N° d'ordre : D. U : 2123

E D S P I C : 518

UNIVERSITE BLAISE PASCAL - CLERMONT II

ECOLE DOCTORALE
SCIENCES POUR L'INGENIEUR DE CLERMONT-FERRAND

Thèse

Présentée par

AHMED BENZERROUK
titre : Master Mécatronique

pour obtenir le grade de

DOCTEUR D'UNIVERSITE

SPECIALITE: Vision pour la robotique

Architecture de Contrôle Hybride pour les Systèmes Multi-robots Mobiles.

Soutenue publiquement le 18 avril 2011 devant le jury:

COMPOSITION DU JURY

M.	Thierry FRAICHARD	Président du jury
M.	Lounis ADOUANE	Encadrant
M.	Rachid ALAMI	Rapporteur
M.	Philippe MARTINET	Encadrant
M.	René ZAPATA	Rapporteur
M.	Nicolas ANDREFF	Directeur de thèse

Table des matières

Remerciements

Un grand merci à mes parents !
Sans leur soutien, ma thèse n'aurait jamais vu le jour.

Cette thèse a été réalisée au LASMEA (LAboratoire des Sciences et Matériaux pour l'Electronique, et d'Automatique), au sein du groupe GRAVIR (Groupe Automatique, Vision et Robotique). Elle n'a pu être menée à bout sans l'aide de certaines personnes auxquelles je veux dire merci.

Ainsi, je tiens à remercier en premier lieu Rachid Alami, Directeur de Recherche CNRS au LAAS, et René Zapata, Professeur à l'université de Montpellier II, pour avoir accepté de rapporter sur cette thèse.

Ma reconnaissance va aussi à Thierry Fraichard pour m'avoir fait l'honneur de présider le jury de soutenance.

Je remercie Nicolas Andreff Professeur à l'université de Franche-Comté, pour avoir dirigé ma thèse les deux premières années avant de rejoindre l'université de Franche-Comté. Je remercie également, Philippe Martinet, Professeur à l'Institut Français de Mécanique Avancée pour avoir accepté de le remplacer la dernière année. Je les remercie tous les deux pour leurs conseils qui ont aidé à l'avancement des travaux de ma thèse, et m'ont permis de prendre du recul pour placer et situer mes travaux.

Je remercie Lounis Adouane, maitre de conférences à Polytech'Clermont-Ferrand pour avoir encadré cette thèse. Je le remercie pour ses conseils, pour avoir toujours été disponible et proche de mes travaux tout en me laissant une grande liberté d'action et de décision.

Mes remerciements vont aussi à Laurent Lequièvre pour tout le temps et les efforts que nous avons investis dans la partie expérimentale. Je ne peux oublier de remercier également l'équipe VIPA qui nous a soutenus : Nadir Karam, Clément Deymier, Hicham Hadj et Datta Ramadasan.

Je veux aussi dire merci à tout ceux qui, de près ou de loin, avec une critique, une remarque ou simplement une discussion le temps d'une pause, ont influencé d'une manière ou d'une autre mes travaux : Omar Ait-Aider, Pierre Avanzini, François Berry, Jonathan Courbon, Roland Lenain, Guillaume Lozenguez, Youcef Mezouar, Jean-Charles Quinton, Benoit Thuilot, et j'en oublie certainement d'autres. Je m'excuse d'avoir omis de les citer.

Enfin, merci à tous ceux que j'ai connus au laboratoire rendant la vie dans celui-ci agréable et conviviale. La liste sera longue, mais les personnes se reconnaîtront à travers les pauses café, soirées poker, week-ends canoe, et toutes les activités qui nous ont unis. J'ai pu nouer avec certains une véritable amitié.

Résumé : La complexité inhérente à la coordination des mouvements d'un groupe de robots mobiles est traitée en investiguant plus avant les potentialités des architectures de contrôle comportementales dont le but est de briser la complexité des tâches à exécuter. En effet, les robots mobiles peuvent évoluer dans des environnements très complexes et nécessite de surcroît une coopération précise et sécurisée des véhicules pouvant rapidement devenir inextricable. Ainsi, pour maîtriser cette complexité, le contrôleur dédié à la réalisation de la tâche est décomposé en un ensemble de comportements/contrôleurs élémentaires (évitement d'obstacles et de collision entre les robots, attraction vers une cible, etc.) qui lient les informations capteurs (provenant de caméras, des capteurs locaux du robot, etc.) aux actionneurs des différentes entités robotiques. La tâche considérée est la navigation en formation en présence d'obstacles (statiques et dynamiques). La spécificité de l'approche théorique consiste à allier les avantages des architectures de contrôle comportementales à la méthode de la structure virtuelle où le groupe de robots mobiles suit un corps virtuel avec une dynamique (vitesse, direction) donnée. Ainsi, l'activation d'un comportement élémentaire en faveur d'un autre se fait en respectant les contraintes structurelles des robots (e.g. vitesses et accélérations maximales, etc.) en vue d'assurer le maximum de précision et de sécurité des mouvements coordonnés entre les différentes entités mobiles. La coopération consiste à se partager les places dans la structure virtuelle de manière distribuée et de façon à atteindre plus rapidement la formation désirée. Pour garantir les critères de performances visés par l'architecture de contrôle, les systèmes hybrides qui permettent de commander des systèmes continus en présence d'évènements discrets sont exploités. En effet, ces contrôleurs (partie discrète) permettent de coordonner l'activité des différents comportements (partie continue) disponibles au niveau de l'architecture, tout en offrant une analyse automaticienne rigoureuse de la stabilité de celle-ci au sens de Lyapunov. Chaque contribution est illustrée par des résultats de simulation. Le dernier chapitre est dédié à l'implémentation de l'architecture de contrôle proposée sur un groupe de robots mobiles Khepera III.

Mots-clés : Systèmes multi-robots mobiles, Navigation en formation, Evitement d'obstacles, Architecture de contrôle, Systèmes hybrides, stabilité au sens de Lyapunov.

Abstract : Inherent difficulty of coordinating a group of mobile robots is treated by investigating behavior-based architectures which aim to break task complexity. In fact, multi-robot navigation may become rapidly inextricable, specifically if it is made in hazardous and dynamical environment. The considered task is the navigation in formation in presence of (static and dynamic) obstacles. To overcome its complexity, it is proposed to divide the overall task into two basic behaviors/controllers (obstacle avoidance, attraction to a dynamical target). Applied control is chosen among these controllers according to sensors information (camera, local sensors, etc.). Theoretic approach combines behavior-based and the virtual structure strategy which considers the formation as a virtual body with a given dynamic (velocity, direction). Thus, activating a controller or another is accomplished while respecting structural robots constraints (e.g. maximal velocities and accelerations). The objective is to insure the highest precision and safety of the coordinated motion between the robots. These ones cooperate by optimizing the way of sharing their places in the formation in order to form it in a faster manner. To guarantee performance criteria of the control architecture, hybrid systems tolerating the control of continuous systems in presence of discrete events are explored. In fact, this control allows coordinating (by discrete part) the different behaviors (continuous part) of the architecture. A complete analysis of this architecture stability is also given thanks to Lyapunov-based theory. Every contribution is illustrated through simulation results. The last chapter is devoted to the implementation of the proposed control architecture on a group of Khepera III robots.

Key-words : Multi-mobile robots system, Formation navigation, Obstacle avoidance, Control architecture, Hybrid systems, Lyapunov-based stability.

Glossaire

SMR : Système Multi-Robots.

FLM : théorème des Fonctions de Lyapunov Multiple.

TT : Temps de Tenue.

TTM : Temps de Tenue Moyen.

V : Fonction de Lyapunov.

v_i : vitesse linéaire du robot i.

