste aux variations de l'environnement, pour réduire l'instrumentation du verger et renforcer l'autonomie du système ;
- le développement de lois de commande réactives, basées sur des capteurs extéroceptifs embarqués, pour l'exécution de demi-tours d'une rangée vers une autre.

Le premier a été validé en expérimentation, dans différents vergers et pour différentes saisons. Les résultats obtenus ont prouvé l'efficacité de cet algorithme dans des contextes arboricoles très variés, démontrant ainsi sa généricité. De plus son implémentation sur processeur graphique (GPU) lui confère une rapidité d'exécution qui en font un outil très pertinent pour la navigation dans les vergers et plus largement pour une application embarquée. Cet algorithme est un élément central de notre stratégie puisqu'il fournit les données pertinentes pour la navigation et notamment la navigation dans la zone de fourrière où l'absence d'informations sensorielles complexifie le problème. Ces informations sont ensuite utilisées pour nourrir les lois de commande réactives permettant de suivre une rangée ou d'effectuer les demi-tours. Ces derniers sont réalisés en suivant une conchospirale qui peut être adaptée en ligne en fonction de la variabilité du verger permettant de réaliser différentes trajectoires : demi-cercle, demi-tour en Ω ou en queue d'aronde. Différents correcteurs ont été synthétisés et analysés en simulation ou expérimentation. Les résultats obtenus ont montré l'intérêt des spirales dans ce contexte particulier de la commande référencée capteur.

Dans le chapitre 3, nous avons considéré la seconde difficulté évoquée plus haut :

la productivité réduite des systèmes autonomes agricoles dédiés aux vergers. Nous avons tenté de répondre à ce problème en proposant d'équiper la base mobile avec un système multi-bras. La dernière contribution de cette thèse porte donc sur la synthèse d'une stratégie de commande référencée vision permettant de contrôler les mouvements du système multi-bras de manière à prendre en compte les contraintes inhérentes au problème : partage de l'espace de travail, limitations physiques en termes de position et de vitesse, collisions avec l'environnement, perte du signal visuel, etc. L'analyse du problème a montré l'intérêt d'une structure de commande à la fois réactive et capable de considérer un grand nombre de contraintes. Dans ce contexte, l'étude bibliographique a montré tout l'intérêt de l'asservissement visuel prédictif qui offre l'avantage de la réactivité apportée par l'IBVS et de la prise en compte de contraintes par la formulation du problème sous la forme d'un NMPC. Nous avons contribué sur ce thème à deux niveaux :

— la comparaison de différents modèles de prédiction des indices visuels en termes de précision ;
— la formulation mathématique des contraintes, notamment celles concernant le partage de l'espace de travail.

La stratégie de commande proposée a été validée en simulation pour différents cas d'application, simples ou plus complexes. Les résultats obtenus ont montré la pertinence et l'efficacité de notre approche pour positionner précisément les organes terminaux.

Les travaux proposés ont permis d'ouvrir un ensemble de perspectives intéressantes. Différents axes sont envisageables pour la navigation :

— L'algorithme de détection a été validé sur des banques d'images prélevées avec le robot. Toutefois, il n'a pas encore été complètement implémenté dans l'architecture de navigation. Un premier travail sera donc de porter cet algorithme sur le robot dans l'optique de réaliser des missions à plus grande échelle.
— La carte topologique proposée reste encore limitée. En effet, les conditions d'adjacence devraient être définies de manière plus précise, en exploitant au mieux les données sensorielles acquises dans l'optique de la mettre à jour et de la rendre plus générique. A plus long terme, il serait aussi intéressant de construire une carte topologique représentant l'exploitation agricole complète.
— Les fonctions de Lyapunov exhibées dans ce manuscrit n'ont permis d'établir que la stabilité locale du système rebouclé avec certains des correcteurs synthétisés. Une étude plus approfondie permettrait peut-être de parvenir à une preuve de stabilité globale.
— Le demi-tour en Ω n'est pas encore complètement réactif du fait de la présence de l'angle β qui mesure l'orientation du robot par rapport au repère lié au dernier arbre de la rangée. Une première solution permettant de limiter les erreurs serait de s'orienter vers l'odométrie visuelle. Une seconde solution, qui nous semble plus pertinente compte tenu du type de navigation mis en place, serait de modifier le profil en distance utilisé pour le rendre indépendant de cette quantité. Dans ce contexte, exploiter des informations a priori ou bien des données sensorielles complémentaires provenant de la perception des autres rangées pourrait être intéressant. De même, le demi-tour en queue d'aronde proposé constitue une première réponse à la réalisation d'une telle manœuvre basé sur l'orientation globale du robot, actuellement fournie par l'odométrie. Nous pensons ici qu'il pourrait être reformulé en définissant un nouveau profil de distance, unifiant les

méthodes utilisées pour réaliser les deux demi-tours.
— La grande majorité des éléments de la stratégie de navigation ont été validés expérimentalement. Afin d'aller vers une expérimentation à grande échelle dans un verger, il reste encore à tester le demi-tour en Ω, à porter une instance plus évoluée de la carte topologique et à améliorer le basculement entre les différentes lois de commande pour mieux garantir la continuité des ordres envoyés au robot.

