Navigation autonome d'un robot agricole

Dimitri Leca

Université Fédérale
Toulouse Midi-Pyrénées

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)

Présentée et soutenue par :

Dimitri LECA

le 7 Avril 2021

Titre :

Navigation Autonome d'un Robot Agricole

École doctorale et discipline ou spécialité :

EDSYS : Robotique 4200046

Unité de recherche :

LAAS - Laboratoire d'Analyse et d'Architecture des Systèmes

Directeur/trice(s) de Thèse :

Mme CADENAT Viviane
M. SENTENAC Thierry

Jury :

M. MARTINET Philippe, Rapporteur
M. PRADALIER Cédric, Rapporteur
M. MEZOUAR Youcef, Examinateur
M. DURAND-PETITEVILLE Adrien, Examinateur
M. DANÈS Patrick, Examinateur
M. ARANA AREXOLALEIBA Nestor, Examinateur

When you spring to an idea, and decide it is truth, without evidence, you blind yourself to other possibilities.

***Robin Hobb**, Assassin's Apprentice*

Résumé

Le travail sur lequel porte cette thèse s'inscrit dans le domaine de la robotique agricole. Il s'agit de développer des stratégies de navigation permettant à un robot mobile d'évoluer et d'intervenir de manière autonome et en toute sécurité dans une exploitation. Ce type d'environnement agricole est fortement évolutif et comporte de nombreux obstacles statiques (batiments, zones de stockage, etc.) et dynamiques (voitures, machines agricoles, opérateurs humains, animaux, etc.). La stratégie de navigation proposée doit donc être à la fois réactive et adaptative. Par conséquent, cette thèse se concentre sur la conception de méthodes de navigation référencées capteurs (LiDAR, vision, ...) et d'évitement d'obstacles en environnements statiques mais aussi fortement dynamiques. De par la diversité des environnements et des cas possibles, nous avons souhaité développer des méthodes qui soient les plus génériques possible, pouvant gérer les cas à la fois statiques et dynamiques. Ainsi, nous introduirons d'abord les spirales, qui permettent d'obtenir des trajectoires d'évitement pertinentes et flexibles. Ensuite, nous présenterons notre méthode de navigation et d'évitement d'obstacles, basée sur une paramétrisation dynamique des spirales en fonction de l'évolution de l'environnement. Nous verrons que de par l'aspect générique des spirales, cette méthode peut être aisément adaptée pour fonctionner dans un cadre statique mais aussi dans un cadre dynamique. Pour finir, ces solutions seront validées en simulation, puis portées sur un robot mobile pour des expérimentations en conditions réelles.

This thesis explores the field of agricultural robotics. It aims at developing navigation strategies allowing a mobile robot to navigate safely and autonomously inside a farm. This kind of agricultural environment is highly evolutive and includes many static obstacles (buildings, storage areas, etc.) and dynamic obstacles (cars, agricultural machines, human operators, animals, etc.). The proposed navigation strategy must therefore be both reactive and adaptive. Consequently, this thesis focuses on the design of sensor-based navigation methods (LiDAR, vision, ...) and obstacle avoidance techniques in static but also highly dynamic environments. Due to the diversity of environments and possible cases, we have developed methods that are as generic as possible, able to handle both static and dynamic cases. Thus, we will first introduce the spirals, which allow to obtain relevant and flexible avoidance trajectories. Then, we will present our method of navigation and obstacle avoidance, based on a dynamic parametrization of the spirals according to the evolution of the environment. We will show that due to the generic aspect of spirals, this method can be easily adapted to avoid both static and dynamic obstacles. Finally, these solutions will be first validated in simulation, then implemented on a mobile robot for experiments in real conditions.

Mots-clefs : navigation autonome, évitement d'obstacle, robotique agricole

Remerciements

Premièrement, je tiens à remercier Viviane Cadenat et Thierry Sentenac pour avoir dirigé mes travaux durant ces trois années de doctorat. Leur encadrement scientifique et technique ainsi que leur soutien humain m'auront permis de mener à bien mon doctorat dans les meilleures conditions. Leur confiance ainsi que le temps qu'ils m'ont accordés a été un point essentiel dans la réalisation de ces travaux.

Je tiens à montrer ma reconnaissance à Philippe Martinet et Cédric Pradalier pour avoir accepté d'être les rapporteurs de cette thèse, ainsi qu'à Youcef Mezouar, Adrien Durand-Petiteville et Nestor Arana Arexolaleiba pour avoir accepté d'en être les examinateurs. Je remercie par ailleurs Patrick Danès, responsable de l'équipe Robotique, Action et Perception, pour m'avoir permis de réaliser cette thèse et avoir accepté d'être le président du jury.

Je souhaiterais aussi remercier les personnes du LAAS et plus particulièrement mes collègues des équipes RAP et Gepetto. Plus qu'un lieu de travail, le LAAS a aussi été un lieu de vie et d'échanges au sein duquel il a été un plaisir d'évoluer. J'espère pouvoir retrouver une telle ambiance professionnelle dans la suite de ma carrière.

Pour finir, je tiens à remercier mes parents, ma famille, ainsi que Lou, Florian et Julien, qui ont su m'encourager et me supporter durant cette période intense mais passionnante de ma vie.

Table des Matières

1 Introduction **17**
- 1.1 L'état et les enjeux futurs de l'agriculture dans notre monde moderne 17
- 1.2 La robotisation dans l'agriculture . . . 19
- 1.3 Le projet *Desherb'eur* . . . 22
- 1.4 La navigation dans un contexte agricole . . . 23
- 1.5 État de l'art des méthodes de navigation référencées capteurs . . . 26
  - 1.5.1 Méthodes basées sur les champs de potentiels . . . 27
  - 1.5.2 Méthodes basées sur les histogrammes (VFH) . . . 29
  - 1.5.3 Méthodes basées sur la vitesse et l'accélération des obstacles (VO et AO) et approche par fenêtre dynamique (DWA) . . . 31
  - 1.5.4 Méthodes dites *Follow the Gap* . . . 32
  - 1.5.5 Méthodes dites *Bug* . . . 32
  - 1.5.6 Évitement d'obstacles par suivi d'ellipse . . . 34
- 1.6 Approche proposée dans ces travaux . . . 36
- 1.7 Plan du manuscrit . . . 38

2 Navigation basée capteurs en environnement statique **41**
- 2.1 Modélisation des spirales et du robot . . . 42
  - 2.1.1 Convention et définition des spirales . . . 43
  - 2.1.2 Modèle du robot et notations . . . 45
- 2.2 Lois de commandes pour le suivi de spirales . . . 46
  - 2.2.1 Modélisation du suivi de spirales . . . 46
  - 2.2.2 Erreurs et dynamiques . . . 47
  - 2.2.3 Présentation des lois de commandes . . . 47
- 2.3 Stratégie de navigation . . . 52
  - 2.3.1 Contrôleur GTG (*Go-To-Goal*) . . . 53
  - 2.3.2 Contrôleur SA (*Spiral Avoidance*) . . . 53
  - 2.3.3 Choix des paramètres des spirales . . . 55
  - 2.3.4 Enchaînement des contrôleurs SA et GTG : conditions d'activation et sens de contournement . . . 59
- 2.4 Simulation de la navigation du robot dans un environnement statique 62
  - 2.4.1 Environnement de déplacement du robot . . . 62

2.4.2 Configuration du robot . . . 62
2.4.3 Analyse de la trajectoire du robot, des erreurs et des commandes 63
2.5 Évolution de la méthode dans le calcul du centre de la spirale 66
2.5.1 Deuxième méthode de calcul du centre de la spirale 66
2.5.2 Simulation . . . 69
2.6 Conclusion . . . 71

3 Navigation basée capteurs en environnement dynamique 75
3.1 Problématiques liées aux environnements agricoles dynamiques et approches proposées . . . 76
3.1.1 Description de l'environnement agricole dynamique 76
3.1.2 Les approches proposées . . . 77
3.2 Description de la stratégie de navigation basée sur la méthode *Distance-Angle Adaptatifs* . . . 79
3.2.1 Calcul des valeurs résiduelles de la distance et l'angle entre le robot et le point de l'obstacle le plus proche 79
3.2.2 Calcul de la distance adaptative et de la distance désirée d_a^* . 81
3.2.3 Calcul de l'angle adaptatif et du sens de contournement 82
3.2.4 Évolution de la méthode de navigation basée spirale en environnement dynamique . . . 83
3.2.5 Validation de la méthode *Distance-Angle Adaptatifs* en environnement dynamique simulé . . . 83
3.3 Description de la stratégie de navigation basée sur la méthode *Enhanced Laser Scan* . . . 88
3.3.1 Détection des obstacles mobiles . . . 89
3.3.2 Fusion et caractérisation dynamique des obstacles mobiles . . 89
3.3.3 Projection des points virtuels . . . 90
3.3.4 Évolution de la méthode de navigation basée spirale 92
3.3.5 Validation de la stratégie de navigation avec la méthode *Enhanced Laser Scan* en environnement dynamique simulé 94
3.4 Conclusion . . . 98

4 Expérimentations en environnement statique et dynamique 101
4.1 Présentation du robot . . . 102
4.1.1 Plateforme robotique . . . 103
4.1.2 L'équipement embarqué . . . 104
4.2 Évaluation et comparaison des méthodes existantes dans la *NavStack* de ROS dans trois environnements statiques de référence 106
4.2.1 Présentation de la Navigation Stack de ROS 106
4.2.2 Mise en place des environnements de simulation 111
4.2.3 Comparaison avec "dwa local planner" 112
4.2.4 Discussion et comparaison avec "teb local planner" 124

4.2.5 Conclusion 125
4.3 Expérimentations dans un environnement statique 126
4.3.1 Première expérimentation avec SCP basé barycentre 127
4.3.2 Seconde expérimentation avec SCP basé sur les points projetés 132
4.3.3 Analyses et conclusions des expérimentations en environnement statique 139
4.4 Expérimentations dans un environnement dynamique 140
4.4.1 Validation de la méthode *Distance-Angle Adaptatifs* 140
4.4.2 Présentation des résultats préliminaires sur la méthode de l'*Enhanced Laser Scan* 146
4.5 Conclusion 151

5 Conclusion **153**
5.1 Détails des contributions 154
5.2 Perspectives futures 157
5.2.1 Adaptation à d'autres cinématiques de robot 157
5.2.2 Développement de nouvelles lois de commandes 158
5.2.3 Intégration dans une manipulation sur site 158

Références bibliographiques **161**

Liste des figures

1.1 Différents types de robots agricoles 20
1.2 Trois robots produits par Naïo Technologies (Crédit Tien TRAN) . . 21
1.3 Drone pulvérisant un pesticide 21
1.4 Lots de travail du projet *Desherb'eur* 24
1.5 Différents types de terrains agricoles 26
1.6 Illustration de la méthode APF 28
1.7 Illustration de la méthode DWA 32
1.8 Illustration de quelques *Bug Algorithms* **(a)** Pas d'obstacle - **(b)** Bug-1 - **(c)** Bug-2 - **(c)** Dist-bug - **(e)** IBA 33
1.9 Illustration d'une ellipse d'influence 34
1.10 Interpolation d'un mur par une ellipse 35

2.1 Modèle de la spirale et les notations associées. 43
2.2 Différentes spirales en fonction du signe et de la valeur de α^*. 45
2.3 Modèle du robot. 46
2.4 Exemple de chemin de Dubins 54
2.5 Calcul du SCP dynamique pour différents types d'obstacles nommés cas (a.), (b.) et (c.). 58
2.6 Définition des angles α_b, α_c et α_g 59
2.7 Choix du sens de contournement 61
2.8 Schéma de l'environnement de simulation 62
2.9 Trajectoire du robot en fonction des différents contrôleurs. 64
2.10 Erreur en angle $e_\alpha(t)$ et erreur en distance $e_d(t)$ 64
2.11 Commandes en vitesse linéaire et angulaire envoyées au robot 65
2.12 Utilisation des contrôleurs 65
2.13 Position du barycentre et du but lors du calcul du SOM 66
2.14 Calcul du SCP 68
2.15 Évolution de la position du SCP (points noirs) lors d'un tour complet de l'obstacle (en bleu) effectué par le robot 69
2.16 Trajectoire finale à la fin de la simulation 71
2.17 Erreur en angle $e_\alpha(t)$ et erreur en distance $e_d(t)$ 71
2.18 Commandes en vitesse linéaire et angulaire envoyées au robot 72

2.19 Utilisation des contrôleurs 72

3.1 Calcul de $d_c^{predicted}(t_i)$ 80
3.2 Environnement de simulation 84
3.3 Robot (ligne noir) et les obstacles (points roses) et leur trajectoires (lignes roses) 85
3.4 Consigne angulaire $\omega(t)$ – Erreurs $e_d(t)$, $e_\alpha(t)$, $e_S(t)$ – Distances $d(t)$,$d^*(t)$ et $d_a^*(t)$ 86
3.5 Angle adaptatif $\delta^\alpha(t)$ 88
3.6 Processus de calcul de la dynamique des obstacles et des points virtuels 91
3.7 Positions et déplacements des obstacles à trois moments clefs 95
3.8 Position des obstacles et du robot et trajectoire du robot à six instants : (a.) t = 2s ; (b) t = 7s ; (c) t = 9s ; (d) t = 16s ; (e) t = 20s ; (f) t = 27.5s 96
3.9 Vitesse linéaire (en $m.s^{-1}$), vitesse angulaire (en $rad.s^{-1}$) et contrôleur utilisé (GTG, ω_A, or ω_B) 96
3.10 Erreur en distance $e_d(t)$, erreur en angle $e_\alpha(t)$, distance au SCP $d(t)$ et distance au point physique le plus proche $d_c(t)$ [physical]. 97

4.1 Base robotique 4MOB de Sterela (Crédit Matthieu Herrb) 102
4.2 Dimensions de la base robotique 4MOB de Sterela 103
4.3 Disposition et portée angulaire des LiDARs sur la plateforme Air-Cobot 104
4.4 Plateforme robotique 4MOB équipé de ses capteurs 105
4.5 La *Navigation Stack* de ROS 107
4.6 Schéma de fonctionnement de l'algorithme DWA 110
4.7 Les trois environnements de simulation 113
4.8 Trajectoires réalisées par le robot dans l'environnement 1, guidé par les algorithmes SA et DWA 115
4.9 Évolution de la distance d_c entre le robot et plus proche point LiDAR détecté 115
4.10 Commande de vitesse angulaire $\omega(t)$ 116
4.11 Trajectoires réalisées par le robot dans l'environnement 2, guidé par les algorithmes SA et DWA 117
4.12 Évolution de la distance d_c entre le robot et plus proche point LiDAR détecté 118
4.13 Commande de vitesse angulaire $\omega(t)$ 118
4.14 Trajectoires réalisées par le robot dans l'environnement 3, guidé par les algorithmes SA et DWA 119
4.15 Évolution de la distance d_c entre le robot et plus proche point LiDAR détecté 120
4.16 Commande de vitesse angulaire $\omega(t)$ 121
4.17 Représentation de l'environnement 127
4.18 Trajectoire finale du robot 129

4.19 Erreur de distance $e_d(t)$ . . . 129
4.20 Erreur angulaire $e_\alpha(t)$ . . . 130
4.21 Commandes de vitesse linéaire $v(t)$ et de vitesse angulaire $\omega(t)$ . . . 130
4.22 Expérimentation à différents instants . . . 132
4.23 Représentation de l'environnement . . . 133
4.24 Photographie de l'environnement et des obstacles présents . . . 134
4.25 Trajectoire effectuée par le robot . . . 135
4.26 Trajectoire du robot . . . 136
4.27 Position des barycentres à quatre moments de commutation ($t \simeq 22s$, $t \simeq 47s$, $t \simeq 60s$ et $t \simeq 100s$) . . . 136
4.28 Erreur angulaire $e_\alpha(t)$ et erreur en distance $e_d(t)$ . . . 137
4.29 Consigne angulaire $\omega(t)$ . . . 137
4.30 Contrôleur utilisé . . . 138
4.31 Images de l'expérimentation à quatre moments . . . 138
4.32 Trajectoire du robot et points les plus proches O_c . . . 141
4.33 Évolutions des variables pendant l'expérimentation . . . 142
4.34 Photographies de trois instants de l'expérimentation . . . 142
4.35 Trajectoire du robot et points les plus proches O_c . . . 143
4.36 Évolutions des variables pendant l'expérimentation . . . 143
4.37 Angle adaptatif $\delta^\alpha(t)$ . . . 143
4.38 Photographies de trois instants de l'expérimentation . . . 144
4.39 Trajectoire du robot et points les plus proches O_c . . . 144
4.40 Évolutions des variables pendant l'expérimentation . . . 145
4.41 Angle adaptatif $\delta^\alpha(t)$ . . . 145
4.42 Photographies de trois moments de l'expérimentation . . . 145
4.43 ELS, la personne étant immobile . . . 147
4.44 Enhanced Laser Scan dans deux cas . . . 147
4.45 Enhanced Laser Scan dans quatre cas, avec une personne se déplacant perpendiculairement au robot. . . . 149
4.46 Enhanced Laser Scan dans quatre cas, avec deux personnes mobiles . . . 150

5.1 Robot Dino, avec ses roues orientées (Crédit Tien TRAN) . . . 157

Chapitre 1

Introduction

1.1 L'état et les enjeux futurs de l'agriculture dans notre monde moderne

Dans les prochaines décennies, il est estimé que la population mondiale passera de 7.7 milliards en 2019 à 8.5 milliards en 2030 (augmentation de 10%), pour atteindre 9.7 milliards en 2050 puis 10 milliards en 2100 [United Nations, 2019]. Parmi les défis inhérents à cette explosion démographique se trouve celui d'une production de nourriture suffisante pour subvenir à cette population grandissante. Au sortir de la Seconde Guerre Mondiale, les états ont basé leur politique sur une agriculture productiviste, avec notamment une augmentation des terres arables et un ancrage de la mono-culture. Ces politiques ont eu pour effet, depuis les années soixante, d'augmenter la nourriture produite dans le monde de 145%, tandis que la superficie des terres arables a augmenté de 11% [Pretty, 2008], jusqu'à couvrir un tiers de la surface terrestre [Steensland and Zeigler, 2018]. Cependant, cette industrialisation de l'agriculture commence à montrer ses limites, notamment de par son impact sur l'environnement et les écosystèmes. En effet, proportionnellement à l'augmentation des surfaces cultivables, la surface agricole irriguée a été multipliée par deux, et l'industrie agricole est devenue responsable de 70% des prélèvements d'eau dans les aquifères, les lacs, les fleuves et les rivières [Leaman, 2012], entraînant une pénurie des ressources aquifères, et une répartition inégale des ressources fluviales. Par ailleurs, l'industrialisation de l'agriculture va de pair avec l'utilisation intensive d'engrais, d'insecticides et de pesticides. C'est notamment le cas du glyphosate, dont plus de 8.6 milliards de tonnes ont été déversées depuis son invention en 1974 [Tarazona et al., 2017], et qui s'est révélé depuis être toxique pour l'environnement et cancérigène pour l'homme [Zhang et al., 2019]. De plus, le passage d'une agriculture locale à une agriculture intensive a entraîné une perte de biodiversité ces cinquante dernières années [FAO et al., 2019], [United Nations, 2019].

L'industrialisation de l'agriculture pose aussi un problème de disponibilité de la main-d'œuvre. En effet, le domaine agricole nécessite une main-d'œuvre ouvrière saisonnière, non-qualifiée, peu payée, dans des zones rurales peu peuplées. Par conséquent, la part d'immigrants dans la main-d'œuvre agricole est importante. Elle est de 24% pour l'Espagne, 37% pour l'Italie et 90% pour la Grèce [Rye and Scott, 2018]. Un phénomène comparable existe outre-Atlantique : aux États-Unis, seulement 24% de la main-d'œuvre agricole est née aux USA, le reste venant notamment du Mexique [Taylor et al., 2012]. La saisonnalité de certaines récoltes, notamment celles des fruits et légumes, combinée à l'exode rural, entraîne une pénurie de main-d'œuvre. La récente crise lié au virus SARS-CoV-2 (COVID19), empêchant tout mouvement démographique, a mis en exergue la fragilité d'une industrie agricole mondialisée, et a montré la nécessité d'une autosuffisance agricole régionale [FAO and CELAC, 2020], [Galanakis, 2020], [Nicola et al., 2020], [Laborde et al., 2020].

Ainsi, depuis quelques années, il devient de plus en plus évident qu'un changement profond est nécessaire dans notre industrie agricole. En particulier, le concept d'agro-écologie [Marioara et al., 2019] propose des solutions à l'impasse vers laquelle se dirige le complexe agro-industriel [Steensland and Zeigler, 2018]. Le terme d'agro-écologie recouvre un ensemble de théories et de pratiques agricoles développées à partir des connaissances en écologie et en agronomie. L'agro-écologie plaide notamment pour une révision en profondeur du système de production agricole au delà de la simple approche productiviste : il s'agit de mettre en place une production raisonnée, soucieuse du développement durable, mais aussi intégrée dans les tissus sociaux-économiques, prenant en considération le bien-être des producteurs. Des initiatives internationales et nationales se développent afin d'analyser et de réfléchir aux possibilités et solutions qu'offre l'agro-écologie [Rudolph et al., 2020]. Ainsi, l'informatisation ainsi que la robotisation des exploitations agricoles s'inscrivent donc parfaitement dans ce cadre agro-écologique [Raghavan et al., 2016], [Clements and Shrestha, 2004].

Il apparaît par conséquent évident que dans les prochaines décennies, une nouvelle industrie agricole devra émerger, et celle-ci devra être économe en eau, en pesticide, ainsi qu'en main-d'œuvre, tout en étant respectueuse socialement et éthiquement. Cependant, une telle transformation ne saurait s'opérer sans une prise de recul technique, éthique et sociétale sur la place de la machine en tant qu'acteur dans la production d'un bien aussi vital que la nourriture, ainsi que la prise de conscience de plusieurs dangers. Premièrement, pour les pays en voie de développement, l'agriculture est d'abord un moyen d'autosuffisance alimentaire : la production de nourriture a pour but direct d'alimenter les populations locales. Cependant, la robotisation de cette agriculture pourrait entraîner une diminution de cette capacité d'autosuffisance. Les OGM (Organismes Génétiquement Modifiés) sont un bon exemple de biotechnologie prometteuse, mais dont l'usage déraisonné s'est montré contre-productif, notamment

dans les pays les plus pauvres [Moseley, 2017]. D'un point de vue strictement personnel, je suis persuadé que le développement et le déploiement de robots agricoles ne peut se faire sans les préoccupations suivantes : (i) le robot devra aider l'agriculteur, et non pas le rendre dépendant à une machine ou à une culture spécifique ; (ii) un changement profond dans les distributions des richesses devra être opéré afin de subvenir aux besoins des gens dont les robots effectueront désormais le travail.

1.2 La robotisation dans l'agriculture

Afin d'apporter des solutions techniques aux problèmes exposés dans la partie précédente, des solutions robotiques ont été conçues. L'idée d'une automatisation de l'exploitation agricole a commencé très tôt à être explorée. Dès 1985, [Harrell et al., 1985] développe la possibilité d'un robot qui récolterait des fruits dans un verger. Dans une optique d'automatisation et de robotisation de l'agriculture, quatre grandes catégories de solutions peuvent être dressées.

La première catégorie est celle des solutions basées sur des robots cartésiens ou bien sur des robots à câbles. Pour les robots cartésiens, nous pouvons notamment citer le projet Open-source Farmbot [Erick et al., 2018] (figure 1.1a), visant à concevoir un robot permettant de gérer une parcelle potagère en optimisant l'espace et en réduisant l'apport en eau et en intrants. Pour les robots à câbles, le projet Tarzan [Davies et al., 2018], [Farzan et al., 2018] (figure 1.1b) propose un robot capable de se déplacer le long d'un câble flexible tendu entre deux pylônes. Une des applications recherchée est le phénotypage rapide d'un champ de coton. De par l'infrastructure utilisée, ces solutions impliquent donc un environnement contrôlé et adapté. Ces modifications de l'environnement peuvent se révéler longues et coûteuses. Elles sont aussi invasives, et peuvent rendre difficile l'accès à la parcelle pour des opérateurs humains ou des machines tierces. Cependant, cela offre l'avantage d'un environnement totalement contrôlé, et permet donc de minimiser le nombre d'évènements imprévus à gérer.

La deuxième catégorie de solutions consiste à automatiser des machines agricoles déjà existantes, principalement des tracteurs. Une telle approche a un double avantage : elle ne nécessite aucun changement structurel au sein de l'exploitation, car le dimensionnement des machines agricoles est déjà adapté au terrain et à l'environnement, et elle permet directement une modularité et une polyvalence de par les outils tractés interchangeables. Les grands acteurs industriels de la machinerie agricole se sont ainsi lancés depuis quelques années dans le développement de tracteurs autonomes. Le constructeur John Deere a ainsi mis sur le marché son premier tracteur autonome en 2017 [Alberto-Rodriguez et al., 2020]. Ces tracteurs autonomes se basent sur des modèles déjà existants, sur lesquels sont rajoutés divers équipements permettant la perception (capteurs proprioceptifs et extéroceptifs), ainsi que l'action (ordinateur embarqué pour la décision) [Thanpattranon et al., 2016]. Ces machines

(a) Robot cartésien FarmBot

(b) Robot à câbles Tarzan

Figure 1.1: Différents types de robots agricoles

sont souvent constituées d'une partie motrice, et d'outils remorqués, nécessitant des manœuvres complexes, mais néanmoins exécutables de manière autonome [Wang and Noguchi, 2016]. En revanche, étant initialement adaptées à une utilisation par un humain, elles sont lourdes et de taille importante, afin de pouvoir notamment intégrer un habitacle, De plus, les outils sont de grande taille dans le but de minimiser le nombre de passages du tracteur et ainsi le temps passé à conduire la machine.

La troisième approche se base sur un constat découlant des possibilités d'automatisation. En effet, une approche autonome permettrait de développer des solutions robotiques alternatives, plus petites et plus légères que les machines agricoles existantes aujourd'hui. Un des principaux avantages serait ainsi le développement d'une agriculture impliquant quasiment zéro compaction des sols, entraînant une augmentation de la productivité des sols [Blackmore et al., 2001]. Ainsi, une flotte de robots légers serait une meilleure alternative aux traditionnels tracteurs avec un poids de plusieurs tonnes [Duckett et al., 2018]. De plus, une approche basée autour de la machinerie agricole déjà existante a pour désavantage d'exclure plusieurs types de cultures, telles que celles des arbres fruitiers (les vignes sont incluses dans cette catégorie), et les exploitations de petites tailles [Bergerman et al., 2016]. Une présence humaine physique, c'est-à-dire un · e conducteur · ice, n'étant plus nécessaire, il est possible d'imaginer et de concevoir des robots autonomes adaptés à tous types d'environnements. Ainsi, sont en cours de développement des robots mobiles de différentes tailles. Dans [Vidoni et al., 2017], [Vroegindeweij et al., 2018] et [Wu et al., 2018] sont présentés des robots de petite taille, respectivement à destination d'exploitations avicoles et d'exploitations maraîchères, ainsi que des robots enjambeurs, capables d'opérer sur des planches et des exploitations viticoles. Nous pouvons citer les trois robots développés par Naïo Technologies, présentés figure 1.2. Le robot Oz (1.2a) est un robot destiné à opérer dans l'inter-rang, au sein d'exploitations maraîchères. Le robot Dino (1.2b) est un robot enjambeur qui intervient *au-dessus* des rangs d'exploitations industrielles. Pour finir, le robot Ted (1.2c) est un robot

enjambeur à destination des exploitations viticoles. Il possède une arche haute, qui lui permet de circuler au-dessus des pieds de vigne.

(a) Robot Oz

(b) Robot Dino

(c) Robot Ted

Figure 1.2: Trois robots produits par Naïo Technologies (Crédit Tien TRAN)

La dernière catégorie concerne les véhicules aériens. En effet, le récent développement des drones offre aussi de nouvelles possibilités au sein d'une exploitation agricole [Gonzalez et al., 2018], [Mogili and Deepak, 2018]. En effet, ces derniers sont capables de survoler une exploitation agricole et de fournir via divers capteurs des informations utiles, telles que le stade de développement des cultures, la présence d'éventuelles maladies, ou le bon fonctionnement de l'irrigation [Mogili and Deepak, 2018]. Ainsi, dans [Puig et al., 2015], un drone équipé d'une caméra RGB est utilisé pour surveiller l'état d'une culture attaquée par des larves. Outre leur tâche de surveillance et de prise d'information, les drones peuvent aussi être utilisés de manière active au sein d'une exploitation agricole, notamment pour réaliser l'épandage de pesticide, comme dans [Hoffmann et al., 2008], [Spoorthi S. et al., 2017] ou [Yallappa et al., 2017] (Figure 1.3). Une flotte de drones peut également être considérée [Yao et al., 2016].

Figure 1.3: Drone pulvérisant un pesticide

Par conséquent, [Eberhardt and Vollrath, 2018] montre qu'une agriculture technologiquement développée a un impact positif significatif sur le niveau de développement d'un territoire. Nous pouvons ainsi citer trois impacts significatifs que la robotique et l'automatique pourront amener dans le domaine de l'agriculture [Bergerman et al., 2016] :

- (i) L'agriculture de précision. Il s'agit d'utiliser des capteurs de haute précision afin de savoir quand semer, irriguer, déposer de l'engrais de façon optimale. Ces capteurs peuvent être simples et mesurer des valeurs physiques immédiates (température, hygrométrie), ou bien être plus complexes (captation d'images hyperspectrales, détecteurs chimiques, etc.)
- (ii) La navigation de machine agricole. Il s'agit de contrôler une machine agricole de façon autonome et précise tout en prenant en compte la cinématique de la machine et les contraintes de l'environnement, afin de réaliser des tâches de navigation au sein ou en périphérie des cultures.
- (iii) La récolte et manipulation de fruits et végétaux. Il s'agit de concevoir des machines pouvant effectuer des actions complexes sur des fruits et végétaux fragiles, comme par exemple récolter des grappes de raisins.

Nous pouvons ainsi imaginer une *ferme du futur*, dans laquelle cohabiteront les quatre catégories de solutions robotiques présentées précédemment et qui impacteront significativement le domaine de l'agriculture. Ainsi, dans un futur plus ou moins proche, les exploitations agricoles pourraient être presque totalement autonomes. Dans ces exploitations, des drones se chargeraient de surveiller les cultures et de transmettre des informations à une flotte de robots mobiles au sol, qui seraient responsables d'effectuer des actions d'entretien, de désherbage, et de récolte des cultures. Cette robotisation serait appuyée par une série de capteurs. Le concept d'*Internet of Things* (IoT) pourra ici trouver son utilité [Virk et al., 2020], en permettant notamment de relier à Internet l'ensemble des acteurs précédemment cités. Des informations utiles, telles que les prévisions météo, pourront être utilisées afin de planifier les actions à effectuer, et des informations sur l'état des cultures pourront être remontées afin d'être agrégées et analysées, permettant de mieux prévoir et optimiser le traitement des cultures.

Dans cette thèse, nous étudierons spécifiquement la problématique de la navigation autonome d'un robot mobile terrestre au sein d'une exploitation agricole, qui est une étape importante et nécessaire vers la réalisation d'exploitations autonomes.

1.3 Le projet *Desherb'eur*

Cette thèse s'inscrit dans le cadre du projet européen *Desherb'eur*. Ce projet est porté par l'établissement national des produits de l'agriculture et de la mer, FranceAgriMer,

qui vise notamment à mettre en œuvre des dispositifs de soutiens techniques et financiers aux acteurs des filières agricoles. Deux entités participent à ce projet : l'entreprise Naïo Technologies, et le laboratoire du LAAS-CNRS. Naïo Technologies est une entreprise créée en 2011 qui conçoit et réalise des robots au service des agriculteurs avec un triple objectif :

- Réduire la pénibilité du travail.
- Limiter l'impact environnemental.
- Améliorer la rentabilité des exploitations.

Le défi de Naïo Technologies est de remplacer l'homme sur les tâches les plus longues ou dangereuses pour la santé mais également de l'assister dans des tâches minutieuses tout en œuvrant pour le respect de l'environnement dans le domaine agricole. C'est pourquoi la société développe des outils innovants issus de la robotique. Au regard du contexte et des enjeux présentés ci-avant, le projet ambitionne de concevoir et mettre en œuvre des systèmes robotiques qui s'appuient le plus possible sur les écosystèmes naturels tout en assurant leur pérennité. Le projet vise le développement de robots de désherbage à destination des secteurs du maraîchage en parcours biologiques et des secteurs industriels de l'horticulture et de la viticulture. À ce jour, trois modèles de robots sont développés et vendus par Naïo Technologies : Oz 1.2a, Dino 1.2b et Ted 1.2c. Dans ce cadre, Naïo Technologies a décidé de s'associer au laboratoire LAAS-CNRS avec les objectifs suivants :

- Optimiser les performances de son robot Oz pour renforcer son positionnement sur le maraîchage bio ($< 10ha$), et développer son positionnement sur le marché de l'horticulture.
- Développer une nouvelle gamme de robots (robot enjambeur à destination du maraîchage industriel et de la viticulture) et de nouveaux outils actifs.
- Se doter d'un outil de production industriel permettant de répondre à la demande croissante.

Le tableau 1.4 présente les axes d'innovation recherchés dans le cadre du projet *Desherb'eur*, les verrous actuels, ainsi que les tâches à effectuer pour progresser sur les axes d'innovation. Plusieurs personnes ont travaillé et collaboré dans le cadre de ce projet. Le travail présenté dans cette thèse s'est quant à lui porté sur le troisième verrou.

1.4 La navigation dans un contexte agricole

Parmi les diverses problématiques émergentes de la conception de robots agricoles, celles de la navigation est une des plus étudiées. En effet, un environnement agricole

Axe d'innovation	Verrous	Facteurs clés de succès
Optimisation de la gamme actuelle	Localisation suffisamment précise du robot dans l'exploitation ou dans la parcelle	En milieu extérieur, exploitation d'un récepteur GPS précis Rajout de cibles visuelles géolocalisées dans l'exploitation, comme amers de localisation
	Variabilité des scènes en extérieur (liée à la saison, à l'éclairage naturel...)	Multiplier les bases d'apprentissage/les cartes SLAM, etc... en fonction des saisons et des conditions météorologiques : phase préalable de choix de la meilleure base en fonction du contexte
	Grande probabilité de rencontrer des situations imprévues (type d'obstacles, formes de parcelles...)	Sécurité indépendante du logiciel de guidage principal : algorithmes de détection et de reconnaissance d'obstacles avancées puis implémentation de stratégies d'évitement d'obstacles avec mémorisation des nouvelles configurations
Développement d'une nouvelle gamme de produits et nouveaux outils actifs	Robots et outils légers, mais robustes, adaptés à un usage agricole très contraignant	Utilisation de l'énergie électrique dans le mouvement des outils Localisation des plans très précise et ni les changements de luminosité ni les ombres portées ne doivent interférer avec le guidage
Industrialisation des solutions développées	Equipements de production adaptés aux différentes gammes de produits	Mise en place d'une infrastructure propre : formation du personnel, équipe dédiée à la production, mise en place du système qualité approprié Mise en place d'un système de test permettant de tester un maximum de situation en conditions réelles
	Adaptation du système de mise à jour pour les grandes séries	Création d'une infrastructure matérielle et logicielle pour le monitoring / mise à jour des robots, pouvant supporter la charge, via USB ou via Web

Figure 1.4: Lots de travail du projet *Desherb'eur*

possède des caractéristiques particulières qui devront être prises en compte. Dans le cadre d'une exploitation agricole, l'environnement est composé à la fois des zones de culture : champs et plantations; mais aussi de l'espace utile : routes, chemins et zones de fourrière (espace en bout de champ, non labouré, permettant aux véhicules d'effectuer des manœuvres). Les problématiques sont différentes pour chacune de ces zones. En effet, les zones de culture sont des espaces souvent structurés ou semi-structurés. Dans la plupart des exploitations industrielles, les rangées de culture sont déjà créées et plantées en utilisant un guidage GPS-RTK offrant une précision centimétrique, et les divers paramètres (distance inter-rang, taille de la zone de fourrière, etc.) sont connus au préalable. Par conséquent, il est possible de définir un protocole de navigation très codifié autour de cartes topologiques [Emmi et al., 2019]. Par exemple, la navigation inter-rang peut

simplement être basée sur les traces GPS-RTK, ou bien sur des capteurs extéroceptifs tels que des LiDARs ou des caméras, permettant au robot de suivre précisément les rangées [Durand-Petiteville et al., 2018]. Les demi-tours en fin de rangée peuvent être gérés par un enchaînement de mouvements menant à un *U-turn* (demi-tour en forme de U), Ω-*turn* (demi-tour en forme d'omega), ou *fishtail-turn* (demi-tour en forme de queue de poisson). Ces mouvements peuvent être gérés par *dead-reckoning* [Bayar et al., 2015], [Bochtis and Vougioukas, 2008], ou bien effectués à l'aide de capteurs extéroceptifs [Durand-Petiteville et al., 2017], [Le Flecher et al., 2017], [Le Flecher et al., 2019].

En revanche, si les zones de cultures peuvent être structurées ou semi-structurées, ce n'est pas le cas de l'espace utile avoisinant. Cet espace comprend toutes les zones utiles d'une exploitation agricole (routes, bâtiments, zones de parking, stockage, manœuvres, etc.). Ce type d'espace possède plusieurs caractéristiques significatives. Premièrement, il s'agit de zones dans lesquelles l'activité humaine est importante, entraînant la présence de nombreux obstacles statiques et dynamiques : des employés vont se déplacer et du matériel peut être entreposé à même le sol temporairement. De gros objets peuvent également être présents, avec notamment la circulation et le stationnement de véhicules et de machines agricoles. Il peut aussi y avoir de grands bâtiments, des hangars, des silos, des murs, etc. Il est aussi possible d'envisager la présence d'animaux domestiques (chiens de bergers) ou de bétail qui peuvent circuler plus ou moins librement et qui peuvent ne pas avoir conscience du danger que peut représenter un robot et ne pas s'écarter à son passage. Une deuxième caractéristique spécifique de ces environnements est la nature du sol. En effet, à la différence des milieux urbanisés, où les sols sont goudronnés ou bitumés, le contexte agricole offre aussi des sols d'une nature différente : en fonction de son usage et des conditions météorologiques, le terrain peut être inégal, terreux, boueux ou accidenté, notamment dans les cultures viticoles. En effet, la figure 1.5 illustre deux exemples de terrains particulièrement difficiles sur lequel un robot agricole est susceptible d'évoluer. L'image 1.5a présente un terrain boueux à la suite de pluies soutenues, sur lequel un véhicule risque de glisser ou déraper. L'image 1.5b montre un vignoble situé sur un terrain escarpé. Pour finir, nous devons considérer un espace de travail qui peut être considérablement grand, parfois plus de 200 ha pour certaines exploitations industrielles.

Ainsi, afin de réaliser une tâche typique, le robot devra aller d'une parcelle à une autre, ce qui l'amènera à devoir se déplacer en autonomie sur une grande distance pouvant atteindre plusieurs centaines de mètres, sur un sol difficile (boue, terrain escarpé, crevasses). Il devra lors de cette navigation éviter des obstacles à la fois statiques et dynamiques. Dans cette optique, il est possible de doter le robot de capteurs proprioceptifs, tels que des gyromètres, magnétomètres, accéléromètres ou systèmes inertiels (IMU). Il doit aussi être équipé de capteurs extéroceptifs, tels que

(a) Boue dans un champ
(Crédit Ian Paterson / CC BY-SA 2.0)

(b) Vignes sur un col escarpé
(Crédit Circe Denyer / CC0 Public Domain)

Figure 1.5: Différents types de terrains agricoles

des caméras, des LiDARs. Des systèmes de localisation GNSS peuvent de plus être utilisés. Ces derniers offrent une précision de l'ordre du mètre (GPS / Galileo), ou bien de l'ordre du centimètre grâce à des systèmes GPS-RTK. Ainsi, dans le cas d'un milieu agricole typique, nous pouvons supposer que l'environnement est connu a priori, et qu'une carte de l'exploitation est disponible pour le robot, ainsi qu'un système de localisation absolue GNSS. Si tel est le cas, des algorithmes de planification de chemins peuvent être utilisés afin de générer une trajectoire menant le robot à son objectif, tout en évitant les obstacles connus au préalable et présents sur la carte. Cependant, comme il a été dit précédemment, un environnement agricole est un espace hautement dynamique, et le robot devra faire face à la présence d'obstacles inconnus. Par conséquent, il sera impératif en complément de doter le robot de capacités de navigation et d'évitement référencés capteurs, afin de surveiller, détecter et gérer les situations imprévues. Ces capacités devront aussi appuyer le robot dans le cas possible de la perte ou de la dégradation du signal GNSS permettant sa localisation. Ainsi, dans cette thèse, nous nous proposons de concevoir et d'implémenter un ensemble de méthodes de navigation référencées capteurs.

1.5 État de l'art des méthodes de navigation référencées capteurs

Cette section a pour but de présenter l'état de l'art actuel des méthodes de navigation référencées capteurs. Nous commençons par introduire l'ensemble des méthodes existantes dans le cas des environnements statiques. Nous présenterons ensuite les incréments qui ont été développés ces dernières années pour chacune de ces méthodes afin de pouvoir aussi gérer les environnements dynamiques.

Dès la genèse de la robotique mobile, la problématique de la navigation autonome a été grandement étudiée et de nombreuses approches pouvant permettre une navigation

sans collision dans un environnement statique ont été développées. Ces dernières ont aujourd'hui prouvé leur fiabilité et leur efficacité. Dans le cas d'environnements connus a priori pour lesquels le robot possède une carte, la problématique est celle de la planification de trajectoires, pour laquelle de nombreuses méthodes ont été proposées [Gonzalez Bautista et al., 2015]. Ces méthodes ne sont utilisées que pour générer une trajectoire menant le robot à son objectif tout en évitant les obstacles connus au préalable. Cependant, lorsqu'aucune carte n'est disponible, que cette dernière est incomplète, ou bien qu'il n'est pas possible pour le robot de se localiser dans cette carte, ce dernier doit se reposer sur des capteurs extéroceptifs. Une navigation se basant uniquement sur les informations renvoyées par les capteurs est appelée navigation référencée capteurs. Parmi ces capteurs extéroceptifs peuvent se trouver divers dispositifs :

- Des dispositifs optiques : le robot peut être équipé d'une ou plusieurs caméras monoculaires. Ces caméras peuvent être installées de façon à former un système de caméras stéréo, donnant accès à l'information de profondeur. Il existe de plus des dispositifs permettant d'obtenir directement une information de profondeur en plus de l'image (caméra RGB-D), les plus connus étant les caméras Realsense d'Intel [Keselman et al., 2017] ou bien les Kinect [Zhang, 2012].

- Des dispositifs laser : ce sont les systèmes de télédétection communément appelés LiDARs (*Light Detection And Ranging*, "Détection et Estimation de Distance par la Lumière"). Ces LiDARs permettent d'obtenir un nuage de points très dense autour du robot, avec une précision de distance en dessous du centimètre. Ces derniers peuvent fournir une information sur un seul plan (LiDARs monoplans), ou bien sur plusieurs plans inclinés de quelques degrés les uns avec les autres (LiDARs multiplans).

- Des dispositifs ultrasons : ces capteurs sont les plus anciens, et consistaient à émettre des ondes ultrasons afin de déterminer une distance. Ces capteurs sont aujourd'hui obsolètes car peu précis et propices aux perturbations.

Dans la plupart des méthodes présentées par la suite, le capteur le plus couramment utilisé est un LiDAR, car il s'agit du dispositif le plus précis et fiable pour obtenir des informations de l'environnement. Pour autant, il peut être généralement possible d'adapter ces méthodes pour qu'elles fonctionnent avec d'autres types de capteurs extéroceptifs, du moment qu'ils fournissent l'information attendue.

1.5.1 Méthodes basées sur les champs de potentiels

Les techniques les plus largement répandues pour effectuer de l'évitement d'obstacles sont basées sur la méthode des champs de potentiel artificiel (APF pour l'anglais *Artificial Potential Fields*), initialement développée par [Khatib, 1985]. Avec cette méthode, les obstacles et l'objectif sont pris en compte pour générer des forces répulsives et

attractives. Le robot est ensuite conduit à travers l'environnement, soumis à ces forces. La figure 1.6, présentée dans [Ahmad et al., 2019], schématise le champ de potentiel dans le cas d'un simple obstacle. Par conséquent, un avantage direct de cette technique est la réactivité. Les champs de potentiels peuvent être générés et modifiés en ligne, sur la base des nouvelles connaissances de l'environnement que le robot est capable d'acquérir pendant la navigation. Cependant, l'article [Koren and Borenstein, 1991] aborde plusieurs limitations inhérentes aux méthodes potentielles, telles que les minima locaux, les objectifs inaccessibles avec des obstacles à proximité (GNRON, Goal Non-Reachable with Obstacle Nearby) et les comportements oscillatoires possibles dans les passages étroits. Diverses techniques ont été mises en application dans le but de résoudre ces problèmes. Afin de prévenir l'apparition de minima locaux, une première approche mise en place par [Borenstein and Koren, 1989] consiste à détecter les moments où le robot se retrouve piégé, puis à alors suivre des procédures d'évitement différents, telles que le suivi de mur. Cette approche a aussi été implémentée par [Mabrouk and McInnes, 2008a]. Par la suite, d'autres solutions ont été proposées. L'article [Mabrouk and McInnes, 2008b] introduit des états dynamiques instables. L'article [Zhu et al., 2006] utilise une méthode de *Simulated Annealing* (recuit simulé), qui autorise le robot à remonter les champs de potentiel afin de sortir d'un minimum local. L'article [Guerra et al., 2016] introduit des petites perturbations dans la commande finale, ou bien [Min Gyu Park and Min Cheol Lee, 2004] ajoute des *collines virtuelles* afin d'échapper aux minima locaux.

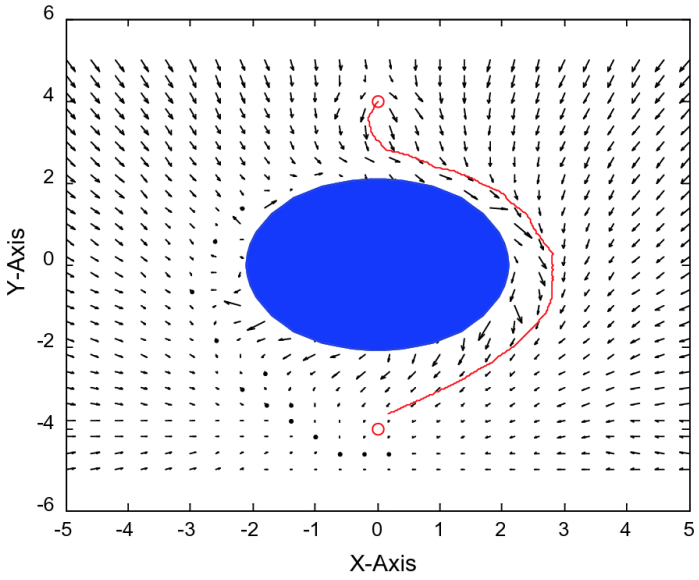


Figure 1.6: Illustration de la méthode APF

Par la suite, cette méthode a été adaptée pour traiter le cas des environnements dynamiques. L'approche commune consiste à concevoir un nouveau potentiel répulsif en tenant compte du contexte dynamique du problème (obstacles et vitesse du robot).

L'article [Ge and Cui, 2002] commence par mesurer la vitesse relative de la cible et des obstacles et conçoit une force répulsive virtuelle afin de prendre en compte les obstacles en mouvement : une fonction d'attraction est définie en fonction de l'amplitude de la vitesse relative entre la cible et le robot à l'instant t. Cependant, il est supposé qu'à tout moment le robot n'est influencé que par un seul obstacle le plus proche, et aucune conclusion n'est donnée sur l'efficacité de cette méthode pour les environnements encombrés. En outre, il ne traite pas des inconvénients communs des méthodes APF tels que les puits de potentiel. En reprenant la même idée que la méthode citée dans l'article [Ge and Cui, 2002], [Lu et al., 2009] ajoute à la fonction d'attraction l'ampleur de l'accélération relative de la cible et des obstacles, et prend également en compte l'accélération des obstacles pour le calcul de la force de répulsion. Cependant, une hypothèse forte est définie selon laquelle les obstacles sont des polygones convexes dont les formes, les positions, les vitesses et les accélérations peuvent être mesurées ou calculées en ligne avec précision. Le problème des minima locaux causés par les environnements dynamiques (obstacles et but) est également pris en compte. Si le robot est piégé dans un minimum local, ce dernier attend que le but ou les obstacles modifient leur mouvement. Sinon, l'environnement est considéré comme statique et des techniques conventionnelles pour éviter les minima locaux sont utilisées. L'article [Zhang et al., 2013] présente une méthode plus complexe pour concevoir des forces de répulsion, en tenant compte de la vitesse et de la direction des obstacles. En particulier, cette méthode permet d'éviter les mouvements d'évitement inutiles lorsqu'il n'y a pas de risque de collision de par la vitesse du robot et des obstacles. Dans [Montiel et al., 2015], un algorithme d'évolution bactérienne (BEA) est utilisé afin de moduler les deux variables scalaires qui représentent les gains proportionnels attractifs et répulsifs des fonctions d'attraction et de répulsion. Un premier algorithme BEA est exécuté au début afin de planifier la trajectoire, et exécuté à nouveau chaque fois qu'un changement d'environnement est détecté. Ainsi, il conduit à une technique flexible de planification de la trajectoire, mais il ne tient pas compte de la dynamique des obstacles elle-même (*c.à.d.* vitesse et direction). Dans [Qixin et al., 2006], une solution est proposée afin de gérer le cas où à la fois l'objectif est les obstacles sont mobiles : un coefficient relatif de danger est calculé pour chaque obstacle et sert à calculer une nouvelle force. Ces méthodes nécessitent toutes un regroupement précis des obstacles et une connaissance précise de leurs dynamiques.

1.5.2 Méthodes basées sur les histogrammes (VFH)

La deuxième classe de méthodes est basée sur des histogrammes de champ de vecteurs (VFH pour l'anglais *Vector Field Histogram*). La méthode VFH a été initialement proposée par [Borenstein and Koren, 1991] et fonctionne en utilisant une représentation statistique de l'environnement du robot via une grille d'histogrammes. Dans un premier temps, une grille d'histogramme cartésien est construite à l'aide des données des capteurs. Ensuite, un histogramme polaire est calculé à partir de

l'histogramme cartésien. Enfin, les secteurs libres candidats sont déterminés à partir de l'histogramme polaire. Le robot est ensuite conduit à travers le secteur choisi. Par conséquent, l'algorithme peut s'adapter au mouvement d'évitement en fonction de la taille du secteur : le robot se déplace au milieu des obstacles dans les passages étroits, et à proximité des obstacles dans les passages larges, ce qui augmente la sécurité en environnement encombré. Il s'agit d'une méthode conçue pour être insensible aux erreurs de lecture qui peuvent apparaître en raison de l'utilisation de capteurs ultrasons peu précis. Elle offre également un mouvement sans oscillation. Cette méthode a connu plusieurs améliorations. La méthode VFH+ a été proposée par [Ulrich and Borenstein, 1998] afin de prendre en compte la dimension du robot et sa vitesse. La méthode VFH* a été introduite par [Ulrich and Borenstein, 2000] et implémente un processus de prédiction pour déterminer le meilleur choix entre plusieurs possibilités de mouvement d'évitement.

Les méthodes basées sur les histogrammes ont également été étudiées pour le cas des environnements dynamiques. La méthode MSV (Mobile and Static Vector field histogram method) a été développée par [You et al., 2008] pour surmonter le problème des obstacles mobiles. La méthode MSV est basée sur la méthode VFH+ et utilise une stratégie spéciale pour gérer les obstacles en mouvement. Une liste de huit mouvements possibles des obstacles mobiles est d'abord établie (en fonction des directions et distances des obstacles mobiles par rapport au robot). Si un obstacle dynamique est détecté, le robot vérifie d'abord si le mouvement de cet obstacle est dangereux et réagit en conséquence en se dirigeant vers un secteur sans obstacle permettant l'évitement. De plus, [Zhu et al., 2012] propose également une amélioration de la méthode VFH. Il s'agit de trouver d'abord tous les secteurs sûrs et prendre leurs directions intermédiaires comme candidats pour la direction finale du mouvement. Ensuite, la menace que représentent les obstacles en mouvement est évaluée pour chaque direction candidate, en utilisant une valeur de menace qui dépend des obstacles et de la dynamique du robot. Une vitesse maximale correspondante est calculée pour chaque direction candidate. Avec cette information, l'algorithme sélectionne la direction finale. Plus récemment, [Babinec et al., 2014] a développé une nouvelle méthode appelée VFH*TDT (VFH* with Time Dependant Tree) qui effectue la projection des obstacles en mouvement directement sur la grille de l'histogramme à un moment futur. Une méthode VFH* classique est alors utilisée, avec prise en compte directe de la dynamique des obstacles à l'intérieur de l'arbre prédictif. Il permet à l'algorithme d'anticiper la trajectoire des obstacles en mouvement et de sélectionner le mouvement le plus sûr pour éviter la collision. Pour finir, la méthode IVFH* développée par [Jie et al., 2010] propose de pondérer l'histogramme afin de prendre en compte les obstacles dynamiques, en déformant la roadmap à l'aide de force attractives et répulsives basées sur les mouvements des obstacles.

1.5.3 Méthodes basées sur la vitesse et l'accélération des obstacles (VO et AO) et approche par fenêtre dynamique (DWA)

Le concept de VO (Velocity Obstacles) [Fiorini and Shiller, 1998] consiste à déterminer l'ensemble des vitesses du robot qui amèneront à une collision dans un temps futur. Une manœuvre d'évitement peut être générée en sélectionnant des vitesses en dehors de l'ensemble des *Velocity Obstacles* en optimisant dynamiquement des paramètres tels que le temps pour atteindre l'objectif ou la distance aux obstacles. Cette méthode fonctionne pour des obstacles se déplaçant à vitesse constante. L'avantage de cette approche est notamment de pouvoir prendre en compte les dynamiques du robot ainsi que les contraintes d'accélération des moteurs. Cette technique a notamment été mise en œuvre sur un fauteuil roulant autonome par [Prassler et al., 2001] et s'est avérée robuste et efficace lors d'expériences réelles. Dans [Berg et al., 2011], les accélérations des obstacles sont prises en compte pour générer des *Acceleration-Velocity Obstacles* (AVO). De plus, [Shiller et al., 2001], [Shiller et al., 2010] et [Large et al., 2005] étend le concept de VO aux obstacles se déplaçant le long d'une trajectoire générale, créant des *Non-Linear Velocity Obstacles* (NLVO). Néanmoins, cette méthode nécessite une expression analytique des trajectoires. Il est à noter que même si la trajectoire de l'obstacle est inconnue, la connaissance de sa vitesse actuelle et de la courbure de sa trajectoire est suffisante pour faire une approximation du premier ordre de la trajectoire et obtenir une expression analytique.

L'approche par fenêtre dynamique (DWA, Dynamic Window Approach) a été initialement développée par [Fox et al., 1997]. L'idée principale de cette approche est de rechercher parmi des couples (v, ω) de vitesses linéaire et angulaire une solution optimale. Tout d'abord, cette méthode considère tous les couples de vitesse possibles (v, ω) qui peuvent être appliqués pendant dans un cycle d'horloge, appelé fenêtre dynamique. Parmi tous ces couples, ceux non admissibles, c'est-à-dire conduisant à une collision ou à une situation dangereuse, sont éliminés. Ensuite, la fenêtre dynamique élimine les vitesses qui ne peuvent pas être atteintes à l'issue d'un court intervalle de temps en prenant en compte les limites d'accélération du robot. Finalement, la commande de vitesse la plus appropriée est choisie en maximisant une fonction de coût prenant en considération, pour une commande (v, ω) appliquée, le cap du robot, sa distance à l'obstacle le plus proche, et sa vitesse. La figure 1.7, tirée de [Ahmad et al., 2019], schématise le fonctionnement de l'algorithme DWA. Des améliorations ont été apportées pour gérer une dynamique d'obstacles plus élevée. L'approche GDW [Brock and Khatib, 1999] est une généralisation de la méthode DWA combinant des méthodes issues de la planification de trajectoire ainsi que de l'évitement d'obstacles. Elle permet l'application de vitesses importantes dans un environnement dynamique inconnu.

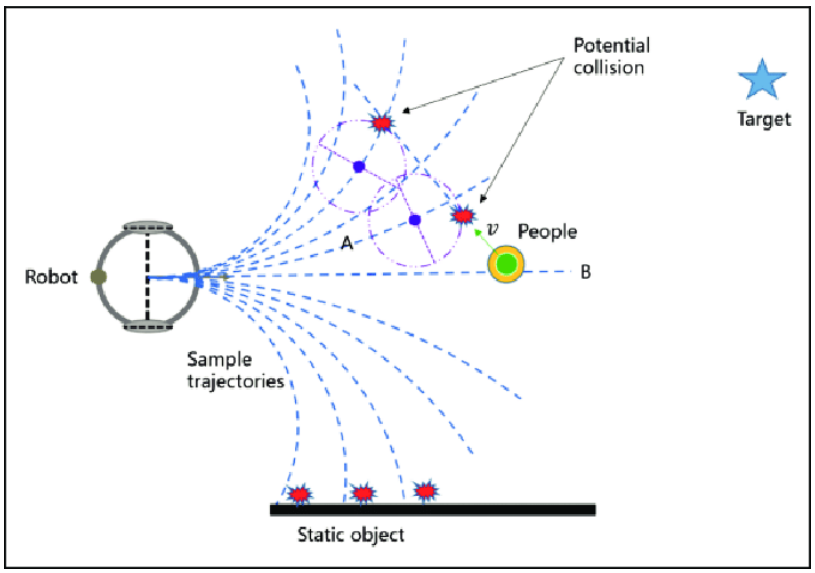


Figure 1.7: Illustration de la méthode DWA

Proche des concepts de VO et de DWA, le concept d'ICS (Inevitable Collision State) introduit par [Fraichard and Asama, 2003], a permis de développer de nouvelles méthodes d'évitement. C'est notamment le cas pour [Hernández-Aceituno et al., 2015], qui utilise une vérification d'état probabiliste afin de générer une trajectoire qui minimise la probabilité de collision, en appliquant un modèle bayésien pour modéliser la trajectoire des obstacles.