ω_i : vitesse angulaire du robot i.

Introduction générale

« En résumé, cet essaim se reproduit, il est autonome, il est capable d'apprentissage, il possède une intelligence distribuée, collective et il est en mesure d'innover pour résoudre des problèmes. Autrement dit, pour ce qui est des caractéristiques concrètes, il est vivant. Voilà une sale nouvelle ! »
Michael Crichton

« LA PROIE », le livre de Michael Crichton [Crichton 03] est l'histoire de chercheurs qui ont créé un essaim composé de millions de robots de l'ordre du nanomètre. Cet essaim avait pour objectif premier d'être une caméra autonome destinée à l'armée. Seulement, il a échappé au contrôle des chercheurs et devient très dangereux pour les humains. Ce scénario de science-fiction est parmi tant d'autres traduisant les peurs des progrès scientifiques et technologiques. Et si nos robots deviennent si autonomes à en perdre le contrôle ?

Loin de ces questions philosophiques posées à la lecture de tels scénarios, cette histoire rappelle aussi combien l'objectif des chercheurs, et notamment des roboticiens, est d'exploiter et de rendre les robots de plus en plus autonomes. Le but final est de leur confier des tâches de plus en plus complexes, et de s'assurer qu'ils les accomplissent avec succès et sans une intervention humaine. Travail pénible d'ouvriers, assistance à la personne, transport, ou exploration sont autant de domaines où un robot peut rivaliser avec l'homme voire réussir là où ce dernier échoue.

Se reposant sur la notion d'essaim, l'aspect multi-robots du scénario de Crichton ne peut échapper au lecteur. Cependant, la coopération de plusieurs robots, notamment mobiles, pour accomplir une tâche ne relève plus de la science fiction mais est aujourd'hui une réalité. Le contrôle d'un tel système constitue la thématique générale des travaux de recherche abordés dans cette thèse. Les avantages

liés à cette coopération sont multiples. Fiabilité, flexibilité, et rapidité induite par la redondance des entités robotiques sont des caractéristiques suffisamment convaincants pour se tourner vers ces systèmes.

Cependant, les enjeux engendrés sont tout aussi nombreux. En effet, la coopération de plusieurs robots signifie qu'ils doivent partager des ressources communes, communiquer, et, pour chaque entité, tenir compte de l'action des autres afin d'accomplir la sienne.

Plus particulièrement, l'objectif de nos travaux de thèse vise à construire une architecture de contrôle pour un système multi-robots évoluant en formation dans des environnements encombrés (présence d'obstacles). Nous voulons aboutir à une structure de contrôle qui soit ouverte. En d'autres termes, nous voulons que cette architecture puisse être aisément adaptée et modifiée au fur et à mesure que la tâche à réaliser devient plus complexe.

L'approche théorique adoptée consiste à s'éloigner d'un contrôle centralisé au profit d'un contrôle distribué sur les entités robotiques. Aussi, les conceptions cognitives prônant l'utilisation de moyens sophistiqués et de planification de tâches sont abandonnées pour tendre vers un contrôle plus réactif où les capteurs des robots sont leurs seules sources d'informations.

Dans ce contexte, nous proposons d'investiguer les architectures de contrôle comportementales dont le but premier est de briser la complexité de la tâche à exécuter en la décomposant en un ensemble de comportements/contrôleurs élémentaires. Cette approche répondra alors à la contrainte d'ouverture exigée. En effet, pour confier une tâche plus complexe au système multi-robots, il suffit d'ajouter les nouveaux comportements/contrôleurs correspondant aux nouvelles tâches dans la base des comportements/contrôleurs déjà disponibles. A cette approche comportementale vient s'allier la méthode de la structure virtuelle où le groupe de robots mobiles suit un corps virtuel avec une dynamique (vitesse, direction) donnée.

Le contrôle de chaque robot du système se fait alors en commutant entre les contrôleurs tout en respectant les contraintes structurelles des entités (vitesses maximales, accélérations maximales, etc.). Pour étudier et analyser la stabilité de l'architecture de contrôle proposée, les potentialités des systèmes hybrides permettant de contrôler des systèmes continus (contrôleurs) en présence d'évènements discrets (conditions de commutation entre contrôleurs) seront exploitées.

L'organisation générale de ce manuscrit est décomposée en quatre chapitres

Chapitre 1

Il est dédié à un état de l'art sur la robotique mobile. Les différentes architectures de contrôle utilisées en robotique mobile sont classifiées. Une attention particulière est accordée au contrôle distribué et réactif des systèmes multi-robots naviguant en formation. Les travaux proposés dans cette thèse sont positionnés par rapport aux travaux existant.

Chapitre 2

Ce chapitre est consacré à l'architecture de contrôle proposée. Les différents blocs constituant cette dernière sont expliqués de façon détaillée. Les contributions apportées sont illustrées par des simulations. Le mécanisme de coopération entre les entités robotiques y est également exposé.

Chapitre 3

En plus d'une introduction aux systèmes hybrides, ce chapitre comprend l'étude de la stabilité au sens de Lyapunov de l'architecture de contrôle proposée. On y trouve également la prise en compte des contraintes structurelles du robot. Il s'agit de déterminer les limites de l'architecture de contrôle proposée en fonction des contraintes cinématiques des robots utilisés.

Chapitre 4

Il est consacré à la présentation détaillée de la plate-forme expérimentale. Il s'articule autour de la plate forme utilisée, les outils développés, et surtout l'implémentation effective des contributions théoriques sur des robots mobiles de type Khepera III.

Le manuscrit se termine par une conclusion générale reprenant les différentes contributions apportées ainsi que les perspectives envisagées.

Chapitre 1

Etat de l'art et positionnement

Ce chapitre est consacré à un état de l'art sur les systèmes multi-robots mobiles. Nous nous focaliserons principalement sur les types et la classification des architectures de contrôle qui leurs sont dédiées. Compte tenu des objectifs et des travaux réalisés, une attention particulière sera accordée aux différentes approches traitant le problème de la navigation en formation.

1.1 Historique

Le *« robot »* doit son nom à Karel CAPEK. Dérivé de « *robota* », il signifie corvée ou travail forcé en tchèque et dans les langues slaves. Ce mot fut alors utilisé pour la première fois dans la pièce de théâtre RUR [1] en 1921 où le robot cherche à détrôner l'homme en prenant la place des ouvriers dans les usines.

Mais il a fallu attendre le début des années 1950, pour que les premiers robots mobiles équipés de capteurs voient le jour. Ainsi, William Grey Walter mit au point des tortues (appelées les Tortues de Bristol) (cf. Figure 1.1). Chacune a un œil photoélectrique qui lui permet de détecter les sources de lumière. Des amplificateurs transmettent le signal reçu par la plaque photoélectrique aux moteurs qui permettent à la tortue de s'approcher de la source de lumière.

FIGURE 1.1 – Les tortues de Bristol par William Grey Walter.

Le premier robot mobile à comporter un planificateur d'actions (et donc une notion d'intelligence) s'appelle *Shakey* et fut développé au *Stanford Research Institute* SRI à la fin des années 1960. Il fut équipé d'une caméra, d'un télémètre et connecté à deux ordinateurs via une liaison radio (cf. Figure 1.2).

En france, le premier robot mobile est né au LAAS-CNRS en 1977. Il se localisait dans une pièce en fusionnant les données de plusieurs types de capteurs (ultrasons, odométrie et télémètre)(cf. Figure 1.3).

Les avancées technologiques réalisées à la fin du siècle dernier et jusqu'à nos jours ont donné naissance à de nouveaux matériaux, des ordinateurs plus rapides, plus petits et moins chers, etc. Ceci a propulsé la recherche en robotique mobile. Ainsi, les robots se déplacent maintenant sur le sol, dans l'air, sous l'eau et même dans l'espace.