Concernant le positionnement des bras pour la manipulation des fruits, les pistes qui s'ouvrent à court terme sont essentiellement techniques :
— les résultats présentés ont été obtenus à l'aide de matlab, en utilisant notamment le solveur intégré. Les temps de calcul actuels ne sont donc pas compatibles avec une application embarquée telle que celle qui est envisagée ici. Ils doivent être réduits. Pour cela, plusieurs pistes sont envisageables. D'un point de vue théorique, un premier travail serait de comparer les modèles de prédiction en considérant à la fois la rapidité d'exécution et la précision fournie. Ensuite, les pistes d'amélioration techniques sont variées : par exemple, une première option serait de décomposer les modèles en processus parallélisables afin de pouvoir exploiter toute la puissance de calcul offerte par les cartes graphiques. Une autre idée pourrait consister à utiliser un "warm start" permettant d'assurer une convergence plus rapide du solveur. Un couplage entre ces deux solutions est également envisageable. Une fois ces améliorations effectuées, il deviendrait possible de tester expérimentalement l'asservissement visuel prédictif proposé.
— A l'heure actuelle, la base mobile et le système multi-bras sont commandés de manière séparée. Une autre étape significative consisterait donc à coordonner le mouvement de ces deux éléments, du moins pour certaines applications (analyse, traitement, etc.).

A plus long terme, la réalisation d'une expérimentation à grande échelle nécessitant des capacités de manipulation et de navigation passera par la prise en compte de domaines connexes à ce travail :
— La perception des fruits à l'aide des caméras embarquées n'a pas fait l'objet des travaux proposés. Ainsi il serait souhaitable de travailler sur la détection des fruits d'une part et l'évaluation de différents indices visuels adaptés à la tâche d'autre part.
— Un travail de planification serait également à fournir dans le but de définir l'ordre des opérations à effectuer (par exemple, l'ordre des fruits à récolter).
— Une étude des moyens de préhension adaptés à la tâche agricole à effectuer doit aussi être menée.

Ces dernières étapes permettront de développer un prototype répondant davantage aux besoins des arboriculteurs, ouvrant la voie à une agriculture plus respectueuse de l'environnement et des hommes.

Liste des publications

- E. Le Flecher, A. Durand-Petiteville, V. Cadenat, T. Sentenac, and S. Vougioukas, “Implementation on a harvesting robot of a sensor-based controller performing a u-turn,” in ECMSM, 2017, pp. 1–6.

- A. Durand-Petiteville, E. Le Flecher, V. Cadenat, T. Sentenac, and S. Vougioukas, “Design of a Sensor-based Controller Performing U-turn to Navigate in Orchards,” in ICINCO, 2017, pp. 172–181.

- A. Durand-Petiteville, E. L. Flecher, V. Cadenat, T. Sentenac, and S. Vougioukas, “Tree detection with low-cost 3D sensors for autonomous navigation in orchards,” RA-L, p. 8, 2018.

- D. Leca, V. Cadenat, T. Sentenac, A. Durand-Petiteville, F. Gouaisbaut, and E. L. Flecher, “Sensor-based Obstacles Avoidance Using Spiral Controllers For an Aircraft Maintenance Inspection Robot,” in European Control Conference, Naples, Italy, 2019, p. 7.

- E. Flécher, A. Durand-Petiteville, F. Gouaisbaut, V. Cadenat, T. Sentenac, and S. Vougioukas, “Nonlinear Output Feedback for Autonomous U-turn Maneuvers of a Robot in Orchard Headlands :,” in Proceedings of the 16th International Conference on Informatics in Control, Automation and Robotics, Prague, Czech Republic, 2019, pp. 355–362.

- E. Flécher, A. Durand-Petiteville, V. Cadenat, and T. Sentenac, “Visual Predictive Control of Robotic Arms with Overlapping Workspace :,” in Proceedings of the 16th International Conference on Informatics in Control, Automation and Robotics, Prague, Czech Republic, 2019, pp. 130–137.

Bibliographie

[Abbas et al., 2017] Abbas, M. A., Milman, R., and Eklund, J. M. (2017). Obstacle avoidance in real time with nonlinear model predictive control of autonomous vehicles. *Canadian journal of electrical and computer engineering*, 40(1) :12–22.

[Allibert et al., 2010] Allibert, G., Courtial, E., and Chaumette, F. (2010). Predictive Control for Constrained Image-Based Visual Servoing. *IEEE Transactions on Robotics*, 26(5) :933–939.

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

[Baeten et al., 2008] Baeten, J., Donné, K., Boedrij, S., Beckers, W., and Claesen, E. (2008). Autonomous Fruit Picking Machine : A Robotic Apple Harvester. In Laugier, C. and Siegwart, R., editors, *Field and Service Robotics*, volume 42, pages 531–539. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Bak and Jakobsen, 2004] Bak, T. and Jakobsen, H. (2004). Agricultural Robotic Platform with Four Wheel Steering for Weed Detection. *Biosystems Engineering*, 87(2) :125–136.

[Barawid Jr et al., 2007] Barawid Jr, O. C., Mizushima, A., Ishii, K., and Noguchi, N. (2007). Development of an autonomous navigation system using a two-dimensional laser scanner in an orchard application. *Biosystems Engineering*, 96(2) :139–149.

[Bargoti et al., 2015] Bargoti, S., Underwood, J. P., Nieto, J. I., and Sukkarieh, S. (2015). A Pipeline for Trunk Detection in Trellis Structured Apple Orchards : A Pipeline for Trunk Detection in Trellis Structured Apple Orchards. *Journal of Field Robotics*, 32(8) :1075–1094.