1.5.4 Méthodes dites *Follow the Gap*

La méthode *Follow the Gap* a été présenté dans [Sezer and Gokasan, 2012]. Cette méthode consiste à diriger le robot vers le centre de l'espace maximum entre deux obstacles. Un incrément de la méthode a été présenté dans [Zohaib et al., 2014b], ayant pour particularité de pouvoir gérer des obstacles en forme de U et de H.

1.5.5 Méthodes dites *Bug*

Les méthodes *Bugs* combinent une stratégie d'évitement activée à proximité immédiate de chaque obstacle avec un mouvement vers la cible en ligne droite lorsqu'il n'y a aucun risque de collision [Kudriashov et al., 2020]. La trajectoire précise que suit le robot lors des phases d'évitement dépend de l'algorithme. Nous allons ainsi présenter les principaux *Bug Algorithms*. Dans l'algorithme **Bug-1**, le robot tourne complètement autour de l'obstacle, à équidistance, afin de trouver le point ayant la distance minimale au but. Ensuite, il continue à suivre le bord jusqu'à ce qu'il atteigne à nouveau ce point, et commence à se déplacer vers le but. Dans l'algorithme **Bug-2**, le robot contourne l'obstacle jusqu'à ce que la pente de la ligne entre sa position actuelle et la

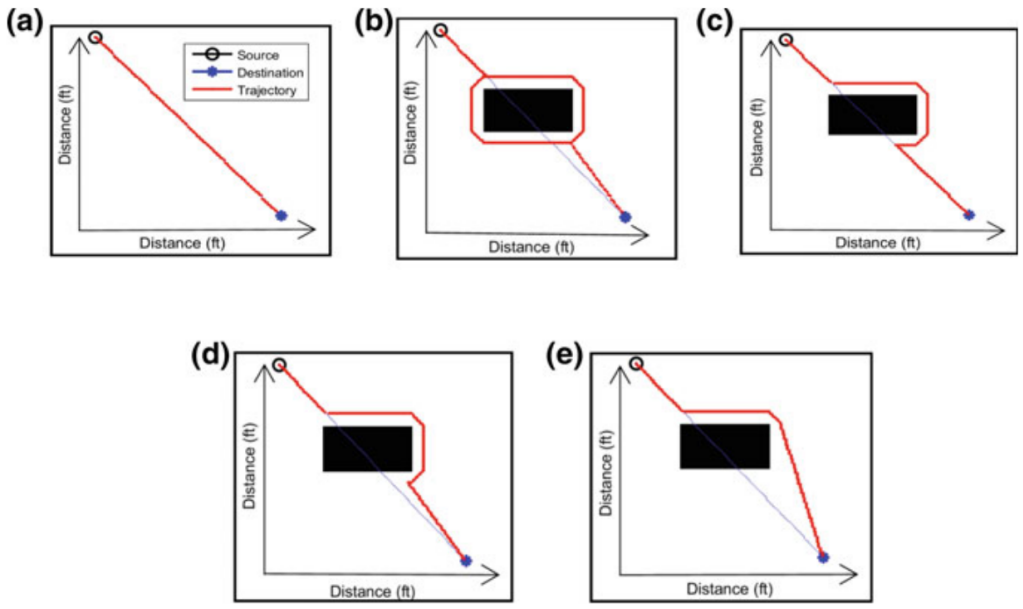


Figure 1.8: Illustration de quelques *Bug Algorithms* **(a)** Pas d'obstacle - **(b)** Bug-1 - **(c)** Bug-2 - **(c)** Dist-bug - **(e)** IBA

destination devienne la même que la pente initiale du chemin de référence vers la destination. Dans l'algorithme **Dist-bug** [Kamon and Rivlin, 1997], le robot continue de suivre le bord jusqu'à ce que la distance entre le point courant et le but diminue. L'algorithme **TangentBug** [Kamon et al., 1998] est un peu plus complexe car il consiste à calculer le *Local Tangent Graph*. Au sein d'un environnement localement connu et constitué d'obstacles polygonaux, le *Local Tangent Graph* est un graphe local constitué des segments liant deux à deux l'ensemble des sommets des polygones détectés dans l'environnement. Le robot utilise ensuite ce *Local Tangent Graph* afin de choisir la direction localement optimale pour éviter les obstacles, tout en se dirigeant vers l'objectif indiqué. Cette technique a été utilisée dans [Mohamed et al., 2011], où un algorithme **TangentBug** est implémenté afin de pallier le risque de minima locaux dans le cadre de l'utilisation d'une méthode basée sur les champs de potentiel artificiels. Cet algorithme a aussi été utilisé dans [Al-Haddad et al., 2011] afin de contrôler un fauteuil roulant automatiquement. Pour finir, l'algorithme **IBA** (Intelligent Bug Algorithm) [Zohaib et al., 2014a] propose une amélioration de la méthode **Dist-bug**, en prenant en considération la position de l'objectif pendant la navigation. La figure 1.8, tirée de [Kudriashov et al., 2020], montre les trajectoires d'évitement lors de l'utilisation des algorithmes **Bug-1**, **Bug-2**, **Dist-bug** et **IBA**. Une comparaison des performances des *Bug Algorithms* est dressée dans [Yufka and Parlaktuna, 2009]. Ces algorithmes ont été implémentés avec succès sur divers types de robot [Fraundorfer et al., 2012]. De par leur fonctionnement, ils garantissent que le robot sera en mesure de rejoindre n'importe quel point atteignable de l'espace. Néanmoins,

l'absence d'anticipation ou de mémoire de l'environnement peut entraîner une trajectoire finale non optimale, notamment dans des espaces encombrés et/ou labyrinthiques.

1.5.6 Évitement d'obstacles par suivi d'ellipse

Les articles [Adouane, 2009], [Adouane et al., 2011], [Vilca Ventura et al., 2012] et [Vilca et al., 2013] proposent un ensemble de méthodes d'évitement et de navigation dans des environnements encombrés basées sur un choix de primitives géométriques elliptiques. Ces ellipses sont déterminées à partir des données acquises de l'environnement et permettent de générer des trajectoires d'évitement. Plus précisément, [Adouane et al., 2011] introduit le concept d'*ellipse d'influence* (cf. figure 1.9). Dans cette approche, les obstacles sont modélisés par des ellipses définies à l'aide de cinq paramètres indiquant notamment les coordonnées de son centre, son demi-grand axe, son demi-petit axe et son orientation. Il sont déterminés à partir de points LiDAR à l'aide d'un algorithme des moindres carrés [Adouane et al., 2011], ou bien d'un filtre de Kalman. Il est alors possible pour chaque obstacle de définir une *ellipse d'influence* en élargissant l'ellipse le modélisant afin de prendre en compte le rayon du robot ainsi qu'une marge arbitraire. L'utilisation d'ellipses offre notamment l'avantage de pouvoir englober des obstacles de forme spéciale, tels que des murs.

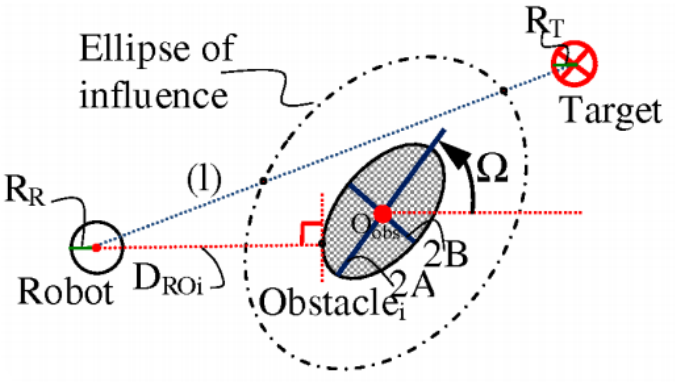


Figure 1.9: Illustration d'une ellipse d'influence

La stratégie de navigation se base alors sur deux contrôleurs. Le premier est chargé de mener le robot vers son objectif, tandis que le second est activé à proximité d'un obstacle dans le but d'effectuer une manœuvre d'évitement. Ce premier contrôleur produit une commande (v, ω) permettant de mener le robot à une position donnée dans un repère absolu. Il se base sur la minimisation des erreurs entre la position du robot et son objectif. Une analyse de stabilité montre que le système rebouclé avec ce contrôleur est asymptotiquement stable. Dans [Adouane et al., 2011], l'auteur propose un contrôleur d'évitement basé sur le concept de cycles limites. En utilisant un couple d'équations différentielles décrivant des cycles limites elliptiques, l'auteur est

capable de générer un contrôleur fonctionnant à vitesse linéaire v constante, capable de contrôler un robot afin de lui faire rejoindre et suivre une trajectoire elliptique donnée. L'asservissement ainsi obtenu est globalement asymptotiquement stable. Afin de savoir quel contrôleur appliquer, [Adouane et al., 2011] implémente la condition d'activation suivante : le contrôleur d'évitement d'obstacle est activé si au moins un obstacle contraint le robot, c'est-à-dire si la ligne (l) joignant le robot à son but coupe une ellipse d'influence, comme le montre la figure 1.9 issue de [Adouane et al., 2011]. Lorsque l'évitement est activé, un repère fixe est attaché à l'obstacle à éviter. La position du robot par rapport à ce repère fixe permet de déterminer le comportement du robot vis-à-vis de l'obstacle (attraction ou répulsion) et le sens de contournement (sens horaire ou sens anti-horaire). Ce choix est maintenu durant toute la durée de l'évitement. Le principe utilisé pour déterminer le sens de contournement consiste simplement à regarder si la plus grande partie de l'obstacle (*i.e.* le centre de l'ellipse) se trouve à droite ou à gauche de la droite reliant le robot à son but. À l'aide du contrôleur d'évitement, le robot rejoint et suit l'ellipse d'influence jusqu'à avoir évité l'obstacle. Le choix des ellipses par rapport aux cercles a pour but de générer des trajectoires de contournement épousant de façon précise des obstacles de formes diverses tels que des murs, comme le montre la figure 1.10 tirée de [Adouane et al., 2011]. Le fait de maintenir un sens de contournement fixe et d'évoluer à vitesse linéaire non nulle assure que le robot ne restera pas bloqué dans un mimimum local.

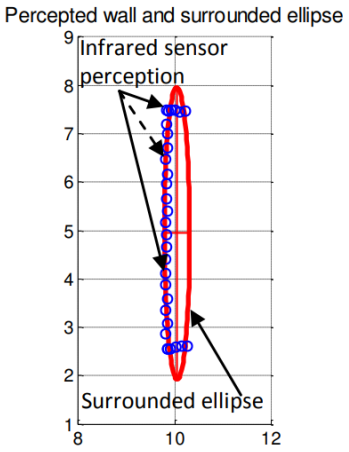


Figure 1.10: Interpolation d'un mur par une ellipse

Cependant, cette méthode n'est pas référencée capteurs au sens où nous l'avons défini. En effet, une fois l'ellipse de référence définie, la méthode consiste à suivre une trajectoire dans le repère lié à l'obstacle. Cela implique deux choses :

- Le suivi de l'ellipse de référence implique de suivre une trajectoire définie par rapport à un repère fixe lié à l'obstacle. Le suivi de cette ellipse nécessite donc une localisation locale précise par rapport à ce repère. Dans un contexte agricole, une telle localisation peut ne pas être disponible, ou bien se révèler imprécise.

- La définition préalable de l'ellipse de référence est très importante. Cependant, cette dernière se fait seulement à partir d'une connaissance locale de l'obstacle. Cette connaissance locale peut être insuffisante pour générer une ellipse anticipant l'entièreté de l'obstacle, y compris les endroits non visibles par le robot au début de l'évitement. Ainsi, l'ellipse générée peut entraîner une trajectoire rentrant en collision avec une partie de l'obstacle.

1.6 Approche proposée dans ces travaux

La section précédente a exposé les méthodes prédominantes dans le domaine de l'évitement d'obstacles et de la navigation dans des environnements statiques et dynamiques. Les avantages et inconvénients de chaque méthode ont été dressés et des comparaisons plus approfondies peuvent être trouvées dans les articles [Kamil and N, 2015], [M.G and Salgoankar, 2017] et [Hoy et al., 2015]. L'approche proposée pour l'évitement d'obstacles en milieu agricole repose sur l'analyse suivante des méthodes listées dans l'état de l'art :

1. Premièrement, les méthodes basées sur les champs de potentiel souffrent notamment du risque de minima locaux. Dans le cadre d'un environnement agricole fortement encombré avec de nombreux obstacles à éviter, le robot peut se trouver souvent dans cette situation et la navigation est alors dans une impasse. De plus lorsque les obstacles sont dynamiques, les solutions proposées basées sur les champs de potentiel nécessitent une détection et un tracking des obstacles mobiles. Or, l'approche a la capacité de ne prendre en compte qu'un nombre réduit d'obstacle mobiles simultanément. Enfin, la distance de sécurité entre le robot et les obstacles n'est pas garantie par les formulations des forces répulsives des obstacles mobiles présentées dans plusieurs articles.

2. Deuxièmement, l'intérêt principal des méthodes DWA et VO réside dans la prise en compte de la dynamique du robot. Or, dans un contexte agricole, avec des sols variés, glissants et dont l'aspect peut changer en fonction du temps, la modélisation de la dynamique du robot est une tâche très complexe. Il est difficile d'assurer qu'en envoyant une commande de vitesse au robot, celui-ci parvienne à l'appliquer réellement sans déraper ou glisser.

3. Troisièmement, les méthodes basées histogrammes (VFH et dérivés) utilisent la localisation pour générer les histogrammes cumulatifs de l'environnement en surmontant la faible précision des capteurs. À l'inverse, nous considérons dans nos cas spécifiques d'utilisation que notre robot est équipé de capteurs extéroceptifs de haute précision, mais avec une localisation éventuellement dégradée (ou inexistante). Ces méthodes ne sont donc pas adaptées à notre cas.

4. Enfin, pour les méthodes consistant à suivre une ellipse de référence autour de l'obstacle [Adouane, 2009], [Adouane et al., 2011], [Vilca Ventura et al., 2012] et [Vilca et al., 2013], il est à noter que cette ellipse reste fixe, une fois définie, et n'est plus remise à jour. Ces techniques supposent donc que l'obstacle puisse être détecté complètement et qu'il soit statique. De plus, le suivi de la trajectoire par rapport à un repère fixe lié à l'obstacle nécessite une localisation du robot relative à ce repère. La difficulté de la méthode relève donc de la disponibilité d'une localisation précise, notamment pour garantir une distance de sécurité robot-obstacle suffisante et éviter la non collision.

Ainsi, l'ensemble des méthodes précédemment citées fonctionnent sur la base d'hypothèses plus ou moins fortes concernant le robot et son environnement. Ces hypothèses peuvent par exemple reposer sur un nombre réduit d'obstacles, ou bien une forme, une géométrie ou une dynamique spécifique des obstacles (par exemple, des obstacles peu nombreux, convexes avec une vitesse linéaire constante dans le temps). Elles peuvent aussi reposer sur une connaissance précise de la dynamique et de la cinématique du robot. Or, dans un contexte agricole avec des obstacles nombreux, volumineux et mobiles, avec des sols souvent glissants ou escarpés, de telles hypothèses peuvent difficilement être faites. Ainsi, un environnement encombré avec de nombreux obstacles dynamiques implique donc des méthodes locales et réactives, mais aussi très génériques.

Dans cette thèse, nous proposons un ensemble de stratégies de navigation réactive référencées capteurs en environnement statique et dynamique, visant à être le plus génériques possible. Ces méthodes seront utilisées en se basant sur les données de divers capteurs extéroceptifs, qui pourront être des caméras RGB-D (exemple : des caméras intel REALSENSE, des caméras SONY) ou des capteurs laser (exemple : Hokuyo URG-04LX). Si les techniques d'évitement d'obstacles telles que celles basées sur les champs de potentiels [Khatib, 1985], [Khatib and Chatila, 1995], ou sur les histogrammes de champs de vecteurs (VFH) [Borenstein and Koren, 1991], [Ulrich and Borenstein, 1998], [Ulrich and Borenstein, 2000] sont connues pour être sensibles aux minima locaux, les approches décrites dans [Adouane, 2009] et [Adouane et al., 2011] ne souffrent pas de ce problème. Cependant, elles ne permettent pas de contrôler précisément la distance robot-obstacle et nécessitent une localisation précise du robot par rapport à l'obstacle. Dans cette thèse, nous proposons une solution référencée capteur adaptable aussi bien aux environnements statiques que dynamiques, peu semsible aux minima locaux et ne nécessitant ni localisation précise par rapport aux obstacles, ni la perception totale de ces derniers. Elle repose sur la définition de spirales de références et leur adaptation au cours de l'évitement.

En effet, le concept de spirale a été initialement formalisé par [Boyadzhiev, 1999]. Dans cet article, les spirales sont définies mathématiquement et il est montré que ces dernières peuvent être paramétrées afin d'offrir un panel de trajectoires très divers.

Ainsi, des techniques de navigation et d'évitement d'obstacles basées sur le suivi de spirales ont été développées. En s'appuyant sur le modèle de la spirale présenté dans [Boyadzhiev, 1999], des techniques d'évitement d'obstacles pour drones ont été proposées par [Mcfadyen et al., 2012b], [Mcfadyen et al., 2012a], [Mcfadyen et al., 2013] et [Mcfadyen et al., 2014]. Dans ce contexte aérien, des spirales coniques sont utilisées pour permettre à un drone de contourner des obstacles. Les spirales ont aussi été utilisées dans [Futterlieb et al., 2014], [Futterlieb, 2017] et [Flecher et al., 2017] afin de doter un robot mobile terrestre de capacité de navigation, de demi-tours et d'évitement. Dans ces ouvrages, le choix de la spirale est principalement motivé par sa généricité, sa versatilité et la possibilité d'appliquer la même méthode avec plusieurs types de capteurs (LiDARs, caméras). Dans cette thèse, notre objectif est donc d'étendre ces travaux effectués sur les spirales, et de nous concentrer sur leur aspect générique et paramétrable afin de développer des méthodes de navigation référencées capteurs en environnement statique et dynamique, pouvant fonctionner dans de nombreux cas et situations différentes.

1.7 Plan du manuscrit

Le manuscrit est organisé selon le plan exposé ci-dessous.

Le deuxième chapitre de cette thèse sera consacré à la présentation de notre technique de navigation en environnement statique. Dans ce chapitre, nous introduirons le concept de spirales et la façon dont ces dernières peuvent être modélisées mathématiquement, ainsi que les divers paramètres permettant de générer des spirales adaptées à l'évitement. Notre première contribution portera donc sur la manière de choisir ces paramètres afin de mener aux trajectoires les plus pertinentes. Ensuite, notre seconde contribution consistera en le développement d'une succession de lois de commandes référencées capteurs permettant à un robot de rejoindre et de suivre des spirales données. Enfin, notre troisième contribution sera centrée sur l'ensemble de la méthode de navigation permettant au robot de naviguer en autonomie dans un environnement statique. Des simulations montreront la pertinence et l'intérêt de notre méthode.

Le troisième chapitre étendra notre méthode d'évitement en spirale aux cas dynamiques. Deux de nos contributions y seront présentées : la première méthode sera appelée *Distance-Angle Adaptatifs*, et consistera à prendre en compte les obstacles dynamiques en observant l'évolution du point le plus proche O_c au cours du temps. Cette évolution nous permettra de moduler certains paramètres de nos spirales, offrant au robot la capacité d'anticiper et d'éviter des obstacles mobiles. Dans une seconde partie, nous présenterons le nouveau concept d'*Enhanced Laser Scan* (ELS), qui correspond à une acquisition de l'environnement, augmentée grâce à l'ajout de nouveaux points artificiels pour prendre en compte la dynamique des

obstacles. Cet *Enhanced Laser Scan* peut alors être utilisé en donnée d'entrée de notre méthode de navigation présentée au chapitre 2, résultant en une méthode de navigation pouvant gérer plusieurs obstacles statiques et dynamiques simultanément.

Enfin, le dernier chapitre sera consacré aux démonstrations et expérimentations. La première partie de ce chapitre s'intéressera, dans un contexte statique, à la problématique de l'implémentation de notre méthode sous l'architecture ROS/Gazebo et à la comparaison avec deux méthodes de navigation disponibles sous ROS : la méthode Dynamic Window Approach (DWA) et la méthode Time-Elastic-Band (TBE). Les deux parties suivantes exposeront des expérimentations effectuées sur la plateforme 4MOB, qui mettront en exergue la pertinence de nos contributions et des méthodes développées dans des conditions expérimentales réelles.

Chapitre 2

Navigation basée capteurs en environnement statique

Ce chapitre présente la problématique du développement de méthodes de navigation référencées capteurs, permettant à un robot mobile de se déplacer au sein d'une exploitation agricole, dans une environnement statique inconnu. Cette situation représente le cas typique dans lequel le robot doit naviguer de son lieu de charge à une parcelle, tout en évitant divers obstacles statiques, prévus ou non, tels que des machines agricoles en stationnement. Comme expliqué en introduction, l'approche d'évitement d'obstacles proposée est référencée capteurs et repose sur le concept de spirale introduit par [Boyadzhiev, 1999]. En effet, lorsqu'un humain se déplace dans un environnement quelconque et cherche à éviter un obstacle fixe, il effectue une trajectoire quasi-circulaire autour de l'obstacle en se maintenant à une distance à peu près constante de celui-ci, et cela jusqu'à ce que l'obstacle soit dépassé. Ainsi, les trajectoires de type spirale sont retenues afin de modéliser un évitement curviligne. Suivant le type de spirale, il est possible de modéliser un large panel de trajectoires excentriques ou concentriques tournant autour d'un point central. En attachant ce point à l'obstacle à éviter, les trajectoires d'évitement générées permettent de contrôler à chaque instant la distance entre le robot et l'obstacle, tout en assurant que le robot reste orienté de la même manière par rapport à l'obstacle, et que ce dernier reste donc visible en permanence pendant la phase d'évitement.

Dans la littérature, cette approche d'évitement en spirale a été implémentée dans le cas d'un drone équipé d'une caméra sphérique [Mcfadyen et al., 2012a] et [Mcfadyen et al., 2012b]. Le drone décrit alors autour du barycentre de l'obstacle une trajectoire d'évitement de type spirale conique. Dans l'article [Futterlieb, 2017], les spirales sont utilisées afin d'offrir une capacité d'évitement à un robot mobile dans le cadre d'une navigation dans un environnement aéroportuaire : un point fixe est placé sur un obstacle, et ce dernier est suivi afin de permettre au robot d'effectuer une spirale autour de ce point. Cependant, ces travaux se restreignent aux cas d'obstacles

convexes de petites tailles.

Dans ce chapitre, nous allons présenter la manière dont les spirales peuvent être utilisées efficacement pour éviter désormais divers types d'obstacles, peu importe leur taille, ou bien leur forme (convexe ou concave). Nous allons notamment présenter une nouvelle approche d'évitement en spirale, basée sur un choix dynamique des centres des spirales, ainsi qu'un ensemble de lois de commande capable d'asservir le robot à une spirale donnée.

Le chapitre est organisé comme-suit. La section 2.1 décrit la modélisation des spirales et leur utilisation dans le cadre d'un évitement d'obstacle statique par un robot mobile. La section 2.2 présente les lois de commande appliquées au robot mobile pour rejoindre et suivre une spirale donnée. La section 2.3 décrit la stratégie de navigation du robot mobile dans un contexte d'environnement statique pour atteindre une position finale en partant d'une position initiale, tout en évitant les obstacles présents sur le chemin.

2.1 Modélisation des spirales et du robot

D'un point de vue mathématique, une spirale est décrite comme une courbe centrée en un point central et s'en éloignant de plus en plus tout en tournant autour de ce dernier. Il est possible de modéliser les spirales de différentes manières. Ainsi, une spirale peut notamment être définie par son équation en coordonnées polaires, exprimant l'évolution de la coordonnée radiale (rayon) r en fonction de l'angle θ entre le point et l'axe polaire (demi-droite d'angle 0°). A titre d'exemple, la spirale d'Archimède est décrite par la relation : $r = a + b \times \theta,\ \ a, b \in \mathbb{R}$ et la spirale logarithmique par : $r = a.b^{\theta},\ \ a, b \in \mathbb{R}$.

Dans le cadre d'un évitement d'obstacle en spirale, la spirale logarithmique est retenue[1]. En effet, ce type de spirales présente comme avantage d'être équiangle. L'angle entre le rayon et sa tangente reste alors constant en tout point de la courbe. Ainsi le robot, équipé de caméras ou de LiDARs, positionne toujours l'obstacle au même endroit de son champ de vision lorsqu'il décrit une telle spirale.

Cette section présente la définition et les conventions utilisées pour modéliser les spirales logarithmiques. Elle décrit également le modèle du robot mobile différentiel à quatre roues.

[1] Une spirale logarithmique dans le cas tridimensionel est appellée conchospirale

2.1.1 Convention et définition des spirales

Les conchospirales ont été introduites dans l'article [Boyadzhiev, 1999] pour décrire la trajectoire d'un insecte volant autour d'une source lumineuse. Ce type de spirale est modélisée par son centre O_s (dénommé Spiral Center Point (SCP) dans la suite de ce manuscrit) et par un point O_t qui se déplace autour de ce centre dans un plan. Elle se définit par rapport au repère orthonormé attaché au centre de la spirale O_s qui s'écrit $F_s = (O_s, \overrightarrow{x_s}, \overrightarrow{y_s}, \overrightarrow{z_s})$. La notation $\overrightarrow{v^*(t)}$ définit le vecteur vitesse appliqué sur le point mobile O_t dont la norme est notée $v^*(t)$. De plus, le vecteur $\overrightarrow{d^*(t)}$, dont la norme est notée $d^*(t)$, est le vecteur connectant les points O_s à O_t. L'angle $\theta^*(t)$ est l'angle entre le vecteur $\overrightarrow{x_s}$ et le vecteur $\overrightarrow{d^*}$. Enfin, l'angle α^* est défini comme l'angle orienté entre les vecteurs $\overrightarrow{v^*(t)}$ et $-\overrightarrow{d^*(t)}$. La figure 2.1 résume l'ensemble de ces notations et décrit le modèle d'une spirale divergente qui évolue dans le sens direct.

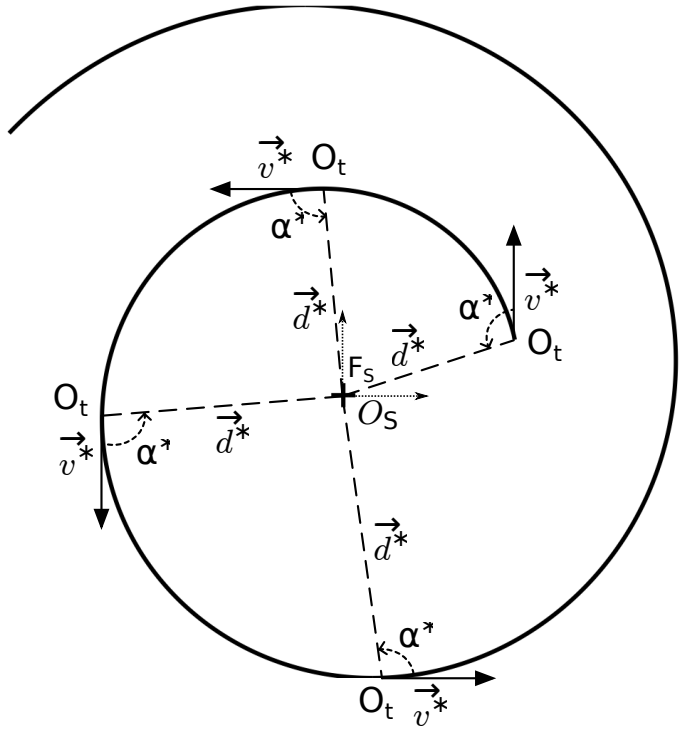


Figure 2.1: Modèle de la spirale et les notations associées.

Dans le repère F_s attaché au point O_s, le vecteur radial du point O_t est donc noté $\overrightarrow{d^*(t)} = (x(t), y(t))$. La spirale est alors décrite par le système d'équations cartésiennes suivant :

$$\begin{cases} x(t) = d^*(t)\cos(\theta^*(t)) \\ y(t) = d^*(t)\sin(\theta^*(t)) \end{cases} \tag{2.1}$$

La variable t définit le temps. Les variables (d^*, θ^*) représentent les coordonnées polaires du point O_p dans le repère F_s. Dans notre cas, $\theta^* \in]-\infty, \infty[$ et $d^* \in [0, \infty[$. La distance $d^*(t)$ est fonction de l'angle $\theta^*(t)$. Afin de respecter la contrainte équiangle, l'angle $\alpha^*(t)$ entre les vecteurs $\overrightarrow{v^*(t)}$ et $\overrightarrow{-d^*(t)}$ doit être constant. Cette contrainte permet, selon les calculs exposés dans [Boyadzhiev, 1999], d'obtenir la relation (2.2) qui correspond à l'équation polaire d'une spirale équiangle dont l'état initial est (d_0, θ_0) :

$$d^*(\theta^*) = d_0 e^{\cot \alpha^* (\theta_0 - \theta^*)} \tag{2.2}$$

Dans le cadre d'une approche d'évitement, nous considérons qu'à la fois l'angle $\alpha^*(t)$ et la vitesse linéaire $v^*(t)$ restent constants dans le temps. Ainsi, il est montré dans [Boyadzhiev, 1999] que si $v^*(t)$ et $\alpha^*(t)$ sont constants alors O_t décrit une spirale dont le centre est O_s. Ils sont donc pour la suite respectivement notés v^* et α^*. A partir de l'équation (2.1), il vient alors[2] :

$$\begin{cases} \dot{x} = \dot{d^*} \cos(\theta) - d^* \sin(\theta)\dot{\theta} \\ \dot{y} = \dot{d^*} \sin(\theta) + d^* \cos(\theta)\dot{\theta} \end{cases} \tag{2.3}$$

Sachant que α^* est l'angle entre $\overrightarrow{v^*}$ et $-\overrightarrow{d^*}$, et que $\overrightarrow{v^*}$ est colinéaire à $\overrightarrow{\dot{d^*}}$, nous pouvons exprimer l'angle α^* selon la relation suivante :

$$\frac{x\dot{x} + y\dot{y}}{\sqrt{x^2 + y^2}\sqrt{(\dot{x}^2 + \dot{y}^2)}} = \frac{\dot{d^*}}{\sqrt{(\dot{d^*})^2 + (\dot{\theta})^2 (d^*)^2}} = -\cos(\alpha^*) \tag{2.4}$$

Ainsi, si la spirale est parcourue à vitesse constante, cela implique :

$$||\dot{d^*}||^2 = (\dot{x})^2 + (\dot{y})^2 = (\dot{d^*})^2 + (\dot{\theta})^2 (d^*)^2 = (v^*)^2 \tag{2.5}$$

En combinant les équations (2.4) et (2.5), nous obtenons la relation suivante :

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

L'analyse de cette équation montre qu'à partir d'une condition initiale (d_0, θ_0) donnée la spirale ne dépend donc que de la valeur du paramètre α^*. Son signe définit le sens de contournement et sa valeur le type de spirale.

Sens de contournement

Le signe du paramètre α^* définit le sens de contournement direct ou indirect (anti-horaire ou horaire) de la spirale : si $\alpha^* \geq 0$, la spirale s'effectue dans le sens direct (sens anti-horaire). En revanche, si $\alpha^* < 0$, la spirale s'effectue dans le sens indirect (sens horaire).

[2] Pour des raisons de lisibilité, la dépendance au temps est omise.

Type de spirale

La valeur du paramètre α^* définit le type de spirale : si $0 \leq \alpha^* < \pi/2$ ou $-\pi/2 < \alpha^* < 0$, la spirale est concentrique (aussi appelée spirale convergente), et le robot se rapproche du centre de la spirale. Si $\pi/2 \leq \alpha^* \leq \pi$ ou $-\pi \leq \alpha^* \leq -\pi/2$, la spirale est excentrique (aussi appelée spirale divergente), et le robot s'éloigne du centre de la spirale. Si $\alpha^* \pm \pi/2$, la spirale est un cercle, et d^* reste constant.

La figure 2.2 montre les diverses formes de spirales en fonction du signe et de la valeur du paramètre α^*.

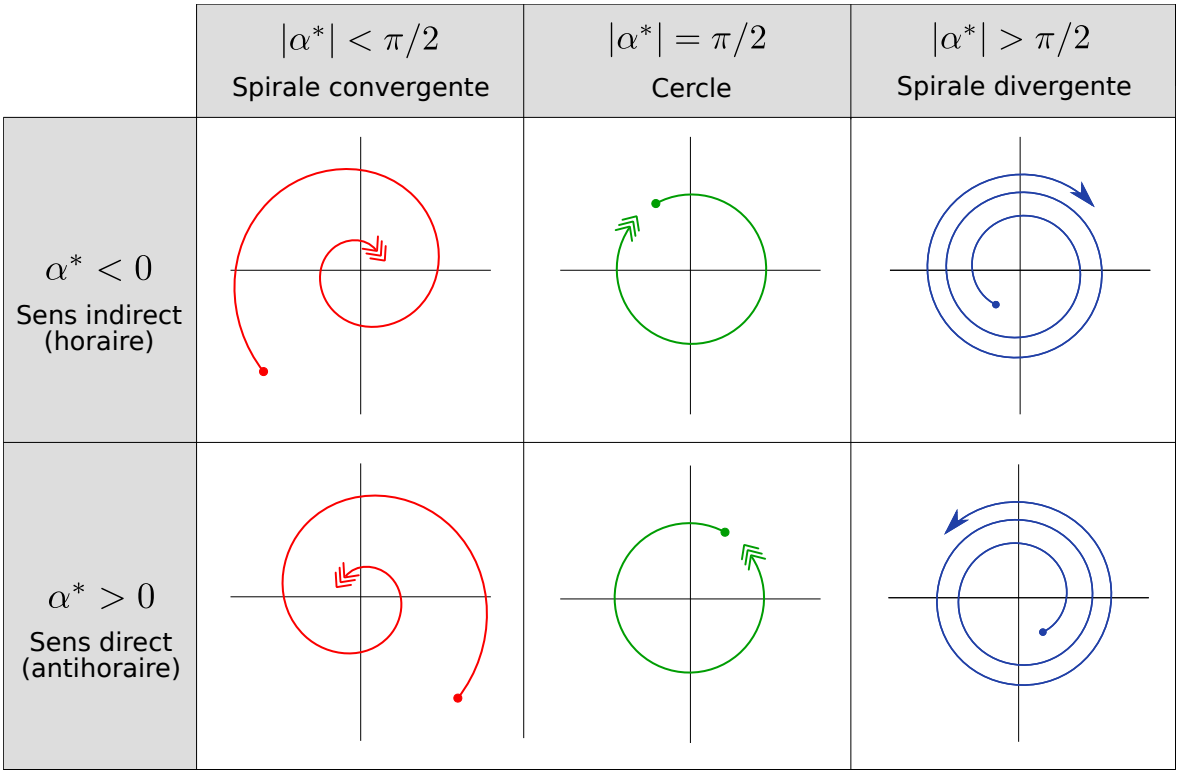


Figure 2.2: Différentes spirales en fonction du signe et de la valeur de α^*.

2.1.2 Modèle du robot et notations

Dans les applications agricoles, un type de robot largement utilisé est un robot différentiel à quatre roues comme le robot Oz de Naïo-Technologies (cf. figure 1.2a) qui a la capacité de tourner sur lui-même. Le modèle de ce type de robot est présenté sur la figure 2.3. $F_w = (O_w, \overrightarrow{x_w}, \overrightarrow{y_w}, \overrightarrow{z_w})$ est le repère fixé au monde, tandis que $F_r = (O_r, \overrightarrow{x_r}, \overrightarrow{y_r}, \overrightarrow{z_r})$ est le repère attaché au robot. Le vecteur $\chi(t) = [x(t), y(t), \theta(t)]^T$ représente la pose du robot dans le repère monde F_w. $x(t)$ et $y(t)$ sont donc les

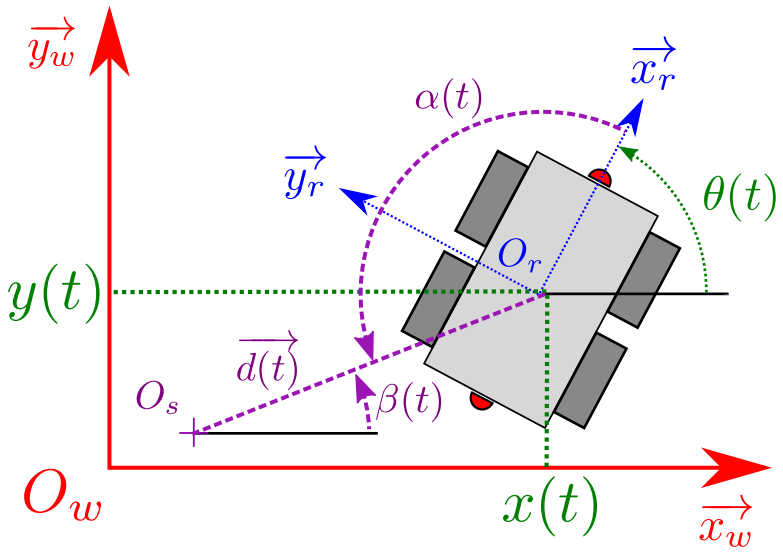


Figure 2.3: Modèle du robot.

coordonnées de O_r dans le repère monde F_w et $\theta(t)$ est l'angle entre les vecteurs $\overrightarrow{x_w}$ et $\overrightarrow{x_r}$. O_s est le centre de la spirale à suivre, tandis que $\overrightarrow{d}$ représente le vecteur joignant O_s à O_r et $\alpha(t)$ est l'angle entre les vecteurs $\overrightarrow{x_r}$ et $-\overrightarrow{d}$. $\beta(t)$ est l'angle entre les vecteurs $\overrightarrow{x_w}$ et $\overrightarrow{d}$. Le robot est commandable à l'aide de deux entrées qui sont la consigne de vitesse linéaire $v(t)$ (en $m.s^{-1}$) et la consigne de vitesse angulaire $\omega(t)$ (en $rad.s^{-1}$).

2.2 Lois de commandes pour le suivi de spirales

Après la modélisation de la spirale et du robot, cette section détaille les lois de commande appliquées au robot pour suivre une spirale donnée. La stabilité et la présence éventuelle de singularité sont également étudiées.

2.2.1 Modélisation du suivi de spirales

Notre stratégie de navigation se base sur le suivi de spirales dont les paramètres seront définis dans la section 2.3.3. À partir du modèle du robot présenté dans la figure 2.3, nous pouvons écrire la relation suivante, reliant les angles $\alpha(t)$, $\beta(t)$ et $\theta(t)$:

$$\alpha(t) = \pi - \theta(t) + \beta(t) \tag{2.7}$$

Il s'ensuit :

$$\dot{\alpha}(t) = -\dot{\theta}(t) + \dot{\beta}(t) = -\omega(t) + \frac{v(t)}{d(t)}\sin(\alpha(t)) \tag{2.8}$$

En utilisant l'équation (2.6), le modèle suivant, dérivé de la cinématique d'un robot différentiel, peut être déduit :

$$\begin{cases} \dot{d}(t) = -v(t)\cos(\alpha(t)) \\ \dot{\alpha}(t) = -\omega(t) + \frac{v(t)}{d(t)}\sin(\alpha(t)) \end{cases} \tag{2.9}$$

Il s'agit donc d'asservir les variables $d(t)$ et $\alpha(t)$ afin de suivre une trajectoire en spirale telle que :

$$\dot{d}^* = -v^* cos(\alpha^*) \tag{2.10}$$

avec v^* une vitesse linéaire constante appliquée au robot et α^* l'angle de suivi de spirale désirée. d^* et α^* sont donc les consignes de la commande. La stratégie de commande est donc de contrôler les variables $d(t)$ et $\alpha(t)$ par rapport aux consignes d^* et α^*.

2.2.2 Erreurs et dynamiques

Avec l'hypothèse que la trajectoire de référence (ou consigne) est la suivante :

$$\begin{cases} \dot{d}^*(t) = -v^* \cos\alpha^* \\ \dot{\alpha}^* = 0 \end{cases} \tag{2.11}$$

et que les variables $d(t)$ et $\alpha(t)$ sont asservies aux consignes d^* et α^*, l'erreur en distance $e_d(t)$ et l'erreur en angle $e_\alpha(t)$ sont alors définies par les relations suivantes :

$$\begin{cases} e_d(t) = d(t) - d^*(t) \\ e_\alpha(t) = \alpha(t) - \alpha^*(t) \end{cases} \tag{2.12}$$

En utilisant les équations (2.11) et (2.12), les dynamiques des erreurs $\dot{e}_d(t)$ et $\dot{e}_\alpha(t)$ sont données par les relations suivantes :

$$\begin{cases} \dot{e}_d(t) = v^*\cos(\alpha^*(t)) - v(t)\cos(\alpha(t)) \\ \dot{e}_\alpha(t) = -\omega(t) + \frac{v(t)}{d(t)}\sin(\alpha(t)) \end{cases} \tag{2.13}$$

$$\begin{cases} \dot{e}_d(t) = v^*\cos(\alpha^*(t)) - v(t)\cos(e_\alpha(t) - \alpha^*(t)) \\ \dot{e}_\alpha(t) = -\omega(t) + \frac{v(t)}{e_d(t)+d^*(t)}\sin(e_\alpha(t) - \alpha^*(t)) \end{cases} \tag{2.14}$$

Notre objectif est de développer des lois de commande afin que le système défini avec les erreurs (e_d, e_α) soit globalement asymptotiquement stable (GAS). Par la suite, nous proposons une série de lois de commande qui satisfont le plus possible cette hypothèse.

2.2.3 Présentation des lois de commandes

Cette sous-section est consacrée à la présentation de trois lois de commande pour la vitesse angulaire $\omega(t)$. Ces lois de commande sont basées sur les erreurs (e_d, e_α), et prendront en considération le fait que la vitesse linéaire v^* reste constante.

Première loi de commande

Cette première loi de commande est basée sur la fonction de Lyapunov $V(e)$ suivante :

$$V(e) = \frac{1}{2}e_d(t)^2 + \frac{1}{2}e_\alpha(t)^2 \tag{2.15}$$

Nous obtenons :

$$\begin{aligned}\dot{V}(e) = e_d(t)[v^* \cos(\alpha^*) - v(t)\cos(e_\alpha(t) + \alpha^*)] \\ + e_\alpha(t)[-\omega(t) + \frac{v(t)}{e_d(t) + d^*(t)} \sin(e_\alpha(t) + \alpha^*)]\end{aligned} \tag{2.16}$$

En choisissant la vitesse linéaire $v(t)$ et la vitesse angulaire $\omega(t)$ telles que :

$$\begin{cases} v(t) = \frac{\cos(\alpha^*)}{\cos(e_\alpha(t)+\alpha^*)} v^* + \lambda_d e_d \\ \omega(t) = \frac{v(t)}{e_d(t)+d^*(t)} \sin(e_\alpha(t) + \alpha^*) + \lambda_\alpha e_\alpha \end{cases} \tag{2.17}$$

avec λ_d et λ_α deux gains réels positifs, nous obtenons :

$$\dot{V}(e) = -\lambda_d e_d^2 - \lambda_\alpha e_\alpha^2 \tag{2.18}$$

D'après la théorie de Lyapunov [Sastry, 1999], le système rebouclé avec cette loi de commande est donc globalement asymptotiquement stable.

Cependant, la première loi commande angulaire $\omega(t)$, notée $\omega_1(t)$, peut se réécrire comme suit :

$$\omega_1(t) = \omega(t) = \frac{v(t)}{d(t)} \sin(e_\alpha(t) + \alpha^*) + \lambda_\alpha e_\alpha \tag{2.19}$$

Nous remarquons que cette commande $\omega(t)$ ne dépend pas de la distance désirée $d^*(t)$. Par conséquent, cette loi de commande ne permet pas d'asservir le robot à une spirale spécifique, si ce dernier n'est pas initialement sur cette spirale.

Deuxième loi de commande

Cette deuxième loi de commande, introduite dans [Leca et al., 2019], vise à résoudre le problème précédent à savoir que la commande $\omega(t)$ ne dépend pas de la distance désirée $d^*(t)$. Afin d'assurer la convergence à la fois des erreurs e_d et e_α vers zéro de manière asymptotique, cette deuxième loi repose sur un retour de sortie. Ainsi, à partir de n'importe quelle situation initiale, le robot est en mesure de rejoindre une spirale donnée par ses paramètres α^* et $d^*(t)$. Étant donné que le système est non linéaire, nous utilisons une linéarisation exacte entrée-état. En utilisant la même formulation des erreurs $e_d(t)$ et $e_\alpha(t)$ des équations (2.12), les dynamiques des erreurs sont alors les suivantes :

$$\begin{cases} \dot{e}_d(t) = v^*[\cos(\alpha^*) - \cos(\alpha(t))] \\ \dot{e}_\alpha(t) = -\omega(t) + \frac{v^*}{d(t)} \sin(\alpha(t)) \end{cases} \tag{2.20}$$

Ces équations peuvent être écrites comme suit :

$$\begin{cases} \dot{e}_d(t) = v^*[\cos(\alpha^*) - \cos(e_\alpha(t) + \alpha^*)] \\ \dot{e}_\alpha(t) = -\omega(t) + \frac{v^*}{e_d(t)+d^*(t)} \sin(e_\alpha(t) + \alpha^*) \end{cases} \tag{2.21}$$

Afin de linéariser ce système d'erreur, nous définissons tout d'abord la transformation $T(e)$ comme suit :

$$z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} e_d(t) \\ v^*[\cos(\alpha^*) - \cos(e_\alpha(t) + \alpha^*)] \end{bmatrix} = T(e) \tag{2.22}$$

Nous remarquons que $T(0) = (0)$ et que le domaine de T est $\mathcal{D}$ défini par :

$$\mathcal{D} = \{(e_d, e_\alpha \in \mathbf{R}^2 || e_d \in \mathbf{R}, e_\alpha \in]-\alpha^*, -\alpha^* + \pi[\} \tag{2.23}$$

Ainsi, T définit un difféomorphisme. Si nous appliquons cette transformation au système d'erreur (2.21), nous obtenons :

$$\begin{cases} \dot{z}_1 = z_2 \\ \dot{z}_2 = v^* \sin(e_\alpha(t) + \alpha^*)\dot{e}_\alpha(t) \end{cases} \tag{2.24}$$

et par la suite :

$$\begin{cases} \dot{z}_1 = z_2 \\ \dot{z}_2 = v^* \sin(e_\alpha(t) + \alpha^*)(-\omega(t) + \frac{v^* \sin(e_\alpha(t)+\alpha^*)}{e_d(t)+d^*(t)}) \end{cases} \tag{2.25}$$

En prenant :

$$\widetilde{\omega}(t) = v^* \sin(e_\alpha(t) + \alpha^*)(-\omega(t) + \frac{v^* \sin(e_\alpha(t) + \alpha^*)}{e_d(t) + d^*(t)}) \tag{2.26}$$

nous obtenons :

$$\begin{cases} \dot{z}_1 = z_2 \\ \dot{z}_2 = \widetilde{\omega}(t) \end{cases} \tag{2.27}$$

$\widetilde{\omega}$ est la loi de commande à générer. À ce stade, le système (2.27) est désormais linéaire, et cette loi de commande peut s'écrire classiquement comme suit :

$$\widetilde{\omega}(t) = -\lambda_1 z_1(t) - \lambda_2 z_2(t) \tag{2.28}$$

avec $\lambda_1, \lambda_2 > 0$, le système (2.27) est asymptotiquement stable. Pour finir, grâce aux équations (2.26) et (2.28), la deuxième loi de commande angulaire $\omega(t)$, notée $\omega_2(t)$ s'écrit de la manière suivante :

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

Soit, en utilisant la relation (2.28) :

$$\omega_2(t) = \frac{\lambda_1 z_1(t) + \lambda_2 z_2(t)}{v^* \sin(e_\alpha(t) + \alpha^*)} + \frac{v^* \sin(e_\alpha(t) + \alpha^*)}{e_d(t) + d^*(t)} \tag{2.30}$$

Au final, nous obtenons :

$$\omega_2(t) = \frac{\lambda_1 e_d(t) + \lambda_2 \dot{e}_d(t)}{v^* \sin(e_\alpha(t) + \alpha^*)} + \frac{v^* \sin(e_\alpha(t) + \alpha^*)}{e_d(t) + d^*(t)} \tag{2.31}$$

Nous proposons le théorème suivant : soient deux scalaires positifs λ_1 et λ_2, le système d'erreurs (2.12) en boucle fermée avec la loi de commande (2.27), (2.26), (2.28), où $z = T(e)$ dans le système (2.22), est localement asymptotiquement stable. Pour preuve, il suffit de constater que z converge asymptotiquement vers zéro, et que $z = T(e)$ définit un difféomorphisme local avec $T(0) = 0$. Par construction, cette loi de commande sera donc capable de maintenir le robot le long de la spirale désirée, et les deux gains λ_1 et λ_2 permettent de régler la vitesse de convergence du système en boucle fermée. Le terme $\sin(e_\alpha(t) + \alpha^*)$ au dénominateur de l'équation (2.31) introduit des singularités lorsque $\alpha(t) = k\pi, k \in \mathbb{N}$. Ce terme apparaît du fait de la dérivation au second ordre de l'erreur $e_d(t)$. Ainsi, dans le cas extrême où le robot fait face à l'obstacle ($\alpha = 0$), le dénominateur $v^* \sin(e_\alpha(t) + \alpha^*)$ (qui est égal à $v^* \sin(\alpha)$) sera proche de zéro, et fera tendre la consigne angulaire $\omega_2(t)$ vers l'infini.

Troisième loi de commande

Cette troisième loi de commande, utilisée dans [Leca et al., 2019], [Flecher et al., 2017] et [Le Flecher et al., 2017] prend en compte la singularité du contrôleur précédent. L'équation (2.31) montre que les singularités sont causées par le terme $\sin(e_\alpha(t) + \alpha^*)$ qui provient de la dérivation au second ordre de l'erreur $e_d(t)$. Ainsi, pour éviter cette singularité, la loi de commande proposée ne dépend pas de cette erreur $e_d(t)$. Une erreur hybride, indépendante de $e_d(t)$, est alors proposée comme suit :

$$e_S(t) = e_\alpha(t) - \alpha_S(t, d) \tag{2.32}$$

Cette nouvelle erreur, $e_S(t)$, est donc fonction de l'erreur angulaire $e_\alpha(t)$ et d'un terme $\alpha_S(t, d)$ qui dépend du temps t et de la distance $d(t)$ entre le robot et l'obstacle.

Afin d'identifier les termes impliqués dans les dynamiques de cette erreur, l'équation (2.32) est dérivée par rapport au temps :

$$\dot{e}_S(t) = \dot{\alpha}(t) - \dot{\alpha}_S(t, d) = -\omega(t) + \frac{v(t)}{d(t)} \sin(\alpha(t)) - \dot{\alpha}_S(t, d) \tag{2.33}$$

Pour faire tendre l'erreur $e_S(t)$ vers zéro, nous imposons une décroissance exponentielle et la loi de commande en vitesse angulaire $\omega(t)$, notée $\omega_3(t)$ s'écrit alors comme

suit :

$$\omega_3(t) = \omega(t) = \lambda_S e_S(t) + \frac{v(t)}{d(t)} \sin(\alpha(t)) - \dot{\alpha}_S(t, d) \tag{2.34}$$

avec λ_S un réel positif. Suivant l'objectif fixé, nous constatons que $\omega_3(t)$ ne possède pas de singularité si le terme $\dot{\alpha}_S(t, d)$ n'en possède pas. Nous proposons donc la définition de $\alpha_S(t, d)$ suivante :

$$\alpha_S(t, d) = \epsilon(t)\alpha_D \tag{2.35}$$

$\epsilon(t)$ est l'erreur entre $d^*(t)$ et $d(t)$ qui est normalisée à ± 1, comme le montre l'équation suivante :

$$\epsilon(t) = \text{sign}(d^*(t) - d(t)) \frac{\min(|d^*(t) - d(t)|, n)}{n} \tag{2.36}$$

avec $n \in \mathbb{N}^{*+}$. $\epsilon(t)$ appartient au domaine $[0, 1]$ si $d^*(t) > d(t)$, ou au domaine $[-1, 0]$ si $d^*(t) < d(t)$. En effet, $\epsilon(t)$ a sa norme égale à 1 lorsque $||d(t) - d^*(t)||$ est plus grand qu'une valeur arbitraire n, et égale à 0 quand $d(t) = d^*(t)$. De plus, α_D est défini comme :

$$\alpha_D = \begin{cases} \text{sign}(\alpha^*) \times \pi - \alpha^* & \text{si } d^* > d \\ \alpha^* & \text{si } d^* < d \end{cases} \tag{2.37}$$

Afin d'analyser la stabilité de cette loi de commande, nous définissons la fonction de Lyapunov suivante :

$$V_S(x_S(t)) = \frac{x_{S1}(t)^2 + x_{S2}(t)^2}{2} \tag{2.38}$$

avec

$$x_S(t) = \begin{bmatrix} x_{S1}(t) \\ x_{S2}(t) \end{bmatrix} = \begin{bmatrix} \alpha(t) - \alpha_D - \alpha_D \times \epsilon(t) \\ d(t) - d^*(t) \end{bmatrix} \tag{2.39}$$

L'état d'équilibre $x_{SE} = [x_{S1}(t), x_{S2}(t)]^T = [0, 0]^T$ correspond au robot suivant la spirale définie par α_D tout en étant à la distance $d^*(t)$, c'est-à-dire quand $\alpha(t) = \alpha_D$ et $d(t) = d^*(t)$. De plus, $V_S(x_S(t)) > 0$ pour $x_S(t) \neq x_{SE}$ et $V_S(x_S(t)) = 0$ quand $x_S(t) = x_{SE}$. Afin d'étudier l'évolution de $V_S(x_S(t))$, nous calculons sa dérivée par rapport au temps :

$$\begin{aligned} \dot{V}_S(x_S(t)) &= \dot{x}_{S1}(t)x_{S1}(t) + \dot{x}_{S2}(t)x_{S2}(t) \\ &= \lambda_S e_S(t)^2 + [\dot{d}(t) - \dot{d}^*(t)][d(t) - d^*(t)] \\ &= \lambda_S e_S(t)^2 - v[cos(\alpha(t) - cos(\alpha_D)][d(t) - d^*(t)] \end{aligned} \tag{2.40}$$

Cette équation (2.40) montre que $\dot{V}_S(x_S(t)) = 0$ pour $x_S(t) = x_{SE}$. Cependant, il ne peut être montré que $\dot{V}_S(x_S(t)) < 0$ pour $x_S(t) \neq x_{SE}$. En effet, même si $\lambda_S e_S(t)^2 < 0$ pour $x_B(t) \neq x_{SE}$, le deuxième terme $v[\cos(\alpha(t) - \cos(\alpha_D)][d(t) - d^*(t)]$ n'est pas toujours positif. Pour certaines configurations initiales, le signe de $\cos(\alpha(t) - \alpha_D)$ est

différent de celui de $d(t) - d^*(t)$. De plus, suivant l'orientation initiale du robot, le terme $||d(t) - d^*(t)||$ peut ne pas diminuer immédiatement lorsque ce dernier commence à bouger. En effet, seule l'entrée $\omega(t)$ est utilisée pour commander deux degrés de liberté $\alpha(t)$ et $d(t)$ d'un robot non-holonome. Ainsi, il peut être nécessaire de préalablement orienter le robot. Durant cette phase la distance $||d(t) - d^*(t)||$ augmente. Ainsi, nous pouvons conclure que la loi de commande n'est pas globalement asymptotiquement stable. En revanche, il est immédiat de constater que :

$$\begin{aligned} &\text{Si } d(t) > d^*(t) : \\ &\qquad \cos(\alpha(t) - cos(\alpha_D) > 0 \text{ si } \alpha(t) > \alpha_D \\ &\text{Si } d(t) < d^*(t) : \\ &\qquad \cos(\alpha(t) - cos(\alpha_D) < 0 \text{ si } \alpha(t) < \alpha_D \end{aligned} \tag{2.41}$$

Cela garantit que la loi de commande est localement asymptotiquement stable une fois que $\alpha(t)$ est au-delà de α_D.

Ainsi, nous avons à notre disposition trois lois de commande permettant au robot de suivre une spirale donnée. Chacune de ces lois possède des avantages et des inconvénients, notamment au niveau des domaines de stabilité et des zones de singularité. Nous allons voir dans le chapitre suivant comment ces dernières peuvent être combinées afin d'obtenir une sortie de contrôle pertinente.