1. pour Rossum's Universal Robot écrite par la même personne Karel CAPEK

FIGURE 1.2 – Le robot Shakey de SRI.

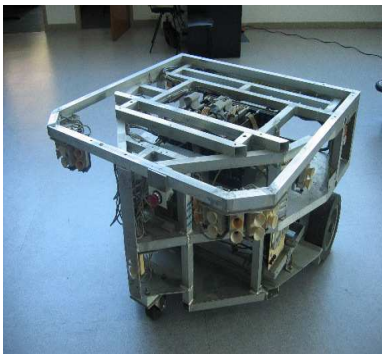


FIGURE 1.3 – Le robot Hilare du LAAS, France.

1.1.1 Pourquoi la robotique mobile ?

Les applications de la robotique étaient en premier lieu principalement industrielles. Elles reposaient sur des bras manipulateurs fixes pour accomplir des tâches de manipulation, soudure, usinage, etc. L'inconvénient majeur de tels robots est d'avoir un domaine atteignable limité par leurs structures.

Doter les robots d'un pouvoir de mobilité ne peut alors être que souhaitable.

En effet, grâce à cette propriété, il devient possible d'augmenter considérablement le domaine atteignable du robot sans augmenter ses dimensions, mais aussi d'élargir l'utilisation de la robotique à d'autres domaines.

Ainsi, l'agriculture a vu se développer des travaux de recherche menant à la commercialisation de tracteurs implantant des fonctionnalités semi-autonomes. On entend par semi-autonome que seul l'ordre d'exécution de la tâche à réaliser est donné par l'homme. Cependant, celle-ci est exécutée de façon complètement autonome. Les véhicules agricoles (tondeuse, tracteurs, moissonneuses-batteuses, etc) AutoGuide de la société AGCO , ainsi que E-drive de Agrocom [Cariou 10] permettent d'avancer parallèlement à la trajectoire précédente et effectuent, en tout autonomie, des corrections de parallélisme sur les lignes droites ou courbes (cf. Figure 1.4).

FIGURE 1.4 – Utilisation de la robotique dans l'agriculture.

Le domaine militaire a aussi bénéficié de cette révolution. Un exemple peut être illustré au travers du robot Packbot [Yamauchi 04]. Ce robot est équipé d'une caméra et peut transporter un bras manipulateur (cf. Figure 1.5). Il est conçu pour détecter et endommager les engins explosifs afin de sauver la vie des soldats.

FIGURE 1.5 – Le robot Packbot, une création de iRobot.

Les robots mobiles ont même précédé l'homme dans l'exploration spatiale. C'est notamment le cas du robot « Sojourner » (cf. Figure 1.6) qui se pose sur le

sol de la planète Mars à bord de la sonde Pathfinder en juillet 1997. Ses capteurs (caméras, capteurs lasers) lui permettent de naviguer en toute autonomie pour exécuter la tâche demandée depuis la Terre.

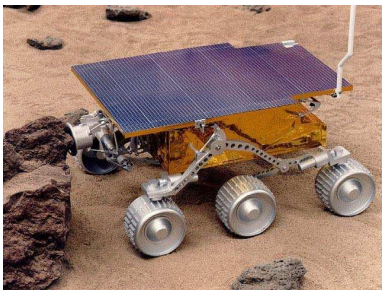


FIGURE 1.6 – Le robot Sojourner utilisé par la NASA sur Mars.

Malgré toutes ces avancées, la mobilité soulève encore de nombreux problèmes. En effet, les robots mobiles sont amenés à évoluer de manière autonome dans des environnements réels qui risquent d'être encombrés ou peu structurés. Plusieurs champs d'investigation pour la recherche dans plusieurs disciplines ont alors vu le jour : en mécanique afin de dimensionner la structure du robot et sa motorisation, en électronique pour choisir les composants adéquats (alimentation des moteurs, cartes mémoire, etc.), en intelligence artificielle et sciences cognitives pour reproduire les connaissances animales sur les robots, en automatique où on retrouve le contrôle régissant le comportement du robot mobile dans son environnement, etc. Le travail réalisé dans cette thèse s'inscrit plutôt dans cette dernière discipline.

Afin d'explorer l'état de l'art et positionner nos contributions dans le cadre des architectures de contrôle existantes, il est d'abord proposé d' exposer certains éléments sur le contrôle de l'entité robotique dont la connaissance préalable est primordiale.

1.2 Le contrôle d'un robot mobile

Le contrôle d'un robot mobile nécessite naturellement la **perception** totale ou partielle de l'environnement où il se trouve, ainsi que sa **localisation** dans cet environnement, afin de **décider** de l'**action** à entreprendre. On retient alors trois éléments principaux indispensables au contrôle du robot mobile : la perception et la localisation, la décision et l'action.

1.2.1 La perception et la localisation

Les êtres humains et animaux disposent généralement de sens (vue, ouïe, toucher, etc) pour percevoir l'environnement dans lequel ils vivent et interagissent. A l'image des vivants, et pour naviguer dans son environnement afin d'accomplir la tâche désirée, le robot doit pouvoir le connaître. Ainsi, pour atteindre un point ou suivre une trajectoire, ou alors éviter un obstacle gênant, les robots mobiles disposent généralement de capteurs permettant de récupérer des stimuli appropriés afin d'alimenter un modèle et produire une représentation de cet environnement. La disposition des capteurs et son influence sur la qualité du contrôle du robot attirent une large communauté inspirée généralement des êtres vivants [Collins 05], [Iida 05]. Ainsi, des capteurs tactiles inspirés des poils des animaux comme les moustaches peuvent s'avérer très efficaces pour les robots mobiles afin de détecter des obstacles ou suivre un mur par exemple [Fend 06]. Par contre, comparer un robot disposant de deux ou quatre capteurs, avec des précisions pouvant être limitées, à un animal ayant des milliers de capteurs risque d'être très loin de la réalité. En conséquence, il faut trouver avec précaution le bon compromis entre les capacités sensorielles des robots et la tâche à accomplir.

On distingue deux types de capteurs en fonction de ce qu'ils mesurent :

1. Les capteurs extéroceptifs : ils décrivent l'interaction avec l'environnement et informent donc sur la localisation relative du robot par rapport à cet environnement. Les caméras, les capteurs GPS « *Global Positioning System* », ultrasons, infrarouges, laser, radars magnétiques etc. font partie de cette catégorie.
2. Les capteurs proprioceptifs : ces capteurs mesurent l'état interne du robot (vitesse, position, couple, charge de la batterie, etc.). On retrouve dans cette catégorie à titre d'exemple, les capteurs d'odométrie permettant de donner une estimation de la position du robot et de son orientation. Ces estimations sont données par rapport à un repère de référence qui est celui du véhicule dans sa configuration initiale sous les hypothèses de roulement sans glissement.