[Bateux and Marchand, 2017] Bateux, Q. and Marchand, E. (2017). Histograms-based visual servoing. *IEEE Robotics and Automation Letters*, 2 :80–87.

[Bayar, 2013] Bayar, G. (2013). Long distance autonomous trajectory tracking for an orchard vehicle. *Industrial Robot : An International Journal*, 40(1) :27–40.

[Bayar et al., 2015] Bayar, G., Bergerman, M., Koku, A. B., and ilhan Konukseven, E. (2015). Localization and control of an autonomous orchard vehicle. *Computers and Electronics in Agriculture*, 115 :118–128.

[Bechar and Vigneault, 2017] Bechar, A. and Vigneault, C. (2017). Agricultural robots for field operations. Part 2 : Operations and systems. *Biosystems Engineering*, 153 :110–128.

[Bergerman et al., 2016] Bergerman, M., Billingsley, J., Reid, J., and van Henten, E. (2016). Robotics in agriculture and forestry. In *Springer handbook of robotics*, pages 1463–1492. Springer.

[Bergerman et al., 2015] Bergerman, M., Maeta, S. M., Zhang, J., Freitas, G. M., Hamner, B., Singh, S., and Kantor, G. (2015). Robot Farmers : Autonomous Orchard Vehicles Help Tree Fruit Production. *IEEE Robotics & Automation Magazine*, 22(1) :54–63.

[Blackmore, 2009] Blackmore, S. (2009). New concepts in agricultural automation. In *Proceedings of HGCA conference*, page 10.

[Blasco et al., 2002] Blasco, J., Aleixos, N., Roger, J., Rabatel, G., and Molto, E. (2002). Robotic Weed Control using Machine Vision. *Biosystems Engineering*, page 9.

[Blok et al., 2019] Blok, P. M., van Boheemen, K., van Evert, F. K., IJsselmuiden, J., and Kim, G.-H. (2019). Robot navigation in orchards with localization based on Particle filter and Kalman filter. *Computers and Electronics in Agriculture*, 157 :261–269.

[Bochtis and Vougioukas, 2008a] Bochtis, D. and Vougioukas, S. (2008a). Minimising the non-working distance travelled by machines operating in a headland field pattern. *Biosystems engineering*, 101(1) :1–12.

[Bochtis and Vougioukas, 2008b] Bochtis, D. and Vougioukas, S. (2008b). Minimising the non-working distance travelled by machines operating in a headland field pattern. *Biosystems engineering*, 101(1) :1–12.

[Bommarco et al., 2013] Bommarco, R., Kleijn, D., and Potts, S. G. (2013). Ecological intensification : harnessing ecosystem services for food security. *Trends in Ecology & Evolution*, 28(4) :230–238.

[Boyadzhiev, 1999] Boyadzhiev, K. N. (1999). Spirals and Conchospirals in the Flight of Insects. *The college mathematics Journal*, 30(1) :23.

[Bradley Hamner et al., 2011] Bradley Hamner, Marcel Bergerman, and Sanjiv Singh (2011). Autonomous Orchard Vehicles for Specialty Crops Production. In *2011 Louisville, Kentucky, August 7 - August 10, 2011*. American Society of Agricultural and Biological Engineers.

[Cadenat, 1999] Cadenat, V. (1999). *Commande référencée multi-capteurs pour la navigation d'un robot mobile*. PhD thesis, Université Toulouse 3 Paul Sabatier.

[Cariou et al., 2010] Cariou, C., Lenain, R., Thuilot, B., Humbert, T., and Berducat, M. (2010). Maneuvers automation for agricultural vehicle in headland. In *AgEng 2010, International Conference on Agricultural Engineering*, page 10, Clermont-Ferrand, France.

[Ceres et al., 1998] Ceres, R., Pons, J., Jiménez, A., Martín, J., and Calderón, L. (1998). Design and implementation of an aided fruit harvesting robot (agribot). *Industrial Robot : An International Journal*, 25(5) :337–346.

[Chaumette, 1990] Chaumette, F. (1990). *La relation vision-commande : theorie et application a des taches robotiques*. PhD thesis, Universite Rennes 1.

[Chaumette, 1998] Chaumette, F. (1998). Potential problems of stability and convergence in image-based and position-based visual servoing. In Thoma, M., Kriegman, D. J., Hager, G. D., and Morse, A. S., editors, *The confluence of vision and control*, volume 237, pages 66–78. Springer London, London.

[Chaumette, 2003] Chaumette, F. (2003). Avancées récentes en asservissement visuel. In *Journée Nationales de la Recherche en Robotique, JNRR'03*, pages 103–108, Clermont-Ferrand, France.

[Chaumette, 2004] Chaumette, F. (2004). Image Moments : A General and Useful Set of Features for Visual Servoing. *IEEE Transactions on Robotics*, 20(4) :713–723.

[Chaumette and Hutchinson, 2006] Chaumette, F. and Hutchinson, S. (2006). Visual Servo Control Part I : Basic Approaches. *IEEE Robotics & Automation Magazine*, page 9.

[Chaumette and Marchand, 2001] Chaumette, F. and Marchand, T. (2001). A redundancy-based iterative approach for avoiding joint limits : application to visual servoing. *IEEE Transactions on Robotics and Automation*, 17(5) :719–730.