2.3 Stratégie de navigation

Cette section est dédiée à la présentation de la stratégie de la tâche de navigation qui s'inscrit dans la catégorie des méthodes réactives. Cette tâche de navigation consiste à atteindre un but O_g dont les coordonnées sont définies par (x_g, y_g) dans le repère monde F_w, tout en évitant les obstacles statiques présents dans l'environnement du robot. Le principe de la tâche repose sur l'enchaînement de deux lois de commandes comme suit :

- Une loi de commande, dite *Go-To-Goal (GTG)*, qui dirige le robot vers une position désirée lorsque l'espace est considéré comme libre. Par la suite, cette loi de commande est nommée *Contrôleur GTG*.
- Une loi de commande d'évitement d'obstacles qui est enclenchée lorsqu'un obstacle est détecté. Cette dernière est basée sur l'utilisation des deux lois de commande pour le suivi de spirale présentées dans l'équation (2.44). Elle est nommée *Contrôleur SA* (pour *Spiral Avoidance*).
- Des seuils et des conditions déterminant l'enchaînement des deux contrôleurs GTG et SA, ainsi que des méthodes permettant de choisir les paramètres des spirales servant pour l'évitement.

Nous considérons donc un robot mobile équipé de capteurs extéroceptifs adéquats capables de fournir une acquisition sous la forme d'un nuage de points à 360° autour de lui dans un plan horizontal. Dans la partie précédente, nous avons présenté diverses lois de commande, obtenues à l'aide de techniques de l'automatique. Le robot est donc en mesure de rejoindre, puis de suivre n'importe quelle spirale, donnée par trois paramètres : son centre, l'angle α^* désiré, et la distance d^* désirée. Dans cette partie, nous présentons une méthode de navigation locale, permettant à un robot d'atteindre un point donné dans l'espace.

Ainsi, nous allons présenter dans cette section une stratégie mettant en œuvre l'ensemble de ces trois points. Cette dernière sera présentée en simulation afin d'en montrer ses capacités et ses limitations.

2.3.1 Contrôleur GTG (*Go-To-Goal*)

Le contrôleur Go-To-Goal est un contrôleur qui a pour but de mener le robot à un but préalablement défini, en faisant l'hypothèse d'un espace de travail totalement libre. Ce but peut être une position, auquel cas l'orientation du robot n'importe pas, ou bien une pose, auquel cas le robot doit avoir une orientation précise. De nombreux contrôleurs de ce type existent dans la littérature. Nous pouvons citer en exemple deux contrôleurs GTG qui pourront être utilisés :

Contrôleur proportionnel

Ce contrôleur est simplement un contrôleur proportionnel maintenant l'erreur entre la direction de l'obstacle et l'orientation du robot égal à zéro. Ainsi, le robot se dirige toujours vers le but. Avec un tel contrôleur, il est possible d'atteindre une position précise, mais pas de contrôler l'orientation du robot à cette position.

Contrôleur basé sur les courbes de Dubins

Ce contrôleur est basé sur un suivi de chemins de Dubins [Dubins, 1957]. Ces chemins permettent de relier deux poses à l'aide d'arcs de cercle reliés tangentiellement par des segments de droite. La figure 2.4 montre comment un chemin de Dubins peut être utilisé pour relier deux poses A et B. Ce contrôleur permet notamment de contrôler l'orientation qu'aura le robot au point d'arrivé.

2.3.2 Contrôleur SA (*Spiral Avoidance*)

Ce paragraphe propose une loi de commande qui fusionne les lois précédentes en fonction de la position relative robot-obstacle. En effet, dans un contexte de navigation et d'évitement, les bornes d'évolution des valeurs des variables $\alpha(t)$ et $d(t)$ ne peuvent pas être fixées. Par conséquent, certaines configurations n'assurent pas la stabilité ou

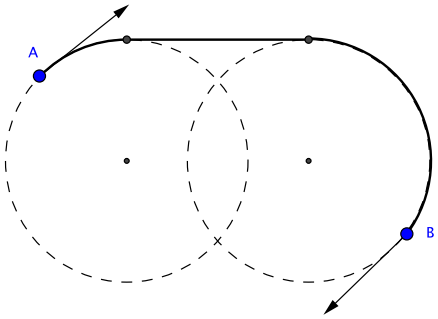


Figure 2.4: Exemple de chemin de Dubins

l'absence de singularité de certaines lois de commande. Une loi de commande de la vitesse angulaire basée sur la combinaison des lois de commandes précédentes selon la configuration robot-obstacle permet de garantir le contrôle du robot dans n'importe quel domaine de l'espace. Cette loi de commande repose sur l'utilisation de la loi de commande $\omega_2(t)$ (2.31) lorsque le robot a rejoint la spirale désirée. Dans ce cas, nous avons la garantie que l'erreur angulaire $e_\alpha(t)$ est faible. La loi de commande $\omega_3(t)$ (2.34) est donc retenue pour amener le robot sur la spirale désirée au moment du déclenchement de l'évitement d'obstacle. La première loi de commande de vitesse angulaire ne permettant pas de s'asservir sur la spirale désirée, elle n'est donc pas utilisée, mais a tout de même été présentée à but exhaustif. Ainsi, pour des raisons de clarté, nous nommons par la suite ω_A la loi de commande précedemment introduite en tant que ω_2, et ω_B la loi de commande précédemment nommée ω_3. En guise de rappel, nous avons donc les contrôleurs suivants (cf. équations (2.31) et (2.34)) :

$$\omega_A(t) = \omega_2(t) = \frac{\lambda_1 e_d(t) + \lambda_2 \dot{e}_d(t)}{v^* \sin(e_\alpha(t) + \alpha^*)} + \frac{v^* \sin(e_\alpha(t) + \alpha^*)}{e_d(t) + d^*(t)} \tag{2.42}$$

$$\omega_B(t) = \omega_3(t) = \lambda_S e_S(t) + \frac{v(t)}{d(t)} \sin(\alpha(t)) - \dot{\alpha}_S(t, d) \tag{2.43}$$

Afin d'appliquer le raisonnement poursuivi ci-dessus, nous introduisons un seuil de commutation e_α^{switch} sur l'erreur en angle e_α. Cela conduit aux conditions de commutation suivantes :

- Si $|e_\alpha(t)| < e_\alpha^{switch}$, la loi de commande en vitesse angulaire retenue est ω_A (cf. équation (2.42))
- Sinon, la loi de commande en vitesse angulaire retenue est ω_B (cf. équation (2.43)).

Il est nécessaire que le seuil de commutation e_α^{switch} soit fixé de sorte que les valeurs calculées de ω_A restent dans des limites admissibles. Ces limites dépendent de plusieurs facteurs : (i) De la vitesse linéaire v^* choisie par l'utilisateur. (ii) De la vitesse angulaire maximale qui peut être envoyée sans endommager le robot ou ses actionneurs. En outre, il permet d'exprimer à quelle distance de la singularité α doit être avant de commuter du contrôleur ω_B au contrôleur ω_A. En résumé, au début de l'évitement de l'obstacle, le robot utilise la loi de commande ω_B pour éviter tout problème de singularité. Le robot converge vers la spirale désirée. Lorsque l'erreur $e_\alpha(t)$ est inférieure au seuil de commutation e_α^{switch}, c'est-à-dire que la valeur de cet angle est suffisamment éloignée de la singularité, la loi de commande ω_A est appliquée au robot. Enfin, afin de prévenir une discontinuité dans la commutation des deux lois de commande, une fonction d'hystéresis est implémentée par moyenne sur une fenêtre glissante. Ainsi, quand une commutation entre les lois de commande $\omega_A(t)$ et $\omega_B(t)$ est activée, la loi de commande résultante $\omega_R(t)$ est calculée comme suit :

$$\omega_R(t) = \frac{i}{p}\omega_A^{last} + \frac{p-i}{p}\omega_B(t) \tag{2.44}$$

ω_A^{last} est la dernière commande de la vitesse angulaire issue de la commande ω_A. $p \in \mathbb{N}^*$ est le nombre d'itérations utilisées pour gérer la commutation, et $i \in [0, ..., p]$.

2.3.3 Choix des paramètres des spirales

Dans la sous-section précédente, nous avons introduit des lois de commandes permettant de suivre une spirale définie par trois paramètres : l'angle désiré α^*, la distance désirée d^* et le centre de la spirale O_s. Dans cette sous-section, nous allons voir comment choisir pertinemment ces paramètres afin d'obtenir des trajectoires d'évitement adéquates.

Choix de l'angle désiré α^*

Le choix de α^* permet de définir la manière dont le robot se déplace par rapport à l'obstacle avec le type de spirale. Un angle plus petit que $\frac{\pi}{2}$ en valeur absolue génère une trajectoire de type spirale intrinsèque (ou spirale convergente). Le robot se rapproche de l'obstacle et cela crée donc un risque de collision. Un angle plus grand que $\frac{\pi}{2}$ génère une spirale extrinsèque (ou spirale divergente). Le robot s'éloigne de plus en plus de l'obstacle et ce qui entraîne une trajectoire plus longue. Ainsi, le choix se porte donc sur un angle $\alpha^* = \pm\frac{\pi}{2}$. Le signe de α^* dépend du sens de contournement autour de l'obstacle qui est discuté dans la suite.

Choix de la distance désirée d^*

La valeur de d^* est fixée à une valeur constante afin de rester cohérent avec le choix de α^*. Cette valeur est choisie afin d'assurer que le robot ne s'approche pas moins

que cette valeur du plus proche obstacle détecté. Elle doit être choisie en adéquation avec la taille du robot, sa maniabilité, et la disposition typique de l'environnement.

Choix du centre de la spirale (SCP, Spiral Center Point)

Le centre de la spirale SCP doit être choisi pour garantir les comportements suivants :

- L'absence de collision, ainsi que la sécurité du robot, des objets et personnes avoisinants. Par conséquent, le robot ne doit pas s'approcher à une distance plus petite qu'une distance de sécurité spécifiée par l'utilisateur. Il est donc important, à l'issue des calculs, de ne pas obtenir un SCP qui soit *à l'intérieur* d'un obstacle.

- L'évitement d'obstacles de taille et de formes quelconques (obstacles ponctuels ou volumineux, concaves ou convexes, voir par exemple des murs entiers). Ainsi, le SCP devra être sélectionné afin d'être adaptable à plusieurs profils d'obstacles.

- Un évitement avec une trajectoire lisse et continue du robot, même lors de fortes variations dans la forme des obstacles. La géométrie des obstacles détectés doit donc être dans la mesure du possible prise en compte dans le calcul des SCP successifs.

- Une commande du robot adaptée à sa cinématique. Les vitesses linéaires et angulaires appliquées au robot doivent être cohérentes et admissibles.

Ainsi, au vu des points précédents, le principe est de définir un centre de spirale dynamique évoluant dans le temps au lieu d'un SCP fixe comme dans les articles [Flecher et al., 2017] et [Futterlieb, 2017]. Dans ces travaux, le barycentre de l'obstacle est pris comme centre d'évitement et gardé constant pendant tout l'évitement. La conséquence est que dans le cas d'obstacles convexes, le SCP se retrouve placé au centre de l'obstacle, et non pas à sa surface. Ainsi, le robot s'approche à une distance moindre que celle désirée de la surface de l'obstacle.

L'idée est donc que le SCP soit mis à jour en fonction des données acquises par les différents capteurs du robot, que ces derniers soient des caméras stéréo, ou des lasers de type LiDARs. Le SCP est calculé pour se déplacer sur l'enveloppe convexe de l'obstacle le plus proche. De cette façon, le SCP suit la forme des obstacles convexes. Il doit aussi être capable de *lisser* les obstacles concaves ainsi que les éventuelles aspérités. De même, il doit gérer plusieurs obstacles à la fois et éviter notamment que le robot ne cherche à se faufiler entre deux obstacles proches. L'évolution du SCP doit donc dépendre du mouvement du robot ainsi que des données capteurs.

Méthode du calcul du centre de la spirale dynamique

La méthode proposée dans [Leca et al., 2019] se base sur le calcul préalable de deux points : le point O_c le plus proche du robot, calculé en utilisant l'algorithme 1, et un barycentre O_b obtenu selon l'algorithme 2, prenant en entrées P la liste des points détectés autour de l'obstacle par les capteurs, ainsi que les variables O_c et d^*. La méthode se résume par les étapes suivantes :

- Étape 1 : Après l'acquisition de données autour du robot, *le point le plus proche en norme cartésienne* O_c est calculé selon l'algorithme 1.

Algorithme 1 Calcul de O_c

Entrées : P (liste de points)
$d_{min} \leftarrow +\infty$
pour chaque Point Pt dans P **faire**
  $d \leftarrow \text{distance}(Pt,\ P)$
  si $d < d_{min}$ **alors**
    $d_{min} \leftarrow d$
    $O_c \leftarrow Pt$
  fin si
fin pour chaque

- Étape 2 : tous les points détectés à l'intérieur du cercle de rayon $2d^*$ (choisi pour considérer la distance de sécurité du robot) centré sur O_c sont utilisés pour calculer *leur barycentre* O_b selon l'algorithme 2.

Algorithme 2 Calcul de O_b

Entrées : P, O_c, d^*
Sortie : O_g
$d_{min} \leftarrow +\infty$
$n \leftarrow 0$
$P_{final} \leftarrow [0\ 0]^T$
pour chaque Point Pt dans P **faire**
  $d \leftarrow \text{distance}(Pt,\ O_c)$
  si $d < 2d^*$ **alors**
    $P_{final} \leftarrow P_{final} + Pt$
    $n \leftarrow n + 1$
  fin si
fin pour chaque
$O_b \leftarrow P_{final}/n$

La spécificité du calcul du barycentre O_b tient au cercle de rayon $2d^*$ centré en O_c utilisé pour obtenir le barycentre. En effet, prendre en compte l'ensemble

des points à l'intérieur de ce cercle permet notamment de considérer différents obstacles placés à une distance inférieure à $2d^*$ l'un de l'autre, et entre lesquels le robot ne peut donc pas aller tout en respectant une distance minimum de d^* par rapport à chacun d'eux.

- Étape 3 : la distance d_b entre O_b et le centre du robot, ainsi que la distance d_c entre O_c et le centre du robot sont calculées. Finalement, le SCP O_s est choisi comme le point le plus proche du robot entre O_c et O_b. Ainsi $O_s = O_c$ si $d_c < d_b$, et $O_s = O_b$ sinon.

La figure 2.5 illustre le calcul du SCP dynamique pour différents types d'obstacles nommés cas (a.), (b.) et (c.), durant l'évitement. En vert, sont représentés les points détectés par les capteurs, tandis que les cercles bleus et rouges matérialisent respectivement la distance de sécurité du robot d^*, et le cercle de rayon $2d^*$ centré en O_c, utilisé pour calculer le barycentre.

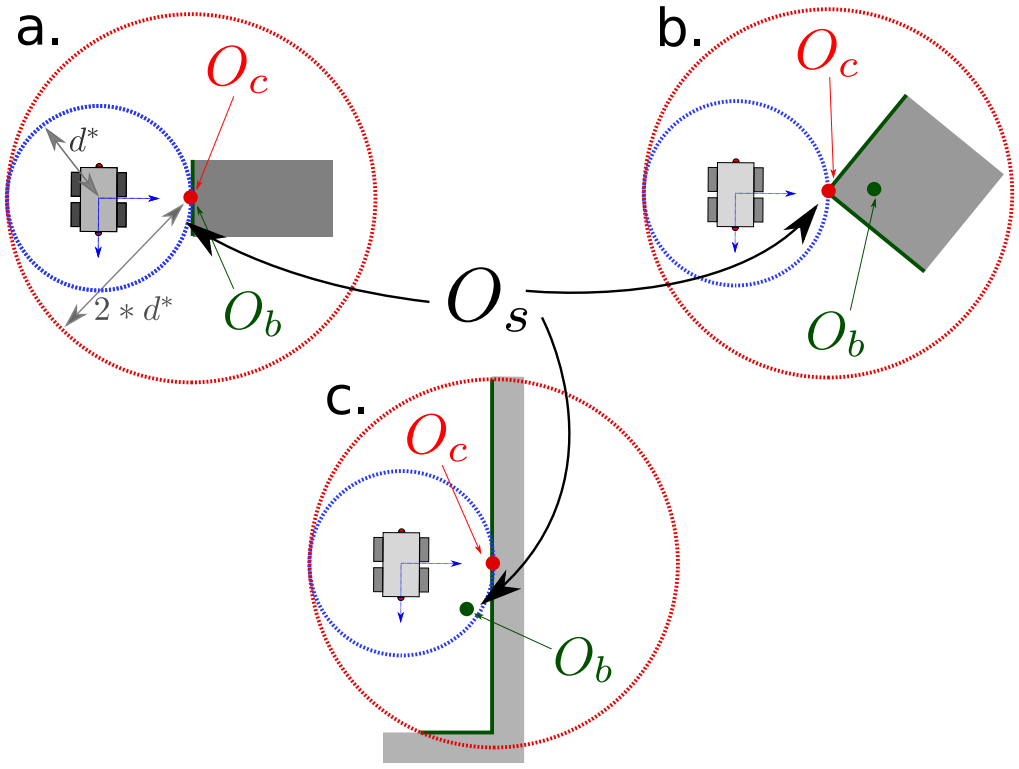


Figure 2.5: Calcul du SCP dynamique pour différents types d'obstacles nommés cas (a.), (b.) et (c.).

Pour le cas (a.), la méthode calcule en premier O_c, qui est donc la projection orthogonale du robot sur le mur. Ensuite, O_b est calculé comme le barycentre de tous les points détectés par le LiDAR à l'intérieur du cercle rouge. Dans ce cas (.a), étant

donné qu'un seul segment est perçu, O_b coïncide avec O_c, ce qui donne $O_s = O_c = O_b$.

Pour le cas (b.), O_c repose sur le coin d'un cube (donc un obstacle convexe). Par conséquent, $O_s = O_c$.

Pour le cas (c.), de manière similaire au cas (a.), le point O_c est obtenu sur le mur. Les points verts inscrits dans le cercle rouge sont donc pris en compte pour calculer le barycentre O_b. Ce dernier étant plus proche du robot que O_c, il s'ensuit $O_s = O_b$.

Pour conclure, dans le cas d'obstacles droits ou convexes, $O_s = O_c$ et O_s suit les bords de l'obstacle. Pour les obstacles concaves, $O_s = O_b$ et O_s est dans l'espace libre. Cette propriété offre l'avantage d'anticiper les futures concavités de l'obstacle, et permet d'éviter les minima locaux. Ainsi, dans le cas par exemple d'un couloir, le robot de cherchera pas à rentrer le couloir si sa taille est inférieure à $2d^*$.

2.3.4 Enchaînement des contrôleurs SA et GTG : conditions d'activation et sens de contournement

Un superviseur permet de gérer l'enchaînement entre les deux lois de commande en fonction des données capteurs. Ces deux lois de commande sont le contrôleur Go-To-Goal (GTG), servant à diriger le robot vers le but, et le contrôleur d'évitement en spirale (SA) quand un risque de collision est détecté. Dans cette sous-section vont donc être présentés les calculs des conditions d'activation pour chaque loi de commande, ainsi que les conditions permettant de choisir un sens de contournement pertinent lorsqu'un obstacle est détecté.

Définition des angles

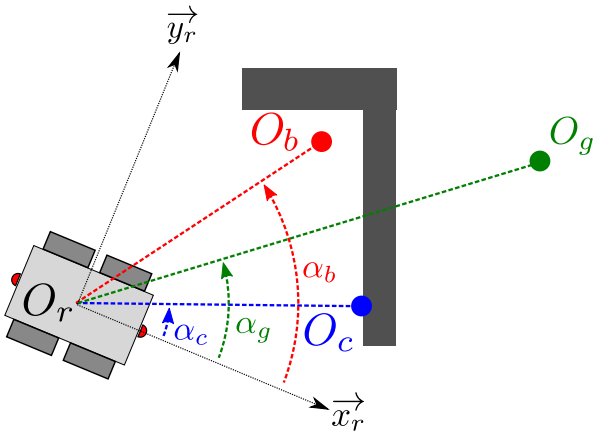


Figure 2.6: Définition des angles α_b, α_c et α_g

Comme le montre la figure 2.6, le calcul du sens de contournement et des conditions d'activation de l'évitement est réalisé à partir de trois angles α_b, α_c et α_g qui sont :

$$\begin{aligned} \alpha_b &= (\overrightarrow{x_r}, \overrightarrow{O_r O_b}) \\ \alpha_c &= (\overrightarrow{x_r}, \overrightarrow{O_r O_c}) \\ \alpha_g &= (\overrightarrow{x_r}, \overrightarrow{O_r O_g}) = \text{atan2}(y_g - y, x_g - x) - \theta \end{aligned} \tag{2.45}$$

Les angles α_b et α_c correspondent respectivement aux angles entre l'axe $\overrightarrow{x_r}$ du robot et les points O_b et O_c, tandis que l'angle O_g correspond à l'angle entre $\overrightarrow{x_r}$ et le point à atteindre O_g.

Conditions d'activation

Deux conditions d'activation des contrôleurs GTG et SA, notées (1) et (2), sont ainsi définies :

$$(1) \begin{cases} d_c < d_c^{tr} \text{ et} \\ \mid \alpha_g - \alpha_c \mid < \pi/2 \end{cases} \qquad (2) \begin{cases} d_b < d_b^{tr} \text{ et} \\ \mid \alpha_g - \alpha_b \mid < \pi/2 \end{cases} \tag{2.46}$$

avec d_c^{tr}, d_b^{tr} les deux distances de déclenchement telles que :

$$\left.\begin{aligned} d_c^{tr} &= d^* + d^*(1 - \tfrac{|\alpha_c|}{\pi/2}) \\ d_b^{tr} &= d^* + d^*(1 - \tfrac{|\alpha_b|}{\pi/2}) \end{aligned}\right\} \text{si contrôleur GTG actif} \tag{2.47}$$

$$\left.\begin{aligned} d_c^{tr} &= 2d^* \\ d_b^{tr} &= 2d^* \end{aligned}\right\} \text{si contrôleur SA actif} \tag{2.48}$$

Le contrôleur d'évitement en spirale est actif si au moins une des conditions (1) ou (2) est vraie. Deux cas doivent être séparés :

- Lorsque le contrôleur GTG est actif, les distances d_c^{tr} et d_b^{tr} évoluent entre d^* et $2d^*$. Ces valeurs dépendent de l'angle avec lequel le robot approche du voisinage de l'obstacle. Typiquement, si un obstacle apparaît directement devant le robot, l'évitement devra être déclenché en amont (à une distance de $2d^*$), tandis que si le robot s'approche de l'obstacle de façon tangentielle, il ne sera pas nécessaire d'anticiper autant.

- Lorsque le contrôleur SA est actif, d_c^{tr} et d_b^{tr} sont tous deux égaux à $2d^*$.

Ainsi, les conditions d'activation sont plus restrictives pour passer du contrôleur SA au contrôleur GTG que pour la transition inverse. Cela a pour effet d'éviter des basculements intempestifs entre les contrôleurs. En effet, une fois le contrôleur SA activé, les valeurs de d_c^{tr} et d_b^{tr} sont fixés à $2d^*$. Le robot cherchant à maintenir une distance d^* entre lui et un des points O_c ou O_b, il ne peut alors quitter l'évitement si et seulement si $|\alpha_g - \alpha_c| < \pi/2$ et $|\alpha_g - \alpha_b| < \pi/2$ (cf. équations (2.46)), c'est-à-dire si l'obstacle est *derrière* le robot.

Choix du sens de contournement (SOM)

Lors de l'enchaînement entre les contrôleurs GTG et SA, un sens de contournement (Sense of Motion, SOM) doit être choisi. Ce SOM est calculé afin d'obtenir la trajectoire d'évitement la plus courte possible en se basant sur la connaissance locale de l'obstacle à l'instant de décision. Par conséquent, les valeurs de α_b et α_g sont comparées afin de choisir le sens de contournement adéquat comme suit :

- Si $\alpha_b \leq \alpha_g$, la plus grande partie de l'obstacle visible se trouve sur le côté droit du robot par rapport au but, le sens de contournement choisi est le sens horaire.
- Si $\alpha_b > \alpha_g$, la plus grande partie de l'obstacle se trouve sur le côté gauche du robot par rapport au but, le sens de contournement choisi est le sens anti-horaire.

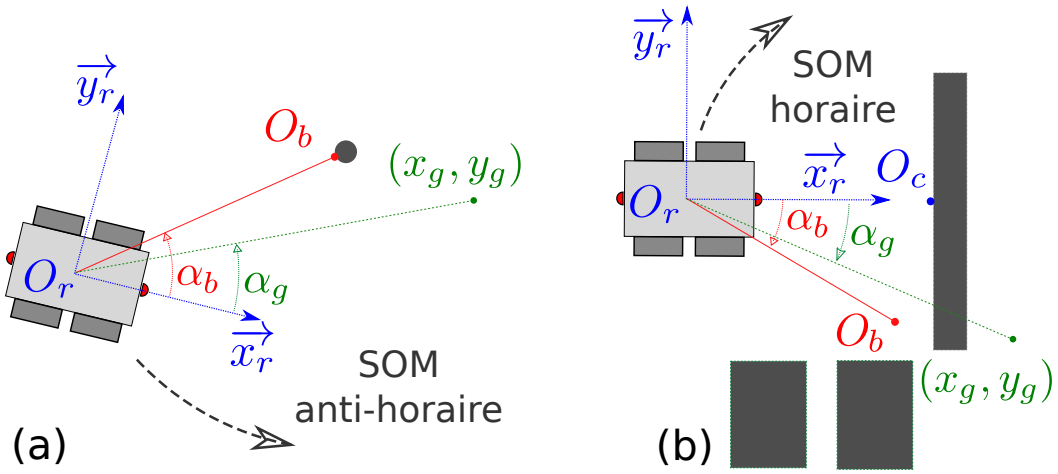


Figure 2.7: Choix du sens de contournement

La figure 2.7 illustre ces deux exemples :

- Exemple (a) : L'obstacle est sphérique et de petite taille, ainsi $O_b \simeq O_c$. Dans ce cas $\alpha_b < \alpha_g$ et un sens de contournement anti-horaire est sélectionné.
- Exemple (b) : Plusieurs obstacles se trouvent à portée du capteur LiDAR. Comme la distance entre ces obstacles est plus petite que $2d^*$, ils sont tous pris en compte pour calculer O_b. Par conséquence, $\alpha_b < \alpha_g$ et un sens de contournement horaire est sélectionné.

Le choix du sens de contournement le plus pertinent à l'instant de la décision est donc fonction de la perception de l'obstacle à cet instant. Ce choix peut donc se révéler ne pas être le meilleur, et cela entraînera seulement une réalisation de la mission de navigation plus lente. Lors de la phase de navigation, quand le robot passe du contrôleur SA au contrôleur GTG, le sens de contournement est remis à zéro, et est recalculé si une nouvelle phase d'évitement est nécessaire.

2.4 Simulation de la navigation du robot dans un environnement statique

L'objectif de cette section est d'évaluer la méthode de navigation avec les deux lois de commande présentées dans un environnement avec des obstacles statiques. Cet environnement et la configuration du robot sont tout d'abord présentés. La trajectoire, les erreurs et les commandes calculées sont ensuite analysées dans cet environnement.

2.4.1 Environnement de déplacement du robot

L'environnement est détaillé sur la figure 2.8. Cet environnement de simulation cherche

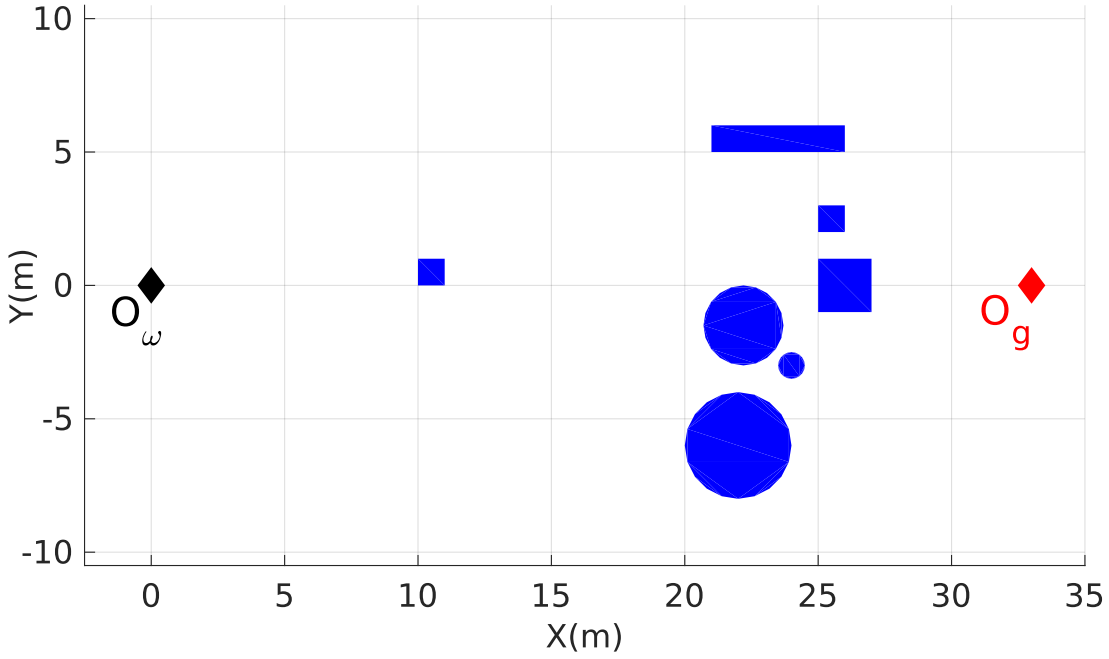


Figure 2.8: Schéma de l'environnement de simulation

à représenter un espace agricole typique : vaste avec des obstacles de grande taille de l'ordre de plusieurs mètres.

2.4.2 Configuration du robot

Le robot démarre d'une position initiale $O_r(0) = O_w$ et doit atteindre un but O_g dont les coordonnées en mètres dans le repère initial du robot sont $[x_g, y_g] = [33, 0]$. Entre le robot et son objectif se trouvent plusieurs obstacles, dont notamment un petit obstacle ponctuel, et un regroupement de plusieurs obstacles.

Ce robot est doté d'un contrôleur ω_R chargé de l'évitement en spirale (*Contrôleur SA*), dont la définition peut se trouver équation (2.44), et d'un *contrôleur GTG* proportionnel de gain unitaire. Le robot évolue à une vitesse linéaire fixe $v^* = 0.5m.s^{-1}$. Les gains des deux contrôleurs ω_A (cf. équation (2.42)) et ω_B (cf. équation (2.43)) sont fixés à $\lambda_S = 0.5$, $\lambda_1 = 0.5$ et $\lambda_2 = 0.5$. De plus, le paramètre n utilisé pour le contrôleur w_B est fixé à $n = 5$.

Pour l'enchaînement entre les deux contrôleurs, le seuil e_α^{thresh} est fixé à $e_\alpha^{thresh} = \pi/12$. Pour les paramètres des spirales, l'angle désiré est donc fixé à $\alpha^* = \pm\frac{\pi}{2}$, tandis que la distance désirée est réglée à $d^* = 3m$. Afin de modéliser au mieux la plateforme d'expérimentations, le capteur LiDAR est configuré avec des paramètres semblables à un capteur LiDAR réel : une résolution angulaire de 0.25° à 360°, et un bruit uniforme de $30mm$ est appliqué sur les données LiDARs. Pour finir, le contrôleur GTG utilisé pour cette simulation est un correcteur proportionnel corrigeant le cap du robot pour l'aligner sur la direction du but.

2.4.3 Analyse de la trajectoire du robot, des erreurs et des commandes

La figure 2.9 montre la trajectoire effectuée par le robot durant l'évitement. La partie noire correspond au moment où le contrôleur GTG est actif. La figure 2.10 montre les erreurs $e_d(t)$ et $e_\alpha(t)$. La figure 2.11 montre l'évolution des vitesses linéaires et angulaires et la figure 2.12 la loi de commande utilisée. La figure 2.13 représente la situation du robot et de l'environnement aux instants précis des commutations entre le contrôleur SA et le contrôleur GTG, et notamment la position des barycentres O_b. Nous voyons donc que le robot parvient correctement à atteindre le but O_g en approximativement $100s$.

Comme le montre la figure 2.9, le robot commence à la pose $[0, 0, 0]$, et le contrôleur GTG conduit le robot vers son but O_g. À une distance de $6m$ du premier obstacle, les conditions de déclenchement 3.11 sont activées.

À cet instant, selon la figure 2.13(a), un sens de contournement anti-horaire est sélectionné, car nous avons $\alpha_b > \alpha_g$. L'évitement débute, et le contrôleur ω_B est utilisé car $e_\alpha(t) \simeq -\pi/2$. Ainsi, les deux erreurs $e_d(t)$ et $e_\alpha(t)$ convergent rapidement vers zéro.

À $t \simeq 19s$, l'erreur $e_\alpha(t)$ est suffisamment faible pour utiliser le contrôleur ω_A.

À $t \simeq 23s$, l'évitement est terminé, et le contrôleur GTG reprend la main pour mener le robot vers le but.

À $t \simeq 34s$, l'évitement est de nouveau déclenché. Comme le montre la

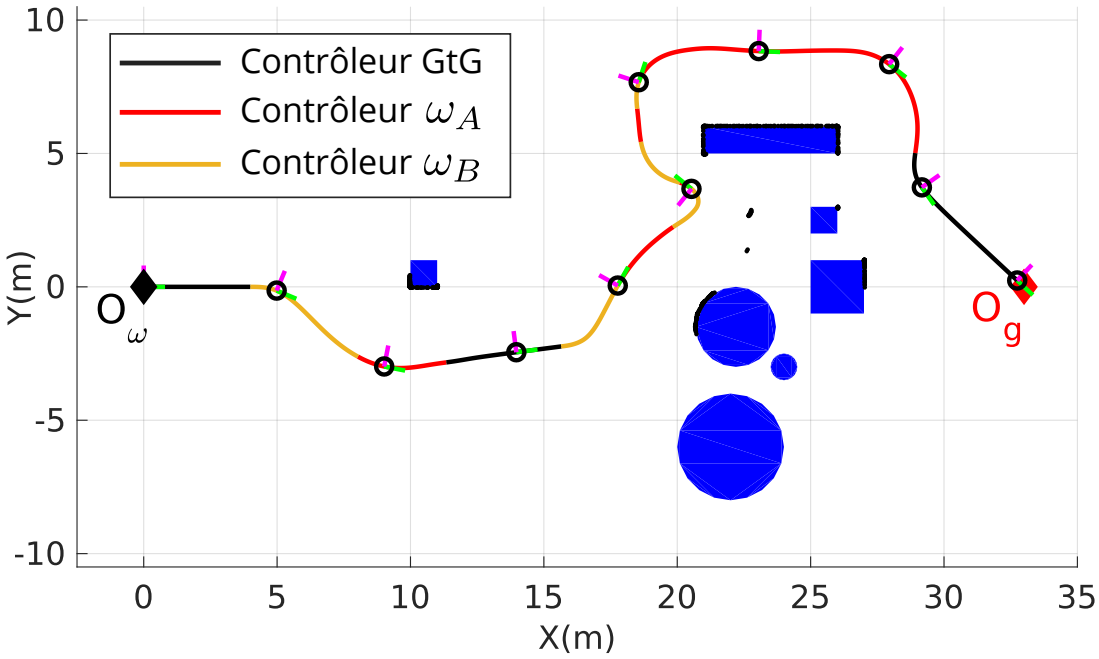


Figure 2.9: Trajectoire du robot en fonction des différents contrôleurs.

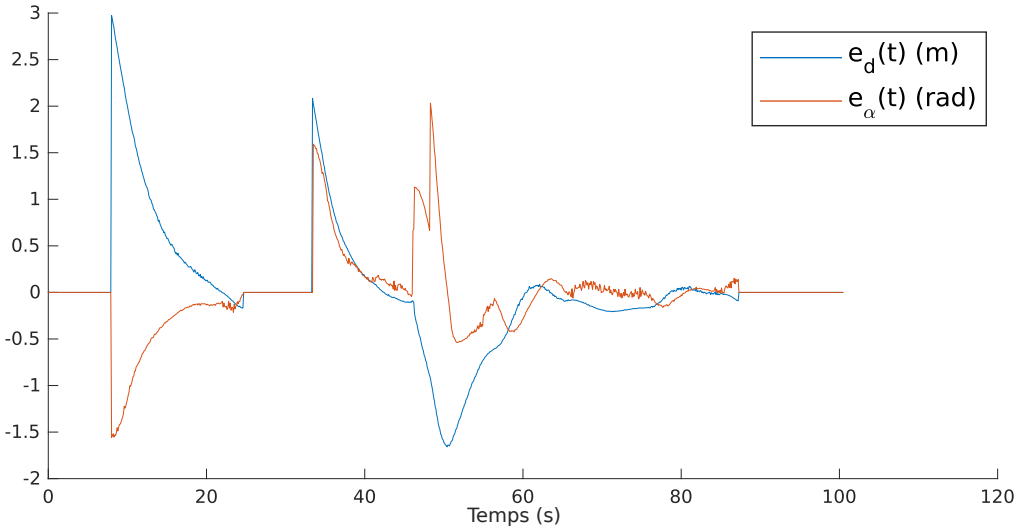


Figure 2.10: Erreur en angle $e_\alpha(t)$ et erreur en distance $e_d(t)$

figure 2.13(b), le barycentre O_b se situe *à droite* du segment joignant le robot à son but, et un sens d'évitement horaire (par la gauche) est choisi. Le robot commence donc à éviter le groupement d'obstacles, dont la dimension est légèrement inférieure à $2d^* = 6m$. Nous observons sur la figure 2.12 quelques commutations entre les contrôleurs ω_A et ω_B, suivant les irrégularités du groupement d'obstacles.

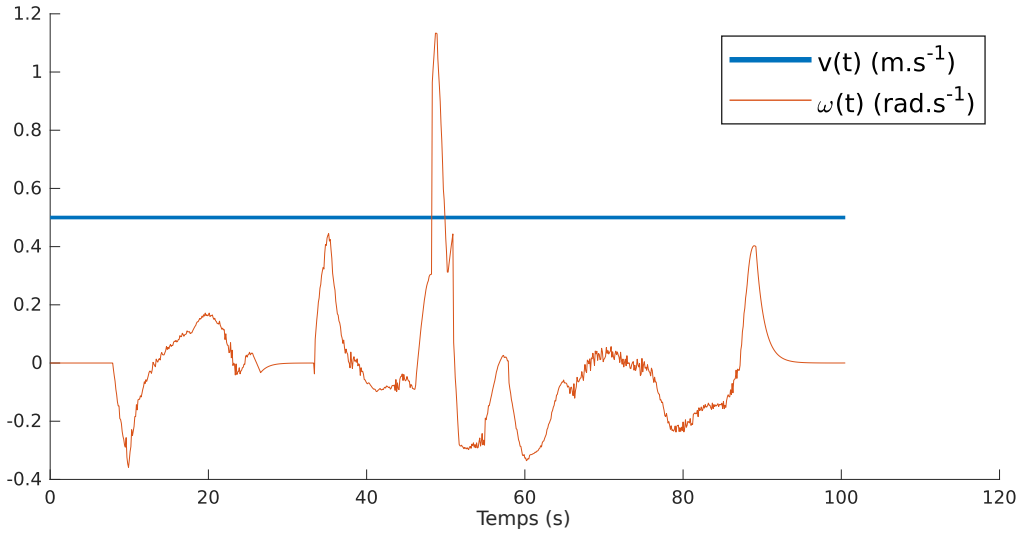


Figure 2.11: Commandes en vitesse linéaire et angulaire envoyées au robot

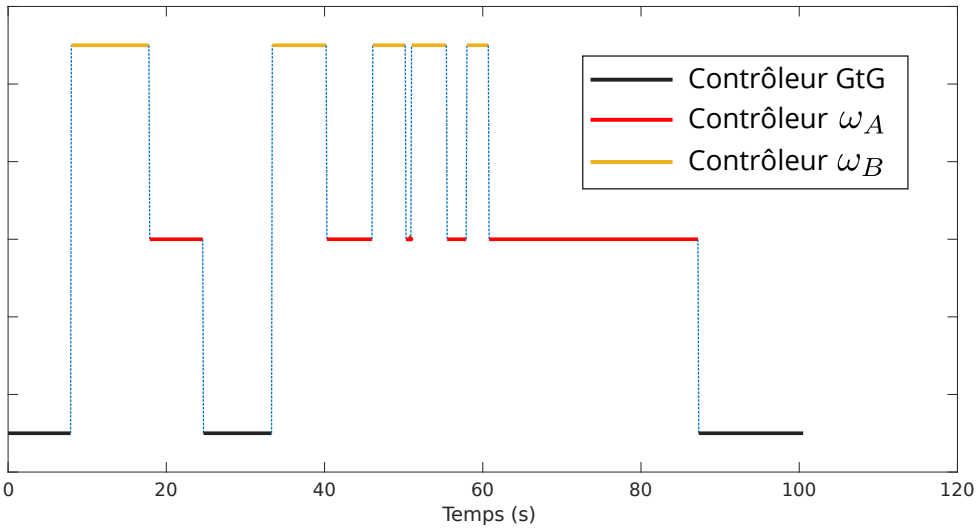


Figure 2.12: Utilisation des contrôleurs

À $t \simeq 50s$, le pic sur les erreurs $e_d(t)$ et $e_\alpha(t)$ correspond à l'évitement de la concavité. Sur la figure 2.9, les SCP successifs sont modélisés par des points noirs. L'intérêt d'utiliser le barycentre apparaît lors de l'évitement de ce groupement d'obstacles. En effet, le barycentre est sélectionné comme SCP sur plusieurs itérations, permettant au robot de ne pas s'engager dans les interstices présents dans le groupement d'obstacles, entre lesquels il ne peut pas passer tout en assurant une distance de sécurité suffisante.

À $t \simeq 88s$, l'évitement est terminé, et le contrôleur GTG reprend le contrôle afin

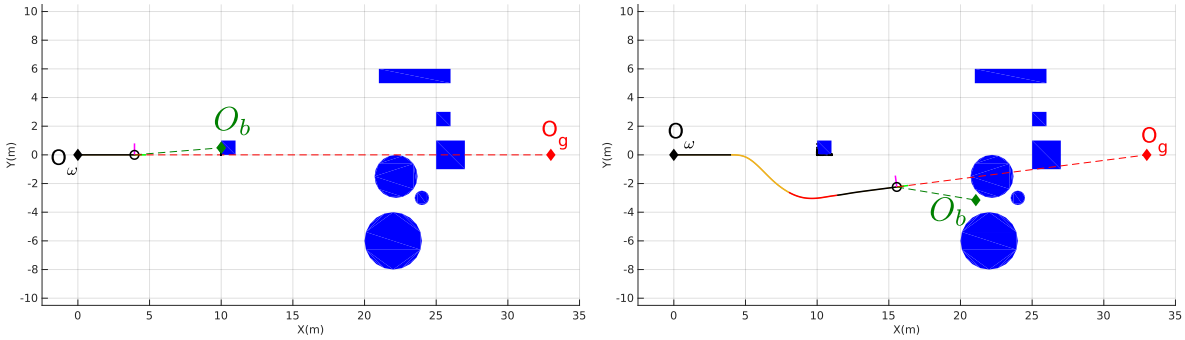


Figure 2.13: Position du barycentre et du but lors du calcul du SOM

de mener le robot à son but, qu'il atteint finalement à $t \simeq 100s$.

Nous voyons donc que notre méthode permet avec succès de mener le robot à un objectif donné. L'utilisation possible du barycentre O_b en tant que SCP permet une forte réactivité du robot dans le cas de groupement d'obstacles de forme globalement concave. Cependant, cette forte réactivité se traduit par des pics ponctuels dans les erreurs $e_\alpha(t)$ et $e_d(t)$, liés aux sauts des SCP successifs, qui se repercutent dans la sortie de commande angulaire résultante.

2.5 Évolution de la méthode dans le calcul du centre de la spirale

L'étude précédente a montré que l'utilisation du barycentre O_b en tant que SCP permettait d'éviter divers types d'obstacle, et notamment des obstacles concaves. Cependant, l'évolution du SCP présente tout de même de fortes discontinuités suivant les obstacles. Ces discontinuités entraînent des pics dans les erreurs en angle et en distance (cf. figure 2.10), et finalement aussi sur les commandes calculées (cf. figure 2.11). Ainsi, une évolution de la méthode précédente a été développée pour assurer une évolution du SCP le long des obstacles sans saut brusque. À l'instar de l'algorithme précédent, nous utilisons le calcul du point LiDAR le plus proche O_c, ainsi que la prise en compte des points situés à une distance de $2d^*$ de O_c.

2.5.1 Deuxième méthode de calcul du centre de la spirale

La méthode proposée pour calculer le centre de la spirale repose sur une liste P de points acquis par le robot à l'instant t donnés dans le repère cartésien attaché au robot. Les étapes suivantes peuvent se visualiser sur la figure 2.14 :

- Étape 1 : Après l'acquisition des points P autour du robot, le point le plus proche du robot en norme euclidienne est calculé. Ce point est dénommé O_c par

Algorithme 3 Calcul de O_p

Entrées : P, $O_c \begin{pmatrix} x_{O_c} \\ y_{O_c} \end{pmatrix}$, d^*
Sortie : O_p
$d_{min} \leftarrow +\infty$
$n \leftarrow 0$
$P_{finals} \leftarrow []$
$O_p \leftarrow []$
pour chaque point Pt dans P **faire**
  $d \leftarrow \text{distance}(Pt,\ O_c)$
  $H \leftarrow []$
  si $d < 2d^*$ **alors**
    $\overrightarrow{v} \begin{pmatrix} x_v \\ y_v \end{pmatrix} = \frac{\overrightarrow{O_cPt}}{||O_cPt||}$
    $\overline{O_cH} = \frac{-x_{O_c}x_v - y_{O_c}y_v}{\sqrt{x_v^2+y_v^2}}$
    $\begin{cases} x_H = x_{O_c} + \frac{\overline{O_cH}}{\sqrt{x_v^2+y_v^2}} x_v \\ y_H = y_{O_c} + \frac{\overline{O_cH}}{\sqrt{x_v^2+y_v^2}} y_v \end{cases}$
    $P_{finals} = [P_{finals}\ \ H]$
  fin si
fin pour chaque
pour chaque Point Pt dans P_{finals} **faire**
  $d \leftarrow \text{distance}(Pt,\ P)$
  si $d < d_{min}$ **alors**
    $d_{min} \leftarrow d$
    $O_p \leftarrow Pt$
  fin si
fin pour chaque

la suite. Le barycentre O_b est aussi calculé selon l'algorithme 2.

- Étape 2 : Pour chaque point P^i se situant à l'intérieur du cercle de rayon $2d^*$ centré sur O_c, O_p^i, le projeté orthogonal de la position O_r du robot sur la droite (O_cP^i) est calculé. Si le projeté O_p^i se trouve en dehors du segment $[O_cP^i]$, ce dernier est pris égal à O_c.

- Étape 3 : Parmi ces projetés orthogonaux, le plus proche du robot en norme euclidienne est sélectionné, et finalement nommé O_p.

- Étape 4 : Le point le plus proche du robot parmi O_c, O_p, et O_b est sélectionné

pour être le centre de la spirale O_s.

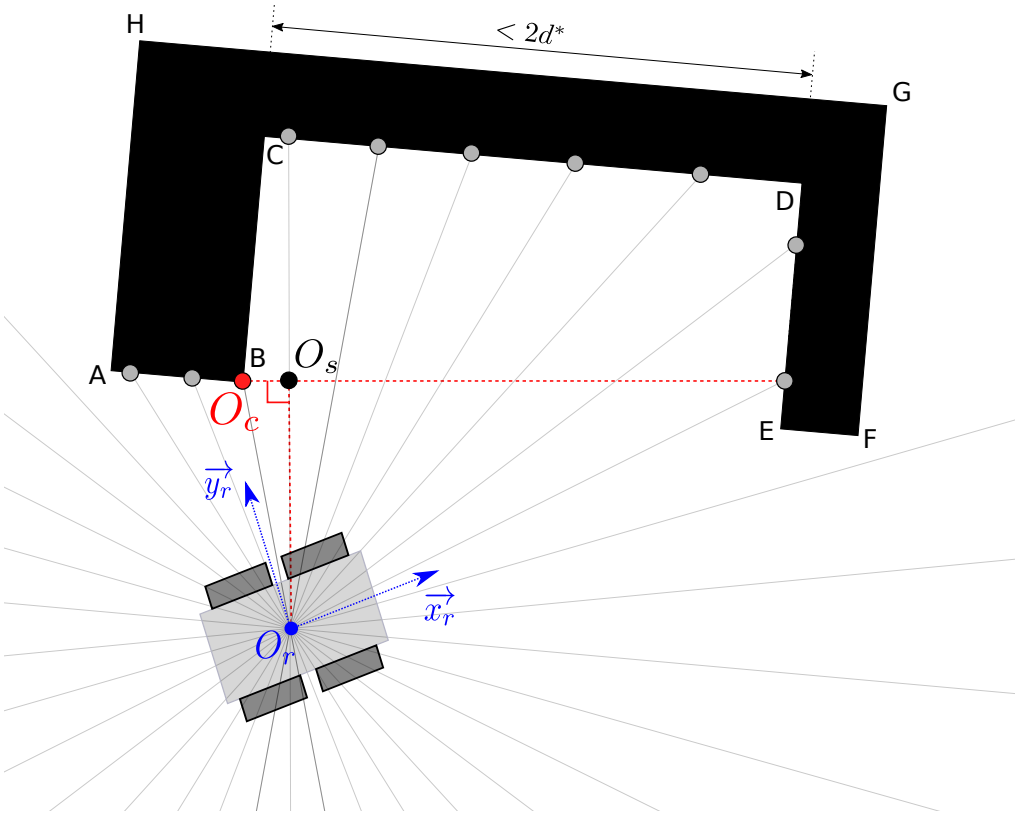


Figure 2.14: Calcul du SCP

La figure 2.14 expose les différentes étapes du fonctionnement de la méthode dans le cas d'un obstacle concave. L'obstacle est un polygone $ABCDEFGH$ avec un creux d'une largeur inférieure à $2d^*$ (par conséquent, $BE < 2d^*$). Ainsi la première étape consiste à trouver O_c le point le plus proche du robot, représenté en rouge sur la figure. À l'issue de cette étape, pour les étapes 2 et 3, le projeté le plus proche du robot se trouve sur le segment liant O_c au point situé à l'autre extrémité de la concavité, proche du point E. Ainsi, lorsque le robot évite l'obstacle, le point le plus proche O_c reste dans un premier temps presque confondu avec le point B, et le SCP O_s, qui est donc toujours choisi comme étant le projeté sur ce segment, suit le segment $[BE]$.

Ainsi, du point de vue de la méthode de suivi de spirale, qui ne nécessite que l'information O_s, cette dernière se comporte pour cet obstacle concave comme pour un obstacle convexe semblable au polygone $AFGH$. La figure 2.15 présente plusieurs cas mettant en exergue l'intérêt d'une telle méthode dans les cas concaves. Cette figure détaille l'évolution du SCP autour de divers obstacles en fonction de la position du robot. Les figures *a.*, *b.* et *c.* montrent ainsi respectivement un obstacle carré avec un

côté vide, un coin intérieur, et un ensemble de petits obstacles. Dans les trois cas, la *taille* des concavités (c'est-à-dire la largeur du creux pour *a.*, la distance entre les deux extrémités pour *b.*, et la distance entre chacun des petits obstacles dans *c.*) est inférieure à $2d^*$. Dans chacun de ces exemples, nous remarquons donc que le SCP suit l'enveloppe convexe de chacun des obstacles et n'effectue aucun saut brusque.

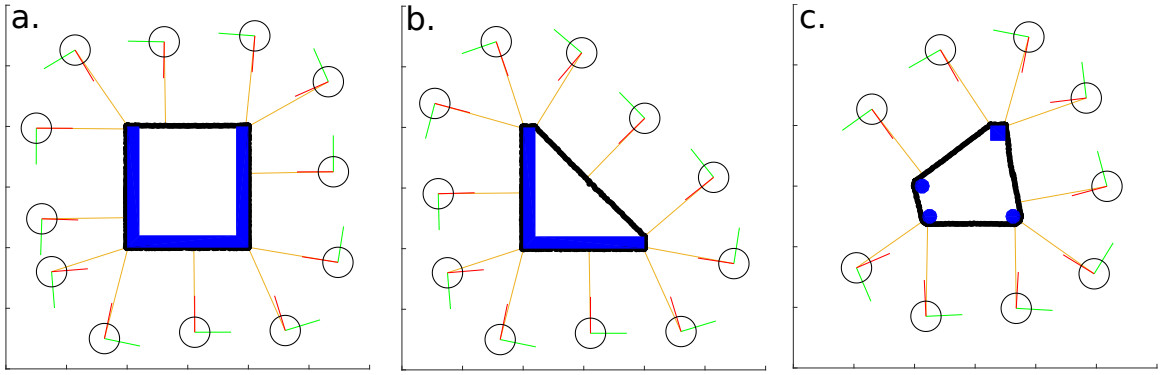


Figure 2.15: Évolution de la position du SCP (points noirs) lors d'un tour complet de l'obstacle (en bleu) effectué par le robot

Stratégie de navigation

La stratégie de navigation reste semblable à celle présentée dans la section 2.3. Pour les conditions de commutation, une troisième condition est ajoutée similaire à celles présentées en 2.46 impliquant O_c et O_b :

$$(3) \begin{cases} d_p < d_p^{tr} \text{ et} \\ \mid \alpha_g - \alpha_p \mid < \pi/2 \end{cases} \tag{2.49}$$

avec d_p, α_p et d_p^{tr} définis de façon similaire à 2.45 et 3.11 :

$$\alpha_p = (\overrightarrow{x_r}, \overrightarrow{O_r O_p}) \qquad d_p = ||\overrightarrow{O_r O_p}|| \tag{2.50}$$

$$d_p^{tr} = \begin{cases} d^* + d^*(1 - \frac{|\alpha_p|}{\pi/2}) & \text{si contrôleur GTG actif} \\ 2d^* & \text{si contrôleur SA actif} \end{cases} \tag{2.51}$$

Ainsi, le contrôleur d'évitement en spirale est actif si au moins une des conditions (1), (2) ou (3) est vraie.

2.5.2 Simulation

L'environnement ainsi que les paramètres sont les mêmes que ceux présentés dans la sous-section 2.4. Le seul ajout concerne le calcul de O_p, et sa prise en compte dans

le processus de navigation. Le robot démarre d'une position initiale $O_r(0) = O_w$ et doit atteindre un but O_g dont les coordonnées dans le repère initial du robot sont $[x_g, y_g] = [33, 0]$. Le robot est doté d'un contrôleur ω_R chargé de l'évitement en spirale (*Contrôleur SA*), dont la définition peut se trouver équation (2.44), et d'un *contrôleur GTG* proportionnel de gain unitaire. Le robot évolue à une vitesse linéaire fixe $v^* = 0.5m.s^{-1}$. Les gains des deux contrôleurs ω_A (cf. équation (2.42)) et ω_B (cf. équation (2.43)) sont fixés à $\lambda_S = 0.5$, $\lambda_1 = 0.5$ et $\lambda_2 = 0.5$. De plus, le paramètre n utilisé pour le contrôleur w_B est fixé à $n = 5$. Pour l'enchaînement entre les deux contrôleurs, le seuil e_α^{thresh} est fixé à $e_\alpha^{thresh} = \pi/12$. Pour les paramètres des spirales, l'angle désiré est donc fixé à $\alpha^* = \pm\frac{\pi}{2}$, tandis que la distance désirée est réglée à $d^* = 3m$. Afin de modéliser au mieux la plateforme d'expérimentations, le capteur LiDAR est configuré avec des paramètres semblables à un capteur LiDAR réel : une résolution angulaire de 0.25° à 360°, et un bruit uniforme de $30mm$ est appliqué sur les données LiDAR. Pour finir, le contrôleur GTG utilisé pour cette simulation est un correcteur proportionnel corrigeant le cap du robot pour l'aligner sur la direction du but.

La figure 2.16 montre la trajectoire effectuée par le robot durant l'évitement. Sur cette figure a aussi été ajoutée en gris clair la trajectoire effectuée lors de la simulation précédente, à titre de comparaison. La figure 2.17 montre les erreurs $e_d(t)$ et $e_\alpha(t)$, la figure 2.18 l'évolution des vitesses linéaires et angulaires et la figure 2.19 la loi de commande utilisée.

Comme le montre la figure 2.16, le robot parvient correctement à atteindre son objectif O_g en approximativement $100s$.

Nous remarquons que jusqu'à l'instant $t \simeq 34s$, le comportement et la trajectoire du robot ne changent pas par rapport à la simulation précédente. En effet, le premier obstacle étant convexe, l'ajout du calcul du point O_p ne change rien à la situation.

En revanche, des différences lors de l'évitement du groupement d'obstacles sont observées. En effet, lors de l'évitement de la partie concave, le point projeté O_p est sélectionné la plupart du temps en tant que SCP, et ce dès le déclenchement de l'évitement à $t \simeq 34s$. Le SCP épouse donc l'enveloppe convexe du groupement d'obstacles. Il est intéressant de comparer le graphe des erreurs 2.17 et le graphe des vitesses envoyées 2.18 avec les mêmes graphes issus de la précédente simulation (respectivement 2.10 et 2.11). Nous remarquons notamment une erreur plus faible en distance et en angle, car le robot anticipe davantage les irrégularités de l'obstacle. Par conséquence, la vitesse angulaire maximum en absolu $\omega(t)$ envoyée n'est plus que d'environ $0.4rad.s^{-1}$, alors qu'elle atteignait presque $1.2rad.s^{-1}$ dans la simulation précédente. De plus, nous remarquons sur la figure 2.19 que le contrôleur ω_A est davantage utilisé, car $e_\alpha(t)$ s'approche moins des zones de singularités de ce dernier.