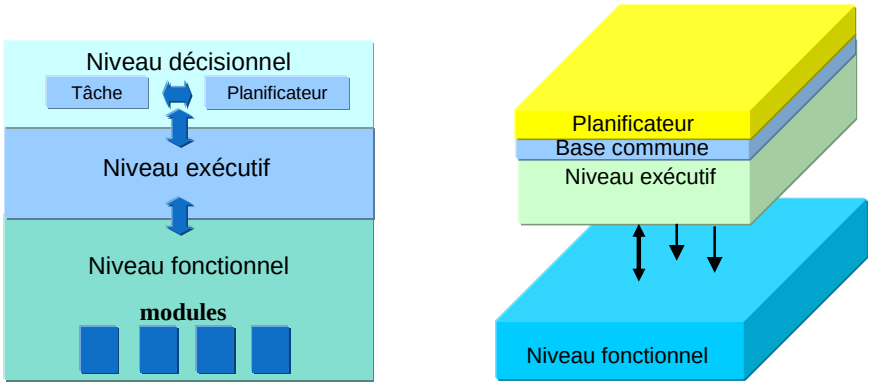
1.2.2 La décision

Dans cette phase, des éléments de décision interviennent pour décider des actions à entreprendre. Celles-ci sont choisies en fonction de la tâche à accomplir et du contexte d'exécution déterminé par les éléments de perception et de localisation. Tout un niveau peut-être attribué à cet élément dans certaines architectures de contrôle étant donné son importance. Ainsi, l'architecture LAAS [2]

2. LAAS Architecture for Autonomous Systems

proposée dans [Alami 98a] contient un niveau décisionnel capable de planifier des actions et superviser leur exécution. Cette exécution relève de la responsabilité d'un niveau appelé niveau fonctionnel où se trouvent des modules qui ont accès aux ressources du robot (capteurs, actionneurs). Ces modules sont sous le contrôle d'un niveau appelé niveau exécutif (cf. Figure 1.7(a)).

Dans l'architecture CLARAty [3] [Estlin 01], on retrouve un niveau décisionnel divisé en deux parties : un planificateur et un niveau exécutif (cf. Figure 1.7(b)). La tâche globale à accomplir est d'abord fournie par le planificateur. Ce dernier brise sa complexité en la décomposant en un ensemble d'objectifs plus élémentaires à accomplir. Ces objectifs sont ensuite coordonnés et exécutés dans un niveau inférieur (le niveau fonctionnel).



(a) Architecture LAAS (b) Architecture CLARAty

FIGURE 1.7 – Les architectures LAAS et CLARAty dans leurs formes simples.

1.2.3 L'action

Les actions décidées dans la phase précédente sont utilisées pour calculer les commandes à appliquer aux robots.

Le robot, immergé dans son environnement, peut-être représenté par un schéma d'asservissement en automatique (cf. Figure 1.8). Ainsi, le bloc *Loi de commande* génère la commande à appliquer aux actionneurs (bloc *Actionneurs*) en fonction des informations fournies par les capteurs, et de la consigne. Cette dernière est la tâche désirée provenant de la phase de décision (tâche planifiée, point à atteindre, etc.).

3. Coupled Layered Architecture for Robotic Autonomy

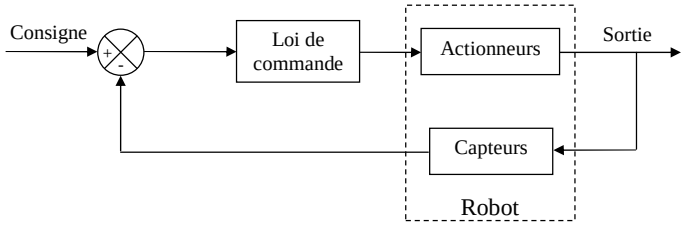


FIGURE 1.8 – Schéma d'un asservissement d'un robot mobile.

Après avoir vu les éléments nécessaires au contrôle d'un robot mobile, à savoir la perception et la localisation, la décision et l'action, nous allons dès à présent s'intéresser à son immersion dans un groupe de robots mobiles dont le but est de coopérer afin de réaliser les tâches demandées.

1.3 Les systèmes multi-robots (SMR)

Ces dernières décennies, l'idée d'utiliser un groupe de robots mobiles dans un seul environnement a attiré un grand nombre de roboticiens, notamment dans le domaine de la commande. L'intérêt porté à ce champ d'investigation est justifié par toutes les tâches possibles et variées qu'offre l'utilisation d'un groupe de robots par rapport à un seul. Les avancées technologiques réalisées dans le domaine de la communication entre systèmes ont grandement contribué aux résultats obtenus. La littérature est si riche qu'il est difficile de rassembler un état de l'art exhaustif sur le contrôle des SMR.

Avant d'aborder certains aspects les plus soulevés dans ce domaine, nous commençons par les nombreuses motivations encourageant à se tourner vers un SMR :

1. comparé à un robot mobile unique, le SMR peut améliorer le temps d'exécution de la tâche, voire la qualité d'exécution de celle-ci. Par exemple, un robot unique peut ne pas avoir la force nécessaire pour pousser un objet lourd. L'utilisation d'un groupe de robots permet alors de diviser cette force sur les différentes entités robotiques qui peuvent détecter les points optimaux où appliquer leurs efforts [Parra-Gonzalez 08], et pousser ensemble dans la direction de mouvement souhaitée [Yamada 01], [Adouane 04]. Un autre exemple est l'utilisation d'un groupe de robots mobiles pour explorer un environnement dans le but d'élaborer une cartographie de cet environnement [Rekleitis 01]. L'idée est que chaque robot en explore une partie différente et établit une carte partielle qui servira à construire la carte globale. La cartographie de l'environnement par un ou plusieurs robots mobiles

est connu dans la littérature sous le nom de *Simultaneous Localization And Mapping* (SLAM).

2. avec un groupe de robots mobiles, on bénéficie d'une distribution des capteurs et des actionneurs sur les entités robotiques, ce qui rend le système plus tolérant à certains défauts de fonctionnement. Par exemple, la panne d'un robot n'entraîne pas l'arrêt systématique de la mission, du fait de la redondance des entités robotiques [Mataric' 95b].
3. l'utilisation d'un groupe de robots peut s'avérer une solution plus simple et moins chère que la conception d'un seul robot dont la structure et le contrôle peuvent vite devenir complexes et encombrants.

Utiliser un groupe de robots mobiles paraît donc une idée séduisante. Cependant, cela entraîne aussi l'apparition de certains aspects liés à l'interaction de plusieurs entités dans le même environnement. La prise en compte de ces aspects est indispensable pour contrôler un SMR. On distingue alors la tâche coopérative à accomplir, les mécanismes de coopération entre entités, et les critères de performances du SMR. Ces points sont sommairement explicités ci-dessous.

1.3.1 Les tâches coopératives en robotique mobile

La tâche coopérative peut être considérée comme l'objectif à atteindre par le système multi-robots. Cet objectif change selon les applications désirées ou le type de robots utilisés.

Pour qu'une tâche soit qualifiée de coopérative, les robots chargés de l'accomplir doivent coordonner leurs actions en interagissant dans le même environnement, et en partageant impérativement des objectifs et des ressources communs : pousser ou porter le même objet, cartographier un environnement, etc. sont des exemples de tâches coopératives dans lesquels chaque robot doit tenir compte des actions des autres en accomplissant la sienne.

L'utilisation d'un SMR doit alors améliorer l'efficacité d'exécution de cette tâche par rapport à la qualité d'exécution offerte par un seul robot. Elle peut aussi rendre la tâche possible, alors que celle-ci est irréalisable par un seul (par exemple, porter ou pousser un objet trop lourd pour un seul robot, explorer une zone trop vaste, etc.). Cela signifie que dans tous les domaines où la robotique mobile peut être utile, la robotique coopérative peut apporter de nouvelles possibilités intéressantes. On peut alors citer l'exploration spatiale [Huntsberger 03], le maintien de formation de véhicules ou d'avions militaires [Balch 99], le transbordement dans les ports et aéroports [Alami 98b], le convoi de véhicules pour le transport urbain de passagers [Bom 05], sauvetage et navigation sous-marins [Bahr 09], etc.

1.3.2 Les mécanismes de coopération

Ces mécanismes représentent la méthodologie utilisée pour la coordination entre les robots afin de présenter ou d'accomplir la tâche coopérative. Il est possible de distinguer trois communautés scientifiques différentes qui abordent la coopération de robots mobiles sans pour autant pouvoir les dissocier complètement [Adouane 05].