[Chen et al., 2018] Chen, X., Wang, S., Zhang, B., and Luo, L. (2018). Multi-feature fusion tree trunk detection and orchard mobile robot localization using camera/ultrasonic sensors. *Computers and Electronics in Agriculture*, 147 :91–108.

[Chong and Żak, 2001] Chong, E. K. P. and Żak, S. H. (2001). *An introduction to optimization*. Wiley-Interscience series in discrete mathematics and optimization. Wiley, 2nd ed edition.

[Choset and Burdick, 1995] Choset, H. and Burdick, J. (1995). Sensor based planning. II. Incremental construction of the generalized Voronoi graph. In *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, volume 2, pages 1643–1648, Nagoya, Japan. IEEE.

[Corke, 2017] Corke, P. (2017). *Robotics, vision and control*, volume 118. Springer Berlin Heidelberg, New York, NY.

[Cronin et al., 2008] Cronin, G. M., Borg, S. S., and Dunn, M. T. (2008). Using video image analysis to count hens in cages and reduce egg breakage on collection belts. *Australian Journal of Experimental Agriculture*, 48(7) :768.

[Dame and Marchand, 2013] Dame, A. and Marchand, E. (2013). Using mutual information for appearance-based visual path following. *Robotics and Autonomous Systems*, 61(3) :259–270.

[De-An et al., 2011] De-An, Z., Jidong, L., Wei, J., Ying, Z., and Yu, C. (2011). Design and control of an apple harvesting robot. *Biosystems Engineering*, 110(2) :112–122.

[Dijkstra, 1971] Dijkstra, E. W. (1971). *A short introduction to the art of programming*, volume 4. Technische Hogeschool Eindhoven Eindhoven.

[Dunn et al., 2003] Dunn, M., Billingsley, J., and Finch, N. (2003). Machine vision classification of animals. In *Mechatronics and Machine Vision 2003 : Future Trends : Proceedings of the 10th Annual Conference on Mechatronics and Machine Vision in Practice*.

[Durand-Petiteville, 2012] Durand-Petiteville, A. (2012). *Navigation référencée multi-capteurs d'un robot mobile en environnement encombré*. PhD thesis, Université Toulouse 3 Paul Sabatier.

[Eaton et al., 2008] Eaton, R., Katupitiya, J., Siew, K. W., and Dang, K. S. (2008). Precision Guidance of Agricultural Tractors for Autonomous Farming. In *Proceedings of IEEE Systems Conference*, pages 1–8, Montreal, QC, Canada. IEEE.

[Emmi et al., 2019] Emmi, L., Dufour, J., Cadenat, V., and Devy, M. (2019). Hybrid topological localization and mapping for autonomous agricultural robots. In *Proceedings of European conference of precision agriculture*, Montpellier, fr.

[Emmi, 2014] Emmi, L. A. (2014). *Contributions to the configuration of fleets of robots for precision agriculture*. PhD thesis, Universidad Complutense de Madrid, Madrid, Spain.

[Fang et al., 2006] Fang, H., Fan, R., Thuilot, B., and Martinet, P. (2006). Trajectory tracking control of farm vehicles in presence of sliding. *Robotics and Autonomous Systems*, 54(10) :828–839.

[FAO, 2019] FAO (2019). *The state of the world's biodiversity for food and agriculture.* Food and Agriculture Organization of the United Nations. OCLC : 1091250294.

[Fleurmond, 2016] Fleurmond, R. (2016). *Asservissement visuel coordonnÃ© de deux bras manipulateurs*. PhD thesis, Université Toulouse 3 Paul Sabatier.

[Foglia and Reina, 2006] Foglia, M. M. and Reina, G. (2006). Agricultural robot for radicchio harvesting. *Journal of Field Robotics*, 23(6-7) :363–377.

[Foley et al., 2011] Foley, J. A., Ramankutty, N., Brauman, K. A., Cassidy, E. S., Gerber, J. S., Johnston, M., Mueller, N. D., O'Connell, C., Ray, D. K., West, P. C., Balzer, C., Bennett, E. M., Carpenter, S. R., Hill, J., Monfreda, C., Polasky, S., RockstrÃ¶m, J., Sheehan, J., Siebert, S., Tilman, D., and Zaks, D. P. M. (2011). Solutions for a cultivated planet. *Nature*, 478 :337–342.

[Folio, 2007] Folio, D. (2007). *Stratégies de commande référencée multi-capteurs et gestion de la perte du signal visuel pour la navigation d'un robot mobile*. PhD thesis, Université Toulouse Paul Sabatier, Toulouse.

[Francois Chaumette, 2002] Francois Chaumette (2002). Asservissement visuel. In *La commande des robots manipulateurs*, volume 3 of *TraitÃ© IC2*, pages 105–150. Chap.

[François Chaumette and Seth Huchinson, 2007] François Chaumette and Seth Huchinson (2007). Visual Servo Control Part II : Advanced Approaches. *IEEE Robotics & Automation Magazine*.

[Futterlieb et al., 2014] Futterlieb, M., Cadenat, V., and Sentenac, T. (2014). A Navigational Framework Combining Visual Servoing and Spiral Obstacle Avoidance Techniques :. In *Proceedings of the 11th International Conference on Informatics in Control, Automation and Robotics*, pages 57–64, Vienna, Austria. SCITEPRESS - Science and and Technology Publications.