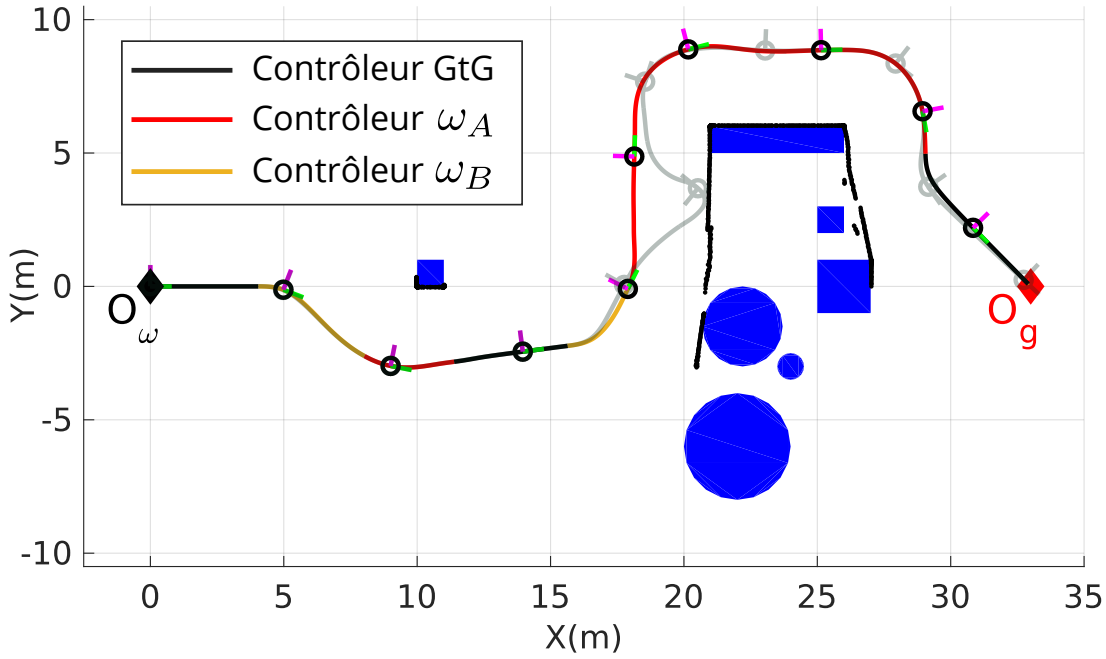


Figure 2.16: Trajectoire finale à la fin de la simulation

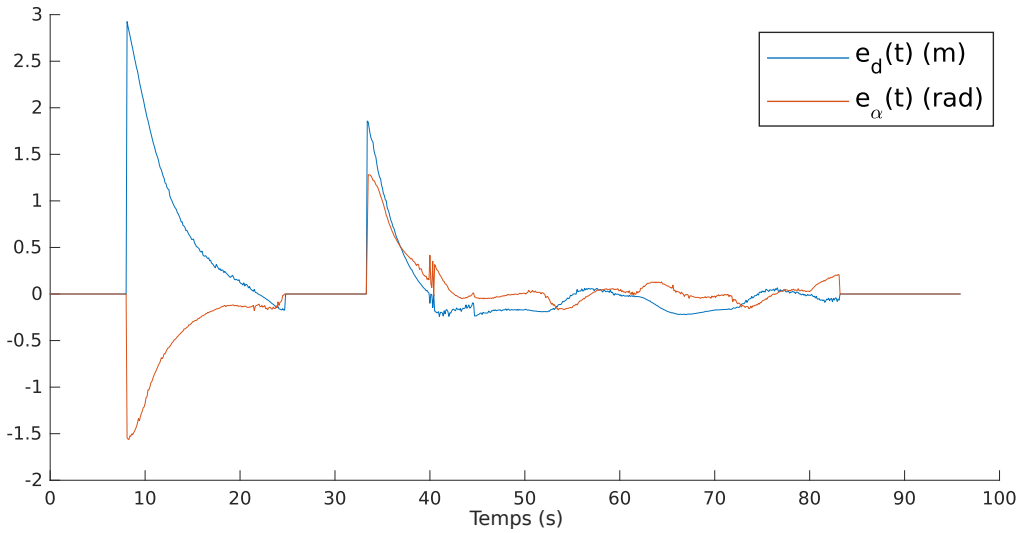


Figure 2.17: Erreur en angle $e_\alpha(t)$ et erreur en distance $e_d(t)$

2.6 Conclusion

Cette section a traité le problème de l'évitement d'obstacles statiques. La méthode de navigation proposée permet de traiter une grande variété d'obstacles et elle est suffisamment efficace pour garantir la non-collision et éviter des problèmes classiques

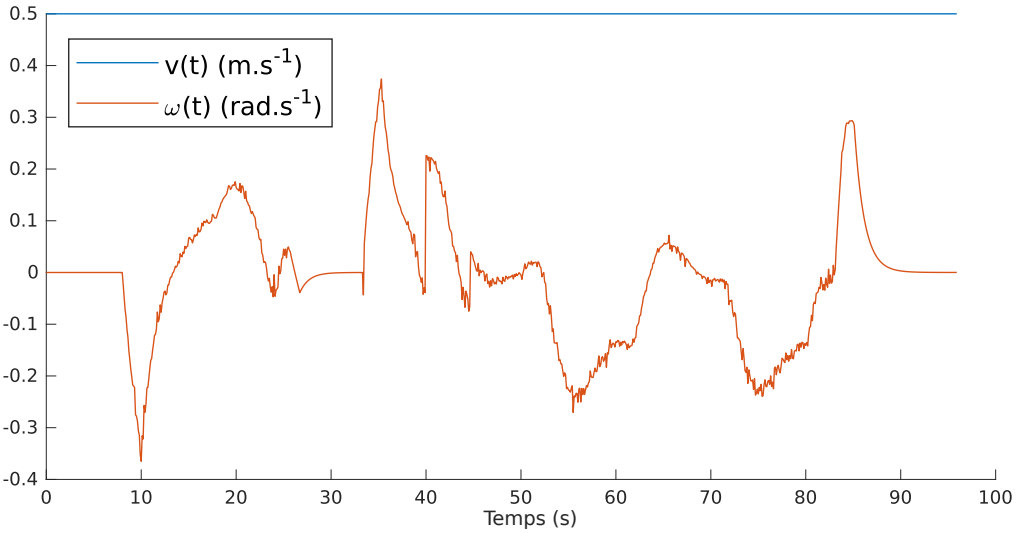


Figure 2.18: Commandes en vitesse linéaire et angulaire envoyées au robot

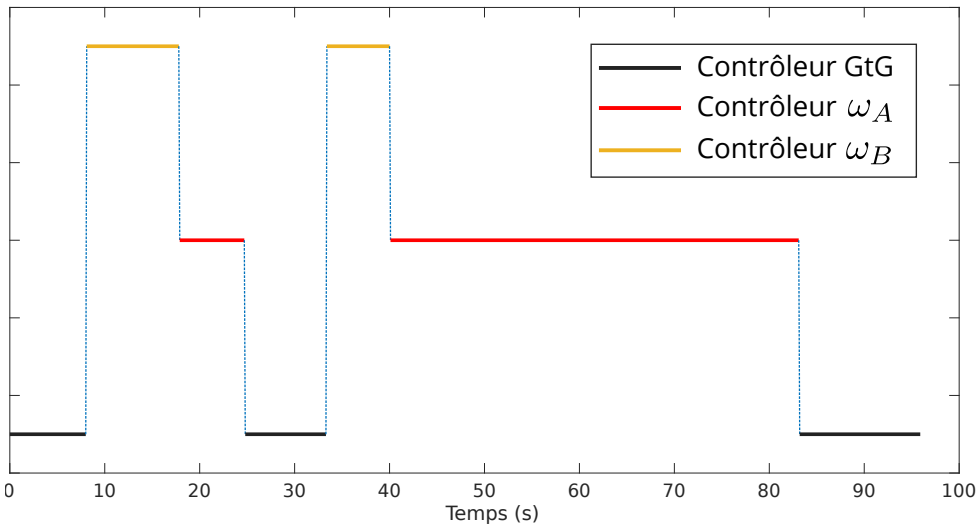


Figure 2.19: Utilisation des contrôleurs

tels que les minima locaux, les singularités, etc. Cette stratégie prolonge les travaux précédents ([Futterlieb, 2017], [Leca et al., 2019]), et consiste à définir et à suivre des spirales appropriées autour des obstacles rencontrés au cours de la mission. Deux contributions principales sont au cœur de cette section : (i) la mise à jour continue des paramètres de la spirale en définissant un barycentre approprié capable de garantir une trajectoire et des entrées de contrôle sans discontinuités, ou en définissant des projetés successifs matérialisant l'enveloppe convexe des obstacles ; et (ii) le couplage de deux contrôleurs référencés capteurs extéroceptifs permettant de suivre une spirale tout en évitant les singularités. La stratégie de navigation est alors basée sur l'enchaînement

de contrôleur d'évitement basé spirale et de contrôleur Go-To-Goal dirigeant le robot vers son objectif. Cet enchaînement est régi par des conditions d'activation adéquates. Cette approche a été implémentée et validée en simulation, et des tests en conditions réelles seront exposés dans le chapitre 4. Cependant, l'approche proposée reste limitée aux objets statiques ou semi-statiques imprévus. Par conséquent, notre prochaine étape est de l'étendre aux obstacles dynamiques.

Chapitre 3

Navigation basée capteurs en environnement dynamique

Le chapitre précédent a détaillé la stratégie de navigation dans un environnement agricole fortement encombré de nombreux obstacles statiques (bâtiments, zones de stockage, etc). L'objectif de ce chapitre est de montrer que cette stratégie de navigation peut s'adapter à des environnements agricoles dynamiques dans lesquels sont aussi présents de nombreux obstacles mobiles (voitures, machines agricoles, opérateurs humains, animaux, etc.).

L'approche générale consiste à utiliser notre méthode présentée dans le chapitre précédent afin de profiter de son caractère générique. La stratégie de navigation et les lois de commande précédemment introduites restent donc quasiment identiques. Cependant, nous allons montrer que modifier en ligne les différents paramètres des spirales générées en fonction des acquisitions capteurs va permettre une prise en compte pertinente des obstacles mobiles. Le calcul des paramètres des spirales, tels que le centre O_s ou les distances et angles désirés d^* et α^*, basé directement sur les données LiDAR sera adapté en fonction de la scène détectée dans l'environnement du robot. En effet, la méthode de calcul des paramètres s'adaptera à cette scène selon si cette dernière présente des obstacles statiques ou dynamiques. Nous obtenons ainsi une stratégie de navigation générique référencée capteurs qui conserve la même structure et qui est capable de s'adapter à un environnement avec des obstacles statiques et mobiles de diverses tailles, formes, vitesses..., etc.

Ce chapitre est donc organisé autour de trois sections. La première section présente les problématiques liées aux environnements agricoles dynamiques et introduit nos deux principales contributions pour ce chapitre. Les sections deux et trois vont respectivement détailler ces deux contributions, en récapitulant les étapes de perception et d'action. Pour chacune de ces sections, une validation de la méthode est proposée en environnement dynamique simulé.

3.1 Problématiques liées aux environnements agricoles dynamiques et approches proposées

3.1.1 Description de l'environnement agricole dynamique

Comme cela a été présenté dans l'introduction du manuscrit, un environnement agricole typique contient divers éléments mobiles sans information a priori : le robot ne possède au préalable aucune connaissance à propos de la taille des obstacles, leur géométrie, leur déplacement (trajectoire, vitesse, accélération) ou le nombre à gérer en simultané. Dans un environnement agricole typique, ces obstacles mobiles peuvent ainsi être :

- des opérateurs humains dont le déplacement est quelconque. Une vitesse de marche standard est de l'ordre du mètre par seconde (entre 2.5 km/h et 5 km/h). Ces opérateurs ne sont pas en interaction avec le robot, mais peuvent tout de même avoir conscience de sa présence et anticiper ses déplacements.

- Des véhicules, agricoles ou classiques qui peuvent être des voitures, des camions, mais aussi des engins agricoles, tels que des tracteurs. Ces véhicules peuvent se déplacer au pas, mais aussi à grande vitesse (plus de 50 km/h).

- Des animaux qui errent librement sur les zones de manœuvres. Ces animaux peuvent être du bétail, des animaux sauvages, ou bien des animaux de compagnie. Ces derniers peuvent avoir un comportement imprévisible à proximité du robot.

Nous constatons que dans une majorité des cas, la vitesse du robot est inférieure aux vitesses des éventuels obstacles mobiles présents dans l'environnement (humains, voitures...). De plus, la plupart des éléments mobiles seront aussi capables de voir et de détecter le robot. Ainsi, nous pouvons faire les hypothèses suivantes :

- Les obstacles mobiles autour du robot ne font que circuler, et ne vont donc pas rester à proximité du robot.

- Les obstacles mobiles ne vont pas volontairement chercher à interférer avec le robot et sa trajectoire. De plus, à l'opposé des environnements urbains, un environnement agricole est non-structuré, et aucune supposition ne peut être faite quant aux zones à l'intérieur desquelles pourront évoluer les obstacles mobiles. Par exemple, il n'est pas possible de faire l'hypothèse que les humains vont n'avoir accès qu'à un espace limité, et les machines à un autre. Tous les types d'obstacles peuvent cohabiter dans la même zone. Ainsi, un robot mobile évoluant au sein d'un tel environnement devra être doté de capacités de perception, afin de détecter les éléments en mouvement, et d'action, afin d'anticiper et de gérer en toute sécurité ces obstacles.

3.1.2 Les approches proposées

Les approches proposées reposent sur deux phases classiques en robotique qui sont la perception et l'action.

La phase de perception détecte, quantifie et qualifie la scène environnant le robot. Elle classe et compte les obstacles statiques et mobiles et quantifie ces derniers à partir des capteurs disponibles sur le robot, le plus souvent des LiDARs. Nos méthodes se basent sur la connaissance de manière locale des déplacements du robot. La première étape consiste à classer les obstacles statiques et mobiles en comparant ce que le robot *voit* à l'instant t avec ce qu'il *devrait voir* par rapport aux données obtenues à l'instant $t-1$ précédent. La comparaison de ces données géométriques datées issues des capteurs permet de séparer les points statiques et mobiles. La deuxième étape est une segmentation effectuée parmi cet ensemble de points détectés comme mobiles, afin de séparer les différents obstacles. En effet, afin d'effectuer un évitement d'obstacle dynamique efficace, il sera nécessaire de traiter les données laser afin de segmenter individuellement les obstacles et de les suivre au cours du temps. Diverses approches existent dans la littérature. Pour la segmentation, une approche basée sur la distance entre les points est la plus utilisée, par exemple par [Dominguez et al., 2011] ou [Peng et al., 2015]. Quant au suivi de la trajectoire des obstacles, les méthodes les plus utilisées sont basées sur des filtres de Kalman étendus ou des filtres à particules [Thuy and Puente Leon, 2009], [Fortin et al., 2012]. Cependant, ces méthodes de suivi impliquent généralement une connaissance de la nature des obstacles et de leur modèle cinématique. À l'issue de cette phase de perception qui repose sur des étapes de détection, quantification et classification des obstacles, le type de scène autour du robot est considéré connu : nombre d'obstacles statiques ou dynamiques, ainsi que la dynamique des obstacles mobiles.

À l'aide des informations déduites de la scène, la phase suivante est le choix d'actions adaptées afin de modifier les consignes envoyées au robot pour sa tâche de navigation. Comme expliqué précédemment, l'approche est d'adapter la tâche de navigation en environnement statique présentée dans le chapitre 2 aux obstacles mobiles par des modifications en ligne des paramètres des spirales. Plus précisément, la spirale d'évitement est définie par trois paramètres : le centre de la spirale O_s, la distance désirée d^* et l'angle désiré α^*. Nous proposons deux méthodes différentes pour modifier en ligne ces paramètres et pour générer des consignes d'évitement en spirale capables de considérer les différentes caractéristiques des obstacles mobiles.

Dans la première méthode, en environnement faiblement encombré, un comportement sécuritaire du robot est privilégié en garantissant la distance robot-obstacle à partir de l'observation de l'évolution du point détecté le plus proche du robot O_c au cours du temps. À l'aide de cette information, il est alors possible de moduler la distance désirée d^* correspondant à la distance désirée entre le robot et le centre

de la spirale (SCP). Cette distance peut être augmentée lorsqu'un obstacle détecté s'approche du robot et donc représente un danger, ou être bien être diminuée si l'obstacle détecté s'éloigne du robot. Nous proposons de formaliser cette stratégie par la méthode appelée méthode *Distance-Angle Adaptatif*. Elle repose sur le principe du seuil adaptatif, introduit par [Emami-Naeini et al., 1988] et [Ding and Frank, 1991] pour détecter les défaillances de systèmes en se basant sur l'étude statistique d'un signal résiduel, comme par exemple dans [Wang et al., 2003] ou [Shi et al., 2005]. Dans [Shi et al., 2005], un seuil adaptatif est calculé à partir de la moyenne et de la variance du signal résiduel pour détecter les défauts dans une installation électro-hydraulique non linéaire. Nous appelons signal résiduel (ou valeur résiduelle) d'une variable la différence entre la valeur effectivement prise par cette variable et sa valeur prédite à partir d'observations passées. En suivant un raisonnement similaire, nous proposons de suivre la valeur résiduelle de la distance et de l'angle entre le robot et le point le plus proche de l'obstacle détecté. Le calcul de la moyenne et de la variance de ces valeurs résiduelles permettent de calculer la *Distance Adaptative* et l'*Angle Adaptatif*, qui donnent le nom à la méthode *Distance-Angle Adaptatifs*. Ces deux valeurs offrent une information utile sur le mouvement de l'obstacle, notamment si celui-ci se rapproche ou s'éloigne du robot, ou bien si l'obstacle se déplace de la gauche vers la droite du robot, ou inversement. Ces données vont permettre de calculer en ligne la distance désirée à respecter entre le robot et l'obstacle et le sens de contournement de ce dernier. Cette méthode a l'avantage de ne nécessiter l'observation que d'un seul point : O_c. Cependant, elle ne prend en compte que la dynamique de l'obstacle le plus proche. Elle sera donc particulièrement adaptée aux cas où peu d'obstacles évoluent dans l'environnement.

La seconde méthode, présentée dans [Leca et al., 2019], est basée sur l'analyse non plus du seul point le plus proche O_c, mais de l'ensemble des points acquis par les capteurs. La comparaison de deux acquisitions proches dans le temps permet de détecter et de séparer les points appartenant aux différents éléments mobiles éventuellement présents. L'application de méthodes de regroupement permet de classifier ces points en différents groupes, correspondant chacun à un obstacle distinct. Il peut ensuite être déduit de l'analyse de ces groupes les caractéristiques des éléments mobiles présents dans l'environnement du robot. Afin de prendre en compte ces obstacles mobiles, nous introduisons le concept d'*Enhanced Laser Scan* (ELS). Cet ELS est formé de l'acquisition LiDAR courante et de points LiDAR virtuels afin de modéliser la future trajectoire de chacun des obstacles en fonction des informations déduites de la comparaison des deux acquisitions. Cet ELS composé de points détectés et virtuels sert d'entrée à l'algorithme de navigation qui en extrait notamment les nouveaux paramètres d'évitement comme le centre de la spirale ou le sens de contournement. Cette méthode permet de considérer des obstacles mobiles se dirigeant vers le robot même si ces derniers ne sont pas encore les plus proches du robot. Le robot peut ainsi anticiper des obstacles mobiles et mieux réagir.

3.2 Description de la stratégie de navigation basée sur la méthode *Distance-Angle Adaptatifs*

Comme indiqué précédemment, la méthode *Distance-Angle Adaptatifs* repose sur deux étapes.

La première est le calcul des valeurs résiduelles [Shi et al., 2005] des variables d_c et α_c, correspondant respectivement d'une part à la distance entre le robot et le point O_c, et d'autre part à l'angle entre l'axe du robot et le point O_c. En effet, comme l'a montré le chapitre 2, parmi les points remarquables extraits de l'environnement, le point O_c représente le point détecté le plus proche du robot. Ainsi, son évolution représente un bon indicateur pour déduire la dynamique de l'obstacle qu'il représente. Il s'agit, à chaque itération, de comparer la distance d_c avec une valeur de distance prédite à partir d'une acquisition précédente et d'une connaissance locale du déplacement du robot. La valeur résiduelle correspond alors à la différence entre la distance d_c mesurée et celle prédite. Cette valeur résiduelle est calculée sur une fenêtre glissante, afin de minimiser les incertitudes liées aux formes de l'obstacle et à l'incertitude sur l'odométrie.

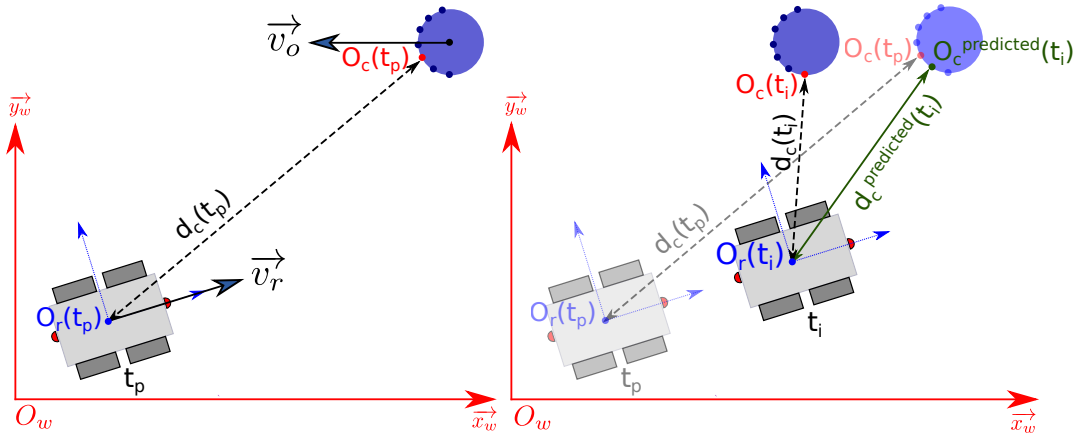
La seconde étape est la détermination de la distance adaptative entre le robot et l'obstacle et plus précisément le point O_c à partir de la valeur résiduelle calculée dans la première étape. Cette distance adaptative permet de calculer la distance à respecter entre le robot et le point le plus proche. Cette deuxième étape comporte aussi le calcul de l'angle adaptatif à partir de la valeur résiduelle de l'angle α_c, qui permet de définir de nouvelles conditions pour choisir le sens de contournement de l'obstacle.

Finalement, les valeurs de la distance désirée et le sens de contournement sont calculés en ligne à chaque itération et introduites dans la méthode de navigation basée spirale présentée chapitre 2. Cela offre la possibilité au robot d'éviter efficacement les obstacles mobiles en environnement faiblement encombré.

3.2.1 Calcul des valeurs résiduelles de la distance et l'angle entre le robot et le point de l'obstacle le plus proche

Le calcul des valeurs résiduelles de la distance d_c et de l'angle α_c entre le robot et le point de l'obstacle le plus proche O_c repose sur deux étapes.

La première étape consiste à prédire la distance $d_c^{predicted}$ et l'angle $\alpha_c^{predicted}$ entre le robot et l'obstacle à l'instant présent en utilisant les données capteurs acquises aux précédents instants. Afin de formuler le problème, nous considérons deux balayages laser. Le premier est acquis à l'instant courant t_i et il est appelé ${}^{F_r(t_i)}P(t_i)$. Il permet de calculer le point le plus proche du robot $O_c(t_i)$ et de déterminer la distance $d_c(t_i)$



(a) Situation à l'instant t_p (b) Situation à l'instant t_i

Figure 3.1: Calcul de $d_c^{predicted}(t_i)$

entre O_r le centre du robot et $O_c(t_i)$, ainsi que l'angle $\alpha_c(t_i)$. Le deuxième balayage laser, désigné par ${}^{F_r(t_p)}P(t_p)$, est obtenu à l'instant précédent $t_p < t_i$. Ce balayage laser permet de prédire la valeur de la distance et l'angle entre le robot et le centre de la spirale à l'instant t_i et de la comparer à sa valeur mesurée ci-dessus $d_c(t_i)$ et $\alpha_c(t_i)$. Ainsi, nous calculons d'abord ${}^{F_r(t_i)}P(t_p)$ qui désigne l'ensemble des points qui seraient perçus au moment t_i si seulement le robot avait été en mouvement pendant l'intervalle de temps $[t_p; t_i]$ et l'obstacle immobile. La matrice de transformation entre les deux repères ${}^{F_r(t_i)}T_{F_r(t_p)}$ est exprimée à l'aide de méthodes de localisation locale (c'est-à-dire l'odométrie, IMU). Même si ces méthodes sont connues pour dériver avec le temps, nous pouvons les considérer fiables pour des intervalles de temps $t_i - t_p$ faibles. Ainsi, il est possible de prédire les coordonnées des points laser de l'acquisition précédente dans le repère correspondant au robot actuel à l'aide du calcul qui suit :

$$ {}^{F_r(t_i)}P(t_p) = {}^{F_r(t_i)}T_{F_r(t_p)} \; {}^{F_r(t_p)}P(t_p) \tag{3.1}$$

À partir de ${}^{F_r(t_i)}P(t_p)$ il est possible de calculer la prédiction du point le plus proche $O_c^{predicted}(t_i)$ et de déduire ensuite la distance prédite $d_c^{predicted}(t_i)$ et l'angle prédit $\alpha_c^{predicted}(t_i)$. La figure 3.1 détaille le calcul de distance prédite $d_c^{predicted}(t_i)$. La figure 3.1a expose la situation à un instant t_p. Le robot avance en ligne droite selon le vecteur $\overrightarrow{v_r}$. Dans l'environnement, un obstacle mobile est détecté selon le vecteur $\overrightarrow{v_o}$. À $t = t_p$, le point le plus proche $O_c(t_p)$ ainsi que la distance $d_c(t_p)$ sont calculés. Le robot se déplace ensuite jusqu'à l'instant $t = t_i$, modélisé sur la figure 3.1b. En considérant connu le déplacement entre les deux positions du robot $O_r(t_p)$ et $O_r(t_i)$, il est possible de calculer le point $O_c^{predicted}(t_i)$ et d'en déduire les variables $d_c^{predicted}(t_i)$ et $\alpha_c^{predicted}(t_i)$.

Il est important de noter que le point $O_c^{predicted}(t_i)$ n'est pas le point $O_c(t_p)$ transféré dans le repère $F_r(t_i)$ du robot, mais bien le point le plus proche prédit

à partir de la position du robot à l'instant t_i parmi l'acquisition ${}^{F_r(t_i)}P(t_p)$. Cela explique donc que sur la figure 3.1, les points $O_c(t_i)$ et $O_c^{predicted}(t_i)$ soient distincts.

À l'aide de la distance prédite $d_c^{predicted}(t_i)$, la seconde étape est le calcul de la valeur résiduelle $r(t_i)$ entre la distance mesurée et la distance prédite comme suit :

$$r(t_i) = \frac{d_c^{predicted}(t_i) - d_c(t_i)}{t_p - t_i} \tag{3.2}$$

Cette valeur résiduelle permet de déterminer si l'obstacle est statique ou non. En effet, pour un obstacle statique, la valeur résiduelle $r(t_i)$ est proche de zéro. Pour un obstacle mobile, cette valeur est non nulle et correspond à la variation de la distance entre le robot et l'obstacle due au déplacement de l'obstacle. Cependant, cette valeur seule prise instantanément n'est pas suffisante pour obtenir une discrimination robuste car plusieurs phénomènes tels que le bruit de mesure, la variation du point O_c liée à la forme de l'obstacle, ou bien les imprécisions dans la méthode de localisation locale peuvent donner une valeur de $r(t_i)$ imprécise. Ainsi, une approche statistique est envisagée, prenant en compte la moyenne et la variance la valeur de $r(t_i)$ sur plusieurs itérations.

3.2.2 Calcul de la distance adaptative et de la distance désirée d_a^*

Comme indiqué précédemment, nous utilisons un raisonnement similaire à la méthode utilisée dans [Shi et al., 2005] afin de calculer une distance adaptative $\delta(t_i)$ reposant sur le calcul de la moyenne η et la variance σ^2 de la distance résiduelle $r(t_i)$. En considérant sa variation sur q temps d'échantillonnage, la moyenne et la variance de la distance résiduelle $r(t_i)$ sont les suivantes :

$$\eta(t_i) = \frac{1}{q}\sum_{i=1}^{q} r(t_i) \tag{3.3}$$

$$\sigma^2(t_i) = \frac{1}{q-1}\sum_{i=1}^{q}(r(t_i) - \eta(t_i))^2 \tag{3.4}$$

À partir des équations ci-dessus, la distance adaptative $\delta(t_i)$ à l'instant t_i est calculée comme suit [Shi et al., 2005] :

$$\delta(t_i) = \eta(t_i) \pm 2.17\sigma(t_i) \tag{3.5}$$

où le coefficient 2.17 correspond à la valeur de confiance à 97%. Ainsi, si $r(t_i)$ appartient à l'intervalle $[\delta^-(t_i), \delta^+(t_i)]$ où $\delta^+(t_i) = \eta(t_i) + 2.17\sigma(t_i)$ et $\delta^-(t_i) = \eta(t_i) - 2.17\sigma(t_i)$, alors l'obstacle est considéré comme étant statique. Autrement, l'obstacle est considéré

comme dynamique.

Par conséquent, la borne supérieure $\delta^+(t_i)$ de l'intervalle de confiance peut être utilisée pour définir une nouvelle consigne de distance désirée d_a^* comme suit :

$$d_a^*(t_i) = d^* + \delta^+(t_i) \tag{3.6}$$

où d^* correspond à la distance d'évitement appropriée dans un environnement statique.

3.2.3 Calcul de l'angle adaptatif et du sens de contournement

Lorsque le contrôleur d'évitement d'obstacles est activé, le robot doit choisir son sens du contournement (SOM) autour de l'obstacle. Dans le cas d'obstacles statiques, les conditions ont été présentées dans le chapitre précédent, sous-section 2.3.4. Ces conditions évoluent pour considérer des obstacles mobiles en observant l'évolution de l'angle adaptatif, noté $\delta^\alpha(t)$. Ce dernier est calculé de la même manière que la distance adaptative $\delta(t)$:

$$\delta^\alpha(t_i) = \eta^\alpha(t_i) + 2.17\sigma^\alpha(t_i) \tag{3.7}$$

avec :

$$\eta^\alpha(t_i) = \frac{1}{q}\sum_{i=1}^{q} r^\alpha(t_i) \tag{3.8}$$

$$(\sigma^\alpha)^2(t_i) = \frac{1}{q-1}\sum_{i=1}^{q}(r^\alpha(t_i) - \eta^\alpha(t_i))^2 \tag{3.9}$$

$$r^\alpha(t_i) = \frac{\alpha_c^{predicted}(t_i) - \alpha_c(t_i)}{t_p - t_i} \tag{3.10}$$

Cette valeur fournit une connaissance basique sur la direction du mouvement des obstacles par rapport au robot. En d'autres termes, elle permet de déterminer si l'obstacle se déplace de la gauche vers la droite relativement au robot, ou l'inverse. Cette information permet alors de calculer un SOM le plus approprié :

- Si $|\delta^\alpha(t)| \leq \delta^\alpha_{threshold}$, les conditions pour le cas statique sont spécifiées
- Sinon, si $\delta^\alpha(t) < -\delta^\alpha_{threshold}$, un sens d'évitement direct (anti-horaire) est spécifié.
- Sinon, si $\delta^\alpha(t) > \delta^\alpha_{threshold}$, un sens d'évitement indirect (horaire) est spécifié.

$\delta^\alpha_{threshold}$ est une constante choisie afin de séparer les obstacles statiques et dynamiques. De plus, de la même manière que précédemment, le sens de contournement est recalculé lorsque le robot change de contrôleurs, ou bien si la distance entre deux SCP consécutifs est plus grande que $2d^*$. Ce cas apparaît typiquement lorsque le SCP change d'obstacles.

3.2.4 Évolution de la méthode de navigation basée spirale en environnement dynamique

Cette sous-section présente l'utilisation de la distance désirée et elle permet aussi de montrer comment le comportement de la méthode de navigation basée spirale peut être modifié à partir de l'évolution de la distance désirée d_a^*.

En effet, la distance désirée d_a^* rentre dans le calcul des seuils de commutation entre le contrôleur Go-To-Goal (GTG) et le contrôleur d'évitement d'obstacle (SA). Ces seuils sont présentés dans le chapitre 2 (sous-section 2.3.4) et sont rappelés ci-dessous. Ils sont désormais calculés avec la distance désirée adaptative d_a^* :

$$\left.\begin{array}{l} d_c^{tr} = d_a^* + d_a^*(1 - \frac{|\alpha_c|}{\pi/2}) \\ d_b^{tr} = d_a^* + d_a^*(1 - \frac{|\alpha_b|}{\pi/2}) \end{array}\right\} \text{si contrôleur GTG actif} \tag{3.11}$$

$$\left.\begin{array}{l} d_c^{tr} = 2d_a^* \\ d_b^{tr} = 2d_a^* \end{array}\right\} \text{si contrôleur SA actif} \tag{3.12}$$

Si un obstacle mobile se dirige vers le robot, à chaque itération, la distance mesurée $d_c(t_i)$ est inférieure à la distance prédite $d_c^{predicted}(t_i)$. Ainsi, la distance résiduelle $r(t_i)$ augmente, tout comme la distance adaptative $\delta(t_i)$. Par conséquent, la distance désirée d_a^* augmente et le robot va éviter l'obstacle à une distance plus grande que si l'obstacle était statique. Ainsi, en plus de s'asservir à la distance adaptative d_a^*, le robot anticipe le mouvement de l'obstacle et commute sur le contrôleur SA à une distance plus élevée que dans le cas statique, permettant au robot de maintenir une bonne distance de sécurité entre lui et les obstacles. Dans le cas où le point O_c s'éloigne du robot, la distance adaptative désirée $d_a^*(t_i)$ peut diminuer et atteindre des valeurs très faibles pouvant être dangereuses pour le robot. Ainsi, par mesure de sécurité, nous donnons à $d_a^*(t)$ une valeur minimale fixe, correspondante à la plus petite distance acceptable entre le robot et un obstacle qui s'éloigne de lui.

3.2.5 Validation de la méthode *Distance-Angle Adaptatifs* en environnement dynamique simulé

Cette simulation a pour objectif de montrer l'apport du calcul et de l'utilisation de la distance adaptative désirée d_a^* par rapport à une distance désirée d^* fixe pour évoluer dans un environnement dynamique, ainsi que la pertinence de l'utilisation de l'angle adaptatif δ^α pour choisir le sens de contournement si un obstacle est détecté comme dynamique. Dans un but de comparaison, nous verrons premièrement si la stratégie de navigation présentée dans le chapitre 2, conçue pour gérer des environnements statiques avec une distance d^* fixe, parvient à effectuer une tâche de navigation en environnement dynamique. Nous comparerons ensuite cette méthode avec notre nouvelle méthode *Distance-Angle Adaptatifs*, basée sur la distance d_a^* et les nouvelles

conditions de choix du sens de contournement introduites dans la sous-section 3.2.3. Le comportement du robot et les valeurs des lois de commande sont détaillés dans les deux cas de figure. Pour commencer, nous décrivons l'environnement dynamique d'évolution du robot qui comporte à la fois des obstacles statiques et mobiles.

Environnement dynamique d'évolution du robot

L'environnement considéré est présenté dans la figure 3.2. Le robot part de $O_r(0) = O_w$, et doit atteindre son objectif $O_g = [33, 0]^T$. Sa vitesse a été fixée à $v^* = 1,5m.s^{-1}$. Entre la position de départ du robot et l'objectif, il y a trois obstacles circulaires. Les deux premiers, désignés par (1) et (2), sont mobiles et se déplacent à la vitesse de $1,5m.s^{-1}$, tandis que le dernier (3) est statique. Afin de mettre en exergue la prise en compte des obstacles mobiles, cette simulation fait uniquement appel au contrôleur ω_2 (cf. équation (2.43)) présenté dans la section 2.2.3. Le gain λ_S est fixé à $\lambda_S = 1$. Le paramètre n (cf. équation (2.43)) est fixé à $n = 5$. La distance nominale désirée est $d^* = 3m$. Afin de simuler le bruit provenant du LiDAR, un bruit gaussien additif avec un écart type $\sigma = 0,03$ a été ajouté sur les points du LiDAR. La boucle principale fonctionne avec un pas de $T_S = 0,02s$. Pour calculer la distance résiduelle, l'intervalle entre t_i et t_p a été choisi tel que $t_i - t_p = 0,2s$, et la quantité d'échantillons utilisée pour calculer les résidus de l'équation (4.5) est $q = 30$.

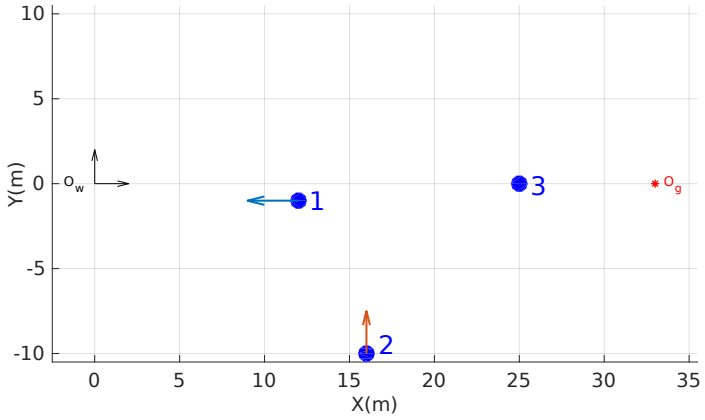
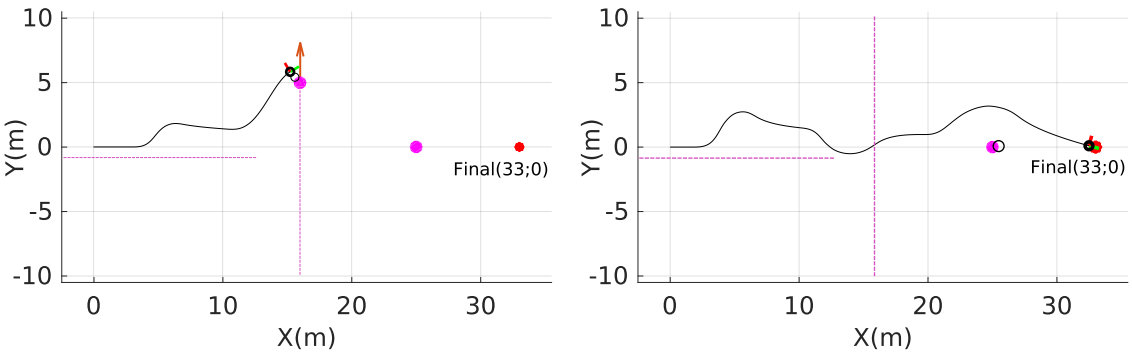


Figure 3.2: Environnement de simulation

Pour souligner l'intérêt de notre approche par rapport aux solutions plus classiques, nous allons nous baser sur une comparaison entre la stratégie de navigation reposant sur un d^* statique (pas d'adaptation en ligne de cette distance) présentée dans le chapitre 2, puis la même stratégie de navigation mais basée sur une distance d^* adaptative. Nous présentons ci-après les résultats correspondants. Les figures 3.3a et 3.3b montrent respectivement la trajectoire suivie par le robot dans le cas statique, et dans le cas adaptatif. Les figures 3.4a et 3.4b montrent, toujours dans le cas statique



(a) Cas d^* statique (b) Cas d_a^* adaptatif

Figure 3.3: Robot (ligne noir) et les obstacles (points roses) et leur trajectoires (lignes roses)

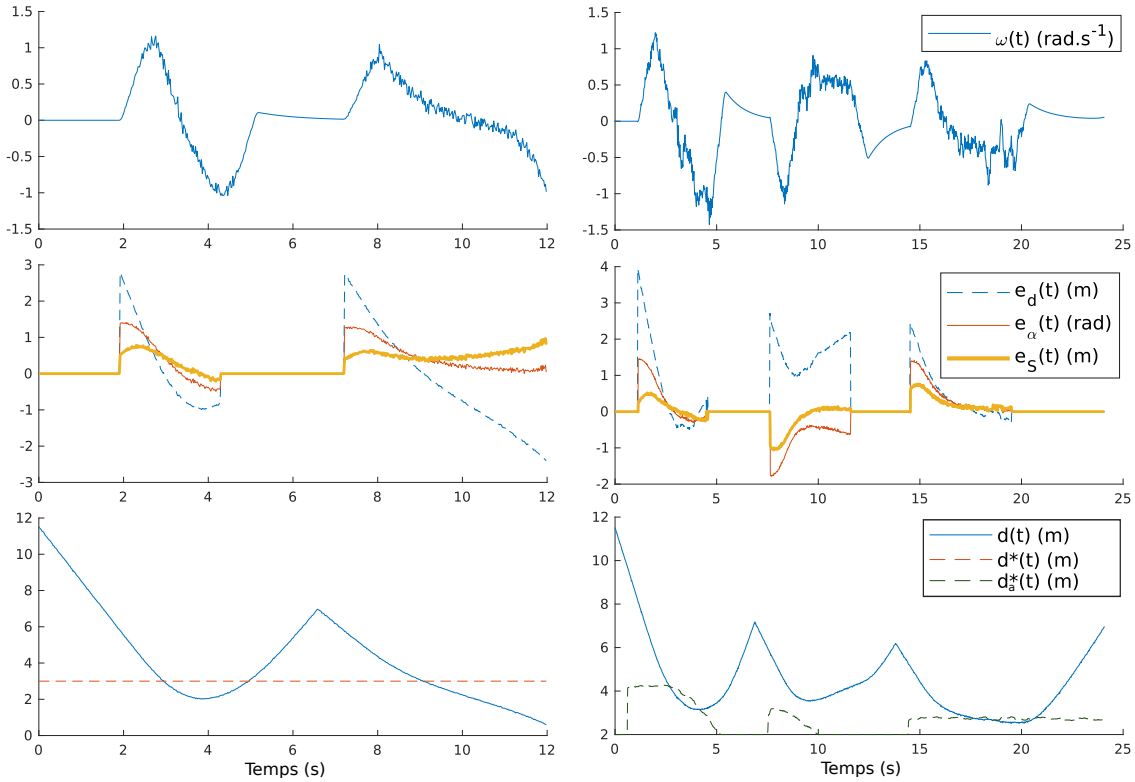
puis dans le cas adaptatif, l'évolution de la consigne angulaire $\omega(t)$, des erreurs $e_d(t)$, $e_\alpha(t)$ et $e_S(t)$, ainsi que l'évolution de la distance $d(t)$, de la distance désirée fixe d^*, et de la distance désirée adaptative $d_a^*(t)$. Nous allons par la suite analyser ces résultats, en observant le déroulement de la simulation lors de l'utilisation de la méthode de navigation développée pour les cas statiques, avec une distance désirée d^* fixe, puis lors de l'utilisation de notre nouvelle méthode *Distance-Angle Adaptatifs*.

Stratégie de navigation en environnement dynamique basée sur la distance fixe d^*

Dans l'environnement dynamique décrit précédemment, la stratégie de navigation repose sur l'utilisation de la distance désirée d^* fixe et un sens de contournement calculés pour un environnement statique. Nous considérons que la distance désirée reste $d^* = 3m$. Le seuil permettant d'activer l'évitement est alors fixé à $2d^* = 6m$. De plus, le SOM est choisi grâce à la méthode présentée pour les cas statiques au chapitre précédent dans la section 2.3.4.

Les figures 3.3a et 3.4a montrent respectivement la trajectoire du robot et des obstacles, la commande angulaire $\omega(t)$, ainsi que les erreurs $e_d(t)$, $e_\alpha(t)$ et $e_S(t)$ et l'évolution de la distance $d(t)$.

À l'instant $t = 2s$, l'obstacle (1) s'approche en dessous de la distance de déclenchement de $2d* = 6m$ et l'évitement est activé. Comme à cet instant, l'obstacle se trouve à la droite du robot, un SOM horaire est sélectionné. À l'instant $t \simeq 4,5s$, l'obstacle est évité et le contrôleur GTG est appliqué. Cependant, même si l'obstacle est évité, on constate qu'étant donné que cet obstacle bouge dans la direction du robot, et que ce dernier déclenche l'évitement à 6m, le robot atteint une distance proche de 2m de l'obstacle, par rapport à une distance désirée d^* de 3m.



(a) Cas d^* statique (b) Cas d^* adaptatif

Figure 3.4: Consigne angulaire $\omega(t)$ – Erreurs $e_d(t)$, $e_\alpha(t)$, $e_S(t)$ – Distances $d(t)$,$d^*(t)$ et $d_a^*(t)$

À l'instant $t \simeq 6,5s$, le robot commence à éviter le second obstacle qui est mobile. À cet instant, cet obstacle se trouve à la droite du robot, mais il se déplace de sa droite vers la gauche. Par conséquent, en fonction de la condition SOM statique, un SOM dans le sens des aiguilles d'une montre est choisi. Ainsi, à cet instant, le robot se retrouve à essayer d'éviter par la gauche un obstacle qui se déplace de sa droite vers sa gauche. Par conséquent, le robot est "traîné" par l'obstacle mobile loin de son but, ce qui entraîne l'échec de la mission.

Stratégie de navigation en environnement dynamique basée sur la méthode *Distance-Angle Adaptatif*

Dans le même environnement dynamique, la stratégie de navigation repose maintenant sur une distance d_a^* et un sens de contournement calculés à partir de la méthode *Distance-Angle Adaptatifs*.

La borne limite inférieure pour la distance désirée d_a^* a été fixée à $2m$. À l'instar de la simulation précédente, les figures 3.3b, 3.4b présentent respectivement la trajectoire du robot et des obstacles, ainsi que la consigne angulaire $\omega(t)$, les erreurs $e_d(t)$, $e_\alpha(t)$ et $e_S(t)$, ainsi que la distance $d(t)$ et la distance désirée adaptative $d_a^*(t)$. La figure 3.5 montre l'évolution de l'angle adaptatif $\delta^\alpha(t)$, ainsi que les seuils $\delta^\alpha_{threshold}$ et $-\delta^\alpha_{threshold}$.

A l'instant $t \simeq 1s$, nous pouvons voir sur la figure 3.4b que l'évitement de l'obstacle (1) se déclenche. En effet, comme cela a été indiqué précédemment et comme le montre la figure 3.2, l'obstacle (1) se déplace vers le robot à une vitesse de $1,5m.s^{-1}$. Par conséquent, la distance désirée $d_a^*(t)$ s'élève rapidement à environ 4 mètres. Ainsi, la distance de déclenchement devient proche de $2d_a^*(t) \simeq 8m$, et est atteinte à $t \simeq 1s$, ce qui permet d'anticiper l'évitement de l'obstacle. L'angle adaptatif, noté $\delta^\alpha(t)$ étant inférieur au seuil, un SOM dans le sens des aiguilles d'une montre est choisi selon l'algorithme statique. Grâce à cette activation précoce, le robot ne s'approche pas à moins de $3m$ du premier obstacle lors de son évitement. De plus, la figure 3.4b montre que la distance adaptative souhaitée $d_a^*(t)$ diminue au fur et à mesure que le robot passe l'obstacle, pour atteindre une valeur inférieure à d^*. La phase d'évitement est donc déclenchée plus tôt que dans le cas précédent et a une durée plus courte.

À l'instant $t = 7,5s$, le deuxième obstacle mobile est détecté et l'évitement est à nouveau activé. Puisque $|\delta^\alpha(t)| > \delta^\alpha_{threshold}$, le signe de δ^α (positif) fournit l'information que l'objet se déplace, et vient de droite à gauche. Grâce à cette information, le SOM est choisi dans le sens inverse des aiguilles d'une montre, afin de pouvoir le contourner par la droite. À l'instant $t \simeq 12s$, l'obstacle (2) est évité.

Le dernier obstacle (3) étant statique, $d_a^*(t)$ reste constant et égal à $d^* \simeq 3m$.

Comme $|\delta^{\alpha}(t)| < \delta^{\alpha}_{threshold}$, un SOM dans le sens des aiguilles d'une montre est fixé. Cet obstacle (3) ne présente aucun danger à $t \simeq 20s$, et le robot atteint son but à $t \simeq 24s$. Enfin, comme on peut le voir sur la figure 3.4b, le contrôleur d'évitement en spirale a réussi à rapprocher $e_S(t)$ de zéro, tout en maintenant le robot à une distance de sécurité de l'obstacle. La loi de commande envoyée au robot est restée dans une plage acceptable.

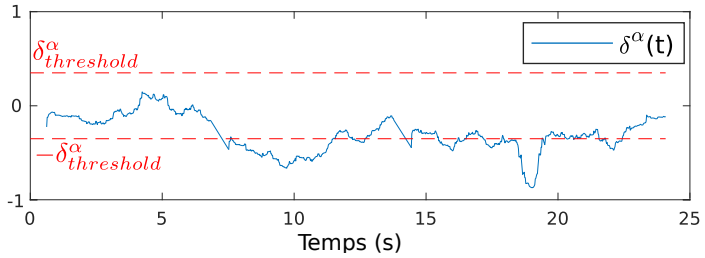


Figure 3.5: Angle adaptatif $\delta^{\alpha}(t)$

Discussions

Les résultats de la simulation ont prouvé l'efficacité et l'intérêt d'une telle méthode par rapport à l'utilisation d'une distance désirée d^* fixe. En revanche, cette méthode souffre d'une importante limitation : elle ne peut prendre en compte qu'un seul obstacle à la fois. Si le robot est en train d'éviter un obstacle, et qu'un second obstacle mobile, plus dangereux, arrive vers le robot, l'algorithme ne s'en rendra compte qu'une fois cet obstacle étant le plus proche du robot, ce qui peut être trop tard selon sa vitesse et sa trajectoire.

3.3 Description de la stratégie de navigation basée sur la méthode *Enhanced Laser Scan*

La seconde approche de navigation en environnement dynamique, préconisée pour un environnement avec de nombreux obstacles, est basée sur la méthode dite *Enhanced Laser Scan* (ELS). Comme son nom l'indique, le principe est d'améliorer les acquisitions effectuées par les LiDARs afin de leur rajouter des points supplémentaires. Cette méthode est introduite afin d'être capable de gérer plusieurs obstacles. Elle se base sur le calcul de la dynamique de chaque point détecté pour artificiellement ajouter des points supplémentaires dont la répartition modélise l'évolution attendue des obstacles mobiles de l'environnement. Ces points ajoutés, appelés points virtuels, sont fusionnés avec le balayage laser initial pour fournir un balayage laser amélioré (ELS, Enhanced Laser Scan). Ce dernier alimente l'algorithme présenté dans le chapitre 2. Cette approche permet au robot d'anticiper l'évolution de chaque obstacle

mobile et de prendre des décisions plus tôt, conduisant au déclenchement plus précoce de l'évitement et à la définition de paramètres d'évitement plus adaptés.

La méthode *Enhanced Laser Scan* repose aussi sur une phase de perception et une phase d'action. La phase de perception se décompose elle même en plusieurs étapes qui sont la détection des obstacles mobiles, leur caractérisation et le calcul des points virtuels. La phase d'action passe par une définition alternative des paramètres d'évitement (centres des spirales O_s, sens de contournement) en fonction des caractéristiques des points améliorés ajoutés et par l'introduction d'une vitesse linéaire variable fonction de l'erreur en angle entre le robot et l'obstacle.

3.3.1 Détection des obstacles mobiles

Afin de détecter les obstacles mobiles, nous appliquons le même principe que celui introduit dans la partie sur la méthode *Distance-Angle Adaptatifs* présentée dans la section 3.2. À la différence de l'approche précédente, la méthode est maintenant appliquée à l'ensemble des points des acquisitions LiDAR et non pas uniquement au point le plus proche O_c. Le principe consiste donc à prédire les coordonnées des points LiDAR d'une acquisition précédente dans le repère correspondant au robot actuel. En effet, en appliquant l'équation (3.1), nous obtenons la prédiction de l'ensemble des points qui seraient perçus au moment actuel t_i, appelés ${}^{F_r(t_i)}P(t_p)$, en prenant en compte l'ensemble des points acquis à un moment t_p passé, ${}^{F_r(t_p)}P(t_p)$, et en considérant le déplacement du robot pendant l'intervalle de temps $[t_p; t_i]$.

Ainsi, l'algorithme 4 détermine pour chaque point dans ${}^{F_r(t_i)}P(t_p)$ son point le plus proche dans ${}^{F_r(t_i)}P(t_p)$. Si cette distance est supérieure à un seuil D, le point est considéré comme mobile. Dans le cas contraire, il est considéré comme statique. Ce seuil doit être réglé par l'utilisateur en fonction de la précision des capteurs présents sur le robot. L'algorithme produit deux matrices de sortie, ${}^{F_r(t_i)}\widetilde{P}(t_i)$ et ${}^{F_r(t_i)}\widetilde{P}(t_p)$. Ces matrices contiennent respectivement le sous-ensemble des points de ${}^{F_r(t_i)}P(t_i)$ et ${}^{F_r(t_i)}P(t_p)$ qui sont considérés comme étant en mouvement, et sont utilisés pour caractériser l'environnement du robot.

3.3.2 Fusion et caractérisation dynamique des obstacles mobiles

L'ensemble des points détectés comme mobiles ${}^{F_r(t_i)}\widetilde{P}(t_i)$ est maintenant analysé en termes de distance pour fusionner éventuellement certains points et en termes de dynamique pour calculer leur vecteur vitesse.

Algorithme 4 Détection des points en mouvement

```
F_r(t_i)P̃(t_i), F_r(t_i)P̃(t_p) ← []
pour chaque Point P1 dans F_r(t_i)P(t_i) faire
  D_min ← +∞
  pour chaque Point P2 dans F_r(t_i)P(t_p) faire
    D_min ← min(D_min, distance(P1, P2))
  fin pour chaque
  si D_min > D alors
    F_r(t_i)P̃(t_i) ← [F_r(t_i)P̃(t_i); P1]
    F_r(t_i)P̃(t_p) ← [F_r(t_i)P̃(t_p); P2]
  fin si
fin pour chaque
```

Séparation/fusion des obstacles

En se basant sur les distances cartésiennes entre les points, les points de la liste ${}^{F_r(t_i)}\widetilde{P}(t_i)$ sont séparés ou rassemblés dans différents groupes représentant chacun un obstacle différent. Pour se faire, une technique de regroupement bien connue comme le regroupement hiérarchique [Rokach and Maimon, 2005] est appliquée. Cette méthode itérative consiste initialement à considérer chaque point comme étant un groupe individuel et à fusionner les deux groupes les plus proches à chaque itération, jusqu'à ce que la plus petite distance entre deux groupes soit supérieure à un seuil fixé. Cette méthode de séparation permet de différencier plusieurs obstacles dont le nombre est initialement inconnu.

Calcul du vecteur vitesse des obstacles

Après le regroupement, le vecteur vitesse de chaque obstacle est estimé. Pour chaque groupe de points définis, les barycentres ${}^{F_r(t_i)}B(t_i)$ et ${}^{F_r(t_i)}B(t_p)$ de leurs points correspondants dans ${}^{F_r(t_i)}\widetilde{P}(t_i)$ et ${}^{F_r(t_i)}\widetilde{P}(t_p)$ sont calculés. Le vecteur vitesse pour chaque ensemble de points est obtenu comme suit :

$$\vec{V}_{R_c}(t_i) = \begin{pmatrix} v_x(t_i) \\ v_y(t_i) \end{pmatrix}_{R_c} = \frac{{}^{F_r(t_i)}B(t_i) - {}^{F_r(t_i)}B(t_p)}{t_i - t_p} \tag{3.13}$$

3.3.3 Projection des points virtuels

Le vecteur vitesse de chacun des obstacles étant estimé, il est possible de calculer les points virtuels. Pour chaque obstacle, les points virtuels vont dépendre de la forme du groupe de points, et de la direction et norme du vecteur vitesse. Ces points virtuels dépendent de l'horizon temporel de projection que nous fixons en fonction de la dynamique des obstacles. Cet horizon étant fixé, les points virtuels sont projetés et

ajoutés afin de modéliser une *enveloppe améliorée* qui va étendre la forme de chaque obstacle le long de son vecteur vitesse.

Horizon temporel de la projection des points virtuels

L'évitement est déclenché à une distance maximale de $2d^*$. En fonction de sa vitesse $v(t)$, le robot peut, dans le pire des cas où il se dirige droit vers le barycentre de l'obstacle, parcourir cette distance en $2d^*/v(t)$ secondes. Nous définissons alors un horizon de temps $t_h = 2d^*/v_{max}$, avec v_{max} la vitesse maximale du robot. Ainsi, chaque point mobile est projeté le long du vecteur $\vec{d_h} = \vec{V}_{R_c}(t_i).t_h$. La norme de ce vecteur prend donc en compte à la fois la vitesse prédite de l'obstacle, mais aussi la distance désirée d^* et la vitesse maximale du robot.

Projection et formation de l'*enveloppe améliorée* de points pour chaque obstacle mobile

Pour chaque obstacle mobile, les points virtuels sont ajoutés au balayage laser initial pour modéliser la trajectoire future de cet obstacle. La méthode consiste à translater, pour chaque groupe d'obstacles, les points d'une distance donnée $||\vec{d_h}||$le long du vecteur $\vec{d_h}$, et à ajouter l'*enveloppe* reliant les points initiaux et les points virtuels. De plus, afin d'éviter que la loi de contrôle angulaire n'atteigne inopinément des valeurs trop élevées, aucun point virtuel ne doit être ajouté à une distance inférieure à d^* autour du robot.