Une partie des travaux sur la robotique coopérative cherche à reproduire les activités des animaux, insectes, etc. En effet, l'homme a toujours eu des inspirations biologiques comme dans la figure (1.9) qui montre une analogie entre la navigation en formation d'avions de chasse acrobatiques et un système biologique composé d'oiseaux volant en formation. Ce type d'approches extrait de la nature est venu enrichir le domaine de la robotique coopérative grâce à l'introduction des architectures de contrôles comportementales [Brooks 85], [Arkin 86]. Ces dernières sont décrites par la relation existante entre les trois éléments cités précédemment (cf. Section 1.2) : perception, décision, et action. Le principe comportemental *« Behavior-based »* sert à examiner les caractéristiques et les règles de la vie sociale de certains animaux et insectes (oiseaux, poissons, fourmis, abeilles, etc) afin de les reproduire sur des robots. Ainsi, dans [Mataric' 95a], des actions de fourragement, rassemblement en troupeaux, dispersion, etc, toutes inspirées de sociétés vivantes ont été reproduites sur de véritables robots mobiles.

FIGURE 1.9 – Analogie avions militaires-oiseaux volant en formation.

L'informatique a aussi des apports importants dans le domaine de la robotique coopérative. Cela a commencé avec l'apparition de la notion d'Intelligence Artificielle (IA), puis de l'Intelligence Artificielle Distribuée produite par un ensemble d'agents coopératifs. Les informaticiens favorisent l'aspect conceptuel du problème à celui de la prise en compte des contraintes physiques des agents dans leur environnement. Cependant, dénuder les agents de leurs contraintes physiques en simulation risque de compliquer le passage à l'implantation réelle des solutions

apportées à cause de l'absence d'une modélisation précise de la dynamique des agents.

L'approche robotique est sans doute la plus apte à la mise en œuvre effective de ses résultats, car elle se base sur un modèle (cinématique, géométrique, dynamique) prenant en compte les contraintes réelles des entités robotiques, et de leur environnement. Contrairement aux approches informatiques et celle inspirée des sociétés vivantes, et grâce à l'utilisation d'un modèle, l'approche robotique offre la possibilité d'étudier les performances du contrôle établi pour le système multi-robots selon certains critères de performances (cf. Section 1.3.3). Cependant, cette étude peut se compliquer avec l'augmentation du nombre des entités robotiques dans le SMR. Même si l'architecture de contrôle proposée dans le cadre de cette thèse (cf. Chapitre 2) est basée sur le principe comportemental, le modèle cinématique offert par l'approche robotique constitue la base de l'étude de ses performances.

1.3.3 Les critères de performance du système

La performance et la justification de l'utilisation d'un SMR peuvent être mesurées en se basant sur différentes caractéristiques :

1.3.3.1 Temps d'exécution de la mission

C'est le temps mis par le SMR pour accomplir la tâche demandée (par exemple, poussée d'objets, cartographie, etc.). Comme première intuition, ce temps d'exécution devrait s'améliorer de façon proportionnelle au nombre de robots grâce au partage des tâches (ex : cartographie d'un environnement [Rekleitis 01], [Sty 01]). Cependant, ce n'est pas une règle générale. Dans [Adouane 05] où la tâche de pousser un objet a été étudiée, il est montré que l'ajout de nouveaux robots mobiles n'améliore pas forcément le temps d'exécution. En effet, des situations de conflits entre les robots peuvent apparaître ce qui affecte directement ce temps. Ces conflits sont dus à la surface limitée de poussée que les robots cherchent à se partager. Un nombre optimal de robots N^* est alors nécessaire. Avoir moins ou plus de N^* robots pourrait nuire au temps d'exécution. Ce dernier a été mesuré dans la figure 1.10 par le nombre d'itérations nécessaires pour que la tâche soit accomplie. N_c désigne le nombre minimal de robots qui doivent pousser la boite au même temps.

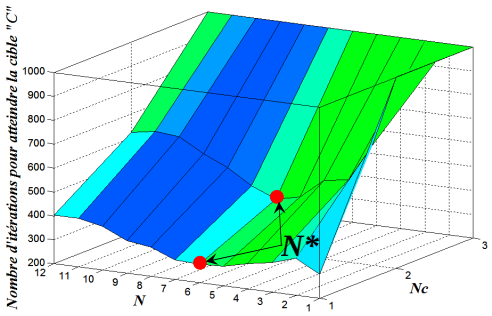


FIGURE 1.10 – Évolution du nombre d'itérations nécessaires pour réaliser la tâche coopérative de poussée d'objets en fonction du nombre de robots utilisés.

1.3.3.2 Stabilité et fiabilité

La stabilité est sans doute une condition impérative pour le bon fonctionnement d'un système automatique. Une architecture de contrôle stable signifie qu'elle assure au SMR d'atteindre la consigne désirée (point à atteindre, trajectoire à suivre, etc.) et donc d'accomplir sa tâche. Il est aussi souhaitable d'étudier la sensibilité du SMR à la défaillance d'une entité robotique et sa capacité à finir la mission même avec la perte totale (ex : arrêt du à une panne) ou partielle (ex : défaillance de certains capteurs) d'un élément de la flottille. Dans l'architecture ALLIANCE [Parker 00], chaque robot r_i calcule un coefficient de motivation globale propre à chaque tâche a_{ij}. Ce coefficient est calculé comme suit

$$\begin{cases} m_{ij}(0) = 0 & et \\ m_{ij}(t) = [m_{ij}(t-1) + impatience_{ij}(t)] \times retour\_capteur_{ij}(t) \times \\ suppression\_activite_{ij}(t) \times reinitialisation\_impatience_{ij}(t) \times accord_{ij}(t) \end{cases}$$

La fonction $impatience_{ij}(t)$ décrit le taux d'impatience du robot. Les autres variables sont de type booléen telles que $retour\_capteur_{ij}(t)$ décrit si les conditions nécessaires pour la réalisation d'une tâche sont réunies ou non ;

$suppression\_activite_{ij}(t)$ indique s'il existe une tâche a_{ik}, telle que $k \neq j$, qui nécessite d'être accomplie à l'instant t ; $reinitialisation\_impatience_{ij}(t)$ est mise à 0 si un autre robot réalise efficacement la tâche que r_i s'impatiente d'accomplir (et à 1 sinon). $accord_{ij}(t)$ détermine si le robot est d'accord pour abandonner la tâche qu'il réalise au profit d'un autre. Ainsi, en mesurant la *motivation* du robot, cette fonction offre à chaque entité la possibilité de vérifier si les autres accomplissent bien leurs tâches. Par exemple si un robot r_l est en panne ou accomplit mal sa tâche, le robot r_i s'impatiente de prendre le relais ce qui donne l'assurance que la tâche sera accomplie.

1.3.3.3 Complexité du calcul et de la communication nécessaires

Même si les avancées technologiques ont donné naissance à des machines avec de puissantes capacités de calcul et des moyens de communication évolués, la gestion d'une flottille constituée d'un grand nombre de robots et la communication entre entités peuvent devenir complexes, voire irréalisables. C'est le cas notamment où une seule unité de calcul doit gérer en temps réel le contrôle de toutes les entités de cette flottille, surtout si elles se déplacent avec des vitesses importantes. En effet, un calcul et une communication coûteux en temps entraînent la diminution de la fréquence de calcul et d'envoi des commandes aux entités. Celles-ci risquent alors de se déplacer en boucle ouverte si le temps entre deux commandes reçues est relativement long (risque de collision, dépassement des consignes, etc.).