[Grewal, 2011] Grewal, M. S. (2011). *Kalman filtering*. Springer.

[Grift et al., 2008] Grift, T., Zhang, Q., Kondo, N., and Ting, K. C. (2008). A review of automation and robotics for the bio- industry. *Journal of Biomechatronics Engineering*, 1(1) :19.

[Gée C, 2015] Gée C (2015). L'agriculture de précision au service de l'agroécologie.

[Hadj-Abdelkader et al., 2008] Hadj-Abdelkader, H., Mezouar, Y., Martinet, P., and Chaumette, F. (2008). Catadioptric visual servoing from 3-d straight lines. *IEEE Transactions on Robotics*, 24(3) :652–665.

[Hajiloo et al., 2016] Hajiloo, A., Keshmiri, M., Xie, W., and Wang, T. (2016). Robust online model predictive control for a constrained image-based visual servoing. *IEEE Transactions on Industrial Electronics*, 63(4) :2242–2250.

[Hansen et al., 2009] Hansen, S., Blanke, M., and Andersen, J. C. (2009). Autonomous Tractor Navigation in Orchard - Diagnosis and Supervision for Enhanced Availability. *IFAC Proceedings Volumes*, 42(8) :360–365.

[Hansson and Servin, 2010] Hansson, A. and Servin, M. (2010). Semi-autonomous shared control of large-scale manipulator arms. *Control Engineering Practice*, 18(9) :1069 – 1076.

[Hart et al., 1968] Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A formal basis for the heuristic determination. *IEEE Transactions On Systems Science And Cybernetics*, page 8.

[Hayashi et al., 2002] Hayashi, S., Ganno, K., Ishii, Y., and Tanaka, I. (2002). Robotic Harvesting System for Eggplants. *Japan Agricultural Research Quarterly : JARQ*, 36(3) :163–168.

[Hayashi et al., 2010] Hayashi, S., Shigematsu, K., Yamamoto, S., Kobayashi, K., Kohno, Y., Kamata, J., and Kurita, M. (2010). Evaluation of a strawberry-harvesting robot in a field test. *Biosystems Engineering*, 105(2) :160–171.

[He et al., 2011] He, B., Liu, G., Ji, Y., Si, Y., and Gao, R. (2011). Auto Recognition of Navigation Path for Harvest Robot Based on Machine Vision. In Li, D., Liu, Y., and Chen, Y., editors, *Computer and Computing Technologies in Agriculture IV*, volume 344, pages 138–148. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Henten et al., 2002] Henten, E., Hemming, J., Van Tuijl, B., Bontsema, J., Kornet, J., Meuleman, J., and Van Os, E. (2002). An Autonomous Robot for Harvesting Cucumbers in Greenhouses. *Journal of Field Robotics*, page 18.

[Heshmati-alamdari et al., 2014] Heshmati-alamdari, S., Karavas, G. K., Eqtami, A., Drossakis, M., and Kyriakopoulos, K. J. (2014). 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.

[Hofstee et al., 2009] Hofstee, J. W., Spätjens, L., and Ijken, H. (2009). Optimal path planning for field operations. In *Proceedings of the 7th European conference on precision agriculture*, pages 511–519.

[Hsu et al., 2006] Hsu, D., Latombe, J.-C., and Kurniawati, H. (2006). On the Probabilistic Foundations of Probabilistic Roadmap Planning. *The International Journal of Robotics Research*, 25(7) :627–643.

[INRA, 2019] INRA (2019). Dictionnaire de l'agroécologie. https://dicoagroecologie.fr/. Accessed : 2019-07-10.

[Isidori, 2013] Isidori, A. (2013). *Nonlinear control systems*. Springer Science & Business Media.

[J. Jin and L. Tang, 2010] J. Jin and L. Tang (2010). Optimal Coverage Path Planning for Arable Farming on 2d Surfaces. *Transactions of the ASABE*, 53(1) :283–295.

[Jasour and Farrokhi, 2009] Jasour, A. M. and Farrokhi, M. (2009). Path tracking and obstacle avoidance for redundant robotic arms using fuzzy NMPC. In *2009 American Control Conference*, pages 1353–1358, St. Louis, MO, USA. IEEE.

[Jianyang Zheng et al., 2005] Jianyang Zheng, Yinhai Wang, and Nihan, N. (2005). Quantitative evaluation of GPS performance under forest canopies. In *Proceedings. 2005 IEEE Networking, Sensing and Control.*, pages 777–782, Tucson, AZ, USA. IEEE.

[Johnson et al., 2009] Johnson, D. A., Naffin, D. J., Puhalla, J. S., Sanchez, J., and Wellington, C. K. (2009). Development and implementation of a team of robotic tractors for autonomous peat moss harvesting. *Journal of Field Robotics*, 26(6-7) :549–571.

[Juman et al., 2016] Juman, M. A., Wong, Y. W., Rajkumar, R. K., and Goh, L. J. (2016). A novel tree trunk detection method for oil-palm plantation navigation. *Computers and Electronics in Agriculture*, 128 :172–180.

[Kanjilal et al., 2014] Kanjilal, D., Singh, D., Reddy, R., and Mathew, J. (2014). Smart farm : extending automation to the farm level. *Int. J. Sci. Technol. Res*, 3(7) :109–113.