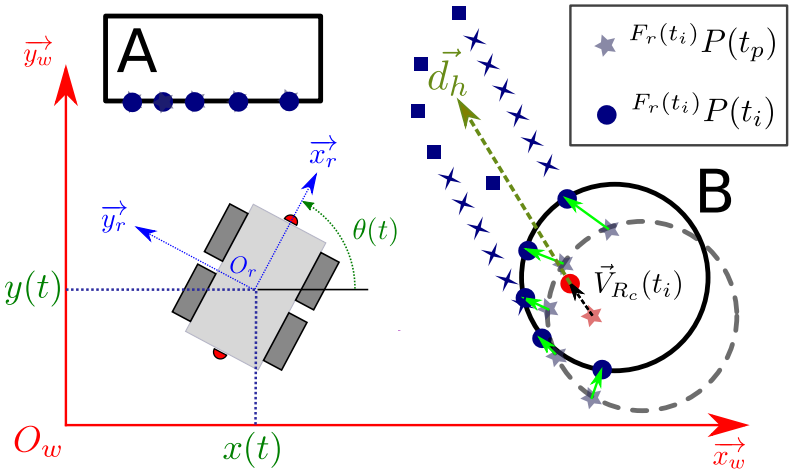


Figure 3.6: Processus de calcul de la dynamique des obstacles et des points virtuels

La figure 3.6 montre comment les points virtuels sont calculés. L'environnement est encombré de deux obstacles. Un obstacle statique rectangulaire (A), et un obstacle dynamique circulaire (B). Les lignes pleines représentent les deux obstacles à

$t = t_i$. Les points bleus solides sont les points ${}^{F_r(t_i)}P(t_i)$ actuellement détectés par le LiDAR. Le cercle en pointillés représente l'obstacle (B) tel qu'il était au moment $t = t_p$, tandis que les points en étoiles grises sont les points ${}^{F_r(t_i)}P(t_p)$, qui ont été détectés au moment précédent t_p, projetés dans le repère actuel du robot R_c. Au début de l'algorithme, chaque point de ${}^{F_r(t_i)}P(t_p)$ est mis en correspondance avec son point le plus proche dans ${}^{F_r(t_i)}P(t_i)$, représenté par les flèches vertes. Si la distance entre ces points appariés est supérieure à un seuil, ces points sont considérés comme étant mobiles. C'est le cas des points situés dans le cercle, mais pas de ceux appartenant à l'obstacle statique (A). A la fin de ce processus, ${}^{F_r(t_i)}\widetilde{P}(t_p)$ contient les cinq points gris étoilés, ${}^{F_r(t_i)}\widetilde{P}(t_i)$ les cinq points bleus situés sur le cercle. Ensuite, le regroupement est effectué et les barycentres ${}^{F_r(t_i)}B(t_i)$ et ${}^{F_r(t_i)}B(t_p)$ sont calculés et représentés respectivement par le point circulaire rouge et le point étoile rouge. On calcule ensuite $\vec{V}_{R_c}(t_i)$, représenté par la flèche noire entre les points rouges, puis le vecteur $\vec{d_h}$ résultant. Ce processus étant effectué, les points virtuels sont calculés et ajoutés au balayage final. Les cinq points bleus ronds sont projetés le long de leur vecteur respectif $\vec{d_h}$, générant les cinq points bleus carrés. Ensuite, l'enveloppe virtuelle est achevée grâce à l'ajout des points bleus en forme de croix, qui viennent relier les points initiaux et les points projetés. La fusion des points bleus carrés et des points bleus en forme de croix forme l'enveloppe virtuelle améliorée pour chaque obstacle mobile. La fusion des enveloppes virtuelles et des points LiDAR initiaux forme l'*Enhanced Laser Scan*.

À partir de cet *Enhanced Laser Scan*, une des méthodes présentées dans le chapitre 2 pour choisir le SCP est utilisée. Pour un instant donnée, le centre de la spirale suivie peut appartenir à l'acquisition LiDAR initiale, pour bien à un des points virtuels rajoutés.

3.3.4 Évolution de la méthode de navigation basée spirale

Cette sous-section présente l'utilisation des caractéristiques de l'enveloppe améliorée de chaque obstacle pour déterminer certains paramètres de la méthode de navigation, comme le sens de contournement. Dans cette méthode, nous propons également de modifier la vitesse linéaire du robot restée constante dans la méthode précédente, selon certaines conditions.

Détermination du sens de contournement (SOM)

Pour les obstacles statiques (section 2.3.4), le sens de contournement (SOM) est basé sur la comparaison entre l'angle α_b et l'angle α_g : si $\alpha_b \leq \alpha_g$, le SOM est défini dans le sens des aiguilles d'une montre. Sinon, il est fixé dans le sens inverse des aiguilles d'une montre.

Pour un environnement dynamique, chaque obstacle doit être considéré, et le vecteur vitesse de chaque obstacle peut être pris en compte à l'instant de la décision, afin de mener à un choix du sens de contournement plus pertinent. Si l'obstacle se trouve à la droite du robot et se déplace de droite à gauche, le calcul précédent produira un SOM dans le sens des aiguilles d'une montre. Un tel SOM ferait en sorte que la trajectoire du robot croise la trajectoire prévue de l'obstacle, ce qui entraînerait une menace de collision. Le calcul proposé anticipe le mouvement de l'obstacle dynamique.

Lorsqu'un obstacle est détecté et que la commutation entre le contrôleur Go-To-Goal et le contrôleur d'évitement d'obstacle est déclenchée, le sens de contournement est déterminé en utilisant O_c et son vecteur vitesse associé $\vec{V}_{R_c}(t) = \begin{pmatrix} v_x(t) \\ v_y(t) \end{pmatrix}_{R_c}$.

- Si $v_y = 0$, les conditions relatives aux obstacles statiques sont appliquées.
- Sinon, si $v_y > 0$, on choisit un SOM dans le sens inverse des aiguilles d'une montre. Cela signifie que l'obstacle va de droite à gauche.
- Sinon, si $v_y < 0$, on choisit un SOM dans le sens des aiguilles d'une montre. Cela signifie que l'obstacle va de gauche à droite.

Réévaluation du sens de contournement (SOM)

Dans les cas complexes (obstacles multiples, dynamique imprévisible des obstacles), il peut arriver que le SOM calculé lors du premier déclenchement du processus d'évitement fasse que le robot soit entraîné par un obstacle se déplacant à la même vitesse que lui. Par conséquent, il est nécessaire de détecter ces situations et de réévaluer le SOM chaque fois que cela est nécessaire. Ces cas peuvent être détectés en vérifiant si $v_x > 0$ et $|v_y| < v_y^0$, avec v_y^0 un seuil fixé par l'utilisateur. Si ces conditions sont remplies, le robot va dans la même direction que l'obstacle. Si ce cas se produit, l'algorithme réévalue le SOM. De plus, à chaque itération, la distance entre deux SCP successifs est calculée. Si cette distance est supérieure à un seuil donné, le nouveau SCP appartient à un nouvel obstacle. Dans ce cas, le SOM est également réévalué.

Modification de la vitesse linéaire

Dans les précédentes parties, la vitesse linéaire v^* du robot restait constante. Cependant, il s'agissait d'environnements statiques pour les méthodes présentées dans le chapitre 2, et d'environnements dynamiques peu encombrés pour la méthode *Distance-Angles Adaptatifs*. Ainsi, nous obtenions une évolution douce du centre des spirales O_s dans la majorité des cas. Cependant, dans le cadre d'un environnement fortement dynamique, l'ajout de points virtuels au sein de l'*Enhanced Laser Scan*, effectué à chaque itération en fonction des nouvelles données acquises par les capteurs, peut

entraîner de soudaines variations du point O_s. Ainsi, pour éviter un comportement brusque et des pics sur la commande angulaire $\omega(t)$, nous proposons de faire varier la vitesse linéaire du robot. Pour un comportement plus lisse, la vitesse linéaire du robot v est donc modulée en fonction de la valeur de l'erreur angulaire $e_\alpha(t)$. Le principe est de faire évoluer la vitesse linéaire entre une vitesse maximale v_{max} et une vitesse minimale v_{min}, selon l'équation suivante :

$$v(t) = v_{max} - (v_{max} - v_{min})\frac{|e_{\alpha(t)}|}{\pi/2} \tag{3.14}$$

Cette formule conduit au comportement suivant : lorsqu'un obstacle se trouve devant le robot ($|e_{\alpha(t)}| \simeq \pi/2$) et que ce dernier doit tourner en urgence, la vitesse linéaire est minimale et égale à v_{min}. Si le robot suit la spirale spécifiée ($|e_{\alpha(t)}| \simeq 0$), la vitesse linéaire est maximale et égale à v_{max}.

3.3.5 Validation de la stratégie de navigation avec la méthode *Enhanced Laser Scan* en environnement dynamique simulé

Cette section a pour objectif d'évaluer la stratégie de navigation avec la méthode *Enhanced Laser Scan* en environnement dynamique simulé, comportant à la fois des obstacles statiques et mobiles.

Environnement dynamique d'évolution du robot

La figure 3.7 illustre l'environnement dynamique dans lequel le robot évolue. Cette zone comporte un obstacle statique (O_5), trois petits obstacles mobiles convexes (O_1, O_2 et O_3) qui peuvent représenter des personnes en train de marcher et un obstacle initialement statique puis mobile (O_4) qui est un obstacle de la taille d'une voiture. La position et la vitesse des obstacles sont données à différents instants : $t_1 = 0s$, $t_2 = 7,5s$ et $t_3 = 12s$. Deux de ces obstacles ont un comportement particulier : à l'instant $t = t_2$, l'obstacle O_2 change sa trajectoire et l'obstacle O_4 ne commence à se déplacer qu'à l'instant t_3.

La mission du robot consiste à partir de sa pose initiale $O_r(0) = O_w$ pour atteindre un objectif O_g dont les coordonnées dans le repère initial du robot sont $[x_g, y_g] = [30, 0]$. La vitesse linéaire du robot $v(t)$ évolue entre $v_{min} = 0,5ms^{-1}$ et $v_{max} = 1,5ms^{-1}$. Pour assurer un mouvement d'évitement sûr, $d^* = 3m$. Les gains du contrôleur ω_A (cf. équation (2.42)) ont fixés à $\lambda_1 = 0,2$ et $\lambda_2 = 0,2$, tandis que le gain pour le contrôleur ω_B (cf. équation (2.43)) est fixé à $\lambda_S = 0,5$. Le seuil e_α^{switch} est fixé à $e_\alpha^{switch} = \pi/12$. Le paramètre n (cf. équation (2.43)) est fixé à $n = 5$. Pour finir, la valeur de v_y^* utilisée pour la réévaluation du SOM est fixée à $v_y^0 = 0,5ms^{-1}$. Pour simuler les LiDAR, un bruit blanc gaussien additif avec un écart type $\sigma = 0,03$ est appliqué à la

sortie de distance LiDAR. L'algorithme fonctionne avec une période d'échantillonnage de $T_s = 0,02s$. En outre, l'intervalle entre t_i et t_p est choisi à 10 itérations, d'où $t_i - t_p = 0,2s$. Pour lisser le contrôle, une moyenne par fenêtre glissante de cinq itérations est appliquée. Le choix du SCP est fait selon la méthode basée sur les barycentres exposée section 2.3.3.

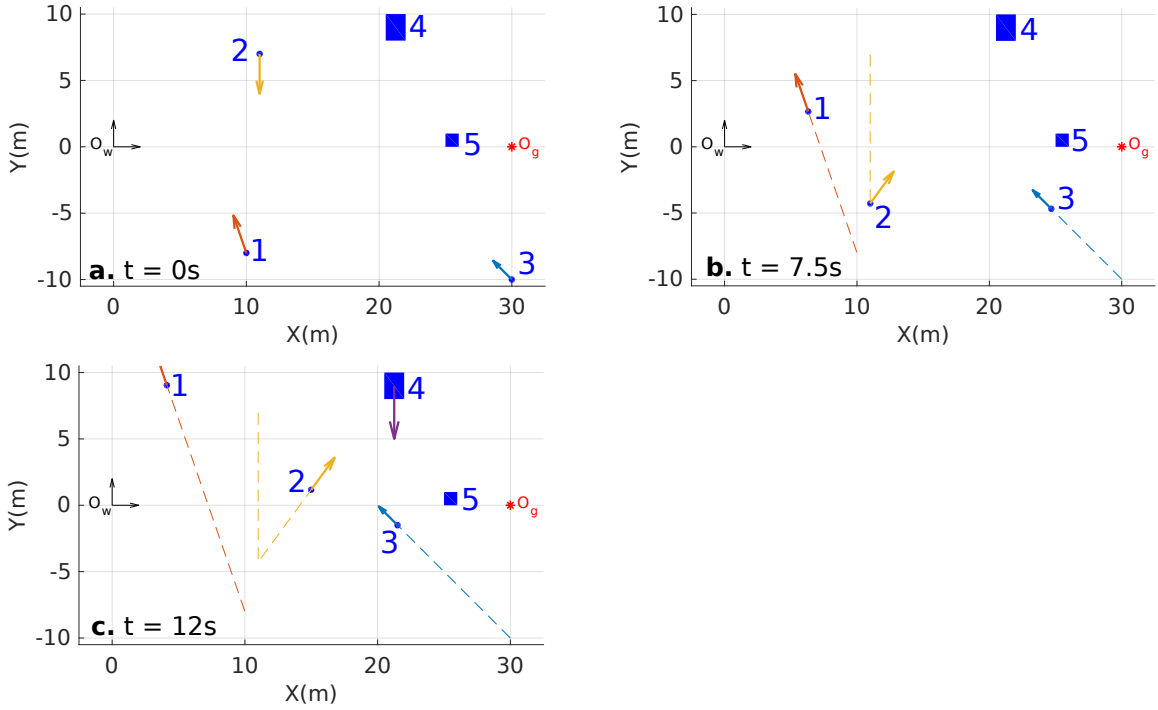


Figure 3.7: Positions et déplacements des obstacles à trois moments clefs

La figure 3.8 affiche la pose du robot et sa trajectoire à quatre moments clefs. Les points rouges représentent l'enveloppe améliorée. Les points de cette enveloppe peuvent être comparés aux vitesses réelles des obstacles indiquées sur la figure 3.7 et représentées par des flèches. A chaque itération, des points laser virtuels sont ajoutés le long de la future trajectoire prévue de chaque obstacle. Les points noirs représentent le centre de la spirale courant.

La figure 3.9 affiche la vitesse linéaire $v(t)$ et la vitesse angulaire $\omega(t)$ ainsi que le type de contrôleurs utilisé. La figure 3.10 présente l'évolution de $e_d(t)$ et de $e_\alpha(t)$, ainsi que de $d(t)$ et de $d_c(t)$ [physical], qui représente le point physique le plus proche du robot.

À l'aide de ces figures, nous commentons l'évitement de chaque obstacle par le robot.

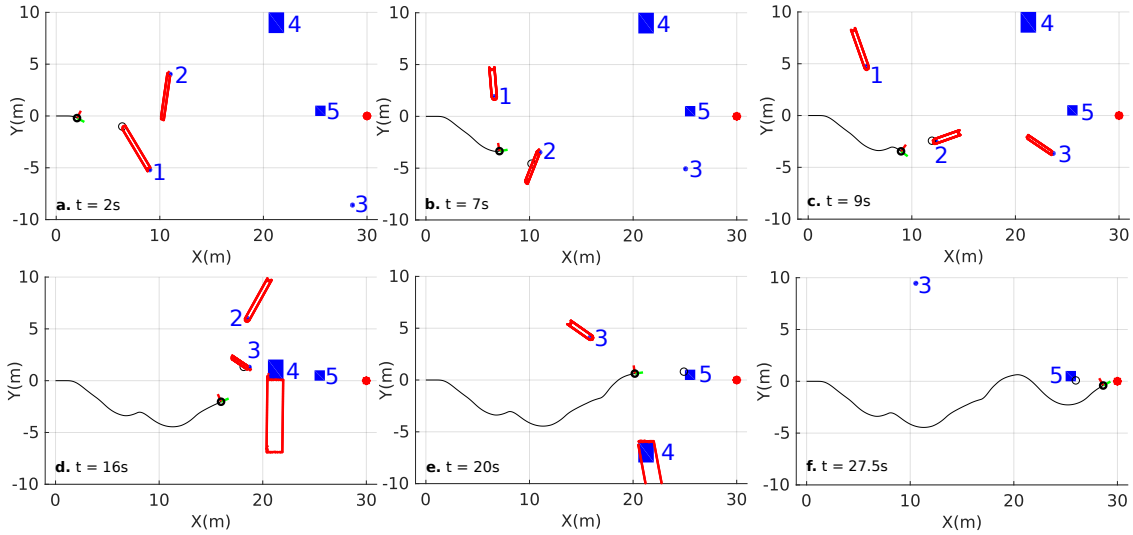


Figure 3.8: Position des obstacles et du robot et trajectoire du robot à six instants : (a.) t = 2s ; (b) t = 7s ; (c) t = 9s ; (d) t = 16s ; (e) t = 20s ; (f) t = 27.5s

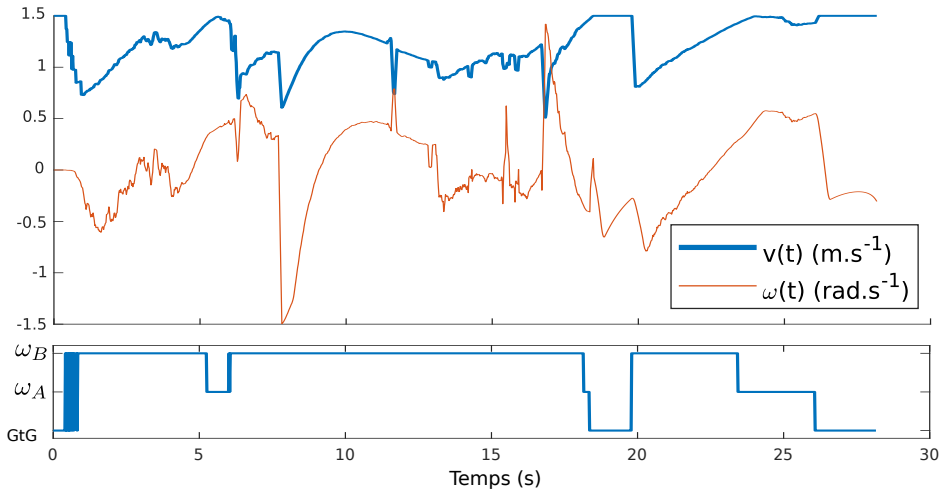


Figure 3.9: Vitesse linéaire (en $m.s^{-1}$), vitesse angulaire (en $rad.s^{-1}$) et contrôleur utilisé (GTG, ω_A, or ω_B)

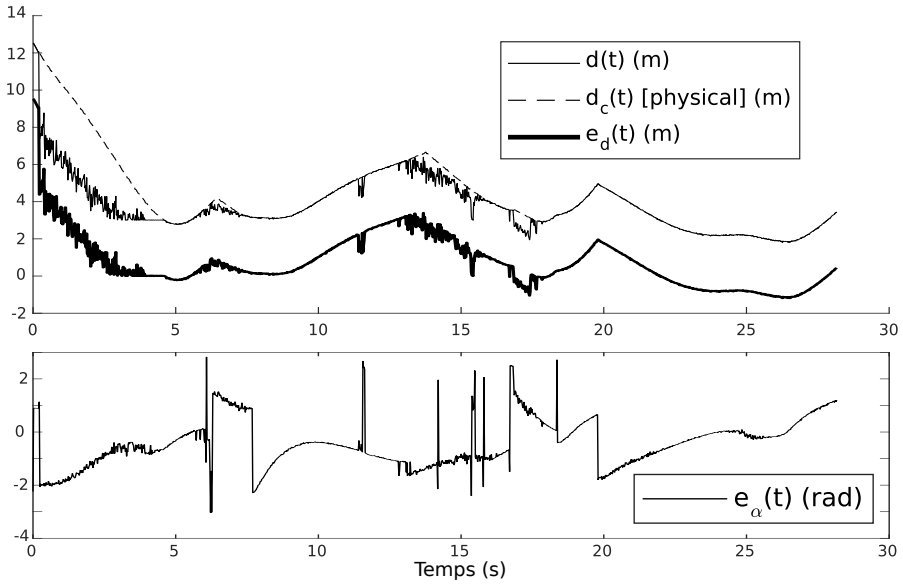


Figure 3.10: Erreur en distance $e_d(t)$, erreur en angle $e_\alpha(t)$, distance au SCP $d(t)$ et distance au point physique le plus proche $d_c(t)$ [physical].

Évitement de l'obstacle O_1

De l'instant $t = 0s$ à $t = 1s$, le robot utilise le contrôleur GTG et se dirige vers le but. Les obstacles O_1 et O_2 sont dans la zone de détection du LiDAR. L'algorithme détecte que ces obstacles sont mobiles et ajoute des points virtuels le long de leurs futures trajectoires prévues, formant finalement l'*Enhanced Laser Scan*. Le point le plus proche O_c est calculé et appartient aux points virtuels ajoutés à partir de l'obstacle O_1. Ce point se trouvant dans la plage de déclenchement de l'évitement, le robot passe du contrôleur GTG au contrôleur SA. À cet instant, grâce au vecteur de vitesse de l'obstacle, l'algorithme détecte que l'obstacle va de droite à gauche, et décide d'un sens de contournement SOM anti-horaire, comme on peut le voir sur la figure 3.8a.

Évitement de l'obstacle O_2

À l'instant $t = 6s$, le SCP devient le point le plus proche O_c qui est un point virtuel dérivé de O_2. Un tel saut dans la position du SCP déclenche la réévaluation du sens de contournement SOM. Comme l'obstacle O_2 va de gauche à droite, on choisit un sens de contournement horaire (cf. figure 3.8c). Comme le montre la figure 3.7, l'obstacle O_2 change soudainement de trajectoire à l'instant $t = 7,5s$. En raison de son choix initial de SOM, le robot tente de l'éviter dans la même direction. La condition évoquée dans la partie 3.3.4 est remplie, et le SOM est réévalué et nouveau sens de contournement, anti-horaire, est calculé comme le montrent les figures 3.8c. et 3.8d. Un pic dans la commande angulaire $\omega(t)$ est observé sur la figure 3.9 et il correspond à la réévaluation du SOM.

Évitement des obstacles O_3 et O_4

À l'instant $t = 16s$, l'obstacle O_3 se déplace de droite à gauche par rapport au robot. Un SOM anti-horaire est fixé. Peu après, l'obstacle O_4 devient l'obstacle détecté le plus proche. Ce saut dans le SCP entraîne une réévaluation du SOM, choisi comme anti-horaire, puisque l'obstacle se déplace de gauche à droite.

Évitement de l'obstacle O_5

À l'instant $t = 20s$, l'obstacle mobile O_4 a été évité, et le robot arrive devant l'obstacle statique O_5. Un SOM anti-horaire est calculé. A l'instant $t = 28.2s$, la simulation se termine avec le robot qui atteint son but. Le robot parvient à atteindre son objectif en 28,2 secondes, en évitant tous les obstacles sans risquer de collision.

Évolution des vitesses linéaire et angulaire du robot

La figure 3.9 montre donc l'évolution de la vitesse linéaire et de la vitesse angulaire au cours de la simulation. Nous voyons notamment plusieurs pics de vitesse angulaire, dus à une augmentation de l'erreur angulaire $e_\alpha(t)$. Ces pics correspondent aussi avec une baisse de la vitesse linéaire $v(t)$, permettant de maintenir $\omega(t)$ dans des limites acceptables. Nous constatons l'utilité de faire varier cette vitesse linéaire avec ce qui se passe à l'instant $t = 7.5$, où l'obstacle O_2 change soudainement de trajectoire. Nous voyons que la vitesse linéaire descend alors proche de v_{min}, afin de laisser du temps au robot pour tourner. Nous voyons ainsi dans la figure 3.10 que la distance $d_c(t)$ entre le robot et le point physique le plus proche ne descend jamais en dessous de trois mètres.

Discussions

Cette simulation met en évidence les avantages de notre méthode dans un environnement encombré fortement dynamique. En utilisant les informations sur l'environnement fournies par l'*Enhanced Laser Scan*, le robot est capable de prendre de meilleures décisions concernant le SOM, ce qui permet d'éviter les obstacles plus rapidement et en prenant en compte leur déplacement. En outre, il peut également gérer une modification soudaine de la direction des obstacles en changeant de SOM, évitant ainsi une possible collision, ou bien d'être *traîné* par l'obstacle. On peut voir sur la figure 3.10 qu'en dépit des nombreux obstacles dynamiques, aucune collision ne se produit. Les commandes $v(t)$ et $\omega(t)$ restent également continues et dans des plages acceptables.

3.4 Conclusion

Ce chapitre a traité le problème de l'évitement d'obstacles mobiles en proposant deux méthodes de navigation qui permettent de gérer une grande variété d'obstacles et de

dynamiques. Ces deux méthodes sont suffisamment efficaces pour garantir la non-collision du robot en environnements dynamiques. Dans la prolongation de la méthode en environnement statique présentée dans le chapitre 2, la force de nos approches dynamiques est la généricité. Ces dernières ont été développées dans l'optique d'être génériques et de pouvoir ainsi gérer la plus grande majorité des cas. Aucune hypothèse n'est donc faite sur le nombre d'obstacles présents, leur taille, leur dynamique, etc. Ainsi, deux contributions ont été présentées dans ce chapitre :

- La première contribution consiste en la mise à jour du paramètre de distance désirée d^* des spirales générées en fonction de l'évolution du point le plus proche O_c, permettant une anticipation de l'évitement dans un environnement dynamique peu encombré.

- l'analyse et la comparaison des acquisitions capteurs successives permet de générer artificiellement des nouveaux points sur les trajectoires futures des éventuels obstacles mobiles. L'ajout de ces points permet à l'algorithme de choisir des centres de spirales prenant en compte le déplacement des obstacles mobiles, et entraîne un comportement d'évitement anticipé, même en présence de plusieurs obstacles mobiles simultanés.

Ces deux méthodes reposent sur une phase de perception consistant à prendre en compte une ou plusieurs acquisitions capteur successives, puis à les comparer afin de détecter les obstacles mobiles. Cette phase de perception se base sur une connaissance locale du déplacement du robot. Elle est donc tributaire de la précision de l'odométrie du robot. Cependant, sur des intervalles de temps assez courts, nous pouvons considérer l'information odométrique comme étant fiable. À la suite de la phase de perception vient la phase d'action. La force principale de nos deux contributions est de se baser sur la méthode d'évitement très générique présentée dans le chapitre 2, et de permettre au robot d'évoluer dans des environnements dynamiques en changeant simplement les paramètres des spirales utilisées pour l'évitement.

Chapitre 4

Expérimentations en environnement statique et dynamique

Les deux chapitres précédents ont été consacrés à la présentation théorique de nos méthodes de navigation référencées capteurs, permettant à un robot mobile de naviguer de manière autonome dans des environnements statiques et dynamiques inconnus. L'ensemble des méthodes présentées ont été validées en simulation, afin de mettre en exergue leur bon fonctionnement et leur pertinence. Cela a permis la mise en place de protocoles expérimentaux permettant de valider dans des situations réelles les approches présentées précédemment. Ainsi, ce chapitre est consacré aux expérimentations effectuées avec le robot "4MOB".

Les environnements dans lesquels vont être effectués les tests sont au nombre de deux. Le premier environnement est un environnement avec un sol goudronné, dans lequel se trouvent plusieurs bâtiments et obstacles (voiture, citerne, etc.). Cet environnement représente un exemple typique de la partie aménagée d'une exploitation agricole, dans laquelle se trouvent des hangars, des véhicules, etc. Le second environnement est un environnement naturel, avec un sol herbeux, boueux et tapissé de feuilles mortes dans lequel se trouvent divers obstacles telles que des arbres ou des tables. Cet environnement représente davantage l'environnement au sein duquel le robot serait susceptible de naviguer pour se rendre sur les parcelles.

Ainsi, les deux premières expérimentations mettent en oeuvre la méthode de navigation avec les deux lois de commande pour un environnement statique présentée dans le chapitre 2. Ces deux expérimentations valideront respectivement la méthode de choix des centres de spirales basée sur le barycentre (cf. section 2.3.3), puis sur le projeté (cf. section 2.5.1), ainsi que l'ensemble des conditions de navigation. Elles auront lieu sur les deux environnements de test présentés dans le paragraphe précédent.

Les deux expérimentations suivantes ont pour but de présenter des résultats préliminaires relatifs aux méthodes exposées dans le chapitre 3 pour les cas dynamiques. Elles auront lieu sur terrain goudronné. Elles valideront premièrement les capacités de perception des méthodes *Distance-Angle Adaptatif* (cf. section 3.2) et *Enhanced Laser Scan* (cf. section 3.3). Pour la méthode *Distance-Angle Adaptatif*, nous verrons notamment que le calcul de l'angle adaptatif et de la distance adaptative s'effectue correctement et permet de prendre des décisions pertinentes pour l'évitement d'obstacles dynamiques. Pour la méthode *Enhanced Laser Scan*, l'expérimentation permettra de mettre en avant le calcul de l'enveloppe améliorée dans un environnement complexe dans lequel se trouvent plusieurs obstacles statiques et dynamiques.

Préalablement à ces deux sections consacrées aux expérimentations, une section sera dédiée à l'évaluation en simulation des méthodes de navigation présentes dans le Middleware ROS à l'aide du simulateur Gazebo. Ces dernières auront lieu au sein de trois environnements considérés comme difficiles à gérer et mettront en évidence les limitations des méthodes *Dynamic Windows Approach* et *Time-Elastic Bands* implémentées au sein de ROS. Cette section permettra notamment d'aborder les problématiques d'implémentation des méthodes, de complexité d'utilisation et de temps de calcul.

Le chapitre est organisé comme suit : la première section présente le robot utilisé pour les expérimentations, ainsi que les capteurs qui l'équipent. La deuxième section présente les méthodes de navigation disponibles sur ROS. Des simulations seront effectuées et une comparaison sera dressée avec notre méthode. La troisième section porte sur les expérimentations effectuées dans des contextes statiques, tandis que la quatrième section présente les résultats préliminaires obtenus dans des environnements dynamiques.

4.1 Présentation du robot

Figure 4.1: Base robotique 4MOB de Sterela (Crédit Matthieu Herrb)

4.1.1 Plateforme robotique

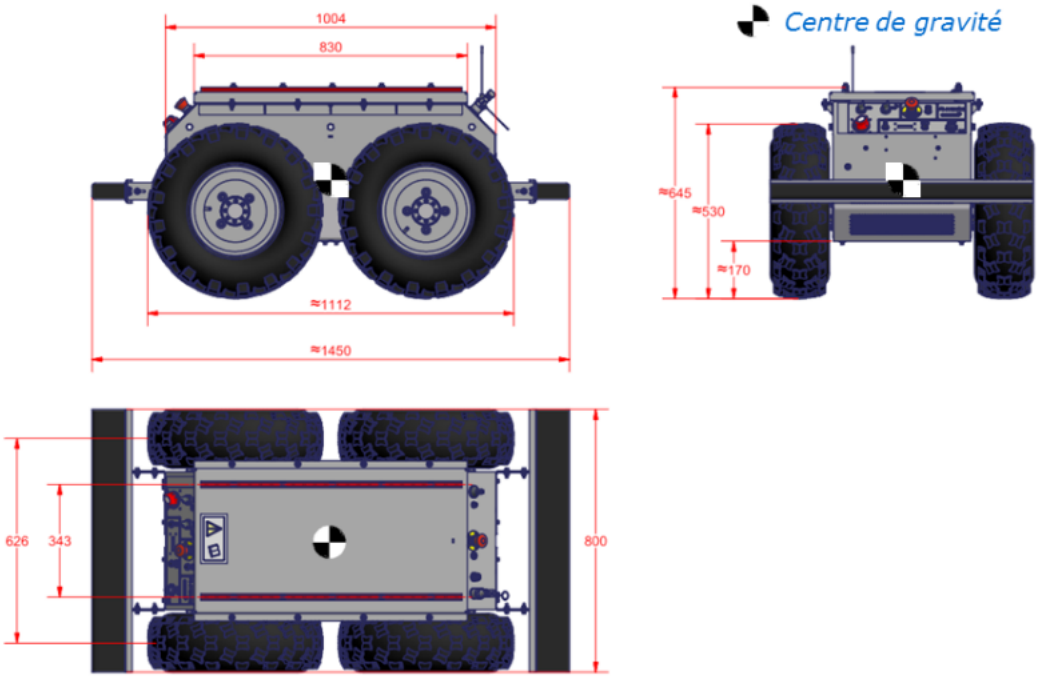


Figure 4.2: Dimensions de la base robotique 4MOB de Sterela

Afin de réaliser les expérimentations, nous avons utilisé une plateforme robotique construite sur la base mobile 4MOB produite par l'entreprise Sterela. Cette dernière est visible sur l'image 4.1. La base robotique 4MOB possède quatre roues indépendantes lui permettant d'effectuer une rotation sur elle-même. En effet, cette plateforme a été conçue afin de pouvoir effectuer des déplacements dans des environnements complexes et sur des sols irréguliers. La plateforme est notamment capable de se déplacer sur des sols herbeux, rocailleux ou boueux. Elle est aussi capable de franchir des obstacles au sol tels que des branches ou des dénivelés brusques, et sous certaines conditions des marches. La plateforme 4MOB pèse 240kg et peut atteindre une vitesse de 5km/h. Elle possède à l'avant et à l'arrière deux pare-chocs en mousse, capables d'arrêter immédiatement le robot si une collision est détectée. Pour finir, la plateforme est alimentée par une batterie lithium-ion, lui donnant une autonomie de 8h seule, réduite à 5h en prenant en compte la charge embarquée. Les dimensions du robot sont données sur la figure 4.2. Par conséquent, cette plateforme tout-terrain robuste a été sélectionnée pour réaliser les expérimentations en conditions réelles.

4.1.2 L'équipement embarqué

Sur cette plateforme 4MOB est installée une charge embarquée, contenant l'ordinateur de bord, ainsi que l'électronique nécessaire au fonctionnement des divers capteurs qui équipent le robot. Afin de mener à bien la tâche de navigation, le robot est doté de capteurs et de fonctionnalités lui offrant des capacités de perception proprioceptives (odométrie, centrale inertielle) et extéroceptives (capteurs laser).

Capteurs Laser LiDAR

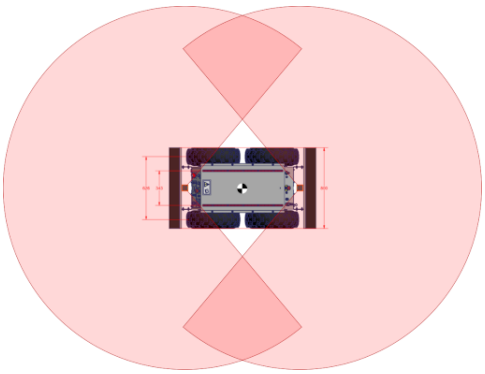


Figure 4.3: Disposition et portée angulaire des LiDARs sur la plateforme Air-Cobot

Le robot est équipé de deux LiDARs Hokuyo UTM-30LX situés à l'avant et à l'arrière du robot, à une vingtaine de centimètres du sol. Au niveau angulaire, ces LiDARs possèdent chacun un champ de vision à 270°, avec une résolution de 0.25°, Chaque LiDAR fournit donc 1080 mesures à chaque acquisition. Ils ont une portée de 30 mètres, avec une précision de l'ordre du demi-centimètre. Leur positionnement à l'avant et à l'arrière du robot permet d'obtenir une information LiDAR à 360° autour du robot. En revanche, le positionnement des LiDARs sur la plateforme entraîne la présence de deux zones aveugles sur les côtés droit et gauche du robot, mais celles-ci se révèlent non gênantes durant la navigation. Ces zones sont schématisées sur la figure 4.3. Ce modèle de LiDAR a été choisi spécifiquement pour sa résistance aux chocs et vibrations, ainsi que pour sa fiabilité en dépit des changements de luminosité extérieure. En outre, il est capable de fournir les informations à une fréquence de 40Hz. Cela en fait donc des capteurs fiables pour une utilisation dans un contexte agricole.

Système odométrique

Le robot possède deux systèmes proprioceptifs, lui permettant de calculer sa pose (odométrie) en temps réel relative à un repère de base. Le premier système se base sur des encodeurs relatifs liés à chaque roue. En connaissant la vitesse de rotation de

chaque roue, il est possible de connaître la vitesse linéaire et angulaire du robot, et en intégrant, connaître sa pose. De plus, le robot est doté d'une centrale inertielle (*Inertial Measurement Unit*, ou IMU), capable de donner la vitesse de rotation et l'accélération linéaire du robot sur les trois axes. Un filtre de Kalman étendu fusionne alors les données provenant de l'odométrie des roues ainsi que de l'IMU, et met à disposition une localisation plus précise du robot par rapport à un repère initial. Cependant, la cinématique du robot implique un glissement des roues lors de virages et de rotations. De plus, la nature des sols sur lequel le robot va être amené à évoluer est propice à des glissements et dérapages incontrôlés du robot. De ce fait, l'odométrie des roues n'est pas une information fiable sur de longues distances.

Ordinateur et système d'exploitation (ROS)

Le robot embarque un ordinateur tournant sous un système d'exploitation Ubuntu 16.04, sur lequel est installé le Middleware ROS Kinetic. Le Middleware ROS est une couche logicielle libre modulaire dont l'objectif est d'offrir une base *universelle* sur laquelle peut fonctionner n'importe quel robot. ROS permet ainsi de faire fonctionner des robots mobiles, des robots humanoïdes, des bras manipulateurs, etc. L'intérêt principal de ROS est la mise à disposition de paquets logiciels prêts à l'emploi, offrant des fonctionnalités telles que des pilotes de capteurs. Ainsi, ce logiciel est intéressant dans le monde de la recherche, car il permet une implémentation extrêmement rapide et un prototypage efficace de nos solutions. Le robot totalement équipé est visible sur l'image 4.4. Sur cette image, nous pouvons voir notamment le LiDAR avant, ainsi que les deux caméras stéréo avant utilisées pour le projet.

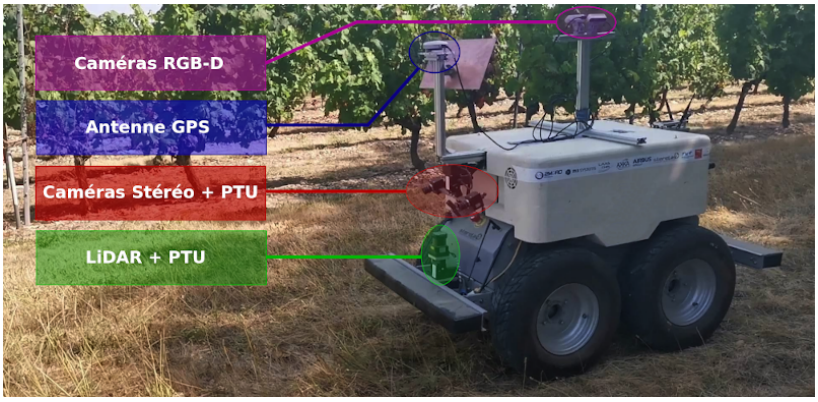


Figure 4.4: Plateforme robotique 4MOB équipé de ses capteurs

4.2 Évaluation et comparaison des méthodes existantes dans la *NavStack* de ROS dans trois environnements statiques de référence

Avant de procéder aux tests expérimentaux, nous avons d'abord porté notre solution sous ROS et avons confronté ses performances aux méthodes d'évitement d'obstacles statiques et de navigation locale implémentées au sein de la *Navigation Stack*[1]. Cette section présente les résultats ainsi obtenus dans différents environnements considérés comme difficiles à gérer par les algorithmes d'évitement d'obstacles, avec la présence de plusieurs obstacles de tailles et de formes différentes. Nous verrons notamment des cas d'obstacles concaves ainsi que le cas où sont présents de nombreux petits obstacles épars. Les méthodes présentes dans ROS, ainsi que notre méthode d'évitement d'obstacles statiques, vont être utilisées pour naviguer dans ces environnements exigeants dans le but de mettre en évidence leurs forces et leurs faiblesses. Ces tests en simulation seront un prélude à la réalisation d'évaluation en conditions expérimentales réelles. Nous analyserons donc en détail les trajectoires réalisées, les choix de navigation effectués, la complexité de mise en place des méthodes, ainsi que le temps de calcul nécessaire. Ces critères permettront notamment de dresser un bilan de ces différentes approches et de les comparer précisément. La première partie sera consacrée à la présentation de la *Navigation Stack* de ROS et de son fonctionnement général. La seconde partie proposera une comparaison entre ces méthodes et notre méthode de navigation basée spirale à l'aide de simulations effectuées sous ROS. Pour conclure, nous mettrons en avant les points forts et les points faibles de chaque méthode pour montrer en quoi notre méthode est adéquate dans un contexte agricole.

4.2.1 Présentation de la Navigation Stack de ROS

La Navigation Stack (en abrégé *NavStack*) est un ensemble de paquets ROS mettant à la disposition des utilisateurs des algorithmes de navigation pour un robot mobile. Le principe de la *NavStack* est d'offrir à l'utilisateur des outils de navigation facilement intégrables sur une plateforme robotique existante. Elle est hautement modulable pour fonctionner avec quasiment toutes les architectures de robots mobiles telle que TurtleBot. Différentes fonctionnalités sont possibles (construction de carte, planification de trajectoires, localisation SLAM, etc.).

La *NavStack* de ROS se base sur le principe de *costmaps*[2]. Il s'agit de la discrétisation d'une partie de l'environnement en une grille de cellules. Chaque cellule possède une valeur précise, déterminée à l'aide d'une carte fournie par l'utilisateur ou

[1] http://wiki.ros.org/navigation
[2] http://wiki.ros.org/costmap_2d

bien calculée à l'aide des données capteurs.

La *NavStack* de ROS possède ainsi des fonctionnalités permettant une localisation et une navigation au sein d'un environnement statique connu à l'aide d'une *costmap* globale. Cette dernière peut être fournie par l'utilisateur ou générée par le robot lui-même à l'aide d'un SLAM. Un planificateur de chemins global s'occupe ensuite de produire une trajectoire amenant le robot à son but en naviguant au sein de l'environnement connu. Plusieurs planificateurs de chemins globaux sont disponibles sous ROS. Ils sont analysés et comparés dans les articles [Cybulski et al., 2019] et [Filotheou et al., 2020].

La *NavStack* possède aussi des fonctionnalités permettant une navigation locale. Les informations fournies par les capteurs (principalement des LiDAR, mais aussi des caméras RGB-D) sont utilisées pour générer une *costmap* locale. Une trajectoire au sein de cette *costmap* locale est alors calculée à l'aide d'un planificateur de chemins local. La figure 4.5 montre un diagramme avec les différents éléments composant la *Navigation Stack* de ROS.

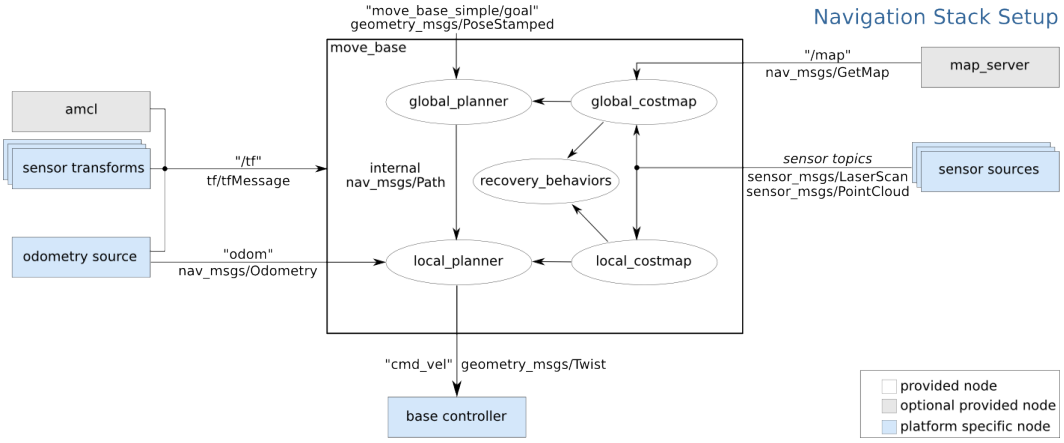


Figure 4.5: La *Navigation Stack* de ROS

Un environnement agricole étant un espace fortement variable, nous avons choisi une stratégie de navigation locale basée sur la commande référencée capteurs. Par conséquent, nous nous concentrerons uniquement sur l'analyse des fonctionnalités locales de la *NavStack* et de deux de ses planificateurs de chemins locaux : il s'agit du paquet `dwa_local_planner`, basé sur l'algorithme DWA (Dynamic Windows Approach) [Fox et al., 1997], et du paquet `teb_local_planner`, basé sur l'algorithme TEB (Time-Elastic Bands) [Rösmann et al., 2012], [Rosmann et al., 2013].

Génération de la *costmap locale*

Lors de la navigation, la *NavStack* garde en mémoire une *costmap locale* qui est mise à jour à chaque nouvelle acquisition fournie par les capteurs[3]. Il s'agit d'une grille de cellules centrée sur le robot. Chaque cellule possède une valeur précise, déterminée par inflation des données laser acquises.

L'inflation est le processus de propagation des valeurs de coût à partir des cellules occupées en fonction de la distance à ces cellules. Pour cela, le coût d'une cellule est défini par cinq symboles spécifiques :

- Le coût "Lethal" (*Mortel*) signifie qu'il y a un obstacle réel (espace de travail) dans une cellule. Donc si le centre du robot se trouve dans cette cellule, alors le robot est évidemment en collision.
- Le coût "Inscribed" (*Inscrit*) signifie qu'une cellule est plus proche d'un obstacle que la circonférence du robot. Le robot est donc certainement en collision avec un obstacle si le centre du robot se trouve dans une cellule dont le coût est égal ou supérieur au coût inscrit.
- Le coût "Possibly Circumscribed" (*Éventuellement circonscrit*) est similaire au coût "inscrit", mais en utilisant le rayon circonscrit du robot comme distance de coupure. Ainsi, si le centre du robot se trouve dans une cellule égale ou supérieure à cette valeur, la collision avec un obstacle ou non dépend de l'orientation du robot.
- Le coût "Freespace" (*Espace Libre*) est supposé être nul, ce qui signifie que rien ne devrait empêcher le robot de s'y rendre.
- Un coût "Unknown" (*Inconnu*) signifie qu'il n'y a pas d'information sur une cellule donnée.

Toutes les autres cellules se voient attribuer un coût dont la valeur se trouve entre "Freespace" et "Possibly Circumscribed" en fonction de leur distance par rapport à une cellule "Lethal" et de la fonction de décroissance fournie par l'utilisateur.

Cette *costmap* est ensuite utilisée par un planificateur de chemins local afin de générer une trajectoire permettant la navigation. Dans notre cas, deux algorithmes seront plus particulièrement utilisés : DWA et TEB.

`dwa_local_planner`

Le paquet `dwa_local_planner`[4] est basé sur le travail effectué dans [Fox et al., 1997]. L'algorithme fournit un contrôleur qui pilote une base mobile dans le plan.

[3] http://wiki.ros.org/costmap_2d
[4] http://wiki.ros.org/dwa_local_planner

L'idée de base de l'approche des fenêtres dynamiques (Dynamic Window Approach, DWA) est résumée par les étapes suivantes :

1. L'espace de contrôle (v, ω) du robot est échantillonné.
2. Pour chaque vitesse échantillonnée, une simulation est effectuée à partir de l'état courant du robot afin de prédire sa trajectoire si la vitesse échantillonnée lui était appliquée durant un court laps de temps.
3. Chaque trajectoire est alors évaluée à l'aide d'une fonction d'évaluation prenant en compte des caractéristiques telles que la distance aux obstacles, la vitesse, ou bien la proximité au but. Les trajectoires menant à une collision sont ainsi automatiquement rejetées.
4. La trajectoire résultant en la meilleure évaluation est sélectionnée, et les commandes de vitesses associées sont envoyées au robot.

Dans le paquet `dwa_local_planner`, la fonction de coût est définie comme suit :

```
cost = path_distance_bias * D1 + goal_distance_bias * D2 + occdist_scale * M
```

Cette fonction de coût est calculée pour chaque trajectoire locale à partir de trois variables extraites de l'environnement et de la *costmap* :

- `D1` : Distance en mètres entre le chemin initial et la position de fin de la trajectoire locale.
- `D2` : Distance en mètres entre le but et la position de fin de la trajectoire locale.
- `M` : Coût maximum de l'obstacle le long de la trajectoire locale.

Elle prend aussi en compte trois paramètres qui doivent être réglés par l'utilisateur :

- `path_distance_bias` : Ce paramètre influe sur la précision avec laquelle le contrôleur doit essayer de rester proche du chemin qui lui a été fourni.
- `goal_distance_bias` : Ce paramètre influe sur la précision avec laquelle le contrôleur doit essayer d'atteindre son objectif local.
- `occdist_scale` : Ce paramètre influe sur l'importance que le contrôleur doit donner à l'évitement des obstacles.

La figure 4.6 montre un schéma du fonctionnement de l'algorithme DWA, avec deux obstacles simples représentés en rouge. Les lignes en pointillés représentent les différentes trajectoires simulées.

Cet algorithme possède plusieurs paramètres qui nécessitent un réglage fin dépendant du robot, de l'environnement et de la tâche. Des guides peuvent être trouvés pour régler ces paramètres[5].

[5] http://kaiyuzheng.me/documents/navguide.pdf

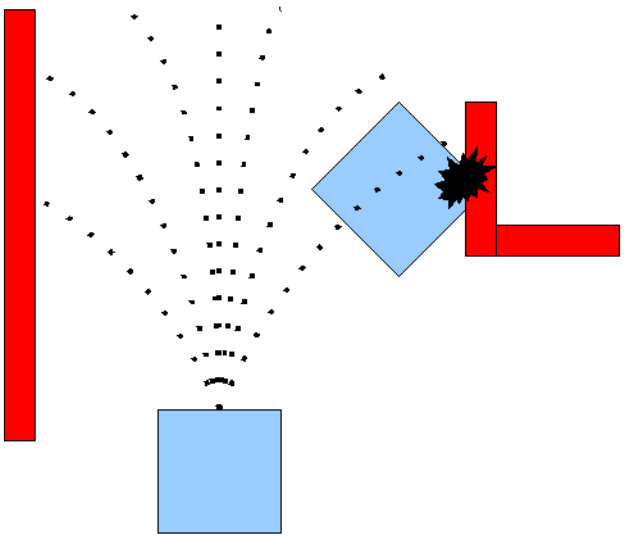


Figure 4.6: Schéma de fonctionnement de l'algorithme DWA

`teb_local_planner`

La méthode TEB (Time-Elastic-Band) a été introduite dans [Rösmann et al., 2012] et [Rosmann et al., 2013] et implémentée dans le paquet `teb_local_planner`. Ce paquet met en œuvre un planificateur de chemins local optimal en ligne pour la navigation et le contrôle des robots mobiles. La trajectoire initiale est générée par un planificateur global. Une trajectoire locale est ensuite optimisée pendant l'exécution en fonction de plusieurs contraintes : la minimisation du temps d'exécution de la trajectoire (objectif temporel optimal), l'évitement des obstacles éventuellement présents, et la conformité aux contraintes cinématiques et dynamiques du robot.

La trajectoire optimale est obtenue en résolvant un problème d'optimisation avec une fonction de coût multi-objectifs. L'utilisateur peut fournir des pondérations permettant de spécifier le comportement en cas d'objectifs conflictuels. Ces pondérations peuvent prendre en compte la distance aux obstacles et la cinématique du robot. La méthode TEB permet notamment de prendre en compte tout types de robots mobiles, même non-holonome.

Différences entre les méthodes DWA et TEB

Les méthodes DWA et TEB reposent toutes deux sur un principe similaire : planifier une trajectoire locale selon un horizon prédéfini qui minimise une fonction de coût. Trouver cette trajectoire nécessite de résoudre un problème d'optimisation complexe. Les méthodes DWA et TEB diffèrent notamment sur l'approche utilisée pour déterminer cette solution :

- La méthode DWA discrétise l'espace des vitesses du robot et simule les trajectoires selon un horizon précis. La précision de cette discrétisation est un paramètre laissé au choix de l'utilisateur. La fonction de coût, dont les poids sont aussi au choix de l'utilisateur, est discrète et opère directement à partir des coûts indiqués dans la *costmap*.

- Contrairement à l'approche DWA, la méthode TEB repose sur une fonction de coût continue. Ainsi, elle ne gère donc pas directement les *costmaps*. La solution implémentée dans ROS actuellement par le paquet `teb_local_planner` consiste à considérer chaque cellule d'obstacle "Lethal" comme un obstacle ponctuel, ce qui limite l'approche à des *costmaps* locales de petite/moyenne taille et à des résolutions assez grossières.

4.2.2 Mise en place des environnements de simulation

Cette section présente les simulations réalisées sous ROS Kinetic à l'aide du simulateur Gazebo 7. La base robotique utilisée est celle du robot TurtleBot, dont le modèle est fourni dans ROS. Cette base robotique est équipée d'un LiDAR qui fournit une information frontale sur 270°.

Cette section vise à comparer notre méthode d'évitement d'obstacles avec les planificateurs locaux `dwa_local_planner` et `teb_local_planner`. Les comparaisons seront notamment basées sur les éléments suivants :

- Réalisation de la tâche de navigation (le robot atteint son but sans collision).

- Temps mis à accomplir la tâche de navigation.

- Distance parcourue.

- Plus petite distance à un obstacle.

Présentation des environnements de simulation

Notre méthode ainsi que les méthodes DWA et TEB sont connues pour fonctionner correctement dans les cas d'obstacles statiques convexes. Ainsi, pour mettre en exergue les différences entre notre méthode d'évitement basée spirales et les méthodes DWA et TEB présentes dans la *NavStack* de ROS, deux environnements ont été simulés et sont visibles figure 4.7 :

- Le premier environnement (figure 4.7a) représente le cas typique d'un obstacle concave présentant un creux, mettant en échec les méthodes d'évitement d'obstacles telles que les champs de potentiels [Koren and Borenstein, 1991].

- Le deuxième environnement (figure 4.7b) représente une dissémination de petits obstacles circulaires. Ces obstacles ne sont pas éloignés de plus de cinq mètres les uns des autres. Ainsi, si nous donnons une distance de sécurité de 2.5 mètres, le robot ne sera pas censé essayer de passer entre deux de ces obstacles.

- Le troisième environnement (figure 4.7c) est un environnement plus grand et plus complexe, regroupant plusieurs obstacles de diverses formes et tailles. À des fins de description, ces obstacles ont été rassemblés en quatre groupes. Le groupe (A) représente deux petits obstacles circulaires. Le groupe (B) représente un obstacle longiligne. L'obstacle (C) est un gros obstacle pouvant modéliser un bâtiment. Pour finir le groupe (D) représente un ensemble d'obstacles rapprochés formant une concavité. Cet environnement, de par sa taille et sa complexité, se rapproche donc d'un cas réel que le robot rencontrera au sein d'une exploitation agricole. Il est également similaire aux environnements considérés par ailleurs dans le manuscrit.

Lors des simulations, le robot partira du point d'origine O_w et devra atteindre la position O_g dont les coordonnées en mètres dans le repère O_w sont $[15, 0]^T$ pour les environnements 1 et 2, et $[30, 0]^T$ pour l'environnement 3.

Choix des paramètres de notre méthode de navigation basée spirale

Afin de régler les paramètres de notre méthode de navigation, nous utilisons les valeurs suivantes : la vitesse linéaire v^* a été fixée à 0,3 $m.s^{-1}$. Pour assurer une distance de sécurité suffisante compte tenu de la taille du robot (environ 150x80cm), $d^* = 2.5m$ est choisi. Les gains pour les contrôleurs ω_A et ω_B sont fixés comme suit : $\lambda_1 = 0,1$, $\lambda_2 = 0,1$, $\lambda_S = 0,5$. Le paramètre n est fixé à $n = 5$. Le seuil de commutation e_α^{switch} est sélectionné comme $e_\alpha^{switch} = \frac{\pi}{12}$. Pour cette expérimentation, l'algorithme de choix du SCP basé sur un calcul de projeté est utilisé.