La figure 1.11 montre la diminution de la qualité de service QoS du réseau de communication utilisé en fonction du nombre des robots. Cette grandeur mesure la capacité du réseau à véhiculer les données dans de bonnes conditions en termes de retard de transmission, perte de paquets, et tâche à réaliser.

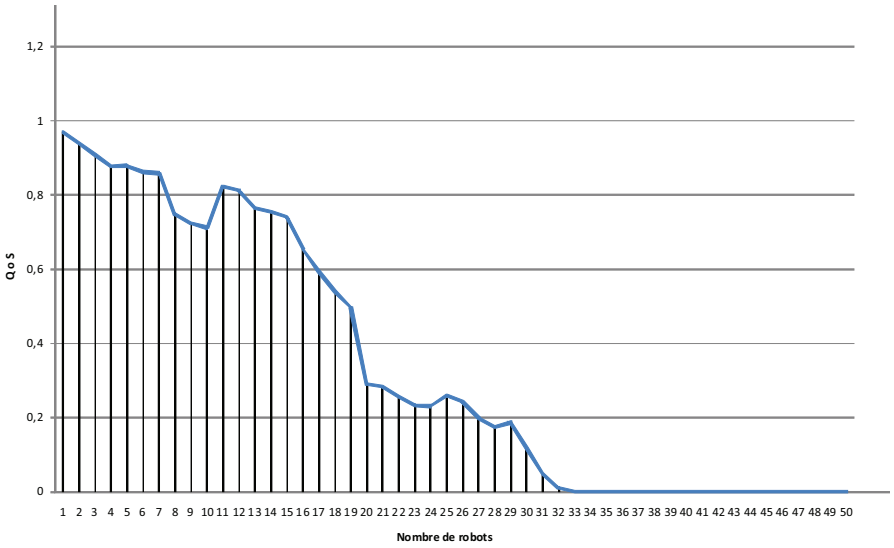


FIGURE 1.11 – Influence du nombre des entités robotiques du SMR sur la qualité de service du réseau.

On remarque que l'augmentation du nombre des robots utilisés affecte sérieusement cette qualité de service.

Dans [Mechraoui 10], l'influence de la communication sur la performance du système est illustrée. Pour cela, la tâche d'une unité de calcul commandant plusieurs entités robotiques pour rejoindre leurs cibles respectives est considérée. La

figure 1.12 montre la trajectoire de l'une des entités devant se déplacer de sa position initiale $(0, 0)$ à sa cible située à $(1, 1)$. On remarque que l'augmentation de la charge du réseau liée à l'augmentation du nombre des robots du SMR affecte directement le comportement de chaque entité. En effet, un réseau trop chargé engendre des retards de communication. L'entité ne reçoit pas ses commandes à temps et échoue dans sa tâche.

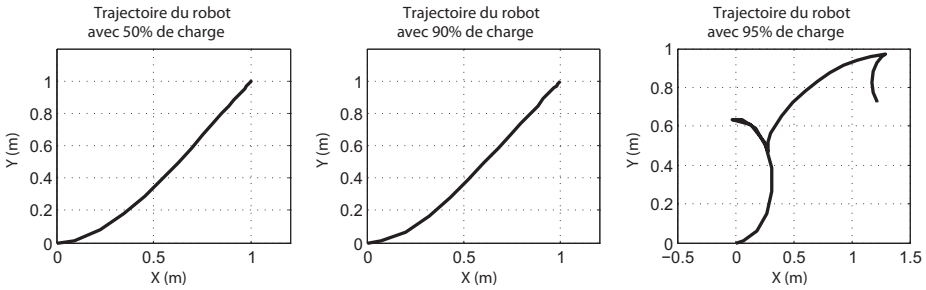


FIGURE 1.12 – Influence de la charge du réseau provoquée par le nombre des robots utilisés sur la performance des entités robotiques.

1.3.3.4 Flexibilité

Il s'agit de la capacité du SMR à accepter l'ajout de nouvelles entités, en cours de la réalisation de la tâche, si le besoin s'en fait sentir. Il faut alors étudier l'influence des nouveaux robots sur les éléments de la flottille déjà opérationnels.

Il est important de noter que ces paramètres sont fortement corrélés. Ainsi, allouer des tâches de façon à minimiser la complexité du calcul (ex : dans le cas d'une unité centrale assurant le calcul, programmer des tâches plus souvent en série qu'en parallèle) signifie l'augmentation du temps d'exécution de la mission et vice versa. Ajouter de nouveaux robots est souvent synonyme d'augmentation de complexité de calcul ce qui n'est pas sans incidence sur le temps alors nécessaire. Il s'agit alors souvent de chercher un compromis qui optimise le plus de critères en cherchant un nombre optimal de robots avec des vitesses maximales prenant en compte les puissances des unités de calcul [Brian 02].

Nous avons vu dans cette section que l'utilisation de plusieurs robots, plutôt que d'un seul, présente de nombreux avantages. Cependant, afin d'accomplir la tâche désirée de façon efficace, il est indéniable d'assurer une bonne coordination entre ces robots. Il faut donc prendre en considération qu'une mauvaise gestion des ressources communes ou une mauvaise coordination des agents a des répercussions directes sur la productivité et les résultats accomplis (tout comme dans

nos sociétés humaines). Afin d'éviter des situations de conflit et de blocage qui peuvent en découler, il est important de mettre en place une stratégie pour allouer les actions aux robots. Cette stratégie peut être centralisée au niveau d'un seul superviseur, ou distribuée sur la totalité ou une partie des entités. Les activités humaines et animales restent parmi les meilleures sources d'inspiration pour établir ces stratégies.

A titre d'exemple, des activités humaines comme les enchères, les négociations, et même des sentiments ont été reproduits sur des robots pour une meilleure allocation des tâches. C'est le cas de l'architecture ALLIANCE que nous avons abordée dans le critère de fiabilité où l'impatience et la motivation sont reproduites sur des robots. Cette architecture a été testée sur une tâche de fourragement [Parker 98], et de poussée d'objet [Parker 94].

D'autres méthodes reprennent un fonctionnement plus collectif de la société humaine habituée à la notion d'hiérarchie : un *courtier* central décide de la répartition et l'attribution des tâches et chaque agent reçoit alors la mission d'accomplir une partie de la tâche globale suivant son aptitude à réaliser cette tâche [Martin 99], [Sycara 96]. MURDOCH [Brian 02], est un processus distribué (cf. Section 1.4.2) sur les robots et utilise des méthodes d'enchère afin de gérer la compétitivité entre les robots.

Pour explorer plus en détail ces coordinations, des architectures de contrôle ont fait l'objet de plusieurs travaux recherches. En effet, le contrôle des entités robotiques constitue le noyau des SMR et sa qualité détermine directement sa performance. Ainsi, la section suivante est dédiée à un état de l'art sur ces architectures de contrôle et leur catégorisation.

1.4 Classification des architectures de contrôle : centralisées, distribuées

Depuis la fin des années 80, la recherche dans le cadre de la robotique coopérative est devenu un centre d'intérêt majeur où plusieurs projets et tâches génériques ont vu le jour (cf. Section 1.3.1). Ces tâches offrent aussi la possibilité d'évaluer les architectures de contrôle proposées dans la littérature, et définir leurs avantages et inconvénients.

Une distinction principale entre les architectures de contrôle porte sur le choix de centraliser le contrôle du système multi-robots ou de le distribuer sur les entités robotiques. On trouve alors des architectures de contrôle centralisées et distribuées.