[Katupitiya, 2014] Katupitiya, J. (2014). An Autonomous Seeder for Broad Acre Crops. *American Society of Agricultural and Biological Engineers*, page 8.

[Kazemi et al., 2010] Kazemi, M., Gupta, K., and Mehrandezh, M. (2010). Path-Planning for Visual Servoing : A Review and Issues. In Morari, M., Thoma, M., Chesi, G., and Hashimoto, K., editors, *Visual Servoing via Advanced Numerical Methods*, volume 401, pages 189–207. Springer London, London.

[Ke et al., 2017] Ke, F., Li, Z., Xiao, H., and Zhang, X. (2017). Visual servoing of constrained mobile robots based on model predictive control. *IEEE Transactions on Systems, Man, and Cybernetics : Systems*, 47(7) :1428–1438.

[Keller et al., 2014] Keller, M., Hoffmann, F., Hass, C., Bertram, T., and Seewald, A. (2014). Planning of Optimal Collision Avoidance Trajectories with Timed Elastic Bands. *IFAC Proceedings Volumes*, 47(3) :9822–9827.

[Kermorgant, 2011] Kermorgant, O. (2011). *Fusion d'informations multi-capteurs en asservissement visuel*. PhD thesis, Universite Rennes 1, Rennes, FR.

[Lars and Pannek, 2016] Lars, G. and Pannek, J. (2016). *Nonlinear model predictive control*. Springer Berlin Heidelberg, New York, NY.

[Latombe, 2012] Latombe, J.-C. (2012). *Robot Motion Planning*. Springer Science & Business Media. Google-Books-ID : nQ7aBwAAQBAJ.

[LaValle, 2006] LaValle, S. M. (2006). *Planning Algorithms*. Cambridge University Press, Cambridge.

[LaValle et al., 2000] LaValle, S. M., Kuffner, J. J., and Jr. (2000). Rapidly-exploring random trees : Progress and prospects.

[Leca et al., 2019] Leca, D., Cadenat, V., Sentenac, T., Durand-Petiteville, A., Gouaisbaut, F., and Flecher, E. L. (2019). Sensor-based Obstacles Avoidance Using Spiral Controllers For an Aircraft Maintenance Inspection Robot. In *European Control Conference*, page 7, Naples, Italy.

[Liu et al., 2017] Liu, C., Lee, S., Varnhagen, S., and Tseng, H. E. (2017). Path planning for autonomous vehicles using model predictive control. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 174–179, Los Angeles, CA, USA. IEEE.

[Liu et al., 2018] Liu, T.-H., Ehsani, R., Toudeshki, A., Zou, X.-J., and Wang, H.-J. (2018). Detection of citrus fruit and tree trunks in natural environments using a multi-elliptical boundary model. *Computers in Industry*, 99 :9–16.

[Longo and Muscato, 2013] Longo, D. and Muscato, G. (2013). Design and Simulation of Two Robotic Systems for Automatic Artichoke Harvesting. *Robotics*, 2(4) :217–230.

[Mansard, 2006] Mansard, N. (2006). *Enchaînement de tâches robotiques.* PhD thesis, Université Rennes 1, Rennes, FR.

[Mansard et al., 2018] Mansard, N., DelPrete, A., Geisert, M., Tonneau, S., and Stasse, O. (2018). Using a Memory of Motion to Efficiently Warm-Start a Nonlinear Predictive Controller. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2986–2993, Brisbane, QLD. IEEE.

[Marchand et al., 1996] Marchand, E., Chaumette, F., and Rizzo, A. (1996). Using the task function approach to avoid robot joint limits and kinematic singularities in visual servoing. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS '96*, volume 3, pages 1083–1090, Osaka, Japan. IEEE.

[MATLAB, 2010] MATLAB (2010). *version 7.10.0 (R2010a).* The MathWorks Inc., Natick, Massachusetts.

[Mcfadyen et al., 2014] Mcfadyen, A., Durand-Petiteville, A., and Mejias, L. (2014). Decision strategies for automated visual collision avoidance. In *International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 715–725, Orlando, FL, USA. IEEE.

[Mcfadyen et al., 2013] Mcfadyen, A., Mejias, L., Corke, P., and Pradalier, C. (2013). Aircraft collision avoidance using spherical visual predictive control and single point features. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 50–56. IEEE.

[Merchan-Cruz and Morris, 2006] Merchan-Cruz, E. and Morris, A. (2006). Fuzzy-GA-based trajectory planner for robot manipulators sharing a common workspace. *IEEE Transactions on Robotics*, 22(4) :613–624.

[Mettin et al., 2009] Mettin, U., Shiriaev, A. S., Freidovich, L. B., and Westerberg, S. (2009). Trajectory planning and time-independent motion control for a kinematically redundant hydraulic manipulator. In *Proceedings of International Conference on Advanced Robotics*, page 6, Munich, Germany. IEEE.

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

[Ministère de la Transition écologique et solidaire, 2018] Ministère de la Transition écologique et solidaire (2018). plan ecophyto ii.

[Morris, 2007] Morris, J. R. (2007). Development and Commercialization of a Complete Vineyard Mechanization System. *HortTechnology*, pages 411–420.