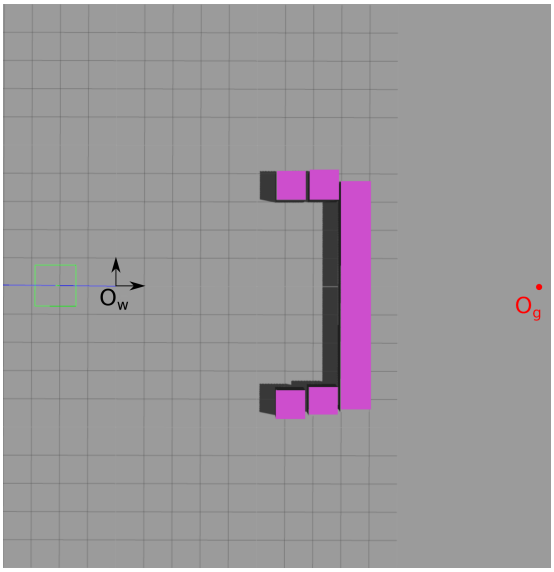
4.2.3 Comparaison avec "dwa local planner"

Génération des *costmaps* et réglages des paramètres

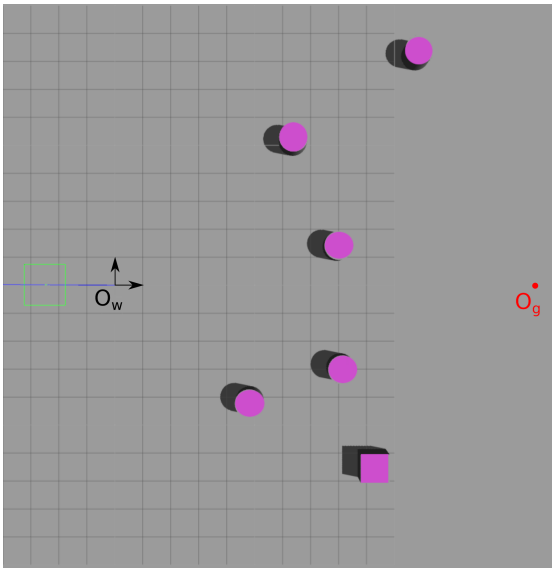
Lors de la navigation, la *NavStack* maintient à jour deux *costmaps* différentes, la *costmap locale* et la *costmap globale*[6]. Le but étant de comparer les méthodes réactives, la *costmap globale* est maintenue totalement vide. Afin d'obtenir un comportement proche du comportement nominal de notre méthode, les paramètres suivants sont utilisés afin de régler la *costmap* locale :

```
robot_radius: 1
obstacle_range: 5
```

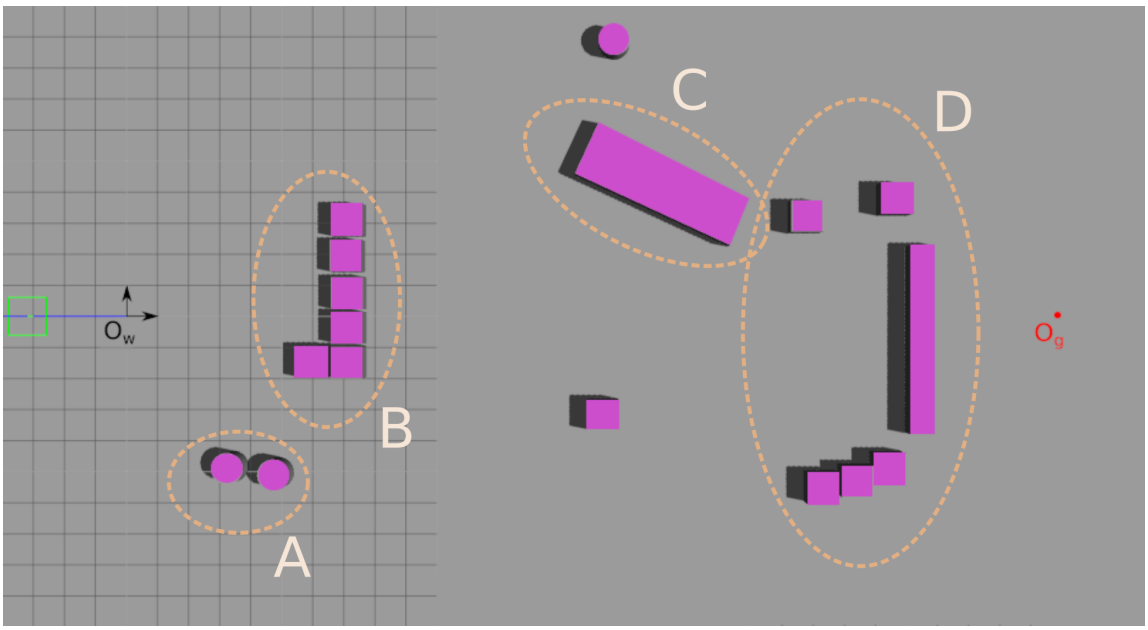
[6] http://wiki.ros.org/costmap_2d



(a) Environnement 1



(b) Environnement 2



(c) Environnement 3

Figure 4.7: Les trois environnements de simulation

```
raytrace_range: 6.0
inflation_radius: 2.5
```

Ces réglages impliquent que les obstacles sont pris en compte à partir d'une distance de cinq mètres (`obstacle_range`)

Le planificateur `dwa_local_planner` nécessite aussi un fichier de configuration. Afin d'obtenir un comportement similaire en vitesses pour les deux méthodes, nous prenons comme paramètres :

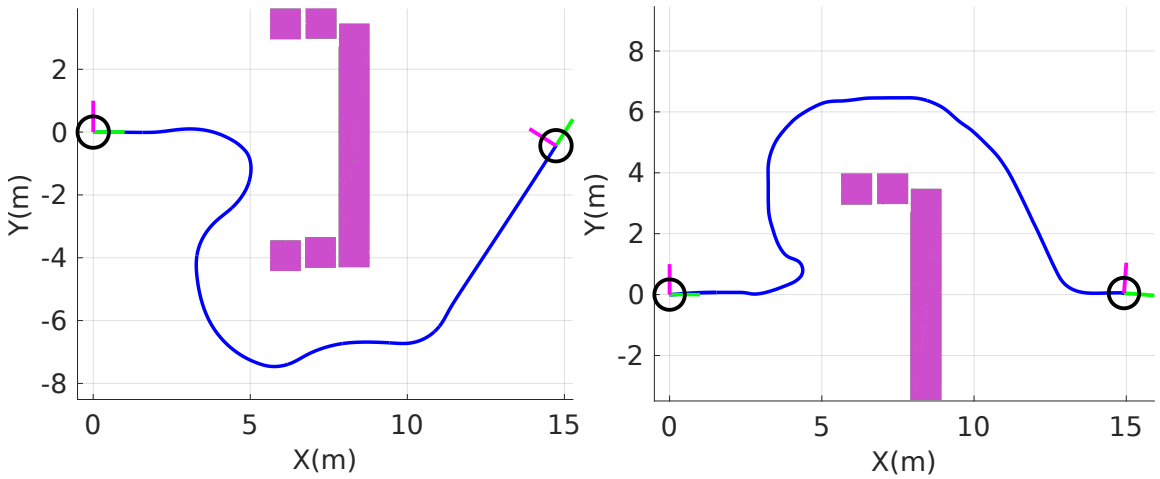
```
DWAPlannerROS:
  max_vel_x: 0.3
  min_vel_x: 0.3
  max_vel_theta: 1
  min_in_place_vel_theta: 0.4
  acc_lim_theta: 3.2
  acc_lim_x: 2.5
  acc_lim_y: 2.5
  planner_frequency: 20.0
  path_distance_bias: 32.0
  goal_distance_bias: 20.0
  occdist_scale: 200
```

Les valeurs des paramètres `path_distance_bias` et `goal_distance_bias` sont celles données par défaut pour le robot TurtleBot. Le paramètre `occdist_scale`, a été réglé à 200 dans le but de privilégier l'évitement de l'obstacle dans le choix de la trajectoire.

Pour chacun des environnements, une simulation est effectuée en utilisant l'algorithme DWA, puis en utilisant notre méthode d'évitement d'obstacles statiques basée spirale (Méthode SA, *Spiral Avoidance*). Nous allons analyser les résultats pour chaque environnement

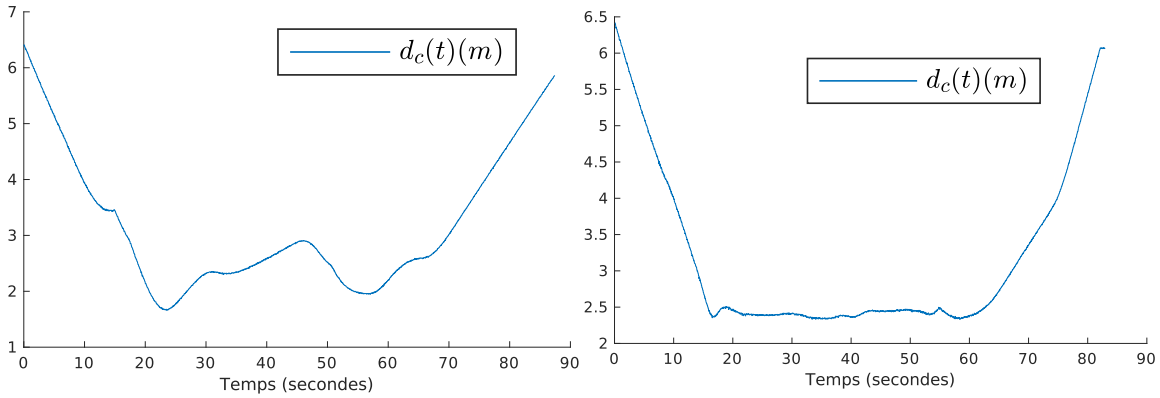
Premier environnement - Obstacle concave

Les figures 4.8a et 4.8b montrent en bleu la trajectoire effectuée par le robot, respectivement en utilisant notre algorithme d'évitement en spirale et en utilisant l'algorithme DWA présent dans la *NavStack* de ROS. Les figures 4.9a et 4.9b présentent l'évolution de la distance d_c entre le robot et le point LiDAR le plus proche détecté, respectivement lors de l'utilisation de l'algorithme SA et de l'algorithme DWA. Pour finir, les figures 4.10a et 4.10b montrent la commande angulaire produite par les algorithmes SA et DWA.



(a) Trajectoire du robot (Méthode SA) (b) Trajectoire du robot (Méthode DWA)

Figure 4.8: Trajectoires réalisées par le robot dans l'environnement 1, guidé par les algorithmes SA et DWA

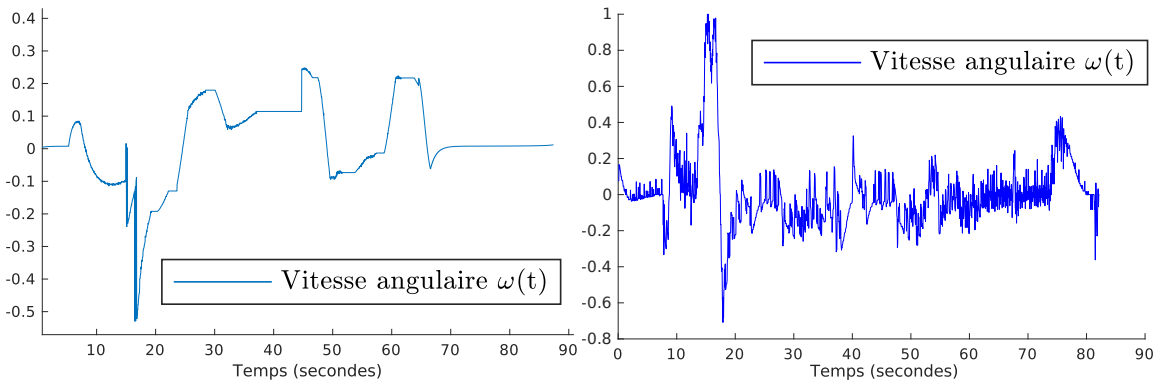


(a) Évolution de d_c (Méthode SA) (b) Évolution de d_c (Méthode DWA)

Figure 4.9: Évolution de la distance d_c entre le robot et plus proche point LiDAR détecté

Nous pouvoir voir sur la figure 4.8 que pour les deux cas le robot parvient à son but, tout en évitant l'obstacle concave présent sur son chemin. Pour l'algorithme SA (figure 4.8a), l'objectif était une position (coordonnées dans le repère O_w), tandis que l'algorithme DWA (figure 4.8b) prend nativement en entrée une pose (position dans le repère O_w et orientation). Une orientation nulle par rapport au repère O_w est alors demandée dans ce cas. De par l'analyse des figures 4.8, 4.9 et 4.10, certaines différences et similarités sont à relever :

- **Sens de contournement** : Nous remarquons que pour la méthode SA, l'évitement de l'obstacle se fait dans un sens anti-horaire, tandis que pour



(a) Commande $\omega(t)$ (Méthode SA) (b) Commande $\omega(t)$ (Méthode DWA)

Figure 4.10: Commande de vitesse angulaire $\omega(t)$

la méthode DWA, l'évitement se fait dans un sens horaire. D'après la figure 4.8, l'obstacle est relativement symétrique horizontalement. Par conséquent, cette différence dans le sens de contournement peut s'expliquer par les conditions différentes pour les deux méthodes. Pour la méthode SA, le choix s'explique par la prise en considération de la forme de l'obstacle lors du calcul de la position du barycentre à l'instant du déclenchement de l'évitement. Dans ce cas ce dernier choisit un sens de contournement par la droite. La méthode DWA quant à elle sélectionne la trajectoire qui minimise le coût au travers de la *costmap*, sans choisir explicitement de sens de contournement.

- **Distance à l'obstacle** : La figure 4.9 montre la distance à l'obstacle le plus proche. Nous pouvons remarquer que pour la méthode SA, le robot se rapproche jusqu'à une distance $d = 2m$ de l'obstacle, au creux de la concavité. En revanche, pour la méthode DWA, le robot parvient à rester à une distance de $d = 2.5m$ de l'obstacle tout au long de l'évitement.

- **Consigne angulaire** $\omega(t)$: Parallèlement à l'observation effectuée dans le point précédent, nous pouvons remarquer sur la figure 4.10 qu'en valeur absolue, la méthode DWA produit une consigne angulaire maximale $\omega(t) \simeq 1rad/s$ à $t \simeq 16s$. De son côté, la méthode SA produit une consigne angulaire presque moitié moindre, proche de $\omega(t) \simeq 0.5rad/s$. Ces comportements résultent d'un compromis que l'utilisateur doit faire lors de la mise en place des méthodes. Ce compromis passe par le paramétrage des gains λ_1, λ_2 et λ_S pour la méthode basée spirale, et par le choix de paramètres pertinents pour la méthode DWA.

- **Distance parcourue et temps de navigation** : La vitesse linéaire étant similaire pour les deux méthodes, nous pouvons voir que le temps de parcours est sensiblement le même lors des deux expérimentations.

- **Fréquence de fonctionnement** : Les simulations ont été lancées sur un

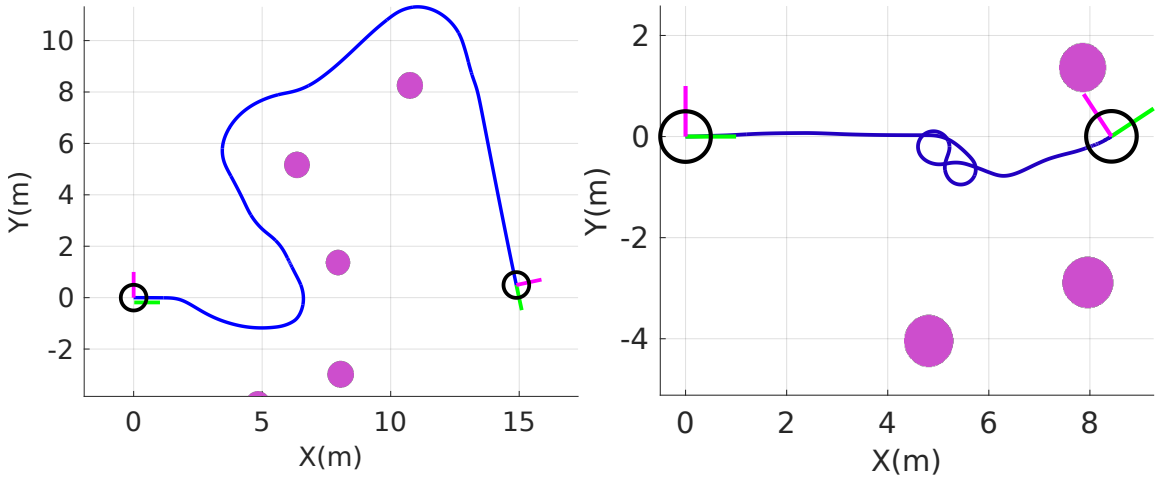
ordinateur possédant un processeur Intel Core i7-7820HQ (Quad Core 2.90GHz). Sur cette machine, la boucle responsable du calcul de la loi de commande SA était en mesure de tourner à une fréquence de 40Hz, tandis que la méthode basée DWA restait limitée à 20Hz. La méthode DWA s'avère ainsi plus coûteuse en temps de calcul, notamment sur des espaces et des robots de taille importante. Ainsi, dans un contexte agricole extérieur avec de gros robots, la méthode DWA nécessitera plus de puissance de calcul que notre méthode basée spirale.

Pour conclure, cette expérimentation met en avant un comportement similaire entre la méthode DWA et notre méthode SA.

Deuxième environnement - Regroupement de petits obstacles

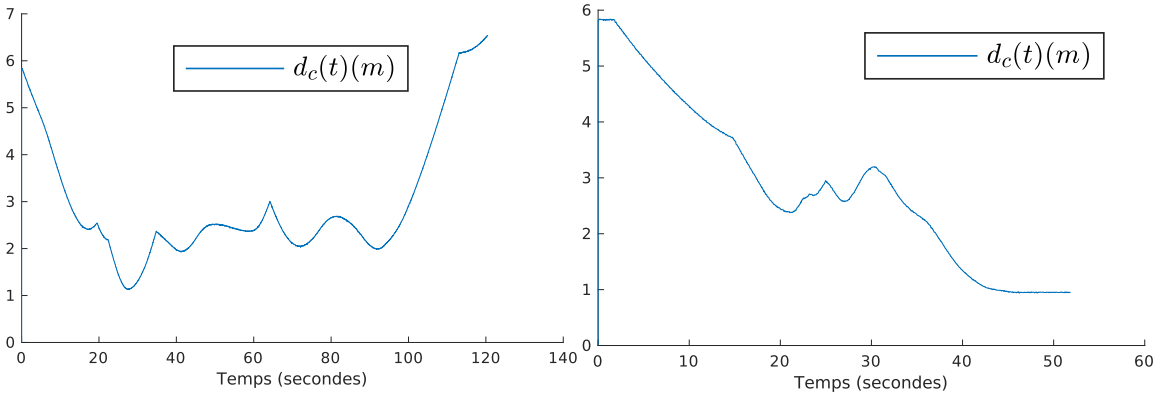
Les figures 4.11a et 4.11b montrent en bleu la trajectoire effectuée par le robot, respectivement en utilisant notre algorithme d'évitement en spirale et l'algorithme DWA présent dans la *NavStack* de ROS. Les figures 4.12a et 4.12b présentent l'évolution de la distance d_c entre le robot et le point LiDAR le plus proche détecté, respectivement lors de l'utilisation de l'algorithme SA et de l'algorithme DWA. Pour finir, les figures 4.13a et 4.13b montrent la sortie en commande angulaire des algorithmes SA et DWA.

Cet environnement, visible sur la figure 4.7b, possède la particularité d'être composé de petits obstacles disséminés, placés à une distance inférieure à cinq mètres les uns des autres, formant une sorte de concavité.



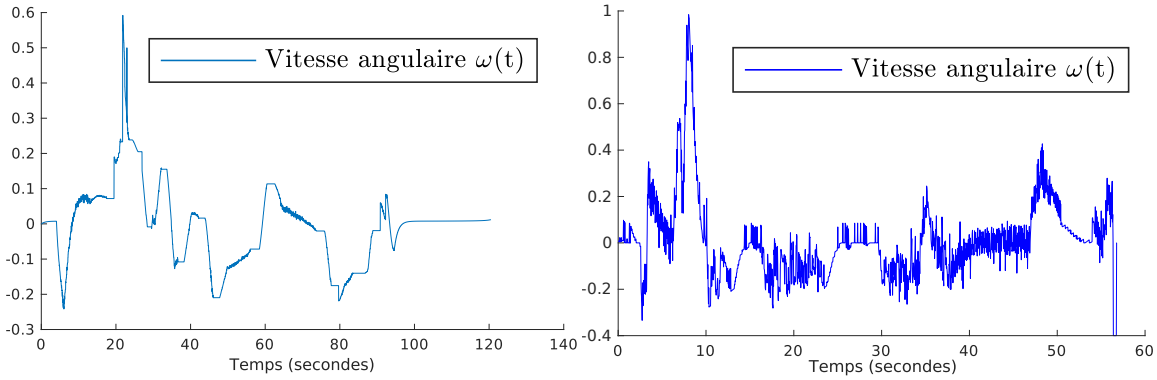
(a) Trajectoire du robot (Méthode SA) (b) Trajectoire du robot (Méthode DWA)

Figure 4.11: Trajectoires réalisées par le robot dans l'environnement 2, guidé par les algorithmes SA et DWA



(a) Évolution de d_c (Méthode SA) (b) Évolution de d_c (Méthode DWA)

Figure 4.12: Évolution de la distance d_c entre le robot et plus proche point LiDAR détecté



(a) Commande $\omega(t)$ (Méthode SA) (b) Commande $\omega(t)$ (Méthode DWA)

Figure 4.13: Commande de vitesse angulaire $\omega(t)$

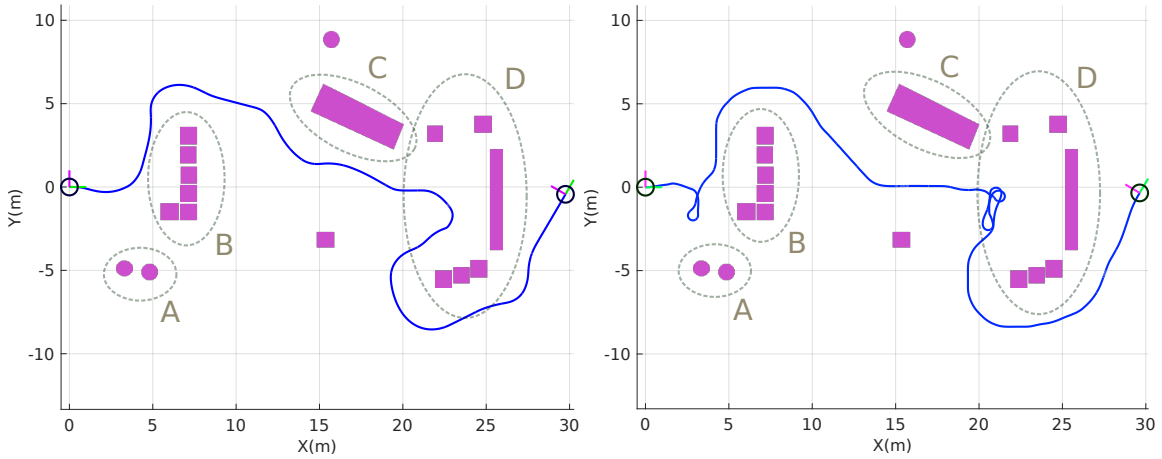
La figure 4.11a montre que la méthode SA parvient à atteindre son but en $t \simeq 100s$. Les figures 4.12a et 4.13a montrent que malgré la complexité de l'environnement, la navigation est effectuée en ne s'approchant pas à moins de $d \simeq 1.2m$ d'un obstacle, tout en gardant une consigne angulaire ne dépassant pas 0.6rad/s.

En revanche, il peut être vu sur la figure 4.11b qu'en utilisant la méthode DWA, le robot ne parvient pas à accomplir de façon correcte la tâche de navigation : il n'atteint pas son but. En effet, nous remarquons qu'arrivé au centre de la concavité formée par les trois petits obstacles ponctuels, le robot adopte un comportement oscillatoire. Dans le cas où le robot se retrouve bloqué, le paquet `dwa_local_planner` met en application deux stratégies de récupération. La première consiste

à faire tourner le robot sur lui-même afin de trouver les espaces libres[7]. La seconde consiste à réinitialiser les *costmaps* présentes en mémoire[8]. Comme nous sommes dans un cas purement réactif dans laquelle une carte globale n'est pas fournie ni maintenue, ces stratégies ne permettent pas de sortir le robot du minimum local, et donnent lieu à des oscillations comme observé dans cette expérimentation. Il finit alors par s'approcher d'un des obstacles à une distance inférieure à 1 mètre. L'algorithme DWA échoue alors à planifier une trajectoire valide. Le robot s'arrête donc. Notre méthode de par sa construction gère mieux la multiplicité d'obstacles, notamment grâce à un choix pertinent du centre des spirales, basées sur un calcul de projeté, permettant de prendre en considération l'enveloppe convexe reliant deux obstacles suffisamment proches.

Troisième environnement

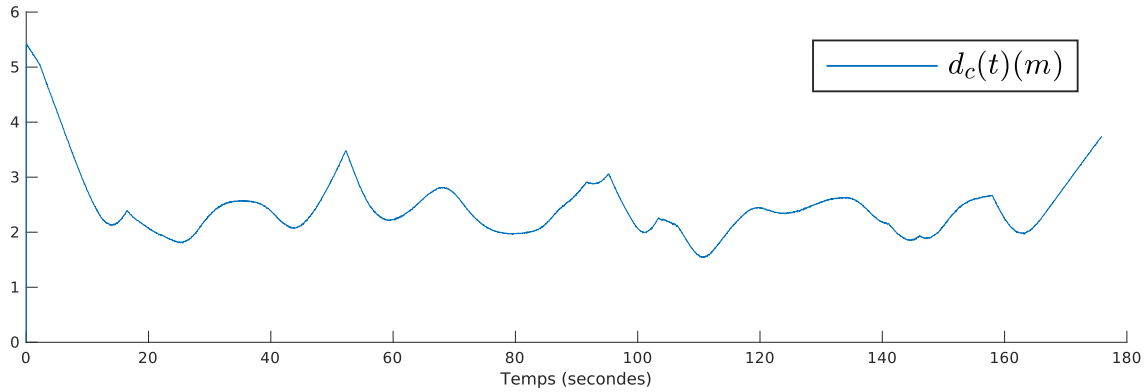
À l'instar des simulations précédentes, les figures 4.14a et 4.14b montrent en bleu la trajectoire effectuée par le robot dans l'environnement. Les obstacles présents sont modélisés en violet. Cet environnement est plus grand et complexe que les deux précédents. Il possède de nombreux obstacles ponctuels, qui peuvent être ou bien former des concavités. Les figures 4.15a et 4.15b présentent l'évolution de la distance d_c entre le robot et le point LiDAR le plus proche détecté, et les figures 4.16a et 4.16b montrent la commande angulaire calculée par les algorithmes SA et DWA.



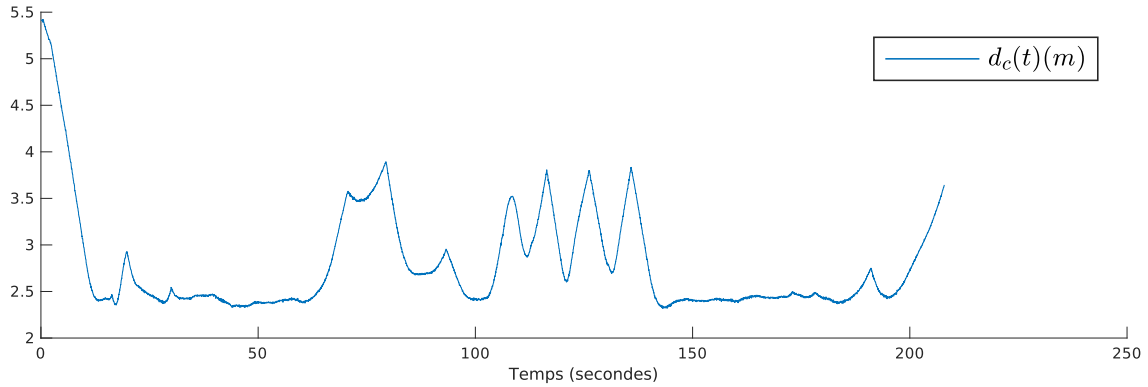
(a) Trajectoire du robot (Méthode SA) (b) Trajectoire du robot (Méthode DWA)

Figure 4.14: Trajectoires réalisées par le robot dans l'environnement 3, guidé par les algorithmes SA et DWA

[7]http://wiki.ros.org/rotate_recovery
[8]http://wiki.ros.org/clear_costmap_recovery



(a) Évolution de d_c (Méthode SA)



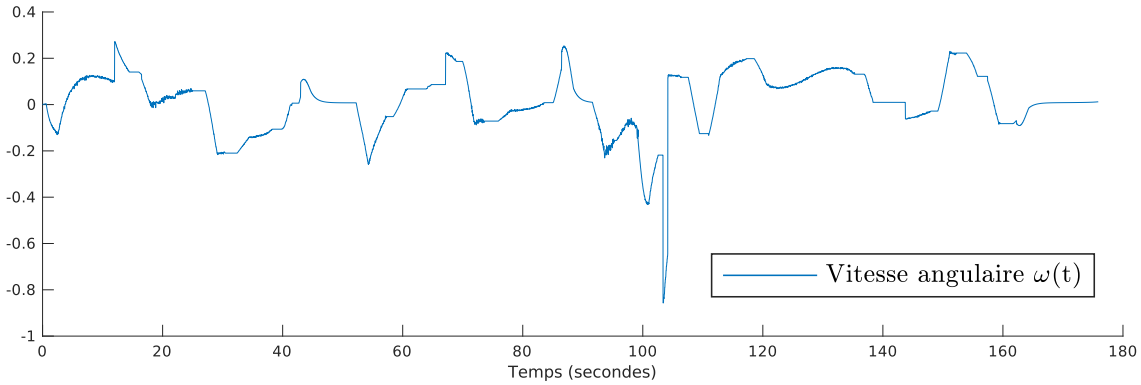
(b) Évolution de d_c (Méthode DWA)

Figure 4.15: Évolution de la distance d_c entre le robot et plus proche point LiDAR détecté

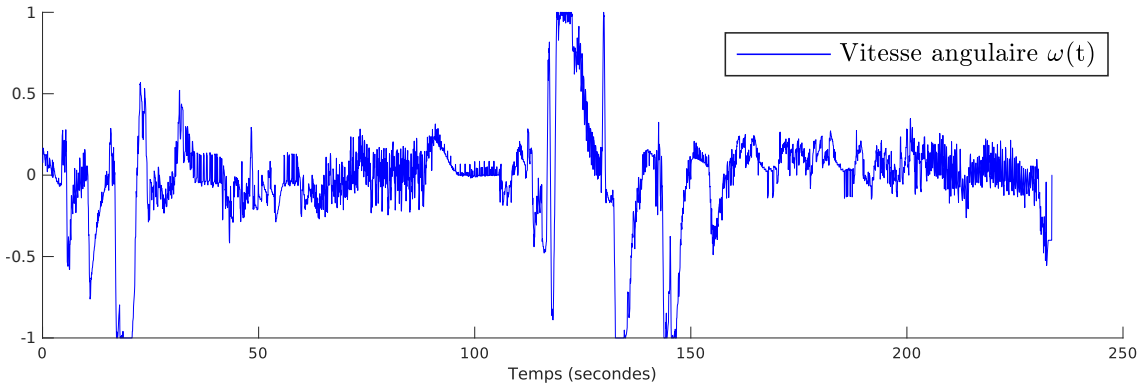
Nous pouvons voir sur la figure 4.14 que dans les deux cas, le robot parvient correctement à son objectif en contournant les divers obstacles présents dans son environnement. Les figures 4.15 et 4.16 montrent respectivement l'évolution de d_c et la commande angulaire générée pour chacune des méthodes. Nous voyons qu'avec les paramètres utilisées pour les méthodes SA et DWA, la distance d_c reste plus proche de la distance désirée $d^* = 2.5m$ pour la méthode DWA. Cependant, cela se fait en contrepartie de commandes angulaires plus fortes. La figure 4.16b montre que la méthode DWA produit plusieurs pics de vitesse angulaire à ± 1 rad/s tandis que la consigne maximale de la méthode SA est d'environ 0.8 rad/s en valeur absolue.

Afin d'analyser plus précisément ces simulations, nous allons décrire le déroulement obstacle par obstacle.

Évitement des groupes (A) et (B) La simulation démarre à $t = 0$, à environ 5.5 mètres de l'obstacle (B). Nous pouvons voir sur la figure 4.14 que dans les deux cas, le



(a) Commande $\omega(t)$ (Méthode SA)



(b) Commande $\omega(t)$ (Méthode DWA)

Figure 4.16: Commande de vitesse angulaire $\omega(t)$

robot suit immédiatement une trajectoire d'évitement. Dans le cas de la méthode SA, un choix explicite du sens de contournement est fait, basé sur la position du barycentre à l'instant de décision. La méthode SA va prendre ici en compte la forme de l'obstacle (B), ainsi que les éléments du groupe (A), et choisir un sens de contournement horaire, en passant par la gauche de l'obstacle (B). Nous remarquons que la méthode DWA produit quant à elle une trajectoire qui commence à contourner le groupement (B) par la droite, pour au final effectuer un demi-tour et éviter l'obstacle (B) dans le sens horaire, comme pour la méthode SA. En effet, la méthode DWA génère une trajectoire en exploitant la connaissance locale de l'environnement. Au début de la mission, cette trajectoire passe par la droite de l'obstacle (B). Cependant, une fois l'obstacle (A) en approche, la méthode détermine que la trajectoire impliquant de contourner l'obstacle (A) est plus coûteuse que celle impliquant de faire demi-tour, ce qui explique la manœuvre effectuée.

Évitement de l'obstacle (C) Une fois l'obstacle (B) évité, les méthodes DWA et SA suivent deux trajectoires quelque peu différentes. La méthode SA dirige le robot

vers le but, tandis que la méthode DWA dirige le robot vers la trajectoire globale générée initialement, et qui était une ligne droite entre la position initiale du robot et le but. Cela a pour conséquence que le robot reste éloigné du l'obstacle (C) pour la méthode DWA, alors que la trajectoire générée par la méthode SA va nécessiter d'éviter l'obstacle (C), impliquant seulement un léger décalage du robot.

Évitement du groupe (D) Le dernier élément à éviter est le groupe (D). Ce dernier est composé de plusieurs obstacles formant une concavité. Nous voyons que dans le cas de la méthode SA, le choix pertinent d'un sens de contournement horaire est effectué. Pour la méthode DWA, nous nous retrouvons dans un cas proche de celui de l'évitement des obstacles (A) et (B) : les évaluations successives proposent des trajectoires impliquant de contourner l'obstacle (D) tantôt dans le sens horaire, tantôt dans le sens anti-horaire. Cela entraîne une phase d'oscillation entre $t \simeq 100s$ et $t \simeq 140s$ (cf. figures 4.15b, 4.16b) au sein de la concavité de l'obstacle (D), dont le robot parvient enfin à sortir à $t \simeq 150s$.

Ces éléments ainsi que les figures 4.15 et 4.16 renforcent certains points déjà évoqués dans les deux précédents environnements et mettent en exergue certains nouveaux comportements :

- **Sens de contournement** : Les deux simulations effectuées sur cet environnement montrent que notre méthode SA est capable de prendre en compte la forme des obstacles afin de déterminer un sens de contournement pertinent au moment de l'évitement à l'aide des informations disponibles sur l'environnement à cet instant. Elle fixe un sens de contournement et le maintient jusqu'à avoir éviter l'obstacle. Par conséquent, étant donné que notre méthode assure une vitesse linéaire positive non nulle, cela permet d'éviter de faire des demi-tours intempestifs ou bien de se retrouver bloqué dans une concavité.

- **Trajectoire et commande angulaire** : Les figures 4.15 et 4.16 montrent respectivement l'évolution de d_c et la commande angulaire générée pour chacune des méthodes. Nous voyons qu'avec les paramètres utilisées pour les méthodes SA et DWA, la distance d_c reste davantage proche de la distance désirée $d^* = 2.5m$ pour la méthode DWA, alors que la méthode SA s'approche à une distance de $d \simeq 2m$. Cependant, cela se fait en contrepartie de commandes angulaires plus élevées. La figure 4.16b montre que la méthode DWA produit plusieurs pics de vitesse angulaire à ± 1 rad/s tandis que la consigne maximale de la méthode SA est d'environ 0.8 rad/s en valeur absolue. Ce compromis entre vitesse angulaire maximale et proximité aux obstacles peut être réglé à l'aide des paramètres pour la méthode DWA, et des gains des contrôleurs pour la méthode SA. Pour finir, nous pouvons notamment voir qu'à vitesse linéaire équivalente, notre méthode SA permet au robot d'atteindre son but en environ 180 secondes, tandis qu'il faut environ 230 secondes avec la méthode DWA.

Cette simulation montre que notre méthode permet de naviguer dans un environnement complexe possédant plusieurs obstacles de tailles et de formes différentes. Notre approche consiste à opérer à vitesse linéaire constante et à choisir un sens de contournement fixe jusqu'à la fin de l'évitement. Ainsi, il est possible d'éviter les grands obstacles tout en empêchant le robot de se retrouver bloqué dans des zones concaves. Cela a pour conséquence de produire une trajectoire plus courte qu'avec la méthode DWA. Notre méthode SA produit de plus une consigne angulaire restant dans des limites acceptables, tout en restant à une distance des obstacles suffisante pour assurer la non collision.

Analyses et conclusion

Ces simulations ont ainsi permis de comparer notre technique d'évitement d'obstacles avec la méthode DWA implémentée dans la *NavStack*. Plusieurs points peuvent être extraits de ces comparaisons. Les deux premiers environnements considérés mettent en évidence les cas nominaux d'un objet concave et d'une scène avec plusieurs objets, tandis que le troisième se rapproche davantage d'un environnement agricole réel. Pour ces trois environnements, la comparaison a porté sur les points suivants :

- **Trajectoires et commandes générées** : Nous avons vu sur le premier environnement que les trajectoires générées par la méthode DWA et par la méthode SA sont similaires en plusieurs points. Pour cet environnement, les deux simulations diffèrent légèrement sur le temps de parcours, la commande angulaire maximale envoyée au robot ainsi que la distance la plus proche à un obstacle. Cependant, ces différences peuvent s'expliquer par les différents gains et paramètres fixés a priori. En effet, un compromis doit être trouvé entre la distance désirée à l'obstacle, la commande angulaire maximale et le temps de parcours. Pour le deuxième environnement, notre méthode SA est capable de générer une trajectoire contournant l'ensemble des obstacles ponctuels, modifiant le centre des spirales le long de l'enveloppe convexe formée autour de ces obstacles, tandis que la méthode DWA se retrouve bloquée dans un minimum local. Dans le cadre d'une navigation basée sur une *costmap* locale, les simulations ont montré que le robot peut ne pas être capable de générer une commande permettant d'accomplir la tâche de navigation tout en respectant les contraintes, malgré l'utilisation de procédures de récupération. Le troisième environnement montre un accomplissement de la tâche pour les deux méthodes. Cependant, il met aussi en évidence que les zones concaves présentent des difficultés pour l'algorithme DWA, alors que notre algorithme SA évite ces zones sans problème à l'aide d'une détermination explicite d'un sens de contournement. Cela entraîne, à vitesses linéaires équivalentes, une trajectoire globalement plus courte pour notre méthode SA que pour la méthode DWA.
- **Choix des paramètres** : La méthode DWA se base d'abord sur la mise à jour d'une *costmap locale*, puis l'application d'un planificateur local. Elle repose

ainsi sur l'ajustement a priori d'une trentaine de paramètres. Il s'agit donc d'une méthode très modulable qui permet de s'adapter à des cas très spécifiques. En revanche, cette capacité à pouvoir être finement réglée peut conduire à une perte de généricité. Dans le cas d'un environnement agricole très variable par nature, il apparaît compliqué de parvenir à attribuer aisément des valeurs pertinentes à tous les paramètres valables pour toutes les situations rencontrées. Cette méthode, quoique classique dans le domaine de la robotique mobile, et l'implémentation qui en est faite dans la *NavStack* semble difficilement utilisable dans notre contexte. Notre approche quant à elle repose seulement sur le choix d'une distance désirée entre le robot et l'obstacle d^* et de trois gains de commande λ_1, λ_2 et λ_S des contrôleurs d'évitement en spirale. Sa mise en œuvre est donc considérablement plus simple.

- **Complexité algorithmique** : L'algorithme DWA, de par son implémentation dans la *NavStack*, implique une discrétisation en une grille (*costmap*) de l'espace environnant le robot. La taille en mètre ainsi que l'échantillonnage de la *costmap* sont des paramètres que l'utilisateur doit déterminer en faisant un compromis entre la précision de la navigation et le temps de calcul. Une seconde discrétisation de l'espace de commande (v, w) est aussi opérée, selon un pas et un horizon donné par l'utilisateur. Ainsi, la méthode DWA nécessite davantage de temps de calcul que la méthode basée spirale, qui se base directement sur les observations LiDAR brutes. Les simulations ont été effectuées sur un ordinateur possédant un processeur Intel Core i7-7820HQ (Quad Core 2.90GHz). La méthode d'évitement en spirale tournait à 40Hz, tandis que la méthode DWA restait limitée à 20Hz. Cette différence en temps de calcul met notamment en avant une des raisons du choix d'une commande référencée capteurs au lieu d'une approche basée sur une carte locale.

Pour conclure, nous proposons une méthode réactive de navigation et d'évitement d'obstacles référencée capteurs, tandis que la méthode DWA implémentée dans la *NavStack* de ROS propose une approche basée sur la construction de la *costmap* locale puis l'utilisation d'un planificateur de chemins local DWA. Les simulations que nous avons menées jusqu'ici montrent que l'approche DWA semble souffrir de plusieurs limitations : complexité dans le choix des paramètres, impossibilité de franchir certains obstacles entraînant un échec de la navigation, et une complexité algorithmique non négligeable.

4.2.4 Discussion et comparaison avec "teb local planner"

Cette section propose une analyse de la méthode TEB (Time-Elastic-Band) telle qu'implémentée dans le paquet ROS `teb_local_planner` et présentée dans les articles [Rösmann et al., 2012] et [Rosmann et al., 2013]. Si l'on compare cette approche avec la méthode DWA présentée dans la précédente partie, l'analyse des

articles [Rösmann et al., 2012], [Rosmann et al., 2013], [Rösmann et al., 2015] et [Rösmann et al., 2016], et la documentation ROS disponible pour le paquet `teb_local_planner` montre que la méthode TEB souffre des mêmes écueils que la méthode DWA analysée précédemment :

- **Choix des paramètres** : À l'instar de la méthode DWA présentée dans la section précédente, la méthode TEB dans son implémentation `teb_local_planner` nécessite environ quatre-vingt paramètres en plus de ceux nécessaires pour la mise en place de la *costmap*[9]. Ces paramètres permettent notamment d'indiquer la configuration du robot, de configurer les paramètres des trajectoires ainsi que la prise en compte des obstacles. Des outils[10] permettent de faire varier ces paramètres en temps réel afin de visualiser les changements dans les trajectoires générées par la méthode TEB. Cette complexité de réglage permet d'obtenir des comportements très fins, mais cela se fait encore au détriment de la généricité.

- **Optimisation en temps** : Comme expliqué plus haut, la méthode TEB repose sur l'optimisation continue d'une fonction de coût. Afin de diminuer le temps de calcul, les contraintes sont relaxées, notamment les contraintes cinématiques liées au robot, ainsi que les contraintes relatives à la distance obstacle-robot. Cependant, dans un contexte agricole avec un environnement en grande partie changeant et inconnu, il semble plus intéressant de respecter les contraintes cinématiques et les distances entre le robot et les obstacles, plutôt que d'optimiser temporellement la trajectoire effectuée.

- **Génération des trajectoires** : La méthode TEB génère une trajectoire locale optimisée temporellement qui minimise une fonction de coût continue. Il n'y a donc aucune garantie que le robot ne tombera pas dans un minimum local, à l'instar de ce qui est montré pour la méthode DWA dans le deuxième environnement.

4.2.5 Conclusion

Cette section a présenté deux des algorithmes de navigation disponibles dans la *NavStack* de ROS : les algorithmes Dynamic-Window-Approach [Fox et al., 1997] (DWA) et Time-Elastic-Bands [Rösmann et al., 2012], [Rosmann et al., 2013] (TEB). L'objectif que nous avons poursuivi était de les comparer à notre approche basée sur l'évitement par spirale (SA). Ces deux méthodes sont sensiblement différentes de notre solution. En effet, les approches DWA et TEB sont des planificateurs de trajectoire locaux. Elles génèrent une trajectoire locale selon un horizon de temps ou de distance précis en fonction des données acquises sur l'environnement entourant le

[9] http://wiki.ros.org/teb_local_planner
[10] http://wiki.ros.org/teb_local_planner/Tutorials/Setup%20and%20test%20Optimization

robot. Ces trajectoires sont déterminées selon la minimisation d'une fonction de coût discrète ou continue, dont les paramètres sont réglés préalablement par l'utilisateur. Notre méthode quant à elle s'inscrit dans le domaine de la commande référencée capteur puisqu'elle exploite directement les données LiDAR acquises pour générer une commande d'évitement.

La mise en place des méthodes DWA et TEB impliquent le réglage de plusieurs dizaines de paramètres, permettant de gérer les *costmaps* ainsi que le fonctionnement algorithmes eux-mêmes. Ce réglage est dépendant du robot ainsi que de l'environnement. Ces paramètres étant fixés avant la navigation, il est difficile d'assurer un comportement correct dans des environnements aussi complexes et variables que des environnements agricoles, comme l'ont montré les simulations réalisées dans les deuxième et troisième environnements. Notre méthode étant référencée capteur, elle nécessite d'une part de choisir des spirales successives déterminées chacune par un centre, une distance et un angle, et d'autre part d'appliquer des lois de commande pour lesquelles trois gains doivent être réglés. Notre méthode s'appuie donc sur un petit nombre de paramètres, ce qui réduit sa complexité de réglage et la rend plus générique. De plus, dans notre approche, l'évitement se fait à vitesse linéaire non nulle et est basé sur un choix spécifique du sens de contournement reposant sur l'ensemble des informations disponibles localement. Cela évite ainsi pour le robot de se retrouver bloqué dans des espaces concaves. De plus, la mise à jour des centres des spirales est basée sur un calcul de projeté. Cela permet d'anticiper les obstacles et de maintenir la commande angulaire dans des limites acceptables pour le système robotisé. Enfin, notre méthode SA étant totalement réactive, elle ne nécessite que peu de puissance de calcul contrairement aux méthodes TEB et DWA, qui requièrent quant à elles la mise à jour de *costmap* et des algorithmes d'optimisation coûteux en temps de calcul.

Ces résultats tendent donc à montrer la pertinence de notre méthode réactive référencée capteurs dans le cadre agricole. Sa généricité, sa simplicité d'utilisation, et sa capacité à gérer des obstacles difficiles font de notre méthode d'évitement un choix intéressant pour la navigation d'un robot dans un environnement agricole statique complexe.

4.3 Expérimentations dans un environnement statique

Les résultats des simulations précédentes ont montré l'intérêt de notre approche. Nous avons donc également validé cette dernière dans des conditions réelles. Cette section présente les deux expérimentations menées dans ce cadre. La première se propose de mettre en en œuvre la stratégie de navigation basée sur des centres des spirales sélectionnés à l'aide du barycentre (cf. sous-section 2.3.3). Elle se déroule sur un terrain

goudronné. La seconde expérimentation repose sur la stratégie de navigation basée sur un choix du SCP calculé grace aux projetés successifs matérialisant l'enveloppe convexe des obstacles (cf. section 2.5.1). Elle se déroule sur un sol herbeux.

4.3.1 Première expérimentation avec SCP basé barycentre

Conditions expérimentales

Cette expérience vise à montrer la capacité du robot à se déplacer dans un environnement possédant des obstacles similaires à ceux qui peuvent être trouvés dans une exploitation agricole. Ainsi, comme le montre la figure 4.17, un environnement a été conçu sur le parking du LAAS-CNRS. Cet environnement est composé de plusieurs obstacles statiques : un groupe de petits obstacles (poubelles, lampe, réservoir, etc...) (A), une voiture garée (B) et deux grands bâtiments (C) et (D).

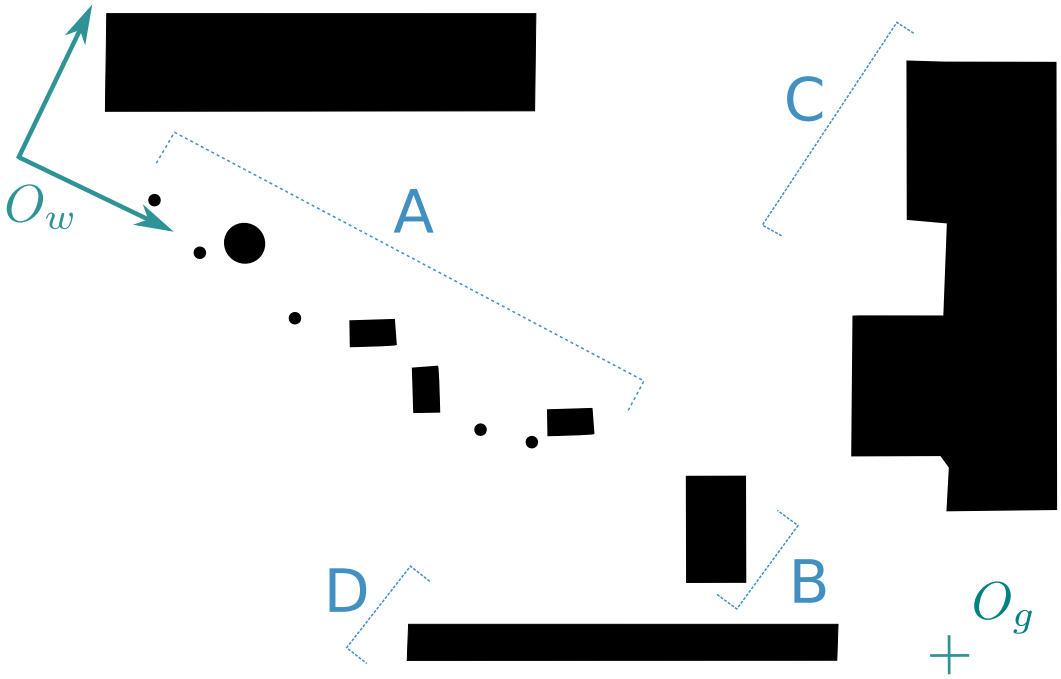


Figure 4.17: Représentation de l'environnement

A partir de sa position initiale O_w, le robot doit se déplacer vers un but précis O_g dont les coordonnées (en mètres) dans le repère initial du robot sont $[x_g, y_g] = [60, -5]$. La vitesse linéaire du robot v^* a été fixée à 0,25 $m.s^{-1}$. Compte tenu de la taille du robot (environ 150x80cm), la distance désirée entre le robot et le centre de la spirale a été fixée à $d^* = 3m$. Les gains pour le contrôleur ω_A (cf. équation (2.42)) sont fixés comme suit : $\lambda_1 = 0,1$, $\lambda_2 = 0,1$. Le gain λ_S pour le contrôleur ω_B (cf.

équation (2.43)) est fixé à $\lambda_S = 0,5$. Le paramètre n (cf. équation (2.34)) est fixé à $n = 5$. Ces lois de commande sont rappelées ci-dessous :

$$\omega_A(t) = \omega_2(t) = \frac{\lambda_1 e_d(t) + \lambda_2 \dot{e}_d(t)}{v^* \sin(e_\alpha(t) + \alpha^*)} + \frac{v^* \sin(e_\alpha(t) + \alpha^*)}{e_d(t) + d^*(t)} \tag{4.1}$$

$$\omega_B(t) = \omega_3(t) = \lambda_S e_S(t) + \frac{v(t)}{d(t)} \sin(\alpha(t)) - \dot{\alpha}_S(t, d) \tag{4.2}$$

Le seuil de commutation e_α^{switch} est sélectionné comme $e_\alpha^{switch} = \frac{\pi}{12}$. Ce seuil détermine quel contrôleur est utilisé (cf. sous-section 2.3.2) :

- Si $|e_\alpha(t)| < e_\alpha^{switch}$, la loi de commande en vitesse angulaire retenue est ω_A.
- Sinon, la loi de commande en vitesse angulaire retenue est ω_B.

Pour le choix du SCP, seul l'algorithme 2 basé sur le calcul du barycentre est utilisé.

Analyse de la trajectoire et des commandes du robot

Les performances du robot sont analysées à travers sa trajectoire, les erreurs, en particulier l'erreur de distance entre le robot et l'obstacle, et la loi de commande résultant des différents contrôleurs. La figure 4.18 montre la trajectoire effectuée par le robot dans cet environnement. Le robot atteint son but en évitant tous les obstacles : la voiture, les obstacles ponctuels et le bâtiment. Les figures 4.19 et 4.20 présentent respectivement l'évolution de l'erreur de distance $e_d(t)$ et de l'erreur angulaire $e_\alpha(t)$. Pour finir, la figure 4.21 affiche l'évolution de la vitesse angulaire ainsi que le contrôleur sélectionné. L'analyse de l'expérimentation et des performances est effectuée obstacle par obstacle.

Évitement d'un groupe d'obstacles (A) Comme le montre la figure 4.18, le robot se trouve initialement au point O_w. La zone devant lui étant dégagée, il se dirige immédiatement vers le but O_g en utilisant le contrôleur Go-To-Goal (GTG).

A l'instant $t = 16s$, l'évitement est déclenché. En utilisant les conditions expliquées dans la section 2.3, un sens de contournement (SOM) dans le sens inverse des aiguilles d'une montre est sélectionné car la plupart des obstacles visibles se trouvent sur le côté gauche du robot. À cet instant, il y a une erreur de distance $e_d(t) \simeq 1,5m$, due aux conditions de déclenchement. Comme le robot est presque face au premier obstacle du groupe (A), il y a également une erreur angulaire $e_\alpha(t) \simeq -1,2rad$. Cette erreur angulaire est supérieure à $e_\alpha^{switch} = \pi/12 \simeq 0,26rad$. Par conséquent, selon la condition 2.3.2, le contrôleur ω_B (cf. équation (2.43)) est utilisé pour lancer l'évitement.

À l'instant $t \simeq 24s$, l'erreur angulaire est suffisamment faible pour basculer sur le contrôleur ω_A (cf. équation (2.43)). Le passage se fait sans aucune discontinuité

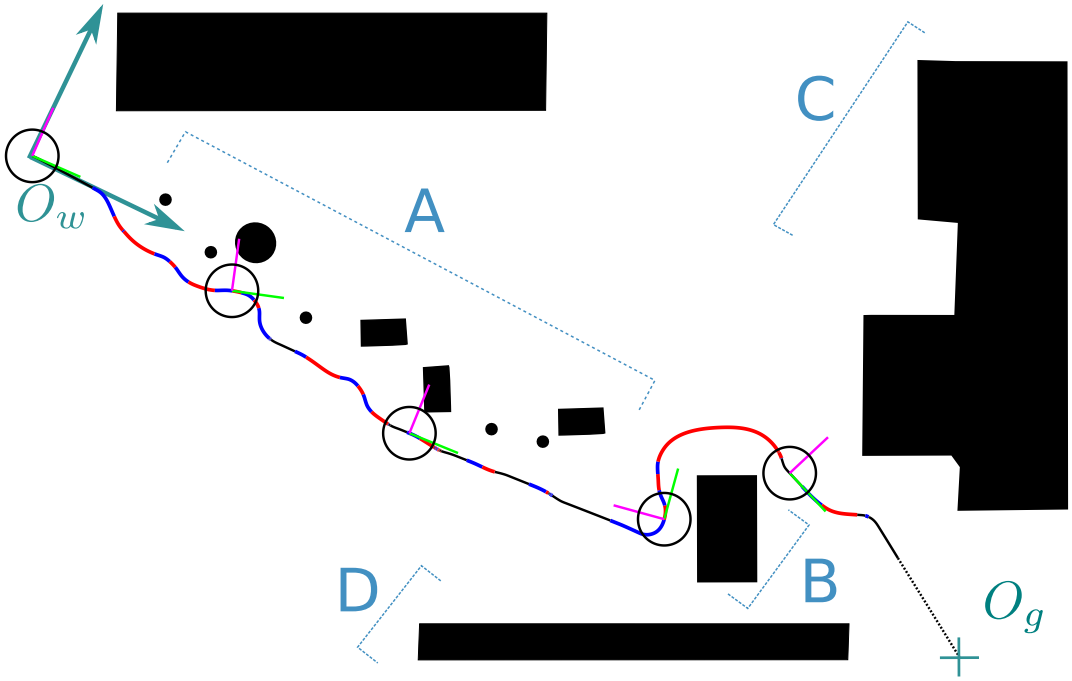


Figure 4.18: Trajectoire finale du robot

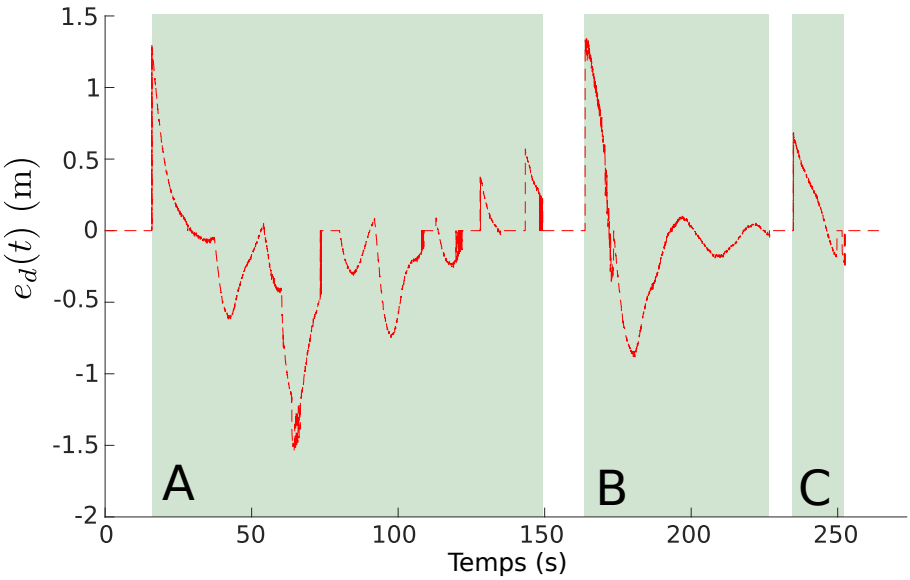


Figure 4.19: Erreur de distance $e_d(t)$

visible sur la loi de commande grâce au dispositif de lissage défini par l'équation (2.44).