1.4.1 Architecture de contrôle centralisée

Comme son nom l'indique, le contrôle du SMR est délocalisé par rapport à la structure physique des robots et se trouve au niveau d'une unité centrale (superviseur dans [Jones 01] ou planificateur central dans [Noreils 93]) qui gère et garantit l'exécution de la tâche. L'unité contient les parties sensorielles (capteurs) afin de collecter les informations de l'environnement, et calcule les consignes de commande (positions, vitesses) qu'elle envoie aux entités robotiques. Elle est aussi responsable de prendre les décisions pour la réalisation de la tâche globale et les communiquer aux robots. Elle doit donc avoir une puissante capacité calculatoire pour satisfaire à toutes ces exigences.

Comme dans [Khoshnevis 98], ce type d'architecture de contrôle a été essentiellement motivé par deux facteurs :

- l'ambition d'alléger la structure physique des robots : en effet, les robots utilisés n'ont besoin ni de capteurs embarqués ni d'une puissante unité de calcul. Ceci réduit considérablement leur coût.
- la concentration des capteurs au niveau d'une unité centrale offre une meilleure connaissance de l'environnement global, ce qui peut assurer une meilleure prise de décision par rapport à des robots se basant sur des informations locales fournies par des capteurs embarqués.

On peut faire l'analogie entre les architectures de contrôle centralisées et la notion d'hiérarchie chez les groupements sociaux, et notamment humaines dans la manière de coordonner les actions entre

- un superviseur central ayant une vision globale de l'environnement et de la tâche à réaliser,
- des robots disposant uniquement des informations utiles à l'exécution de la partie de la mission qui leur est confiée par le superviseur.

1.4.2 Architecture de contrôle distribuée

Par opposition aux architectures de contrôle centralisées, les ressources (capteurs, unités de calcul, etc.) sont dans ce cas distribuées sur tous les éléments du SMR. Chaque robot n'utilise alors que ses propres capteurs et sa propre unité de calcul et de traitement. Il doit aussi pouvoir communiquer et partager des informations avec les autres robots afin de les informer et de s'informer sur la progression de la mission.

La distribution des ressources peut être très avantageuse si la mission du SMR devient trop complexe et fastidieuse. En effet, la complication de la tâche du SMR entraîne naturellement la complication du contrôle nécessaire si bien qu'il

devient difficile voire impossible sa réalisation par un superviseur [Lerman 01]. Par exemple, le simple ajout de nouveaux robots signifie de nouveaux agents (avec tous les calculs relatifs) que le superviseur doit gérer.

Ce type d'architectures de contrôle est donc venu dans l'ambition de pallier aux limitations des architectures de contrôle centralisées. Néanmoins, elles partagent la même source d'inspiration qui reste les sociétés vivantes. Ainsi, le projet CEBOT (CEllular roBOTics System) [Fukuda 89] est un système distribué inspiré de l'organisation des cellules d'un organisme biologique. Les entités robotiques sont alors assimilées à des cellules couplées entre elles. Ces robots cellules sont continuellement reconfigurable afin de répondre aux éxigences de l'environnement. Ce système est cité à titre d'example, beaucoup d'autres systèmes s'inscrivant dans le cadre des systèmes distribués peuvent être trouvés dans la littérature [Jin 94], [Parker 98], [Yamaguchi 01], [Zhiqiang 06].

1.4.3 Architecture de contrôle centralisée ou distribuée ?

Chaque type d'architectures de contrôle présente ses avantages et ses inconvénients. Il est alors difficile voire impossible de privilégier un type plutôt qu'un autre sans connaître la nature de la mission confiée au SMR ni le contexte de sa réalisation.

En effet, l'unité centrale du contrôle centralisée dispose d'informations globales de l'environnement ce qui est un atout non négligeable d'aide à la décision et de coordination d'actions entre les éléments du SMR. Cependant, cette unité constitue aussi la limitation de ce type d'architectures du fait qu'elle doit gérer le contrôle et la communication de toutes les entités robotiques. Si puissante soit-elle, elle n'est pas à l'abri des goulots d'étranglement dans le cas où la mission demandée requiert la coopération d'un nombre important de robots mobiles. En effet, contrôler et communiquer avec tous les robots mobiles en temps réel deviendrait impossible car le temps de calcul risque de s'allonger laissant les robots en boucle ouverte. De plus, le SMR dépend entièrement de cette unité, ce qui signifie qu'un défaut du superviseur central implique l'arrêt complet du système.

Les architectures de contrôle distribuées permettent de tirer profit de la distribution des ressources : chaque robot se contente de prendre une décision relative à lui, en fonction des informations locales de l'environnement délivrées par ses propres capteurs et de sa communication avec ses voisins. Cette distribution allège la tâche des unités de calcul. De plus, les défauts de fonctionnement sont mieux gérés et la panne d'un robot ne signifie pas l'arrêt complet du SMR. En revanche, l'absence d'un superviseur global et d'une information globale sur l'environnement rend difficile la coordination des robots et l'optimisation de l'exécution de la mission.

Le tableau 1.1 résume les différences observées principalement au niveau des trois modules nécessaires à la commande des robots mobiles, à savoir : la perception et la localisation, la décision, et l'action (cf. Section 1.2).

En pratique, beaucoup de SMR n'ont pas un contrôle strictement centralisé ou strictement distribué et utilisent des architectures de contrôle hybrides centralisées/distribuées pour bénéficier des avantages des deux approches. Par exemple, dans [Beard 01], [Das 02], [Feddema 02], [Stilwell 05], des robots pourtant autonomes, sont soumis à un planificateur central.

Modules	Architecture centralisée	Architecture distribuée
Perception et localisation	L'unité centrale utilise des capteurs centralisés et calcule la localisation absolue dans l'environnement de chaque robot. Les robots ne connaissent pas leur environnement.	Les robots utilisent leurs capteurs embarqués et se localisent de façon relative par rapport à leurs consignes.
Décision	L'unité centrale décide, calcule et génère directement des variables de commande (vitesses) qu'elle envoie aux robots.	Les robots génèrent leurs propres consignes de type chemin, trajectoire, points de passages, etc.
Action	Les entités robotiques appliquent directement les variables de commande calculées par l'unité centrale à leurs moteurs.	Les robots assurent le respect des consignes qu'ils ont générées à travers des lois de commande avec un asservissement en boucle fermée sur ces consignes.

TABLE 1.1 – Architecture centralisée versus distribuée.

La classification des architectures de contrôle peut alors être illustrée comme dans la figure 1.13. Cette figure sera mise à jour avec les nouvelles classifications qu'on verra dans les prochains paragraphes, afin de résumer les travaux réalisés dans ce domaine et pouvoir positionner les contributions apportées dans cette thèse.

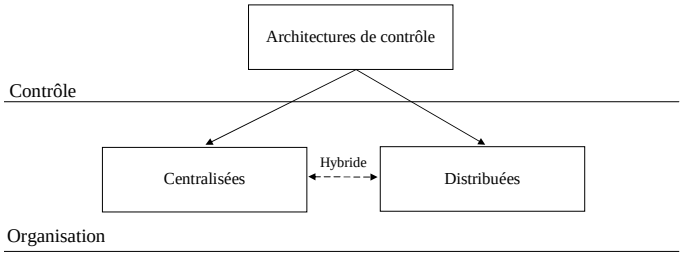


FIGURE 1.13 – Classification des architectures de contrôle selon leur organisation centralisée ou distribuée.

1.5 Classification des architecture de contrôle distribuées : cognitives, réactives

On a vu les deux approches principales utilisées pour la coordination et le contrôle d'un SMR, en l'occurrence les architectures de contrôle centralisées et distribuées. Nous verrons dans les chapitres suivants que nous cherchons à rendre l'architecture de contrôle proposée la plus distribuée possible. En effet, nous souhaitons que le SMR soit flexible au sens qu'on doit toujours être en mesure d'ajouter autant de nouvelles entités qu'on le souhaite (cf. Section 1.3.3), ce qu'un superviseur central ne tolère pas forcément (cf. Section 1.4.3). Pour cette raison, nous allons nous focaliser sur les architectures de contrôle distribuées.