[Oberti et al., 2016] Oberti, R., Marchi, M., Tirelli, P., Calcante, A., Iriti, M., Tona, E., Hočevar, M., Baur, J., Pfaff, J., Schütz, C., and Ulbrich, H. (2016). Selective spraying of grapevines for disease control using a modular agricultural robot. *Biosystems Engineering*, 146 :203–215.

[Ohi et al., 2018] Ohi, N., Lassak, K., Watson, R., Strader, J., Du, Y., Yang, C., Hedrick, G., Nguyen, J., Harper, S., Reynolds, D., Kilic, C., Hikes, J., Mills, S., Castle, C., Buzzo, B., Waterland, N., Gross, J., Park, Y.-L., Li, X., and Gu, Y. (2018). Design of an Autonomous Precision Pollination Robot. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7711–7718, Madrid. IEEE.

[Oksanen and Visala, 2009] Oksanen, T. and Visala, A. (2009). Coverage path planning algorithms for agricultural field machines. *Journal of Field Robotics*, 26(8) :651–668.

[Osborne et al., 2012] Osborne, A., Blake, C., Fullen, B. M., Meredith, D., Phelan, J., McNamara, J., and Cunningham, C. (2012). Prevalence of musculoskeletal disorders among farmers : A systematic review. *American Journal of Industrial Medicine*, 55(2) :143–158.

[Pissard-Gibollet, 1993] Pissard-Gibollet, R. (1993). *Conception et commande par asservissement visuel d'un robot mobile*. PhD thesis, Paris, ENMP.

[Pretty, 2008] Pretty, J. (2008). Agricultural sustainability : concepts, principles and evidence. *Philosophical Transactions of the Royal Society B : Biological Sciences*, 363(1491) :447–465.

[Quinlan and Khatib, 1993] Quinlan, S. and Khatib, O. (1993). Elastic bands : connecting path planning and control. In *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pages 802–807, Atlanta, GA, USA. IEEE Comput. Soc. Press.

[Reed et al., 2001] Reed, J., Miles, S., Butler, J., Baldwin, M., and Noble, R. (2001). AE—Automation and Emerging Technologies. *Journal of Agricultural Engineering Research*, 78(1) :15–23.

[Robmann et al., 2009] Robmann, J., Krahwinkler, P., and Bücken, A. (2009). Mapping and Navigation of Mobile Robots in Natural Environments. In Krüger, T. and Wahl, F. M., editors, *Advances in Robotics Research*, pages 43–52. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Rodriguez-Angeles and Nijmeijer, 2001] Rodriguez-Angeles, A. and Nijmeijer, H. (2001). Coordination of two robot manipulators based on position measurements only. *International Journal of Control*, 74(13) :1311–1323.

[Rosa et al., 2008] Rosa, U., Cheetancheri, K., Gliever, C., Lee, S., Thompson, J., and Slaughter, D. (2008). An electro-mechanical limb shaker for fruit thinning. *Computers and Electronics in Agriculture*, 61(2) :213–221.

[Rossmann et al., 2009] Rossmann, J., Schluse, M., Schlette, C., Buecken, A., Krahwinkler, P., and Emde, M. (2009). Realization of a highly accurate mobile robot system for multi purpose precision forestry applications. In *International Conference on Advanced Robotics*, page 6, Munich, Germany.

[Ruizenaar and Smit, 2017] Ruizenaar, R. and Smit, A. (2017). A method for distributing feed over a plurality of separate feeding locations and a feeding system therefor.

[Rybus et al., 2017] Rybus, T., Seweryn, K., and Sasiadek, J. Z. (2017). Control System for Free-Floating Space Manipulator Based on Nonlinear Model Predictive Control (NMPC). *Journal of Intelligent & Robotic Systems*, 85(3-4) :491–509.

[Rye and Scott, 2018] Rye, J. F. and Scott, S. (2018). International Labour Migration and Food Production in Rural Europe : A Review of the Evidence : International labour migration. *Sociologia Ruralis*, 58(4) :928–952.

[Ryo et al., 2004] Ryo, Tsubota, Noboru, Noguchi, Akira, and Mizushima (2004). Automatic guidance with a laser scanner for a robot tractor in an orchard. In *Automation Technology for Off-Road Equipment Proceedings of the 2004 Conference*. American Society of Agricultural and Biological Engineers.

[Sammons et al., 2005] Sammons, P. J., Furukawa, T., and Bulgin, A. (2005). Autonomous pesticide spraying robot for use in a greenhouse. In *Proceedings of Australian Conference on Robotics and Automation*, volume 1.

[Schor et al., 2016] Schor, N., Bechar, A., Ignat, T., Dombrovsky, A., Elad, Y., and Berman, S. (2016). Robotic Disease Detection in Greenhouses : Combined Detection of Powdery Mildew and Tomato Spotted Wilt Virus. *IEEE Robotics and Automation Letters*, 1(1) :354–360.

[Shalal et al., 2015] Shalal, N., Low, T., McCarthy, C., and Hancock, N. (2015). Orchard mapping and mobile robot localisation using on-board camera and laser scanner data fusion Part A : Tree detection. *Computers and Electronics in Agriculture*, 119 :254–266.

[Sharifi and XiaoQi Chen, 2015] Sharifi, M. and XiaoQi Chen (2015). A novel vision based row guidance approach for navigation of agricultural mobile robots in orchards. In *2015 6th International Conference on Automation, Robotics and Applications (ICARA)*, pages 251–255, Queenstown, New Zealand. IEEE.