De $t = 16s$ à $t = 150s$, le robot évite la succession de petits obstacles qui se

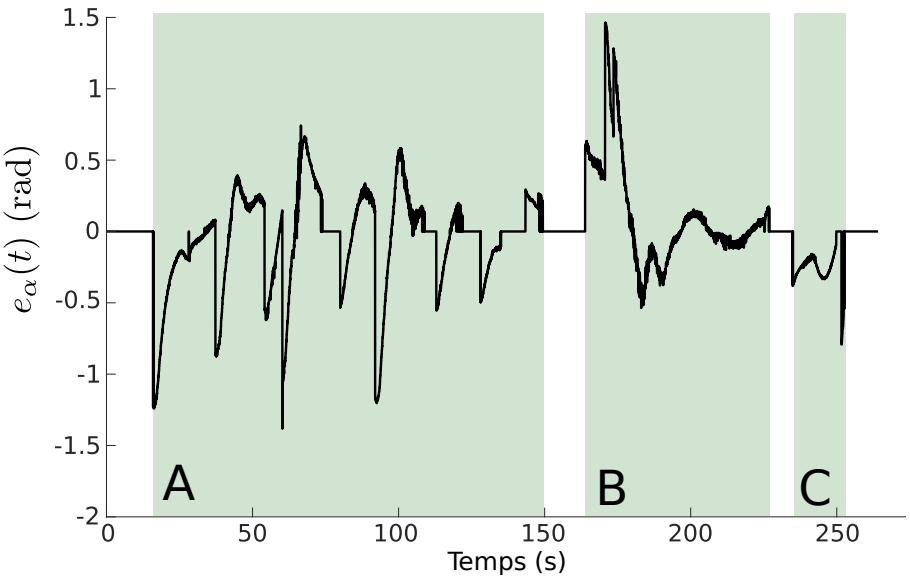


Figure 4.20: Erreur angulaire $e_\alpha(t)$

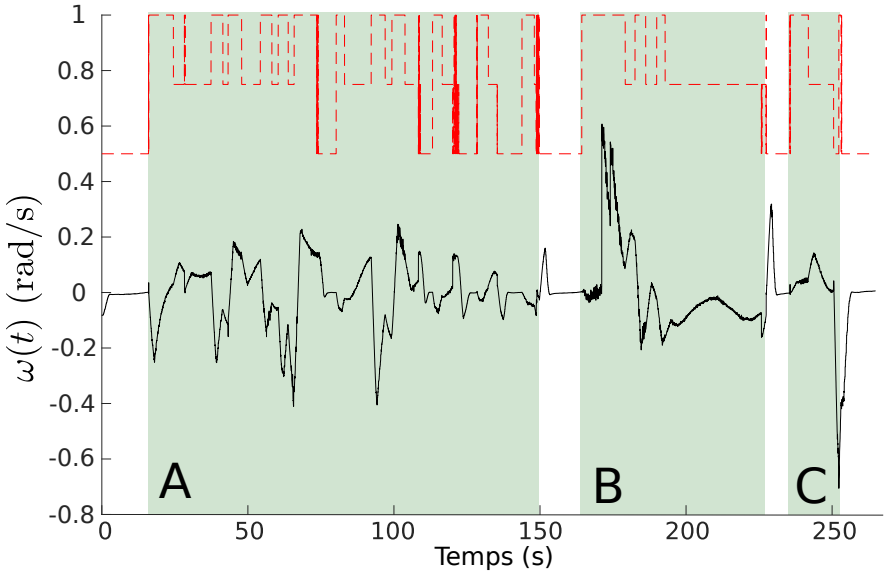


Figure 4.21: Commandes de vitesse linéaire $v(t)$ et de vitesse angulaire $\omega(t)$

trouvent dans le groupe (A). Le robot bascule ensuite plusieurs fois entre les deux contrôleurs d'évitement afin de contourner chacun des petits obstacles. L'élément (A) étant un groupe de petits obstacles proches, on peut voir sur la figure 4.18 que, grâce au choix du SCP, le robot est capable d'éviter ce groupe dans son ensemble. En d'autres termes, il ne cherche pas à entrer dans l'espace présent entre chaque obstacle, évitant ainsi le risque de se coincer dans un minimum local. Comme le SCP évolue avec l'environnement, les erreurs $e_d(t)$ et $e_\alpha(t)$ varient autour de 0.

À l'instant $t \simeq 150s$, le robot considère que le groupement d'obstacle (A) est évité, et se dirige vers l'objectif en utilisant le contrôleur GTG.

Évitement de l'obstacle (B) A l'instant $t \simeq 164s$, le robot détecte l'obstacle voiture (B). Il décide d'un sens de contournement dans le sens des aiguilles d'une montre et commence à utiliser le contrôleur ω_B avant de passer au contrôleur ω_A. Le robot aurait pu éviter (B) dans le sens inverse des aiguilles d'une montre en passant par l'espace entre (B) et (D), mais comme cet espace était inférieur à $2d^*$, la sécurité n'était pas garantie. Cela montre l'intérêt d'utiliser O_b pour déterminer le SOM à l'opposé de prendre le point le plus proche O_c. Comme la distance entre (B) et (D) était suffisamment petite, une partie de (D) a été prise en compte pour le calcul du point O_b, en le déplaçant vers le côté droit du robot et conduisant à un SOM dans le sens des aiguilles d'une montre.

À partir de $t \simeq 192s$ et $t \simeq 225s$, le contrôleur ω_A est capable d'amener et de maintenir les erreurs proches de 0.

À l'instant $t \simeq 227s$, la voiture est évitée et le robot enclenche le contrôleur GTG jusqu'à la détection du bâtiment (C) à $t \simeq 235s$.

Évitement de l'obstacle (C) et fin de l'expérimentation À cet instant, le robot évite l'obstacle (C) en tournant autour du coin. Enfin, il passe au contrôleur GTG et se dirige vers le but.

Cette expérimentation valide plusieurs comportements souhaités :

- *Distance de sécurité :* Comme le montre la figure 4.19, l'erreur de distance $e_d(t)$ n'est jamais inférieure à -1,5m. Le robot n'est pas jamais allé à moins de 1,5 mètres d'un obstacle. Malgré un environnement inconnu et très variable, le robot a réussi à maintenir une distance de sécurité par rapport à chaque obstacle. En outre, l'évitement est déclenché à une distance supérieure à d^*, afin de laisser au robot un temps suffisant pour tourner avant d'atteindre $d = d^*$.

- *Faible sensibilité à la taille ou à la forme des obstacles :* Le robot a réussi à éviter un grand groupe d'obstacles, une voiture et un grand bâtiment. Le robot peut éviter des obstacles présentant des coins intérieurs et extérieurs, des groupements d'obstacles ainsi que de grands murs.

- *Choix de sens de contournement pertinents :* Le sens de contournement est déterminé pour produire le chemin le plus court en tenant compte des connaissances que le robot possède de l'environnement à l'instant de la décision et évitant d'amener le robot dans des passages étroits. En outre, les conditions de

(a) Début de l'expérimentation

(b) Fin de l'évitement de (A)

(c) Début de l'évitement de (B)

(d) Fin de l'expérimentation

Figure 4.22: Expérimentation à différents instants

commutation ne prolongent pas l'évitement plus loin que nécessaire. Dès que le robot considère que la zone avant est libre, il revient à un contrôleur GTG. Enfin, le fait de choisir un nouveau SOM à chaque fois que le contrôleur SA est activé, et de le maintenir constant pendant l'évitement empêche le robot d'être bloqué dans un minima local.

4.3.2 Seconde expérimentation avec SCP basé sur les points projetés

La seconde expérimentation se déroule dans un environnement différent et sur un terrain herbeux, boueux et accidenté. Cette expérimentation a pour objectif d'évaluer la stratégie de navigation avec un contrôleur d'évitement d'obstacles basé sur un choix du centre de la spirale O_s utilisant le projeté, dont la méthode est présentée dans le chapitre 2 et la sous-section 2.5.1.

Conditions expérimentales

Cette expérience vise à montrer la capacité du robot à se déplacer dans un environnement semblable à celui à un environnement agricole, en détectant et en évitant les obstacles. Ainsi, un environnement a été conçu au LAAS-CNRS afin de

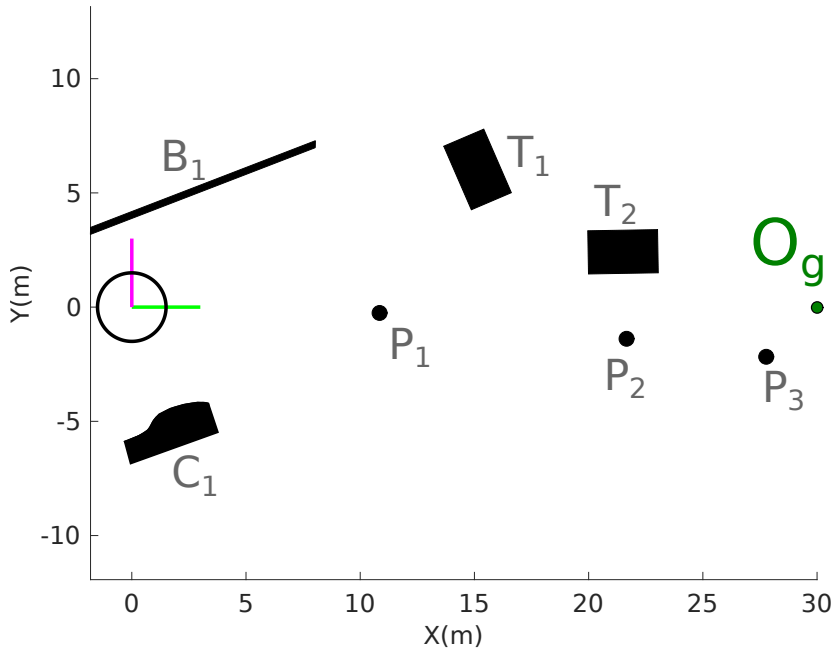


Figure 4.23: Représentation de l'environnement

reproduire les situations auxquelles le robot pourrait être confronté dans une zone agricole : le terrain d'expérimentation est un terrain herbeux, boueux, sur lequel se trouvent des irrégularités (branches d'arbres, petites pierres, feuilles mortes, etc.).

La figure 4.23 montre une représentation de l'enviromment. À partir de sa position initiale O_w, le robot doit se déplacer vers un but précis O_g dont les coordonnées dans le repère initial du robot sont $[x_g, y_g] = [0, 30]$. Entre le robot et son but se trouvent plusieurs obstacles :

- Élément B_1 : il s'agit d'une barrière, située à gauche de la pose initiale du robot.
- Élément C_1 : il s'agit d'une colline assez haute pour être détectée par les LiDARs du robot.
- Éléments T_1 et T_2 : ce sont deux tables de forme rectangulaire.
- Éléments P_1, P_2 et P_3 : Il s'agit de trois petits obstacles ponctuels, de forme cylindrique.

La figure 4.24 montre une photographie de l'environnement de navigation, prise depuis la pose de départ du robot. Sur cette photographie sont indiqués les différents obstacles. La vitesse linéaire v^* a été fixée à 0,3 $m.s^{-1}$. Pour assurer une distance de sécurité suffisante compte tenu de la taille du robot (environ 150x80cm), $d^* = 2m$ est choisi. Les gains pour les contrôleurs ω_A (cf. équation (2.42)) et ω_B (cf. équation (2.43)) sont

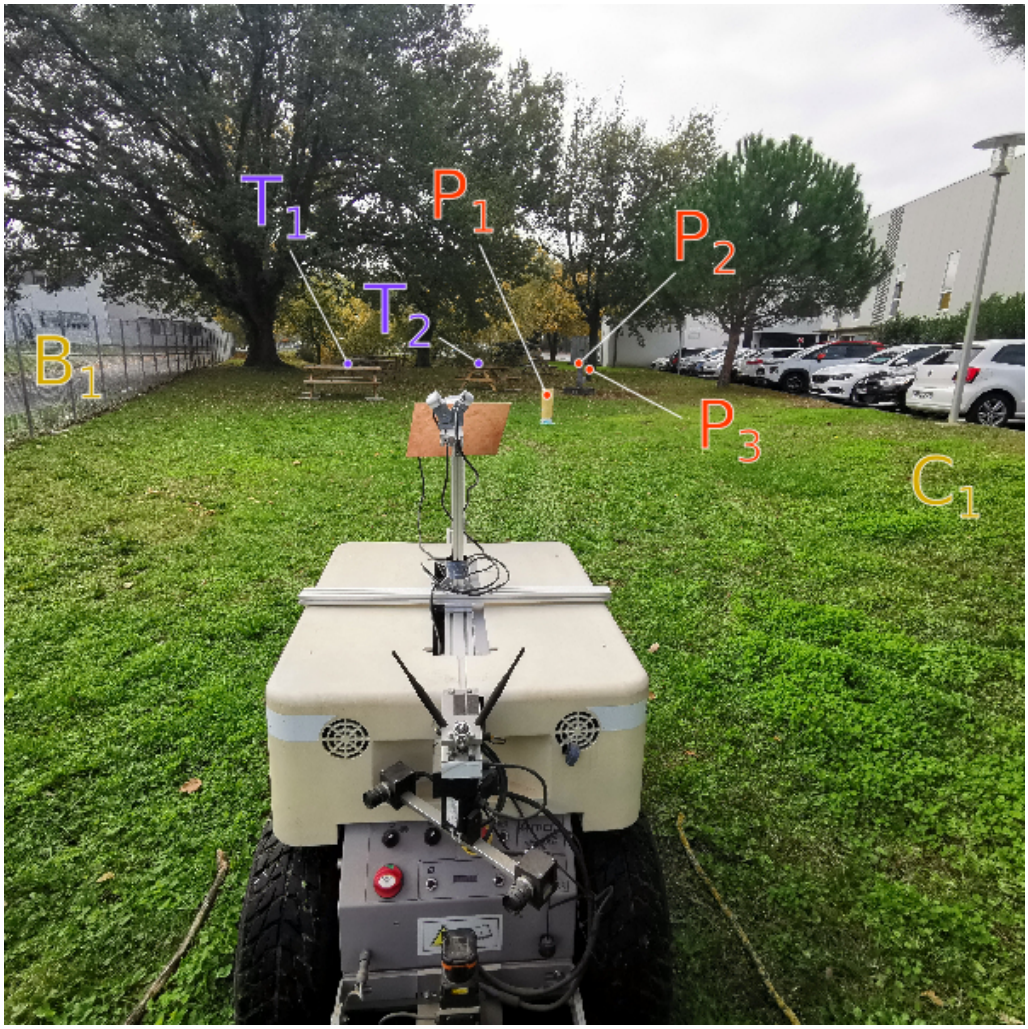


Figure 4.24: Photographie de l'environnement et des obstacles présents

fixés comme suit : $\lambda_1 = 0,1$, $\lambda_2 = 0,2$, $\lambda_S = 0,5$. Le paramètre n (cf. équation (2.34)) est fixé à $n = 5$. Le seuil de commutation e_α^{switch} est sélectionné comme $e_\alpha^{switch} = \frac{\pi}{12}$. Pour cette expérimentation, l'algorithme 3 de choix du SCP, basé sur un calcul de projeté, est utilisé.

Analyse de la trajectoire et des commandes du robot

Les performances du robot sont analysées à travers sa trajectoire, l'erreur en distance $e_d(t)$ et en angle $e_\alpha(t)$, et la loi de commande résultant des différents contrôleurs. La figure 4.25 montre la trajectoire finale du robot. L'usage des différents contrôleurs est modélisé par des couleurs. Les figures 4.26a et 4.26b fournissent respectivement les positions successives du point le plus proche du robot O_c et des centres successifs des spirales O_s. La figure 4.28 détaille l'évolution de l'erreur de distance $e_d(t)$ et de

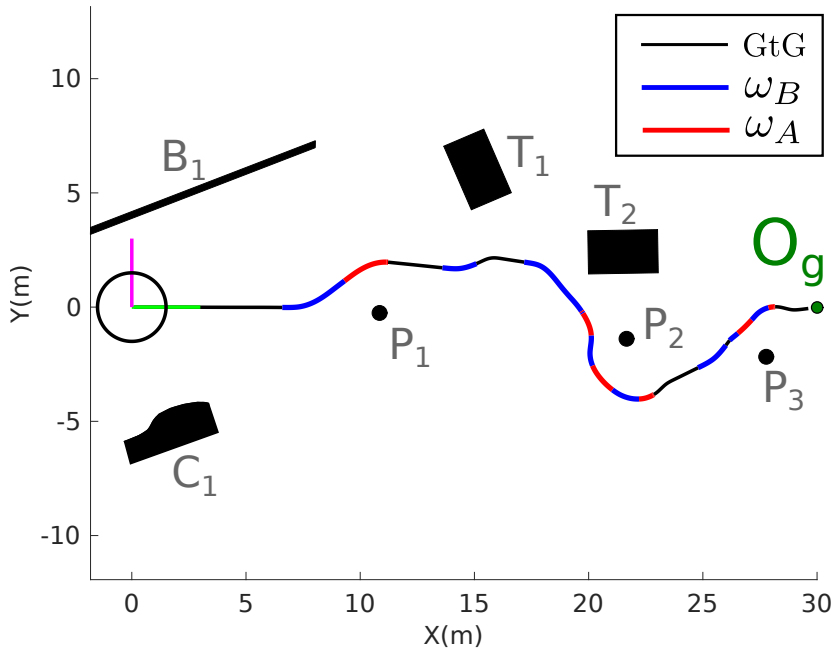


Figure 4.25: Trajectoire effectuée par le robot

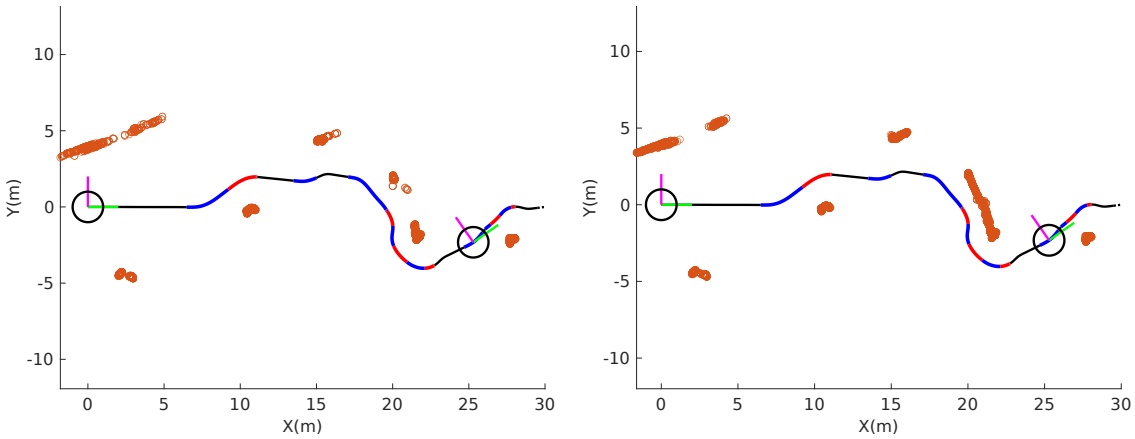
l'erreur angulaire $e_\alpha(t)$. La figure 4.29 illustre l'évolution de la vitesse angulaire et la figure 4.30 le contrôleur courant. Pour finir, sur la figure 4.27 est représentée la position du barycentre O_b à quatre instants correspondant à la commutation vers le contrôleur d'évitement. Sur la figure 4.25 est illustré le robot qui atteint son but en évitant tous les obstacles. Le déroulement est détaillé obstacle par obstacle.

Évitement de l'obstacle (P1) À l'instant $t = 0s$, le robot démarre et commence à se diriger en ligne droite vers son but.

À l'instant $t \simeq 22s$, l'obstacle (P1) se trouve à moins de $2d^* = 4m$ de distance. L'évitement est donc déclenché. Comme le montre la figure 4.27, le point O_b se trouve *à droite* de O_g. Le sens de contournement est choisi dans le sens horaire. Le contrôleur ω_B est tout d'abord utilisé pour amener le robot sur la spirale définie, puis une commutation est effectuée vers le contrôleur ω_A afin de suivre la spirale. Comme le montre la figure 4.28, les erreurs $e_d(t)$ et $e_\alpha(t)$ convergent bien vers zéro.

À l'instant $t = 40s$, l'évitement de l'obstacle (P1) est terminé et le robot enclenche le contrôleur GTG.

Évitement de l'obstacle (T1) À l'instant $t \simeq 48s$, l'obstacle (T1), bien que situé sur le côté du robot, est détecté comme étant dangereux et doit être évité. Ainsi, le



(a) Positions du point le plus proche O_c (b) Positions du centre de la spirale O_s

Figure 4.26: Trajectoire du robot

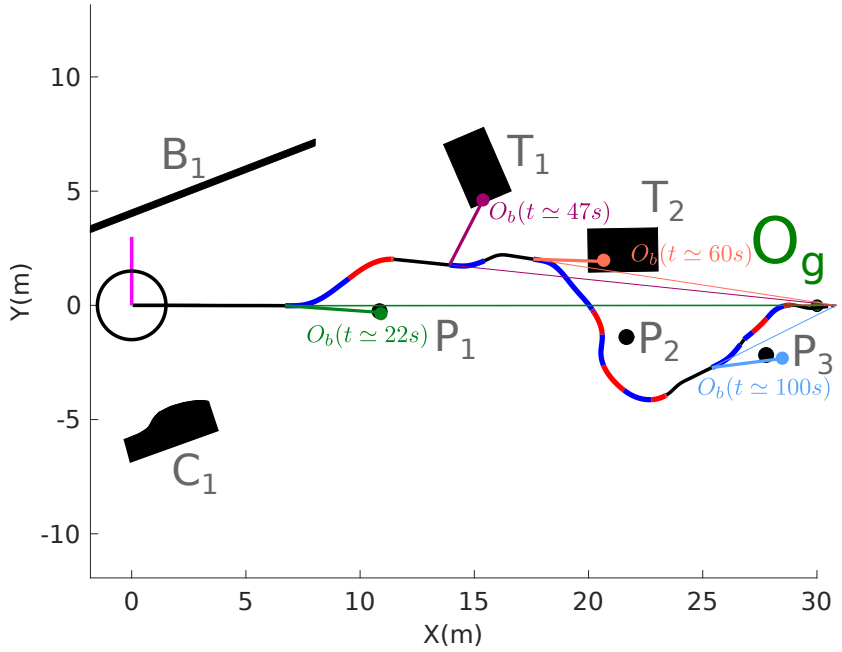


Figure 4.27: Position des barycentres à quatre moments de commutation ($t \simeq 22s$, $t \simeq 47s$, $t \simeq 60s$ et $t \simeq 100s$)

contrôleur ω_B est brièvement enclenché jusqu'à l'instant $t = 52s$.

Évitement des obstacles (T2) et (P2) À l'instant $t \simeq 60s$, le robot arrive à proximité de l'obstacle (T2) et déclenche la procédure d'évitement. Un sens de contournement anti-horaire est sélectionné. Comme le montrent les figures 4.23 et 4.24, les obstacles (T2) et (P2) sont deux obstacles distincts, mais situés à une

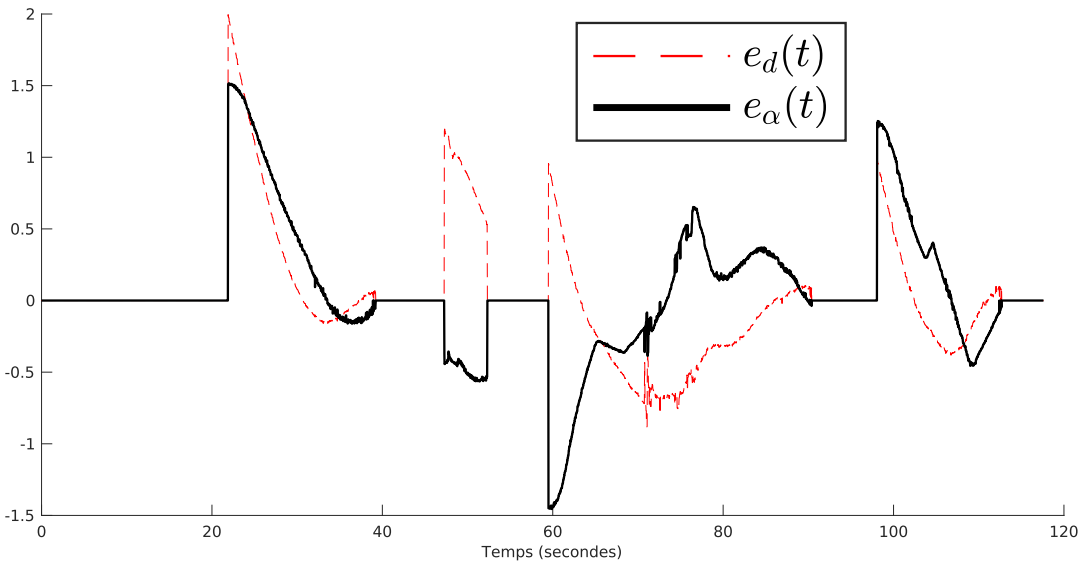


Figure 4.28: Erreur angulaire $e_\alpha(t)$ et erreur en distance $e_d(t)$

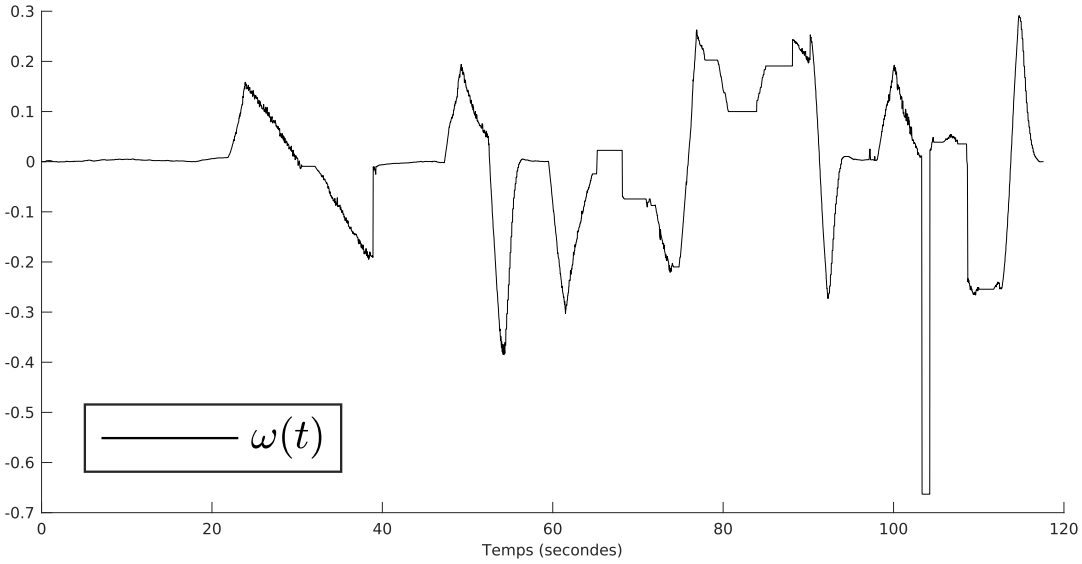


Figure 4.29: Consigne angulaire $\omega(t)$

distance inférieure à $2d^*$ l'un de l'autre. Ainsi, comme l'indique la figure 4.26b et grâce à la méthode de sélection des SCP présentée section 2.5.1, les SCP successifs sont pris de façon continue afin d'évoluer sur l'enveloppe convexe liant (T2) à (P2). Le robot évite ainsi les obstacles (T2) et (P2) comme un obstacle unique.

À l'instant $t = 90s$, l'obstacle (P2) est évité, et le contrôleur GTG est enclenché. Nous pouvons voir sur la figure 4.29 qu'entre $t \simeq 60s$ et $t \simeq 90s$, la consigne angulaire ne dépasse pas 0.3 rad/s en valeur absolue.

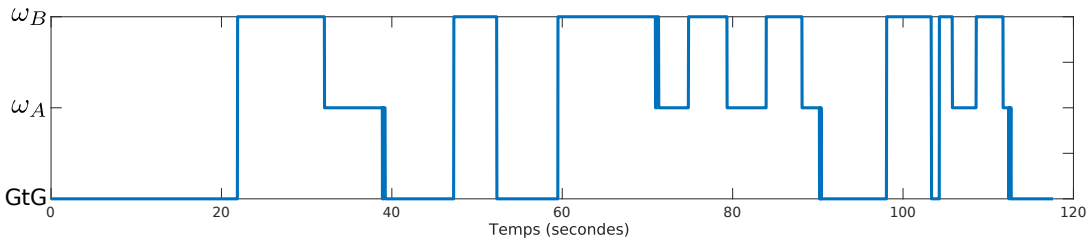


Figure 4.30: Contrôleur utilisé

Évitement de l'obstacle (P3) et fin À l'instant $t \simeq 98s$, l'obstacle (P3) est détecté comme dangereux et le contrôleur SA est activé. À l'instant $t \simeq 112s$, cet obstacle est évité et le robot retourne sur le contrôleur GTG pour atteindre son but à l'instant $t \simeq 120s$.

La figure 4.31 montre quatre photographies prises pendant l'expérimentation, à différents instants.

(a) Évitement de l'obstacle (P1) ($t \simeq 30s$)

(b) Évitement de l'obstacle (T1) ($t \simeq 50s$)

(c) Évitement de (T2) et (P2) ($t \simeq 120s$)

(d) Évitement de l'obstacle (P3) ($t \simeq 100s$)

Figure 4.31: Images de l'expérimentation à quatre moments

Conclusion

À l'instar de l'essai précédent, cette expérimentation a mis en exergue la capacité de notre méthode de navigation à conduire le robot vers son but tout en respectant les points suivants :

- *Distance aux obstacles* : La figure 4.28 montre que l'erreur en distance $e_d(t)$ ne descend pas en dessous d'environ 0.6 mètre. Cela signifie que le robot ne s'est pas retrouvé à une distance inférieure à 1.4 mètres de l'obstacle le plus proche.
- *Faible sensibilité à la taille ou à la forme des obstacles* : le robot est capable d'éviter des obstacles de tailles diverses. Notre méthode de choix du SCP basée sur les projetés permet de *regrouper* des obstacles séparées d'une distance inférieure à $2d^*$. La navigation est ainsi plus fluide. La figure 4.29 met aussi en évidence que la commande angulaire maximale générée lors de l'utilisation du contrôleur SA ne dépasse pas 0.4 rad/s.

4.3.3 Analyses et conclusions des expérimentations en environnement statique

Les deux expérimentations en environnement statique ont montré la pertinence de la méthode de navigation dans le cas d'obstacles statiques et elle a mis en exergue les contributions suivantes :

- *Faible sensibilité à la nature du sol* : notre méthode est référencée capteur et ne fait aucune hypothèse sur la dynamique du robot. Ainsi, elle fonctionne à faible vitesse sur des terrains bétonnés, mais aussi sur des sols agricoles.
- *Approche générique* : les deux expérimentations font apparaître différents types d'obstacles, de tailles et de formes variées. Elles font notamment apparaître des situations considérées comme "difficiles" pour les algorithmes de navigation, avec notamment des regroupements de petits obstacles, et des obstacles concaves. Nous voyons que grâce à l'approche très générique de notre méthode, le robot est capable de gérer ces cas, sans se retrouver bloqué ni devoir s'approcher trop près d'un obstacle.
- *Choix du contrôleur de navigation* : l'application des conditions de commutation et de choix du sens de contournement permet respectivement l'anticipation des obstacles et un choix de contournement sélectionné en fonction des connaissances locales de l'environnement.
- *Commandes de vitesse adaptées* : La gestion de l'évolution du SCP permet d'obtenir une sortie de commande qui reste dans des bornes acceptables compte tenu du robot utilisé. En fonction du robot utilisé et des limites de ces actionneurs, il est possible pour l'utilisateur de choisir une vitesse linéaire v^* et les valeurs

des gains des contrôleurs λ_1, λ_2 et λ_S afin d'obtenir un comportement du robot plus adapté à la tâche.

4.4 Expérimentations dans un environnement dynamique

Cette section a pour but de présenter des résultats expérimentaux préliminaires effectués en conditions réelles afin de valider les contributions théoriques présentées dans le chapitre 3. Les deux approches d'évitement d'obstacles basées sur la méthode *Distance-Angle Adaptatifs* et la méthode *Enhanced Laser Scan* sont évaluées.

4.4.1 Validation de la méthode *Distance-Angle Adaptatifs*

Afin de valider la méthode *Distance-Angle Adaptatifs*, proposée dans la section 3.2, trois expérimentations sont proposées, visant à mettre en valeur le comportement de la méthode dans des cas simples, avec un seul obstacle mobile. L'obstacle considéré est un humain marchant à une vitesse moyenne de 3 km/h. La première expérimentation est le cas où un humain se dirige face au robot. Nous souhaitons ainsi valider le fait que la distance désirée adaptative $d_a^*(t)$ va alors augmenter et que l'évitement est déclenché plus tôt que dans le cas statique. Les deux expérimentations suivantes portent sur le cas où un obstacle se déplace d'un côté à l'autre du robot de façon à couper sa trajectoire. Nous souhaitons valider que le choix du sens de contournement sera pertinemment en fonction de la valeur de l'angle adaptatif $\delta^\alpha(t)$ (cf. sous-section 3.2.3) et que le robot n'est pas *attrapé* par l'obstacle.

Configuration du robot

La vitesse linéaire v^* a été fixée à 0.3 $m.s^{-1}$ et $d^* = 2.5m$ est choisi. Les gains pour les contrôleurs sont fixés comme suit : $\lambda_1 = 0.1$, $\lambda_2 = 0.1$, $\lambda_S = 0.5$. Le paramètre n (cf. équation (2.34)) est fixé à $n = 5$. Le seuil de commutation e_α^{switch} est sélectionné comme $e_\alpha^{switch} = \frac{\pi}{12}$. Pour le calcul du sens de contournement, le seuil $\delta^\alpha_{threshold} = 0.05$ est fixé (cf. équation 3.2.3).

Cas d'un obstacle approchant face au robot

Dans ce cas, le robot se déplace à une vitesse $v^* = 0.3m/s$ pour atteindre un but situé devant lui. L'obstacle quant à lui arrive vers le robot de manière frontale. La figure 4.32 montre la trajectoire du robot depuis sa position initiale, ainsi que l'évolution du point O_c. La figure 4.33a présente l'évolution des deux erreurs $e_\alpha(t)$ et $e_d(t)$ ainsi que la loi de commande angulaire $\omega(t)$, tandis que la figure 4.33b expose l'évolution de la distance adaptative $d_a^*(t)$, de la distance au point le plus proche $d_c(t)$,

et du contrôleur utilisé.

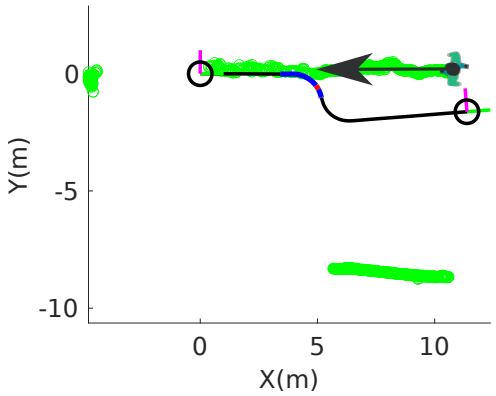


Figure 4.32: Trajectoire du robot et points les plus proches O_c

Au début de l'expérimentation, le point le plus proche se trouve à l'arrière du robot (sur un mur). Ce point n'étant pas mobile, nous avons $d_a^*(t) \simeq d^* = 2.5m$.

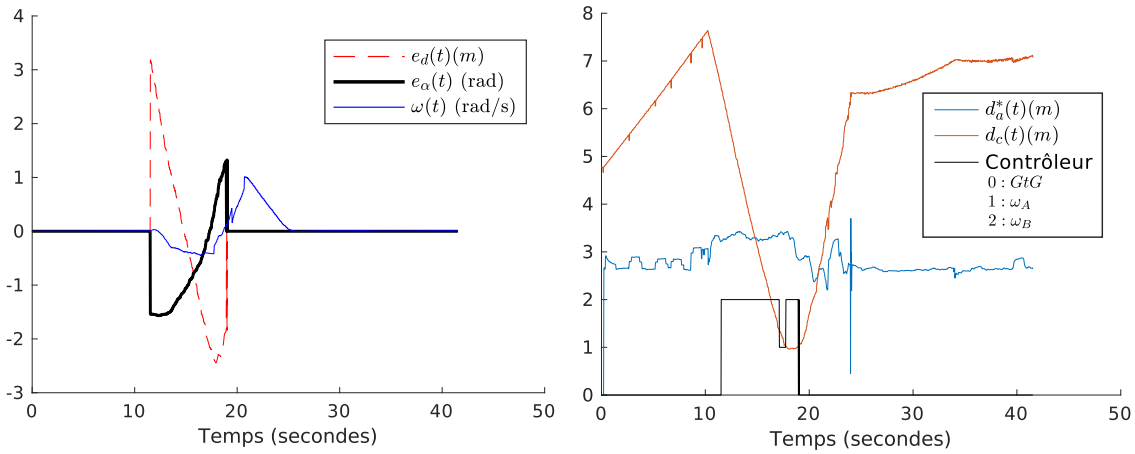
À l'instant $t = 8s$, le point le plus proche appartient désormais à l'obstacle qui se dirige droit vers le robot. La figure 4.33b montre que la valeur de $d_a^*(t)$ augmente jusqu'à atteindre environ 3.3m.

À l'instant $t = 12s$ l'obstacle arrive à une distance inférieure à 6.5m qui est le seuil de commutation calculé à partir de la valeur de $d_a^*(t)$. Grâce au calcul de la distance adaptative, le robot commence donc à éviter l'obstacle à une distance de 6.5 mètres de ce dernier, au lieu de seulement $2d^* = 5$ mètres dans le cas statique.

À l'instant $t = 19s$, l'obstacle est évité, et le contrôleur GTG reprend la main. À cet instant là, la distance adaptative $d_a^*(t)$ diminue car l'obstacle s'éloigne maintenant du robot. La figure 4.34 montre trois photographies prises pendant l'expérimentation. Nous constatons que le robot s'écarte en amont de la personne marchant vers lui et il évite ainsi toute collision.

Cas d'un obstacle approchant par le côté droit du robot

Pour cette expérimentation, l'obstacle arrive par le côté droit du robot. Il peut donc croiser la trajectoire du robot. La figure 4.35 montre la trajectoire du robot depuis sa position initiale ainsi que l'évolution du point O_c. La figure 4.36a présente l'évolution des deux erreurs $e_\alpha(t)$ et $e_d(t)$ ainsi que la commande angulaire $\omega(t)$, tandis que la figure 4.36b montre l'évolution de la distance adaptative $d_a^*(t)$, de la distance au point le plus proche $d_c(t)$ et le choix contrôleur utilisé. Pour finir, la figure 4.37 montre l'évolution de l'angle adaptatif $\delta^\alpha(t)$. Le calcul de cet angle est présenté



(a) $e_d(t)$, $e_\alpha(t)$ et $\omega(t)$ (b) $d_a^*(t)$, $d_c(t)$ et contrôleur utilisé

Figure 4.33: Évolutions des variables pendant l'expérimentation

Figure 4.34: Photographies de trois instants de l'expérimentation

sous-section 3.2.3, et est rappelé ci-dessous :

$$\delta^\alpha(t_i) = \eta^\alpha(t_i) + 2.17\sigma^\alpha(t_i) \tag{4.3}$$

avec :

$$\eta^\alpha(t_i) = \frac{1}{q}\sum_{i=1}^{q} r^\alpha(t_i) \tag{4.4}$$

$$(\sigma^\alpha)^2(t_i) = \frac{1}{q-1}\sum_{i=1}^{q}(r^\alpha(t_i) - \eta^\alpha(t_i))^2 \tag{4.5}$$

$$r^\alpha(t_i) = \frac{\alpha_c^{predicted}(t_p) - \alpha_c(t_i)}{t_p - t_i} \tag{4.6}$$

La figure 4.36a montre que le robot utilise tout d'abord le contrôleur GTG.

À l'instant $t \simeq 16s$, le robot commute sur le contrôleur SA. À cet instant, un choix de sens de contournement SOM est réalisé. La figure 4.37 illustre que l'obstacle ayant une trajectoire de la gauche vers la droite du robot, la valeur de $\delta^\alpha(t)$ est négative et inférieure à $-\delta^\alpha_{threshold}$. Ainsi, nous utilisons les conditions présentées dans le chapitre 3, avec $\delta^\alpha_{threshold} = 0.05$:

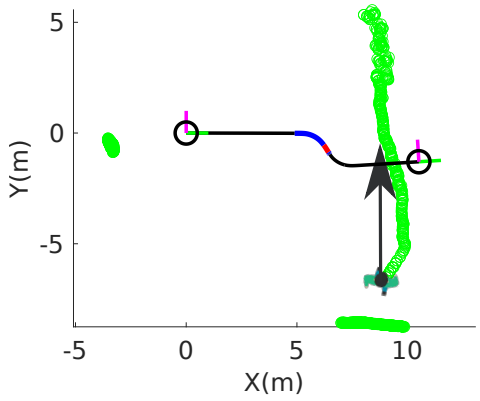
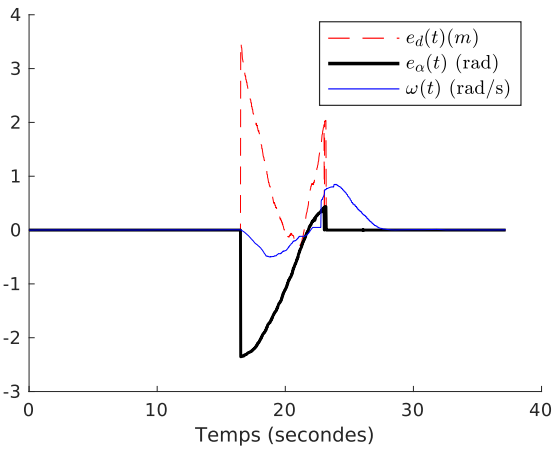
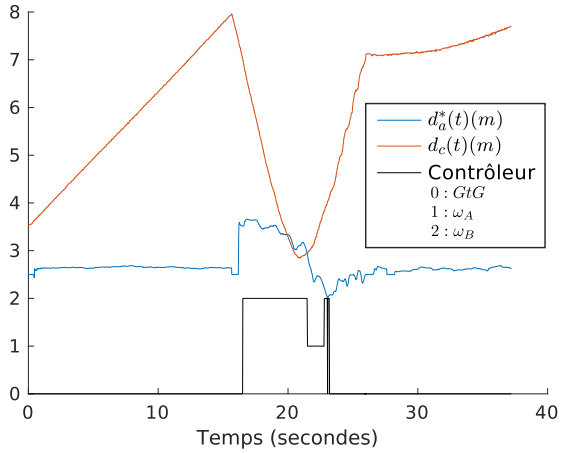


Figure 4.35: Trajectoire du robot et points les plus proches O_c



(a) $e_d(t)$, $e_\alpha(t)$ et $\omega(t)$



(b) $d_a^*(t)$, $d_c(t)$ et contrôleur utilisé

Figure 4.36: Évolutions des variables pendant l'expérimentation

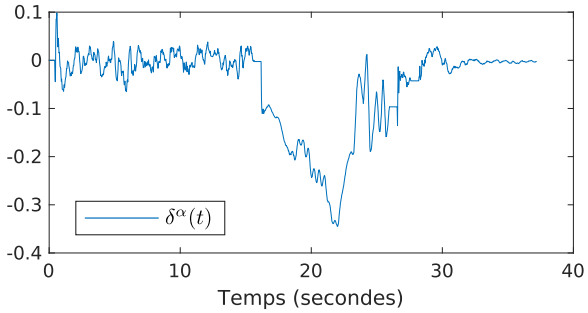


Figure 4.37: Angle adaptatif $\delta^\alpha(t)$

- Si $|\delta^\alpha(t)| \leq \delta^\alpha_{threshold}$, les conditions pour le cas statique sont spécifiées
- Sinon, si $\delta^\alpha(t) < -\delta^\alpha_{threshold}$, un sens d'évitement direct (anti-horaire) est spécifié.

Figure 4.38: Photographies de trois instants de l'expérimentation

- Sinon, si $\delta^{\alpha}(t) > \delta^{\alpha}_{threshold}$, un sens d'évitement indirect (horaire) est spécifié.

Nous avons donc, à $t \simeq 16s$, $\delta^{\alpha}(t) < -\delta^{\alpha}_{threshold}$. Ainsi, un sens de contournement anti-horaire est choisi. Cela permet ainsi de ne pas s'approcher à moins de 3 mètres de l'obstacle, comme le montre la figure 4.36b. La figure 4.38 montre ainsi trois photographies prises durant l'expérimentation. Nous voyons donc le robot tourner vers la droite et contourner la personne dans un sens anti-horaire.

Cas d'un obstacle approchant par le côté gauche du robot

Ce cas est l'inverse du précédent : l'obstacle s'approche de la gauche vers la droite du robot. La figure 4.39 montre la trajectoire du robot depuis sa position initiale ainsi que l'évolution du point O_c. La figure 4.40a présente l'évolution des deux erreurs $e_{\alpha}(t)$ et $e_d(t)$ ainsi que la loi de commande angulaire $\omega(t)$, tandis que la figure 4.40b expose l'évolution de la distance adaptative $d^*_a(t)$, de la distance au point le plus proche $d_c(t)$, et du contrôleur utilisé. Pour finir, la figure 4.41 montre l'évolution de $\delta^{\alpha}(t)$.

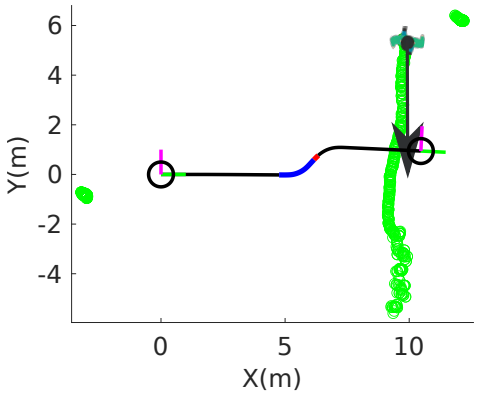
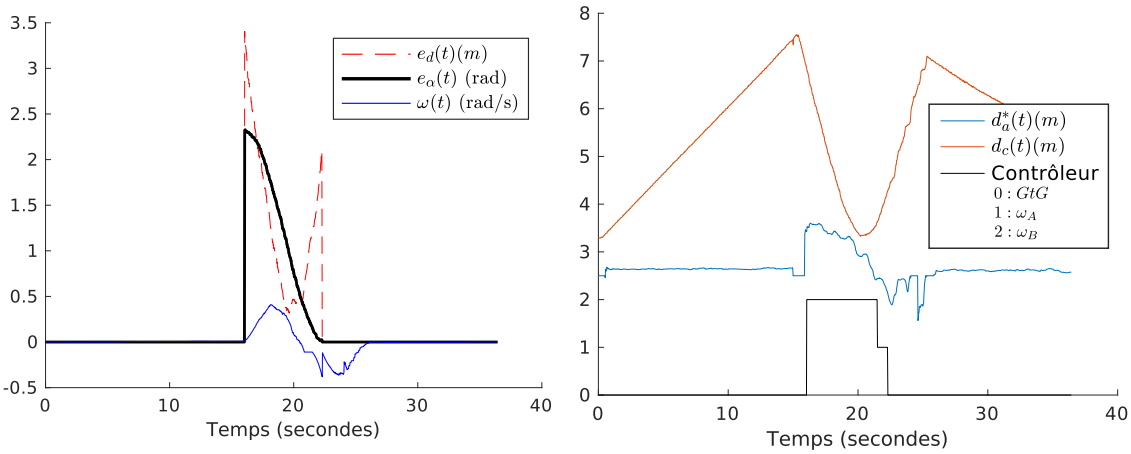


Figure 4.39: Trajectoire du robot et points les plus proches O_c

Le déroulement de l'expérimentation est semblable. À l'instant $t \simeq 17s$, le robot enclenche l'évitement. À cet instant là, les valeurs de $\delta^{\alpha}(t)$ sont positives, plus grandes que $\delta^{\alpha}_{threshold}$. Ainsi, un sens de contournement horaire est choisi, évitant ainsi une la collision. La figure 4.42 montre ainsi trois photographies prises pendant d'évitement.



(a) $e_d(t)$, $e_\alpha(t)$ et $\omega(t)$

(b) $d_a^*(t)$, $d_c(t)$ et contrôleur utilisé

Figure 4.40: Évolutions des variables pendant l'expérimentation

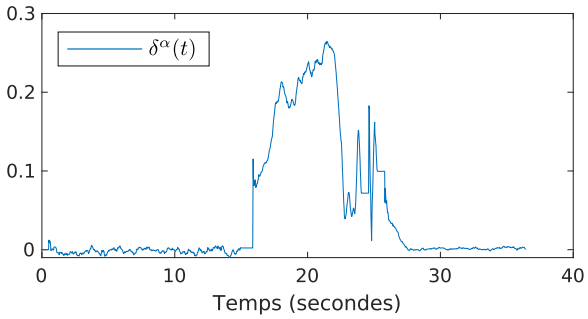


Figure 4.41: Angle adaptatif $\delta^\alpha(t)$

Figure 4.42: Photographies de trois moments de l'expérimentation

Conclusion sur la méthode *Distance-Angle Adaptatifs*

Dans cette sous-section, nous avons validé sur des cas simples la pertinence de notre contribution basée sur l'utilisation de la méthode *Distance-Angle Adaptatifs* :

- La distance adaptative telle qu'introduite dans l'équation (3.6) permet une anticipation du mouvement de l'obstacle, et donc un déclenchement anticipé de l'évitement, ce qui réduit les risques de collision.

- La prise en compte de l'angle adaptatif $\delta^{\alpha}(t)$ permet au robot d'anticiper si l'obstacle vient de la droite vers la gauche, ou inversement. De ce fait, lors du déclenchement de l'évitement, le robot peut choisir un sens de contournement pertinent vis-à-vis de la dynamique de l'obstacle, ce qui réduit les risques de percuter l'obstacle ou bien d'être attiré et traîné par ce dernier.

4.4.2 Présentation des résultats préliminaires sur la méthode de l'*Enhanced Laser Scan*

Dans cette sous-section sont présentés des résultats préliminaires concernant la méthode basée sur l'*Enhanced Laser Scan*. Le but de ces expérimentations est de valider la capacité de la méthode à générer une acquisition laser améliorée correcte dans divers cas, en présence de plusieurs obstacles statiques et dynamiques. Nous souhaitons notamment valider le fait qu'une discrimination est bien effectuée entre obstacles statiques et dynamiques et que les points virtuels sont ajoutés correctement.

Ainsi, les expérimentations suivantes reposent sur le même principe. Le robot est placé immobile dans un environnement où une ou plusieurs personnes vont être présentes et vont bouger tout d'abord de manière contrôlée, puis ensuite de manière *aléatoire* afin de simuler un contexte agricole, où des humains peuvent se déplacer à proximité du robot. La figure 4.44a présente une visualisation d'une acquisition LidAR. En vert sont représentés les points LiDAR. Le robot est représenté par le repère *base_link*. Les points LiDAR montrent différents objets et bâtiments dans l'environnement du robot, ainsi qu'un petit groupement de points en face du robot, représentant la personne pour l'instant immobile. Les éléments entourant le robot étant totalement statiques, aucun point virtuel n'est rajouté à l'*Enhanced Laser Scan*, qui se retrouve être l'acquisition LiDAR de base.

Dans les expérimentations suivantes, le calcul de l'ELS est effectué à une fréquence de 40Hz. La distance D utilisée dans l'algorithme 4 est fixée à $D = 0.1m$, et la vitesse maximale théorique du robot, utilisée pour le calcul du vecteur de projection $\overrightarrow{d_h}$ (sous-section 3.3.3) est fixée à $v_{max} = 1.5m/s$.

Obstacle se déplaçant face au robot

Nous considérons ici le cas d'un obstacle se déplaçant droit vers le robot. L'obstacle est initialement devant le robot, à une distance supérieure à une dizaine de mètres. Cet obstacle, un humain, marche vers le robot, s'arrête, puis repart ensuite pour s'éloigner du robot. Les figures 4.44a et 4.44b représentent l'ELS lorsque respectivement la personne se rapproche de l'obstacle, et lorsque la personne s'éloigne de l'obstacle. Les points verts représentent l'acquisition LiDAR de base, et les points rouges représentent les points virtuels rajoutés. La fusion des deux forme alors l'*Enhanced Laser Scan*.

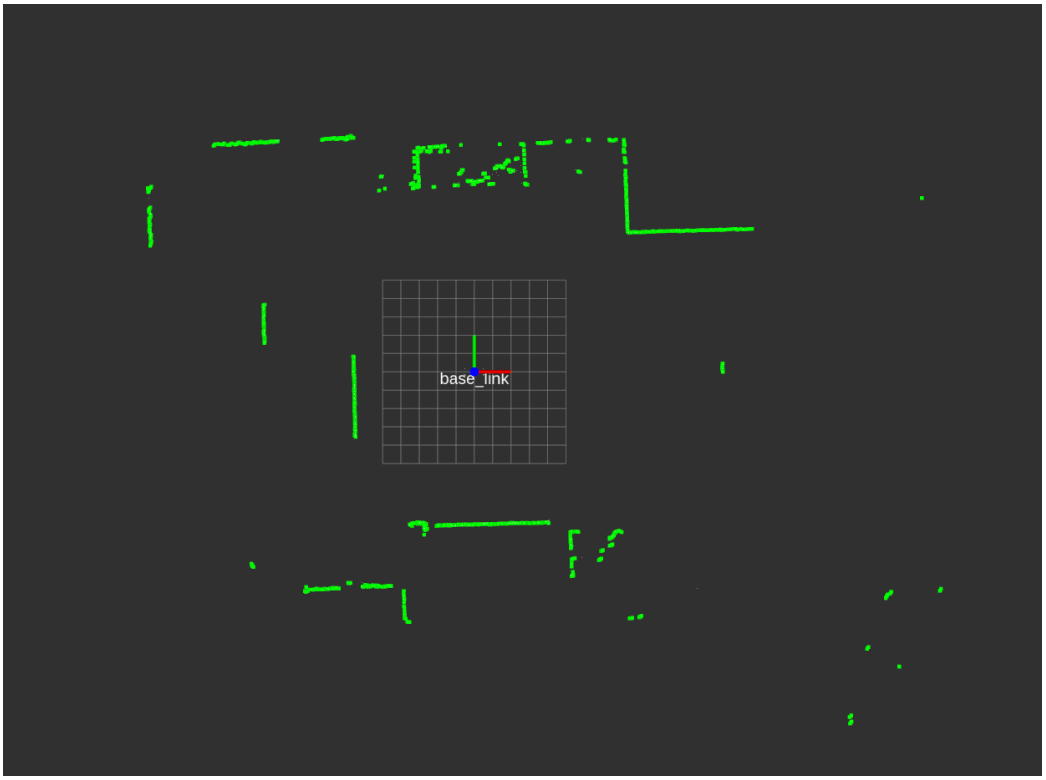
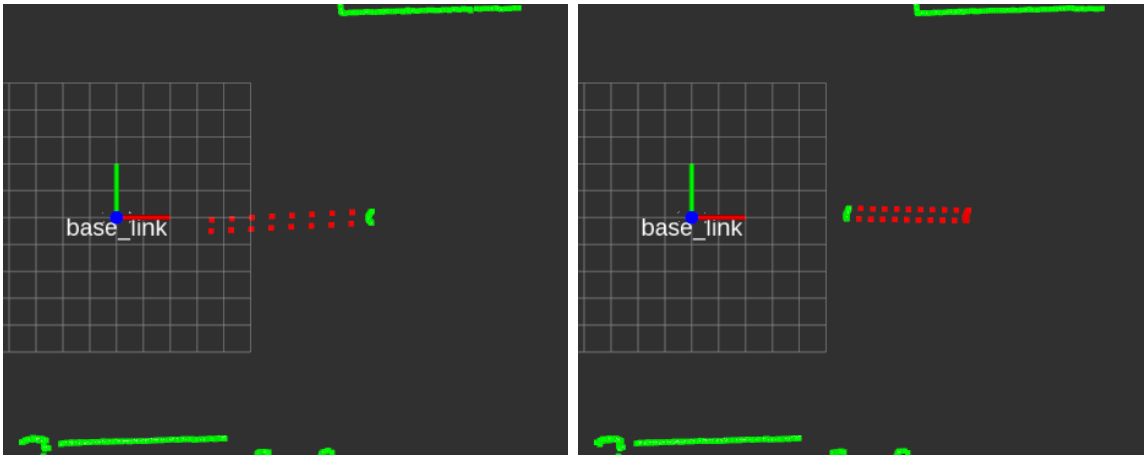


Figure 4.43: ELS, la personne étant immobile



(a) ELS, personne marchant vers le robot (b) ELS, personne s'éloignant du robot

Figure 4.44: Enhanced Laser Scan dans deux cas

Nous constatons donc que lorsque l'obstacle se dirige vers le robot (cf. figure 4.44a), des points virtuels sont rajoutés pour former une *bande* le long de la trajectoire future

prédite de l'obstacle. Ces points sont rajoutés à l'acquisition LiDAR initiale pour former l'*Enhanced Laser Scan*. De même, lorsque la personne s'éloigne du robot (cf figure 4.44b), la bande formée par les points virtuels est ajoutée du côté opposé au robot. Ainsi, si le robot était autorisé à bouger et devait effectuer une tâche de navigation similaire à celle présentée dans les expérimentations précédentes, un des points virtuels de l'ELS serait pris comme SCP dans le premier cas (cf. figure 4.44a), et l'évitement serait déclenché alors que l'obstacle se trouve encore à une grande distance, afin d'anticiper son déplacement. Nous voyons de plus qu'à la périphérie du robot, les éléments rectilignes que nous pouvons observer en haut et en bas des figures 4.44a et 4.44b, correspondant à des bâtiments entourant le robot, ne sont pas assimilés comme étant dynamiques par notre méthode, et ne produisent pas de points virtuels ajoutés à l'ELS.

Obstacle se déplaçant vers la droite ou la gauche du robot

Dans ce deuxième cas, une personne se déplace latéralement par rapport au robot. Les figures 4.45 présentent l'ELS à quatre différents instants. La figure 4.45a montre l'état initial de l'expérimentation : la personne se trouve immobile à gauche du robot. Ainsi, aucun point virtuel n'est ajouté. Sur la figure 4.45b, la personne se met à marcher dans une direction perpendiculaire à l'orientation du robot. L'algorithme détecte alors l'obstacle en mouvement et ajoute donc des points virtuels le long de la trajectoire prédite. Sur les figures 4.45c et 4.45d, la personne a changé de direction et se dirige de la droite vers la gauche du robot.

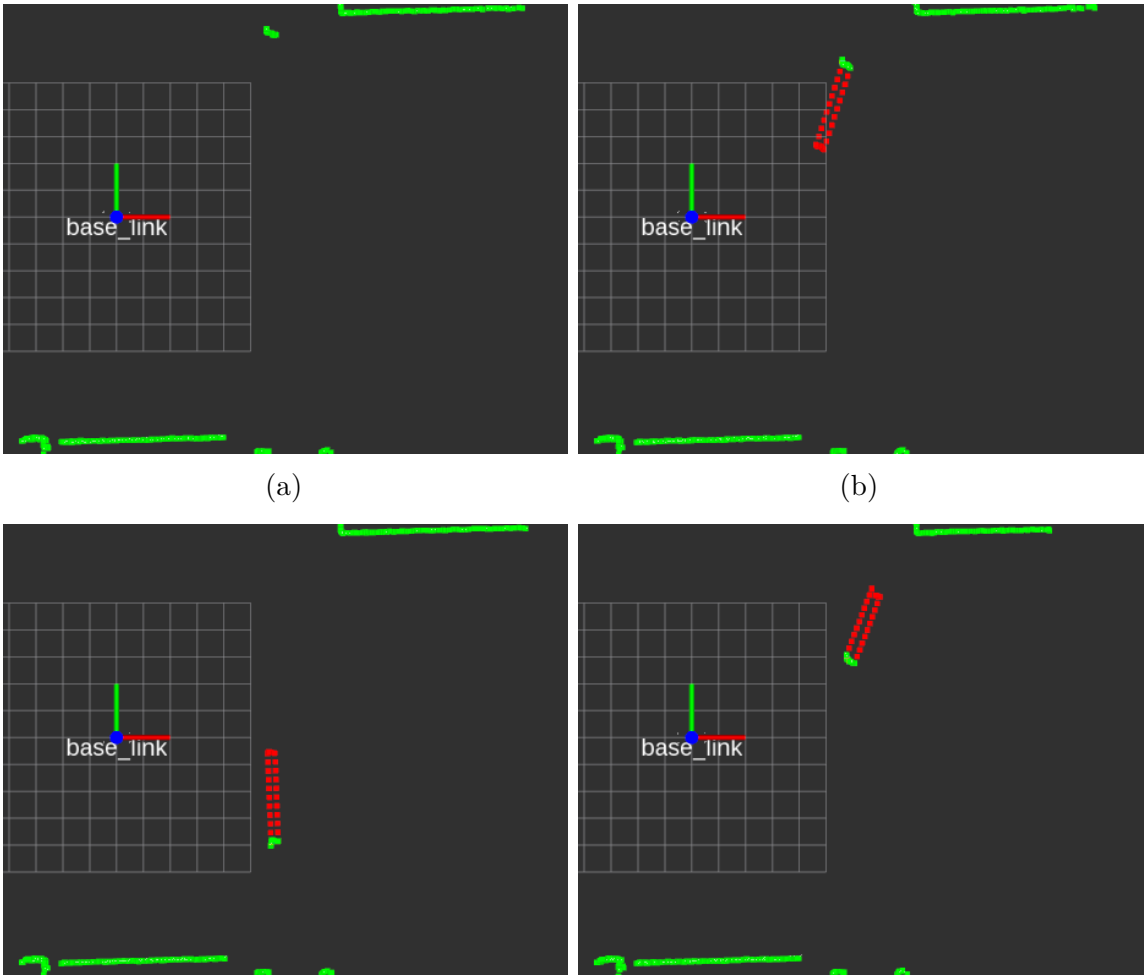
Afin de rajouter les points virtuels le long de la trajectoire prédite de l'obstacle, la méthode passe par le calcul du vecteur vitesse de chaque obstacle (cf. sous-section 3.3.2), qui permet d'en déduire le vecteur $\vec{d_h}(t_i)$ le long duquel sont ajoutés les points :

$$\vec{V}_{R_c}(t_i) = \begin{pmatrix} v_x(t_i) \\ v_y(t_i) \end{pmatrix}_{R_c} = \frac{^{F_r(t_i)}B(t_i) - ^{F_r(t_i)}B(t_p)}{t_i - t_p} \tag{4.7}$$

Ce vecteur est donné dans le repère du robot, et il est notamment utilisé pour calculer le sens de contournement, selon les conditions exposées sous-section 3.3.4 :

- Si $v_y = 0$, les conditions relatives aux obstacles statiques sont appliquées.
- Sinon, si $v_y > 0$, on choisit un SOM dans le sens inverse des aiguilles d'une montre. Cela signifie que l'obstacle va de droite à gauche.
- Sinon, si $v_y < 0$, on choisit un SOM dans le sens des aiguilles d'une montre. Cela signifie que l'obstacle va de gauche à droite.

Ainsi, même si dans ces expérimentations le robot reste immobile, nous pouvons déjà observer que :



(a) (b)

(c) (d)

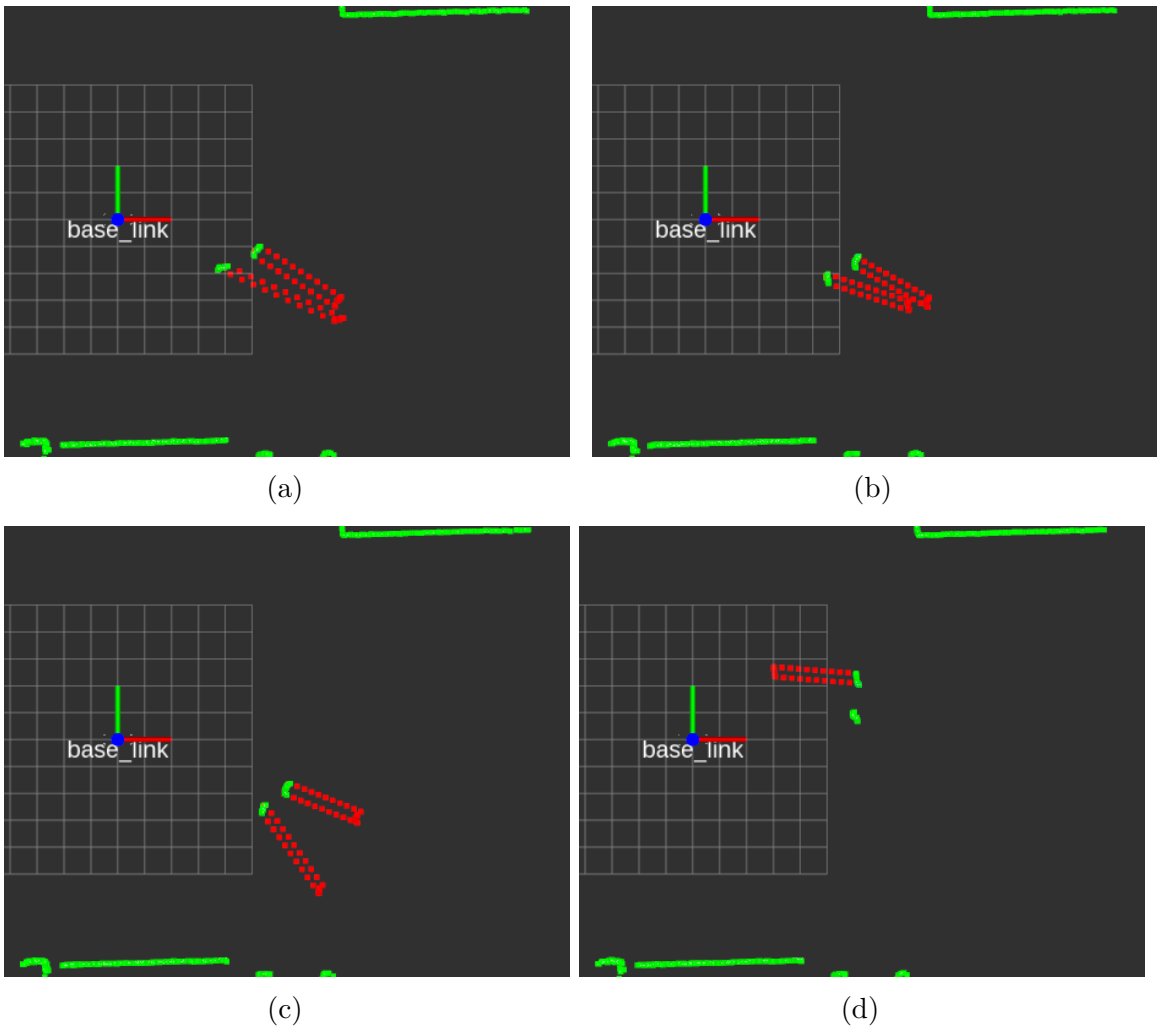
Figure 4.45: Enhanced Laser Scan dans quatre cas, avec une personne se déplacant perpendiculairement au robot.

- Dans le cas de la figure 4.45b, les points semblent être projetés parallèlement à l'axe y_r du robot, en se rapprochant du robot. Ainsi, nous avons donc $v_y < 0$. Cela impliquerait que si l'évitement devait être déclenché à cet instant, le robot choisirait un sens de contournement horaire, ce qui permettrait ainsi de passer *derrière* l'obstacle et de l'éviter correctement.

- Dans le cas de la figure 4.45c, la situation inverse se produit. Nous avons $v_y > 0$, et le sens de contournement serait un sens anti-horaire.

Présence de deux obstacles aux déplacements aléatoires

Dans cette expérimentation, deux personnes se déplacent dans l'environnement du robot pour représenter deux personnes travaillant sur une tâche. Les déplacements de ces personnes ne sont pas structurés et sont indépendants. Le but de cette expérimentation est de valider le fait que plusieurs obstacles mobiles peuvent être gérés simultanément.



(a) (b)

(c) (d)

Figure 4.46: Enhanced Laser Scan dans quatre cas, avec deux personnes mobiles

Les figures 4.46 montrent l'ELS dans quatre cas différents. Dans les quatre cas, excepté celui montré sur la figure 4.46d, les deux personnes sont mobiles. Nous remarquons que les deux obstacles sont correctement discriminés, et séparés l'un de l'autre, mais aussi de l'environnement plus lointain, qui reste détecté comme

statique. Les points virtuels sont donc rajoutés séparément pour chaque obstacle, afin de modéliser pour chacun leur trajectoire distincte.

Conclusion sur la méthode *Enhanced Laser Scan*

Bien que réalisées avec le robot immobile, les expérimentations effectuées valident plusieurs points de nos contributions :

- La méthode de génération de l'ELS est capable en situation réelle de discriminer les points appartenant à des éléments statiques de l'environnement (obstacles, ou bien éléments de *fond*), et les points appartenant à des éléments mobiles.
- Parmi les points discriminés comme appartenant à des obstacles mobiles, chaque obstacle est capable d'être isolé grâce à l'algorithme de regroupement. Cela peut être vu dans l'expérimentation avec deux personnes présentes : les points appartenant à chacune de ces personnes sont séparés avec succès, et chaque personne est donc traitée comme un obstacle seul, avec sa propre dynamique.
- Pour chaque obstacle détecté, les paramètres tels que le vecteur vitesse et l'horizon de projection sont calculés. Ces paramètres sont alors utilisés pour calculer la position des points virtuels à rajouter dans l'acquisition LiDAR. Les trois expérimentations ont montré que ces points virtuels étaient ainsi ajoutés de manière pertinente, de façon à représenter la future trajectoire de l'obstacle.
- L'*Enhanced Laser Scan* est mis à jour en ligne, permettant de prendre en compte des changements brusques dans la dynamique des obstacles. Cela est visible dans la troisième expérimentation, où l'ESL est mis à jour pour refléter les changements dans les mouvements des deux personnes.

4.5 Conclusion

Ce chapitre a porté sur la réalisation d'expérimentations sur un robot mobile afin de valider l'ensemble des solutions qui ont été proposées :

- Les méthodes de navigation en environnement statique présentées dans le chapitre 2 ont fait l'objet de deux expérimentations sur deux sols de nature différente, et validant la pertinence des deux méthodes de choix de centres de spirales, basées respectivement sur un calcul de barycentre (cf. algorithme 2) et de projetés (cf. algorithme 3). Les deux expérimentations ont mis en valeur un comportement fluide du robot, un évitement anticipé des obstacles de formes et de tailles divers, et une prise de décision pertinente quant au sens d'évitement.
- Les méthodes de navigation en environnement dynamique présentées dans le chapitre 3 ont quant à elles fait l'objet de validation préliminaires. Pour la

méthode *Distance-Angle Adaptatifs*, une série de test a permis de montrer le bon fonctionnement du calcul de la distance et de l'angle adaptatif en situation réelle, et par conséquent l'augmentation de la distance d'évitement désirée en cas d'obstacles mobiles dangereux, ainsi qu'un choix du sens de contournement prenant en compte le mouvement de l'obstacle. Pour la méthode *Enhanced Laser Scan*, les expérimentations réalisées avec le robot immobile ont permis de montrer le bon fonctionnement du calcul de l'*Enhanced Laser Scan*, que cela soit avec un obstacle unique ou bien avec plusieurs obstacles.

Chapitre 5

Conclusion

Cette thèse s'est intéressée à la problématique de la navigation et de l'évitement d'obstacles en environnement agricole statique et dynamique. Le cas d'étude typique est celui d'un robot mobile devant se déplacer de son lieu de charge vers une parcelle sur laquelle il doit travailler. Une solution à ce problème apparaît être une stratégie de navigation globale basée sur une localisation absolue GNSS et une carte métrique. Cependant, la présence de nombreux obstacles statiques, difficiles à référencer dans une telle carte et d'obstacles mobiles, difficiles à insérer en temps réel dans une carte métrique, montre les limites de cette approche globale. Ainsi, une stratégie de navigation locale référencées capteurs basée sur une localisation relative et une carte topologique se trouve être pertinente dans ce contexte.

Il existe dans la littérature de nombreuses approches de navigation référencées capteurs permettant une navigation en environnement statique ou dynamique, telles que les méthodes basées sur les champs de potentiel, ou bien sur des histogrammes. Cependant, ces méthodes souffrent de certaines limitations. Elles demandent notamment de faire des hypothèses sur l'environnement (nombre, taille et dynamique des obstacles) et sur la dynamique du robot. Cependant, dans un contexte agricole, il est difficile de faire de telles hypothèses. Ainsi, cette thèse s'est intéressée au concept de spirales, introduites initialement par [Boyadzhiev, 1999], et dont l'intérêt dans un contexte d'évitement d'obstacles est la flexibilité. En effet, les spirales sont définies grâce à trois paramètres qui peuvent être adaptés en ligne afin de génerer des trajectoires d'évitement de manière adaptative et réactive en fonction d'une part des positions, tailles et formes des obstacles, mais aussi de leur dynamique. Ainsi, elles offrent une approche très générique, adaptable à de nombreuses situations et environnements.

Dans cette logique, cette thèse présente notre méthode de navigation générique référencée capteurs basée sur les spirales en environnement statique et dynamique. Nous avons tout d'abord introduit dans le chapitre 2 une méthode d'évitement

référencée capteurs en environnement statique. Cette méthode s'appuie sur une paramétrisation des spirales en fonction des données acquises par les capteurs, ainsi que sur des contrôleurs permettant à un robot de rejoindre et de suivre ces spirales. Cette méthode a ensuite servi de base pour le développement de solutions d'évitements d'obstacles dynamiques. Ces approches, présentées dans le chapitre 3, reposent sur deux phases : la phase de perception et la phase d'action. La phase de perception utilise les acquisitions successives des capteurs pour détecter la présence d'éléments statiques au sein de l'environnement, tandis que la phase d'action correspond à la modification en ligne des paramètres des spirales en fonction des informations obtenues sur les obstacles dynamiques. La généricité de notre méthode a permis le développement de deux approches : la première se base uniquement sur l'observation du point le plus proche du robot et permet une navigation sécuritaire dans une scène faiblement encombrée, tandis que la seconde prend en compte l'ensemble des éléments de la scène et est donc davantage adaptée pour des environnements encombrés. En modifiant respectivement en ligne la distance désirée d^* et le centre de la spirale O_s, ces deux approches permettent ainsi de gérer des environnements dynamiques complexes. Afin de valider ces points dans des situations pratiques, des expérimentations réelles sur une plateforme robotique ont été menées.

5.1 Détails des contributions

Le chapitre 1 a porté sur une analyse précise du contexte agricole et de ses contraintes inhérentes. Les environnements agricoles sont des espaces encombrés, riches en obstacles statiques et mobiles inconnus. Ces obstacles peuvent être de taille et de forme quelconque. De plus, la nature des sols sur lesquels le robot devra évoluer ajoute une contrainte supplémentaire, ces derniers pouvant être glissants, rocailleux, escarpés, etc. À l'issue de ces analyses, il a été montré qu'en complément d'un système de navigation GPS couplé avec une carte topologique, une approche référencée capteurs reste indispensable pour mener à bien des tâches de navigation, afin de gérer la présence imprévue d'obstacles statiques ou mobiles. Ainsi, pour répondre à cette problématique de généricité, nous nous sommes intéressés aux méthodes réactives, se basant sur des acquisitions de capteurs extéroceptifs.

Le chapitre 2 s'est donc intéressé au développement de méthodes de navigation et d'évitement d'obstacles référencées capteurs dans les cas statiques. Les travaux effectués dans ce chapitre se sont basés sur le concept de spirales équiangles développé par [Boyadzhiev, 1999], qui offre des trajectoires pertinentes et adaptables à notre problématique. Ainsi, la première section de ce chapitre a présenté le modèle de la spirale. En particulier, il a été montré qu'une spirale peut être définie par trois paramètres : un centre O_s, une distance désirée d^*, et un angle désiré α^*. La seconde section de ce chapitre a introduit trois lois de commande, permettant à un robot de rejoindre et de suivre une spirale définie préalablement par ces trois paramètres. Les

avantages et inconvénients de chacune de ces lois ont été discutés. La troisième section a ensuite présenté notre méthode de navigation dans son ensemble. Trois éléments ont été discutés :

- Le choix des paramètres d^* et α^* en fonction des données acquises de l'environnement et du type de robot utilisé.
- Le choix du centre des spirales. Deux solutions ont été proposées pour répondre à ce problème. Ces dernières partent du même principe : faire évoluer le centre de la spirale en fonction de la forme des obstacles. Afin de gérer les cas difficiles d'obstacles concaves ou de regroupements d'obstacles, la première solution propose de choisir les centres des spirales à l'aide d'un calcul de barycentre, la seconde propose un choix des centres basé sur un calcul de projeté. Ces deux solutions ont pour avantage d'offrir une évolution fluide du centre des spirales, notamment pour les éléments difficiles, tels que des groupements de petits obstacles, ou bien des obstacles concaves.
- La stratégie de navigation, qui se base sur un contrôleur d'évitement en spirale et un contrôleur *Go-To-Goal*. Le contrôleur d'évitement en spirale est composé de deux des lois de commandes présentées dans la section 2.2 et d'une condition de commutation, permettant de rester dans les limites de stabilité et hors des zones de singularité. Cette stratégie de navigation comporte aussi des conditions de commutation pertinentes permettant au robot de passer du contrôleur *Go-To-Goal* au contrôleur d'évitement et inversement, et d'un algorithme servant à choisir le sens de contournement.

Pour conclure le chapitre, chacun de ces points a été validé en simulation, afin de mettre en exergue la pertinence des solutions proposées.

Le chapitre 3 s'est concentré sur l'adaptation dans des cas dynamiques de la méthode présentée au chapitre précédent. Cette adaptation a été réalisée en modifiant en ligne les paramètres des spirales générées. Ainsi, deux méthodes ont été proposées :

- La première méthode dite *Distance-Angle Adaptatifs* consiste à observer l'évolution du point de plus proche en fonction du temps et du déplacement du robot. Cette observation permet ensuite de modifier en ligne la valeur de d^*, correspondant à la distance désirée entre le robot et le centre de la spirale. Cela permet une distance d'évitement et un seuil de commutation adaptatif plus grand lorsque l'obstacle est dynamique et se dirige vers le robot. L'utilisation similaire de l'angle adaptatif permet aussi un choix du sens de contournement anticipant les mouvements de l'obstacle. Cette méthode est indiquée dans le cas d'environnements peu encombrés.
- Le seconde méthode dite de l'*Enhanced Laser Scan*. Cette méthode consiste à modifier le choix du centre des spirales en ajoutant artificiellement des points

virtuels aux acquisitions fournies par les capteurs. Ces points sont ajoutés sur la trajectoire prédite des obstacles détectés comme mobiles, permettant au robot de gérer plusieurs obstacles dynamiques simultanément dans son entourage.

Ces deux méthodes ont été validées en simulation afin d'exposer leur pertinence, leurs forces et faiblesses.

Le chapitre 4 est consacré à l'implémentation des méthodes développées sur une plateforme robotique et la réalisation d'expérimentations dans des conditions réelles, afin de valider en pratique les solutions présentées dans les chapitres 2 et 3.

- Les méthodes de navigation en environnement statique présentées dans le chapitre 2 ont fait l'objet de deux expérimentations. La première a été effectuée sur sol goudronné avec de grands obstacles (bâtiments, voiture, etc.). La seconde a été effectuée sur un sol herbeux et boueux. Ces deux expérimentations consistaient pour le robot à se rendre vers un but en se basant sur son odométrie, tout en évitant les obstacles dans l'environnement. Elles ont montré la capacité de notre méthode de navigation à produire un comportement fluide et sécuritaire du robot sur divers types de sols, ainsi qu'à prendre des décisions pertinentes pour le déclenchement de l'évitement et le choix du sens de contournement.

- Les méthodes de navigation en environnement dynamique présentées dans le chapitre 3 ont quant à elles fait l'objet d'expérimentations préliminaires dans le but de valider certains points présentés. Pour la méthode *Distance-Angle Adaptatifs*, une série de tests a permis de montrer que le calcul de la distance et de l'angle adaptatif permet une anticipation du mouvement d'un obstacle lorsque celui-ci est le plus proche du robot, menant à un déclenchement précoce de l'évitement et un choix du sens de contournement prenant en compte le mouvement de l'obstacle. Pour la méthode *Enhanced Laser Scan*, des expérimentations réalisées avec le robot immobile, ont montré que notre algorithme était bien en mesure de : (i) détecter les points mobiles dans l'environnement ; (ii) regrouper ces points relativement aux différents obstacles ; (iii) calculer les caractéristiques dynamiques de chacun des obstacles; (iv) ajouter des points artificiels pour chaque obstacle mobile afin de constituer l'*Enhanced Laser Scan*.

Pour conclure, nous avons proposé dans cette thèse un ensemble d'approches référencées capteurs, permettant à un robot mobile de naviguer en autonomie dans des environnements agricoles statiques et dynamiques. Les forces de nos approches sont leur adaptabilité et leur généricité :

- Aucune hypothèse n'est faite sur le nombre, la taille ou la forme des obstacles. Nos algorithmes d'évitement peuvent gérer des obstacles de diverses tailles (humains, batiments, etc.) et formes (convexes, concaves, etc.). Les groupements de petits obstacles sont aussi correctement gérés.

- Pour les obstacles dynamiques, aucune hypothèse n'est faite sur la dynamique de ces obstacles (direction, vitesse, etc.). La prise en compte des éléments mobiles est faite directement au niveau des acquisitions capteurs.
- Le fait de pouvoir régler les gains des contrôleurs de suivi de spirales, les paramètres des spirales, et la vitesse linéaire du robot permet à nos méthodes de s'adapter et de fonctionner à la fois sur des robots de petite taille, mais aussi sur des robots de grande taille.

5.2 Perspectives futures

Les résultats théoriques exposés dans les chapitres 2 et 3 ont été appuyés par des résultats pratiques présentés dans le chapitre 4. Ces résultats ont montré le bon fonctionnement de notre méthode de navigation en environnement statique, ainsi que des résultats prometteurs pour nos méthodes en environnement dynamique. Par conséquent, les travaux effectués dans cette thèse offrent des perspectives intéressantes.

5.2.1 Adaptation à d'autres cinématiques de robot

Figure 5.1: Robot Dino, avec ses roues orientées (Crédit Tien TRAN)

Les méthodes de navigation présentées aux chapitres 2 et 3 se basent sur un modèle cinématique différentiel. Cependant, il est possible de concevoir des robots agricoles avec des cinématiques moins classiques, comme par exemple le robot Dino de Naïo Technologies, visible sur l'image. Sur ce robot à quatre roues, chacune est orientable indépendamment. L'image 5.1 montre notamment le robot avec toutes ses roues orientées à 45°. Cela lui permet ainsi une très grande flexibilité en terme de déplacement. Ainsi, il peut dans cette orientation tourner sur lui-même sans frottement. De plus, cela peut lui permettre, dans notre contexte, de maintenir son

orientation constante tout en évitant un éventuel obstacle. Adapter notre méthode à cette particularité peut offrir certaines opportunités à explorer, notamment en terme de réduction des frottements et de l'augmentation de la précision odométrique.

5.2.2 Développement de nouvelles lois de commandes

Dans la section 2.2, nous avons introduit deux lois de commandes permettant à un robot de rejoindre et de suivre une spirale quelconque. Cependant, chacune de ces lois souffre de certaines limitations. La première possède des zones de singularité qui seront forcément traversées dans un contexte d'évitement, tandis que la seconde n'a pas été prouvée globalement asymptotiquement stable sur l'ensemble de l'espace. Ainsi, nous avons proposé une solution se basant sur l'usage de l'une ou l'autre de ces lois de commande en fonction de l'erreur angulaire. Des travaux futurs pourront se porter sur de développement et la définition de lois de commande globalement asymptiquement stables sur tout le domaine d'évolution du robot, sans présence de singularité. Des pistes de recherche, basées notamment sur la théorie de Lyapunov, pourront être explorées.

5.2.3 Intégration dans une manipulation sur site

Dans le chapitre 4, des résultats préliminaires ont été exposés concernant les méthodes *Distance-Angle Adaptatifs* et les méthodes basées sur l'*Enhanced Laser Scan*. Les expérimentations réalisées ont permis de valider plusieurs points pratiques de ces méthodes. Par conséquent, ces solutions devront être expérimentées dans le cadre de tâches de navigation typiques d'applications agricoles. De plus, comme cela a été indiqué dans l'introduction, cette thèse s'est déroulée dans le cadre du projet *Desherb'eur*. Ce projet a impliqué diverses personnes, qui ont notamment travaillé sur des problématiques de navigation globale [Emmi et al., 2019], ainsi que des problématiques de suivie de rangée et de demi-tour en fin de rangée ([Le Flecher et al., 2017], [Le Flecher et al., 2019]). Dans le cadre de la validation des jalons du projet *Desherb'eur*, des résultats ont été obtenus sur le parcours d'une vigne au long-cours sans évitement d'obstacles, et sur l'évitement d'obstacles dans la vigne. Ainsi, une démonstration long-cours combinant les travaux respectifs des gens ayant contribué sur ce projet pourrait faire l'objet de futurs travaux. Cette démonstration simulerait un cas de navigation et d'utilisation réel du robot :

- Le robot partirait de sa zone de stockage et devra se rendre à l'entrée d'une parcelle, dont la position sera indiquée sur une carte topologique. Il utilisera pour cela une méthode de navigation globale basée GNSS, mais aussi les méthodes d'évitement d'obstacles présentées dans cette thèse afin de gérer les éléments imprévus qui seront présents sur son chemin.

- Une fois arrivé à l'entrée de la parcelle, le robot parcourrait les rangées successives, effectuant les demi-tours en fin de rangée selon les méthodes spécifiées.
- Une fois toutes les rangées parcourues, le robot devra retourner à sa zone de stockage, évitant de la même façon les obstacles statiques et dynamiques présents sur son chemin.

Pour conclure, l'objectif de cette thèse a été d'apporter des solutions techniques et scientifiques à la problématique de la navigation d'un robot autonome au sein d'un environnement agricole. Ces solutions font aussi partie des réponses à une problématique beaucoup plus globale et importante, visant à rendre les exploitations agricoles totalement autonomes, où chaque étape, de la semence à la récolte, serait automatisée. Cependant, il est important de rappeler que de part les contributions scientifiques exposées, cette thèse s'inscrit aussi dans une problématique économique et sociale beaucoup plus contemporaine : relever le défi de nourrir une population humaine grandissante, tout en diminuant de façon urgente l'impact environnemental et écologique que peut avoir de nos jours l'industrie agricole.

Références bibliographiques

[Adouane, 2009] Adouane, L. (2009). Orbital Obstacle Avoidance Algorithm for Reliable and On-Line Mobile Robot Navigation. In *9th Conference on Autonomous Robot Systems and Competitions*, Castelo-Branco, Portugal.

[Adouane et al., 2011] Adouane, L., Benzerrouk, A., and Martinet, P. (2011). Mobile Robot Navigation in Cluttered Environment using Reactive Elliptic Trajectories. In *18th IFAC World Congress*, Milano, Italy.

[Ahmad et al., 2019] Ahmad, H., Mohamad Pajeri, A. N. F., Othman, N. A., Saari, M. M., and Ramli, M. S. (2019). Analysis of mobile robot path planning with artificial potential fields. In Md Zain, Z., Ahmad, H., Pebrianti, D., Mustafa, M., Abdullah, N. R. H., Samad, R., and Mat Noh, M., editors, *Proceedings of the 10th National Technical Seminar on Underwater System Technology 2018*, pages 181–196, Singapore. Springer Singapore.

[Al-Haddad et al., 2011] Al-Haddad, A. A., Sudirman, R., and Omar, C. (2011). Guiding wheelchair motion based on eog signals using tangent bug algorithm. In *2011 Third International Conference on Computational Intelligence, Modelling Simulation*, pages 40–45.

[Alberto-Rodriguez et al., 2020] Alberto-Rodriguez, A., Neri-Muñoz, M., Ramos-Fernández, J. C., Márquez-Vera, M. A., Ramos-Velasco, L. E., Díaz-Parra, O., and Hernández-Huerta, E. (2020). Review of control on agricultural robot tractors. *International Journal of Combinatorial Optimization Problems and Informatics*, 11(3):9–20.

[Babinec et al., 2014] Babinec, A., Duchoň, F., Dekan, M., Pásztó, P., and Kelemen, M. (2014). Vfh*tdt (vfh* with time dependent tree): A new laser rangefinder based obstacle avoidance method designed for environment with non-static obstacles. *Robotics and Autonomous Systems*, 62(8):1098 – 1115.

[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.

[Berg et al., 2011] Berg, J. V. D., Snape, J., Guy, S. J., and Manocha, D. (2011). Reciprocal collision avoidance with acceleration-velocity obstacles. In *2011 IEEE International Conference on Robotics and Automation*, pages 3475–3482.

[Bergerman et al., 2016] Bergerman, M., Billingsley, J., Reid, J., and van Henten, E. (2016). *Robotics in Agriculture and Forestry*, pages 1463–1492. Springer International Publishing, Cham.

[Blackmore et al., 2001] Blackmore, B., Have, H., and Fountas, S. (2001). A specification of behavioural requirements for an autonomous tractor. In *International Symposium on Fruit, Nut and Vegetable Production Engineering conference*.

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

[Borenstein and Koren, 1989] Borenstein, J. and Koren, Y. (1989). Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(5):1179–1187.

[Borenstein and Koren, 1991] Borenstein, J. and Koren, Y. (1991). The vector field histogram-fast obstacle avoidance for mobile robots. *Robotics and Automation, IEEE Transactions on*, 7(3):278–288.

[Boyadzhiev, 1999] Boyadzhiev, K. N. (1999). Spirals and conchospirals in the flight of insects. *The College Mathematics Journal*, 30(1):pp. 23–31.

[Brock and Khatib, 1999] Brock, O. and Khatib, O. (1999). High-speed navigation using the global dynamic window approach. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, volume 1, pages 341–346 vol.1.

[Clements and Shrestha, 2004] Clements, D. and Shrestha, A. (2004). New dimensions in agroecology for developing a biological approach to crop production. *Journal of Crop Improvement*, 11:1–20.

[Cybulski et al., 2019] Cybulski, B., Wegierska, A., and Granosik, G. (2019). Accuracy comparison of navigation local planners on ros-based mobile robot. In *2019 12th International Workshop on Robot Motion and Control (RoMoCo)*, pages 104–111.

[Davies et al., 2018] Davies, E., Garlow, A., Farzan, S., Rogers, J., and Hu, A. (2018). Tarzan: Design, prototyping, and testing of a wire-borne brachiating robot. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7609–7614.

[Ding and Frank, 1991] Ding, X. and Frank, P. (1991). Frequency domain approach and threshold selector for robust model-based fault detection and isolation. *IFAC Proceedings Volumes*, 24(6):271 – 276. IFAC/IMACS Symposium on Fault Detection, Supervision and Safety for Technical Processes (SAFEPROCESS'91), Baden-Baden, Germany, 10-13 September 1991.

[Dominguez et al., 2011] Dominguez, R., Onieva, E., Alonso, J., Villagra, J., and Gonzalez, C. (2011). Lidar based perception solution for autonomous vehicles. *International Conference on Intelligent Systems Design and Applications, ISDA*, pages 790–795.

[Dubins, 1957] Dubins, L. E. (1957). On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79(3):497–516.

[Duckett et al., 2018] Duckett, T., Pearson, S., Blackmore, S., and Grieve, B. (2018). Agricultural robotics: The future of robotic agriculture. *ArXiv*, abs/1806.06762.

[Durand-Petiteville et al., 2018] Durand-Petiteville, A., Le Flecher, E., Cadenat, V., Sentenac, T., and Vougioukas, S. (2018). Tree detection with low-cost three-dimensional sensors for autonomous navigation in orchards. *IEEE Robotics and Automation Letters*, 3(4):3876–3883.

[Durand-Petiteville et al., 2017] Durand-Petiteville, A., Le Flécher, E., Cadenat, V., Sentenac, T., and Vougioukas, S. (2017). Design of a sensor-based controller performing u-turn to navigate in orchards. pages 172–181.

[Eberhardt and Vollrath, 2018] Eberhardt, M. and Vollrath, D. (2018). The effect of agricultural technology on the speed of development. *World Development*, 109:483 – 496.

[Emami-Naeini et al., 1988] Emami-Naeini, A., Akhter, M. M., and Rock, S. M. (1988). Effect of model uncertainty on failure detection: the threshold selector. *IEEE Transactions on Automatic Control*, 33(12):1106–1115.

[Emmi et al., 2019] Emmi, L., Dufour, J., Cadenat, V., and Devy, M. (2019). *Hybrid topological location and mapping for autonomous agricultural robots*, chapter 95, pages 767–774.

[Erick et al., 2018] Erick, M., Fiestas, S., Sixto, R., and Prado, G. (2018). Modeling and simulation of kinematics and trajectory planning of a farmbot cartesian robot. In *2018 IEEE XXV International Conference on Electronics, Electrical Engineering and Computing (INTERCON)*, pages 1–4.

[FAO and CELAC, 2020] FAO and CELAC (2020). *Food security under the COVID-19 pandemic*. FAO.

[FAO et al., 2019] FAO, Portocarrero-Aya, M., and Hinkes, C. (2019). *The State of the World's Biodiversity for Food and Agriculture*. FAO.

[Farzan et al., 2018] Farzan, S., Hu, A., Davies, E., and Rogers, J. (2018). Modeling and control of brachiating robots traversing flexible cables. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1645–1652.

[Filotheou et al., 2020] Filotheou, A., Tsardoulias, E., Dimitriou, A., Symeonidis, A., and Petrou, L. (2020). Quantitative and qualitative evaluation of ros-enabled local and global planners in 2d static environments. *Journal of Intelligent and Robotic Systems*, 98.

[Fiorini and Shiller, 1998] Fiorini, P. and Shiller, Z. (1998). Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research*, 17:760–.

[Flecher et al., 2017] Flecher, E. L., Durand-Petiteville, A., Cadenat, V., Sentenac, T., and Vougioukas, S. (2017). Design of a sensor-based controller performing u-turn to navigate in orchards. In *International Conference on Informatics in Control, Automation and Robotics*, Madrid, Spain.

[Fortin et al., 2012] Fortin, B., Noyer, J. C., and Lherbier, R. (2012). A particle filtering approach for joint vehicular detection and tracking in lidar data. In *2012 IEEE International Instrumentation and Measurement Technology Conference Proceedings*, pages 391–396.

[Fox et al., 1997] Fox, D., Burgard, W., and Thrun, S. (1997). The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33.

[Fraichard and Asama, 2003] Fraichard, T. and Asama, H. (2003). Inevitable collision states. a step towards safer robots? volume 18, pages 388 – 393 vol.1.

[Fraundorfer et al., 2012] Fraundorfer, F., Heng, L., Honegger, D., Lee, G. H., Meier, L., Tanskanen, P., and Pollefeys, M. (2012). Vision-based autonomous mapping and exploration using a quadrotor mav. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4557–4564.

[Futterlieb, 2017] Futterlieb, M. (2017). *Vision based navigation in a dynamic environment*. PhD thesis, Université de Toulouse.

[Futterlieb et al., 2014] Futterlieb, M., Cadenat, V., and Sentenac, T. (2014). A navigational framework combining visual servoing and spiral obstacle avoidance techniques. In *Informatics in Control, Automation and Robotics (ICINCO), 2014 11th International Conference on*, volume 02, pages 57–64.

[Galanakis, 2020] Galanakis, C. (2020). The food systems in the era of the coronavirus (covid-19) pandemic crisis. *Foods*, 9.

[Ge and Cui, 2002] Ge, S. and Cui, Y. (2002). Dynamic motion planning for mobile robots using potential field method. *Autonomous Robots*, 13(3):207–222.

[Gonzalez et al., 2018] Gonzalez, F., Mcfadyen, A., and Puig, E. (2018). Advances in unmanned aerial systems and payload technologies for precision agriculture. In *Advances in agricultural machinery and technologies*, pages 133–155. CRC Press.

[Gonzalez Bautista et al., 2015] Gonzalez Bautista, D., Pérez, J., Milanes, V., and Nashashibi, F. (2015). A review of motion planning techniques for automated vehicles. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–11.

[Guerra et al., 2016] Guerra, M., Efimov, D., Zheng, G., and Perruquetti, W. (2016). Avoiding local minima in the potential field method using input-to-state stability. *Control Engineering Practice*, 55:174 – 184.

[Harrell et al., 1985] Harrell, R., Adsit, P., and Slaughter, D. (1985). Real-time vision-servoing of a robotic tree fruit harvester. *Trans. ASAE*, 85:1–15.

[Hernández-Aceituno et al., 2015] Hernández-Aceituno, J., Acosta, L., and Piñeiro, J. (2015). Application of time dependent probabilistic collision state checkers in highly dynamic environments. *PLOS ONE*, 10:e0119930.

[Hoffmann et al., 2008] Hoffmann, W., Lan, Y., Wu, W., and Fritz, B. (2008). Development of a spray system for an unmanned aerial vehicle platform. *Applied Engineering in Agriculture*, 25.

[Hoy et al., 2015] Hoy, M., Matveev, A. S., and Savkin, A. V. (2015). Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey. *Robotica*, 33:463–497.

[Jie et al., 2010] Jie, D., Xueming, M., and Kaixiang, P. (2010). Ivfh* : Real-time dynamic obstacle avoidance for mobile robots. In *2010 11th International Conference on Control Automation Robotics Vision*, pages 844–847.

[Kamil and N, 2015] Kamil, F. and N, K. (2015). A review on motion planning and obstacle avoidance approaches in dynamic environments. *Advances in Robotics and Automation*, 04.

[Kamon et al., 1998] Kamon, I., Rimon, E., and Rivlin, E. (1998). Tangentbug: A range-sensor-based navigation algorithm. *The International Journal of Robotics Research*, 17(9):934–953.

[Kamon and Rivlin, 1997] Kamon, I. and Rivlin, E. (1997). Sensory-based motion planning with global proofs. *IEEE Transactions on Robotics and Automation*, 13(6):814–822.

[Keselman et al., 2017] Keselman, L., Iselin Woodfill, J., Grunnet-Jepsen, A., and Bhowmik, A. (2017). Intel realsense stereoscopic depth cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–10.

[Khatib and Chatila, 1995] Khatib, M. and Chatila, R. (1995). An extended potential field approach for mobile robot sensor-based motions. In Press, I., editor, *Intelligent Autonomous Systems (IAS)*, pages 490–496, Karlsruhe, Germany.

[Khatib, 1985] Khatib, O. (1985). Real-time obstacle avoidance for manipulators and mobile robots. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, volume 2, pages 500–505.

[Koren and Borenstein, 1991] Koren, Y. and Borenstein, J. (1991). Potential field methods and their inherent limitations for mobile robot navigation. In *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, pages 1398–1404 vol.2.

[Kudriashov et al., 2020] Kudriashov, A., Buratowski, T., Giergiel, M., and Małka, P. (2020). *Introduction to Mobile Robots Navigation, Localization and Mapping*, pages 7–38. Springer International Publishing, Cham.

[Laborde et al., 2020] Laborde, D., Martin, W., Swinnen, J., and Vos, R. (2020). Covid-19 risks to global food security. *Science*, 369(6503):500–502.

[Large et al., 2005] Large, F., Laugier, C., and Shiller, Z. (2005). Navigation among moving obstacles using the nlvo: Principles and applications to intelligent vehicles. *Autonomous Robots*, 19:159–171.

[Le Flecher et al., 2017] Le Flecher, E., Durand-Petiteville, A., Cadenat, V., Sentenac, T., and Vougioukas, S. (2017). Implementation on a harvesting robot of a sensor-based controller performing a u-turn. In *IEEE International Workshop of Electronics, Control, Measurement, Signals and their Application to Mechatronics (2017 ECMSM)*, pages art. 7945895 – 6 p., San Sebastian, Spain. IEEE.

[Le Flecher et al., 2019] Le Flecher, E., Durand-Petiteville, A., Gouaisbaut, F., Cadenat, V., Sentenac, T., and Vougioukas, S. (2019). Nonlinear output feedback for autonomous u-turn maneuvers of a robot in orchard headlands. pages 355–362.

[Leaman, 2012] Leaman, D. (2012). *The State of the World's Land and Water Resources for Food and Agriculture (SOLAW) - Managing Systems at Risk*, volume 66. FAO and Earthscan.

[Leca et al., 2019] Leca, D., Cadenat, V., and Sentenac, T. (2019). Sensor-based algorithm for collision-free avoidance of mobile robots in complex dynamic environments. In *2019 European Conference on Mobile Robots (ECMR)*, pages p.1–6, Prague, Czech Republic.

[Leca et al., 2019] Leca, D., Cadenat, V., Sentenac, T., Durand-Petiteville, A., Gouaisbaut, F., and Le Flécher, E. (2019). Sensor-based obstacles avoidance using spiral controllers for an aircraft maintenance inspection robot. In *2019 18th European Control Conference (ECC)*, pages 2083–2089.

[Lu et al., 2009] Lu, Y., Yixin, Y., and Cheng-Jian, L. (2009). A new potential field method for mobile robot path planning in the dynamic environments. *Asian Journal of Control*, 11(2):214–225.

[Mabrouk and McInnes, 2008a] Mabrouk, M. and McInnes, C. (2008a). An emergent wall following behaviour to escape local minima for swarms of agents. *IAENG International Journal of Computer Science*, 35.

[Mabrouk and McInnes, 2008b] Mabrouk, M. and McInnes, C. (2008b). Solving the potential field local minimum problem using internal agent states. *Robotics and Autonomous Systems*, 56(12):1050 – 1060. Towards Autonomous Robotic Systems 2008: Mobile Robotics in the UK.

[Marioara et al., 2019] Marioara, R. et al. (2019). Agro-ecology: Concept, characteristics and socio-economic benefits. *Agricultural Economics and Rural Development*, 16(1):65–76.

[Mcfadyen et al., 2012a] Mcfadyen, A., Corke, P., and Mejias, L. (2012a). Rotorcraft collision avoidance using spherical image-based visual servoing and single point features. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 1199–1205.

[Mcfadyen et al., 2014] Mcfadyen, A., Corke, P., and Mejias, L. (2014). Visual predictive control of spiral motion. *IEEE Transactions on Robotics*, 30(6):1441–1454.

[Mcfadyen et al., 2012b] Mcfadyen, A., Mejias, L., and Corke, P. (2012b). Visual servoing approach to collision avoidance for aircraft. In *28th Congress of the International Council of the Aeronautical Sciences 2012*, Brisbane Convention & Exhibition Centre, Brisbane, QLD.

[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.

[M.G and Salgoankar, 2017] M.G, M. and Salgoankar, A. (2017). A survey of robotic motion planning in dynamic environments. *Robotics and Autonomous Systems*, 100.

[Min Gyu Park and Min Cheol Lee, 2004] Min Gyu Park and Min Cheol Lee (2004). Real-time path planning in unknown environment and a virtual hill concept to escape local minima. In *30th Annual Conference of IEEE Industrial Electronics Society, 2004. IECON 2004*, volume 3, pages 2223–2228 Vol. 3.

[Mogili and Deepak, 2018] Mogili, U. R. and Deepak, B. B. V. L. (2018). Review on application of drone systems in precision agriculture. *Procedia Computer Science*, 133:502 – 509. International Conference on Robotics and Smart Manufacturing (RoSMa2018).

[Mohamed et al., 2011] Mohamed, E., El-Metwally, K., and Hanafy, A. (2011). An improved tangent bug method integrated with artificial potential field for multi-robot path planning. pages 555–559.

[Montiel et al., 2015] Montiel, O., Orozco-Rosas, U., and Sepúlveda, R. (2015). Path planning for mobile robots using bacterial potential field for avoiding static and dynamic obstacles. *Expert Systems with Applications*, 42(12):5177 – 5191.

[Moseley, 2017] Moseley, W. G. (2017). A risky solution for the wrong problem: Why gmos won't feed the hungry of the world. *Geographical Review*, 107(4):578–583.

[Nicola et al., 2020] Nicola, M., Alsafi, Z., Sohrabi, C., Kerwan, A., Al-Jabir, A., Iosifidis, C., Agha, M., and Agha, R. (2020). The socio-economic implications of the coronavirus and covid-19 pandemic: A review. *International Journal of Surgery*, 78.

[Peng et al., 2015] Peng, Y., Qu, D., Zhong, Y., Xie, S., Luo, J., and Gu, J. (2015). The obstacle detection and obstacle avoidance algorithm based on 2-d lidar. In *2015 IEEE International Conference on Information and Automation*, pages 1648–1653.

[Prassler et al., 2001] Prassler, E., Scholz, J., and Fiorini, P. (2001). A robotics wheelchair for crowded public environment. *IEEE Robotics Automation Magazine*, 8(1):38–45.

[Pretty, 2008] Pretty, J. (2008). Agricultural sustainability: Concepts, principles and evidence. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, 363:447–65.

[Puig et al., 2015] Puig, E., Gonzalez, F., Hamilton, G. S., and Grundy, P. (2015). Assessment of crop insect damage using unmanned aerial systems: A machine learning approach.

[Qixin et al., 2006] Qixin, C., Yanwen, H., and Jingliang, Z. (2006). An evolutionary artificial potential field algorithm for dynamic path planning of mobile robot. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3331–3336.

[Raghavan et al., 2016] Raghavan, B., Nardi, B., Lovell, S. T., Norton, J., Tomlinson, B., and Patterson, D. J. (2016). Computational agroecology: Sustainable food ecosystem design. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '16, page 423–435, New York, NY, USA. Association for Computing Machinery.

[Rokach and Maimon, 2005] Rokach, L. and Maimon, O. (2005). *Data mining and knowledge discovery handbook*, chapter Clustering methods, pages 321–352. Springer US.

[Rösmann et al., 2012] Rösmann, C., Feiten, W., Wösch, T., Hoffmann, F., and Bertram, T. (2012). Trajectory modification considering dynamic constraints of autonomous robots. In *Robotics; Proceedings of ROBOTIK 2012; 7th German Conference on*, pages 1–6. VDE.

[Rosmann et al., 2013] Rosmann, C., Feiten, W., Wosch, T., Hoffmann, F., and Bertram, T. (2013). Efficient trajectory optimization using a sparse model. In *Mobile Robots (ECMR), 2013 European Conference on Mobile Robots*, pages 138–143. IEEE.

[Rudolph et al., 2020] Rudolph, M., Muchesa, E., and Kroll, F. (2020). Towards an eco-social food system: The shift from industrial agriculture to agro-ecology in south africa. *International Journal of Sustainable Development Research*, 6(2):22.

[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. *Sociologia Ruralis*, 58.

[Rösmann et al., 2015] Rösmann, C., Hoffmann, F., and Bertram, T. (2015). Planning of multiple robot trajectories in distinctive topologies. In *2015 European Conference on Mobile Robots (ECMR)*, pages 1–6.

[Rösmann et al., 2016] Rösmann, C., Hoffmann, F., and Bertram, T. (2016). Integrated online trajectory planning and optimization in distinctive topologies. *Robotics and Autonomous Systems*, 88.

[Sastry, 1999] Sastry, S. (1999). *Lyapunov Stability Theory*, pages 182–234. Springer New York, New York, NY.

[Sezer and Gokasan, 2012] Sezer, V. and Gokasan, M. (2012). A novel obstacle avoidance algorithm: "follow the gap method". *Robotics and Autonomous Systems*, 60(9):1123 – 1134.

[Shi et al., 2005] Shi, Z., Gu, F., Lennox, B., and Ball, A. (2005). The development of an adaptive threshold for model-based fault detection of a nonlinear electro-hydraulic system. *Control Engineering Practice*, 13(11):1357 – 1367.

[Shiller et al., 2010] Shiller, Z., Gal, O., and Fraichard, T. (2010). The nonlinear velocity obstacle revisited: the optimal time horizon. *Guaranteeing Safe Navigation in Dynamic Environments Workshop*.

[Shiller et al., 2001] Shiller, Z., Large, F., and Sekhavat, S. (2001). Motion planning in dynamic environments: obstacles moving along arbitrary trajectories. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, volume 4, pages 3716–3721 vol.4.

[Spoorthi S. et al., 2017] Spoorthi S., Shadaksharappa, B., Suraj S., and Manasa, V. K. (2017). Freyr drone: Pesticide/fertilizers spraying drone - an agricultural approach. In *2017 2nd International Conference on Computing and Communications Technologies (ICCCT)*, pages 252–255.

[Steensland and Zeigler, 2018] Steensland, A. and Zeigler, M. (2018). World population prospects 2019. Technical report, Global Harvest Initiative.

[Tarazona et al., 2017] Tarazona, J., Court Marques, D., Tiramani, M., Reich, H., Pfeil, R., Istace, F., and Crivellente, F. (2017). Glyphosate toxicity and carcinogenicity: a review of the scientific basis of the european union assessment and its differences with iarc. *Archives of toxicology*, 91.

[Taylor et al., 2012] Taylor, J., Charlton, D., and Yunez-Naude, A. (2012). The end of farm labor abundance. *Applied Economic Perspectives and Policy*, 34:587–598.

[Thanpattranon et al., 2016] Thanpattranon, P., Ahamed, T., and Takigawa, T. (2016). Navigation of autonomous tractor for orchards and plantations using a laser range finder: Automatic control of trailer position with tractor. *Biosystems Engineering*, 147:90 – 103.

[Thuy and Puente Leon, 2009] Thuy, M. and Puente Leon, F. (2009). Non-linear, shape independent object tracking based on 2d lidar data. In *2009 IEEE Intelligent Vehicles Symposium*, pages 532–537.

[Ulrich and Borenstein, 1998] Ulrich, I. and Borenstein, J. (1998). Vfh+: Reliable obstacle avoidance for fast mobile robots. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 2, pages 1572–1577. IEEE.

[Ulrich and Borenstein, 2000] Ulrich, I. and Borenstein, J. (2000). Vfh*: local obstacle avoidance with look-ahead verification. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, volume 3, pages 2505–2511 vol.3.

[United Nations, 2019] United Nations, Department of Economic & Social Affairs, P. D. (2019). World population prospects 2019. Technical report, United Nations, Department of Economic and Social Affairs, Population Division.

[Vidoni et al., 2017] Vidoni, R., Gallo, R., Ristorto, G., Carabin, G., Mazzetto, F., Scalera, L., and Gasparetto, A. (2017). Byelab: an agricultural mobile robot prototype for proximal sensing and precision farming. In *ASME 2017 International Mechanical Engineering Congress and Exposition*. American Society of Mechanical Engineers Digital Collection.

[Vilca et al., 2013] Vilca, J. M., Adouane, L., and Mezouar, Y. (2013). *On-Line Obstacle Detection Using Data Range for Reactive Obstacle Avoidance*, pages 3–13. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Vilca Ventura et al., 2012] Vilca Ventura, J. M., Adouane, L., and Mezouar, Y. (2012). On-line Obstacle Detection using Data Range for Reactive Obstacle Avoidance. *Advances in Intelligent Systems and Computing*.

[Virk et al., 2020] Virk, A. L., Noor, M. A., Fiaz, S., Hussain, S., Hussain, H. A., Rehman, M., Ahsan, M., and Ma, W. (2020). *Smart Farming: An Overview*, pages 191–201. Springer International Publishing, Cham.

[Vroegindeweij et al., 2018] Vroegindeweij, B., Blaauw, S., Ijsselmuiden, J., and Van Henten, E. (2018). Evaluation of the performance of poultrybot, an autonomous mobile robotic platform for poultry houses. *Biosystems Engineering*, 174.

[Wang and Noguchi, 2016] Wang, H. and Noguchi, N. (2016). Autonomous maneuvers of a robotic tractor for farming. In *2016 IEEE/SICE International Symposium on System Integration (SII)*, pages 592–597.

[Wang et al., 2003] Wang, X., Kruger, U., and Lennox, B. (2003). Recursive partial least squares algorithms for monitoring complex industrial processes. *Control Engineering Practice*, 11(6):613 – 632.

[Wu et al., 2018] Wu, X., Aravecchia, S., and Pradalier, C. (2018). Design and Implementation of Computer Vision based In-Row Weeding System. working paper or preprint.

[Yallappa et al., 2017] Yallappa, D., Veerangouda, M., Maski, D., Palled, V., and Bheemanna, M. (2017). Development and evaluation of drone mounted sprayer for pesticide applications to crops. In *2017 IEEE Global Humanitarian Technology Conference (GHTC)*, pages 1–7.

[Yao et al., 2016] Yao, L., Jiang, Y., Zhao, Z., Shuaishuai, Y., and Quan, Q. (2016). A pesticide spraying mission assignment performed by multi-quadcopters and its simulation platform establishment. pages 1980–1985.

[You et al., 2008] You, B., Qiu, J., and Li, D. (2008). A novel obstacle avoidance method for low-cost household mobile robot. In *2008 IEEE International Conference on Automation and Logistics*, pages 111–116.

[Yufka and Parlaktuna, 2009] Yufka, A. and Parlaktuna, O. (2009). Performance comparison of bug algorithms for mobile robots. In *Proceedings of the 5th international advanced technologies symposium, Karabuk, Turkey*, pages 13–15.

[Zhang et al., 2019] Zhang, L., Rana, I., Shaffer, R. M., Taioli, E., and Sheppard, L. (2019). Exposure to glyphosate-based herbicides and risk for non-hodgkin lymphoma: A meta-analysis and supporting evidence. *Mutation Research/Reviews in Mutation Research*, 781:186 – 206.

[Zhang et al., 2013] Zhang, Q., Yue, S.-g., Yin, Q.-j., and Zha, Y.-b. (2013). Dynamic obstacle-avoiding path planning for robots based on modified potential field method. In Huang, D.-S., Jo, K.-H., Zhou, Y.-Q., and Han, K., editors, *Intelligent Computing Theories and Technology*, pages 332–342, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Zhang, 2012] Zhang, Z. (2012). Microsoft kinect sensor and its effect. *IEEE multimedia*, 19(2):4–10.

[Zhu et al., 2006] Zhu, Q., Yan, Y., and Xing, Z. (2006). Robot path planning based on artificial potential field approach with simulated annealing. In *Sixth International Conference on Intelligent Systems Design and Applications*, volume 2, pages 622–627.

[Zhu et al., 2012] Zhu, Y., Zhang, T., Song, J., Li, X., and Nakamura, M. (2012). A new method for mobile robots to avoid collision with moving obstacle. *Artificial Life and Robotics*, 16(4):507–510.

[Zohaib et al., 2014a] Zohaib, M., Pasha, S. M., Javaid, N., and Iqbal, J. (2014a). Iba: Intelligent bug algorithm – a novel strategy to navigate mobile robots autonomously. In Shaikh, F. K., Chowdhry, B. S., Zeadally, S., Hussain, D. M. A., Memon, A. A., and Uqaili, M. A., editors, *Communication Technologies, Information Security and Sustainable Development*, pages 291–299, Cham. Springer International Publishing.

[Zohaib et al., 2014b] Zohaib, M., Pasha, S. M., Javaid, N., Salaam, A., and Iqbal, J. (2014b). An improved algorithm for collision avoidance in environments having u and h shaped obstacles. *Studies in Informatics and Control*, 23(1):97–106.