[Shim et al., 2012] Shim, T., Adireddy, G., and Yuan, H. (2012). Autonomous vehicle collision avoidance system using path planning and model-predictive-control-based active front steering and wheel torque control. *Proceedings of the Institution of Mechanical Engineers, Part D : Journal of Automobile Engineering*, 226(6) :767–778.

[Siegwart and Nourbakhsh, 2004] Siegwart, R. and Nourbakhsh, I. R. (2004). *Introduction to autonomous mobile robots*. Intelligent robots and autonomous agents. MIT Press, Cambridge, Mass.

[Steensland and Zeigler, 2017] Steensland, A. and Zeigler, D. M. (2017). Global Harvest Initiative, Washington, D.C., October 2017. *Global harvest initiative report*, page 72.

[Subramanian et al., 2006] Subramanian, V., Burks, T. F., and Arroyo, A. (2006). Development of machine vision and laser radar based autonomous vehicle guidance systems for citrus grove navigation. *Computers and electronics in agriculture*, 53(2) :130–143.

[Tahri and Chaumette, 2005] Tahri, O. and Chaumette, F. (2005). Point-based and region-based image moments for visual servoing of planar objects. *IEEE Transactions on Robotics*, 21(6) :1116–1127.

[Taylor et al., 2012] Taylor, J. E., Charlton, D., and Yunez-Naude, A. (2012). The End of Farm Labor Abundance. *Applied Economic Perspectives and Policy*, 34(4) :587–598.

[Taïx et al., 2006] Taïx, M., Souères, P., Frayssinet, H., and Cordesses, L. (2006). Path Planning for Complete Coverage with Agricultural Machines. In Yuta, S., Asama, H., Prassler, E., Tsubouchi, T., and Thrun, S., editors, *Field and Service Robotics*, volume 24, pages 549–558. Springer-Verlag, Berlin/Heidelberg.

[Thrun, 1998] Thrun, S. (1998). Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1) :21–71.

[Todt et al., 2000] Todt, E., Rausch, G., and Suarez, R. (2000). Analysis and classification of multiple robot coordination methods. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 4, pages 3158–3163, San Francisco, CA, USA. IEEE.

[Trevelyan, 1989] Trevelyan, J. P. (1989). Sensing and control for sheep shearing robots. *IEEE Transactions on Robotics and Automation*, 5(6) :716–727.

[Turner et al., 2001] Turner, A., Doxa, M., O'Sullivan, D., and Penn, A. (2001). From Isovists to Visibility Graphs : A Methodology for the Analysis of Architectural Space. *Environment and Planning B : Planning and Design*, 28(1) :103–121.

[van der Lely, 1998] van der Lely, C. (1998). Apparatus for milking animals. US Patent 5,816,190.

[Van Henten et al., 2003] Van Henten, E., Hemming, J., Van Tuijl, B., Kornet, J., and Bontsema, J. (2003). Collision-free Motion Planning for a Cucumber Picking Robot. *Biosystems Engineering*, 86(2) :135–144.

[Van Kuilenburg, 2015] Van Kuilenburg, J. M. (2015). Feed control system, feeding system and method for feeding animals.

[Vougioukas, 2007] Vougioukas, S. G. (2007). Reactive Trajectory Tracking for Mobile Robots based on Non Linear Model Predictive Control. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3074–3079, Rome, Italy. IEEE.

[Vougioukas, 2018] Vougioukas, S. G. (2018). Agricultural Robotics. *Annual Reviez of Control, Robotics, and Autonomous Systems*, page 28.

[Vougioukas et al., 2016] Vougioukas, S. G., Arikapudi, R., and Munic, J. (2016). A Study of Fruit Reachability in Orchard Trees by Linear-Only Motion. *IFAC-PapersOnLine*, 49(16) :277–280.

[Wang et al., 2012] Wang, T., Xie, W., Liu, G., and Zhao, Y. (2012). Quasi-min-max model predictive control for image-based visual servoing. In *2012 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 98–103.

[Zhang et al., 2014] Zhang, J., Maeta, S., Bergerman, M., and Singh, S. (2014). Mapping orchards for autonomous navigation. In *2014 Montreal, Quebec Canada July 16, 2014*, page 1. Proceedings of American Society of Agricultural and Biological Engineers.

[Zhao et al., 2016a] Zhao, Y., Gong, L., Huang, Y., and Liu, C. (2016a). A review of key techniques of vision-based control for harvesting robot. *Computers and Electronics in Agriculture*, 127 :311–323.

[Zhao et al., 2016b] Zhao, Y., Gong, L., Liu, C., and Huang, Y. (2016b). Dual-arm Robot Design and Testing for Harvesting Tomato in Greenhouse. *IFAC-PapersOnLine*, 49(16) :161–165.

[Zhu et al., 2010] Zhu, W., Zhong, F., and Li, X. (2010). Automated Monitoring System of Pig Behavior Based on RFID and ARM-LINUX. In *Proceedings of the Third International Symposium on Intelligent Information Technology and Security Informatics*, pages 431–434, Jian, China. IEEE.

[Zion et al., 2014] Zion, B., Mann, M., Levin, D., Shilo, A., Rubinstein, D., and Shmulevich, I. (2014). Harvest-order planning for a multiarm robotic harvester. *Computers and Electronics in Agriculture*, 103 :75–81.