Ces dernières sont classées à leur tour en fonction de la relation entre la perception, la décision (planification), et l'action à l'intérieur de l'architecture de contrôle. On trouve ainsi principalement deux catégories d'architectures : cognitives et réactives. La différence principale entre les deux approches se trouve dans la manière dont le SMR gère une situation imprévisible : dans les approches cognitives, les actions des robots sont exclusivement le résultat d'un raisonnement « intelligent » issu de couches décisionnelles et exécutives supérieures connaissant cette situation, tandis que dans les approches réactives, les actionneurs des robots réagissent directement aux stimuli de l'environnement (informations relevées par les capteurs) en temps réel. Des architectures de contrôle hybrides réactives/cognitives constituent là aussi une troisième solution dont l'objectif est de rassembler les bienfaits de ces deux types que nous allons décrire plus en détail dans les paragraphes suivants.

1.5.1 Architectures de contrôle cognitives

Cette approche repose sur une décomposition fonctionnelle des différentes tâches à exécuter. Cette décomposition suit la structure représentée dans la fi-

gure 1.14. Dans ce type d'architectures, il est impératif d'avoir une phase de planification basée sur un modèle de l'environnement. Ce modèle, généralement stocké dans la mémoire du robot, est alimenté à l'aide de capteurs et permet de planifier la tâche désirée avant d'appliquer une quelconque action à ses moteurs [Grassi Junior 06]. Pour satisfaire à ces exigences, des moyens sophistiqués doivent être intégrés dans le robot utilisé. On trouve alors au moins :

- une mémoire pour stocker la carte de l'environnement. Le robot doit aussi être en mesure de mettre à jour ces informations. A titre d'exemple, l'ajout d'un élément dans l'environnement (un objet encombrant, un autre robot, etc) implique la mise à jour de la carte stockée dans la mémoire du robot afin de prendre en compte cet élément dans la planification de la tâche.
- une unité de calcul assez puissante pour répondre à la planification et/ou replanification des trajectoires.

Pour le cas des SMR, une modélisation de l'environnement peut exister mais n'est pas impératif [Iocchi 01]. Dans ce cas, d'autres contraintes sont imposées au comportement global du SMR. Ces contraintes font que les entités robotiques doivent continuellement traiter un grand flux d'informations échangées entre elles, via des protocoles de communication évolués [Brian 02].

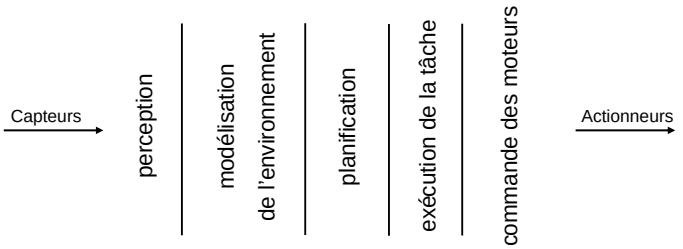


FIGURE 1.14 – Décomposition classique des architectures de contrôle cognitives.

1.5.2 Architectures de contrôle réactives

Ce type d'architectures de contrôle est venu s'opposer aux architectures de contrôle cognitives dans le but d'obtenir un robot ou un SMR qui répond en temps réel aux changements imprévisibles de l'environnement sans recourir à une planification préalable. Les concepts de modélisation de l'environnement et de raisonnement utilisés dans les architectures de contrôle cognitives sont abandonnés au profit d'un contrôle complètement distribué sur les entités robotiques et une stratégie de stimuli-réponses (perception-action). Cela signifie qu'il y a un lien direct entre les informations fournies par les capteurs et les actions appliquées aux moteurs. Pour un SMR, chacun doit répondre individuellement aux changements

qui peuvent l'affecter sans passer par un niveau décisionnel supérieur engageant une réorganisation complète du système.

Les travaux menés dans le contrôle réactif des robots mobiles doivent une grande partie de leur développement aux sociétés vivantes comme celles d'insectes. Ces derniers n'ont pas une grande capacité de raisonnement. Cependant, ils arrivent à reproduire des comportements complexes à partir de la coordination d'un ensemble de comportements élémentaires de base [Anderson 90].

Brooks [Brooks 85] est certainement l'un des premiers à avoir introduit cette notion de réactivité au contrôle des robots mobiles. Contrairement aux architectures de contrôle cognitives décomposées classiquement en blocs fonctionnels verticaux (cf. Figure 1.14), il propose de définir l'architecture en blocs horizontaux (cf. Figure 1.15) appelés *comportements*. Chaque comportement traduit une tâche élémentaire qui constitue une partie de la tâche globale désirée du robot. Cette approche ascendante, consistant à assembler plusieurs comportements élémentaires afin de réaliser des tâches plus complexes, permet de rendre le SMR apte à accomplir de nouvelles missions en incrémentant simplement le bloc de comportements existant avec les nouveaux correspondant aux nouvelles tâches désirées.

Cette structure permettant une mise à jour incrémentale est souhaitable devant la complexité grandissante des missions accomplies par les robots. En effet, en revoyant les compétences exigées du SMR à la hausse, il ne sera plus nécessaire de reconstruire entièrement une nouvelle architecture à chaque fois que la tâche à réaliser devient plus complexe. Il suffira alors d'ajouter les nouveaux comportements désirés.

Comme l'un des objectifs de nos travaux est d'avoir une architecture de contrôle ouverte, c'est à dire que l'on peut mettre à jour en incrémentant de nouveaux comportements, il est proposé d'explorer plus loin les travaux sur les approches comportementales afin d'en tirer profit (cf. Section 1.6.2).

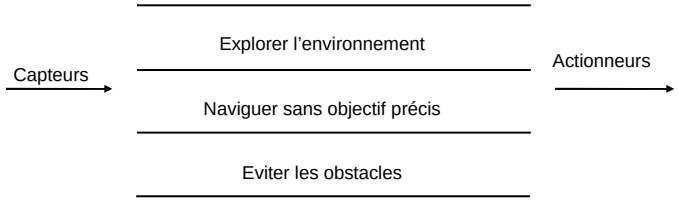


FIGURE 1.15 – Décomposition horizontale des architectures de contrôle.

Une autre illustration élégante du contrôle réactif est l'enveloppe ZVD (Zone Virtuelle Déformable entourant le robot) proposée par Zapata [Zapata 94] et

permettant d'éviter des obstacles. On retrouve le lien direct entre les capteurs et les actionneurs dans la mesure où les commandes sont générées en cherchant à minimiser les déformations de la ZVD provoquées par les obstacles. Cette méthode sera exposée plus en détail dans la section 2.3.2 (cf. page 72).

1.5.3 Approches cognitive versus réactive

Comme nous pouvons le constater, chaque école présente à travers ses spécificités des caractéristiques intéressantes. L'avantage principal de l'approche cognitive est qu'elle peut intégrer plusieurs principes de haut niveau : raisonnement, communication de haut niveau, etc. et elle s'appuie sur une modélisation de l'environnement lui permettant de prendre des décisions optimales avant d'entreprendre une quelconque action. Cependant, ces avantages sont rattrapés par certains inconvénients :

- les principes de haut niveau nécessitent des temps de calculs importants et des moyens à coûts élevés,
- l'environnement d'évolution doit être modélisable. En effet, un environnement fortement dynamique nécessite des mises à jour fréquentes ce qui peut devenir rapidement ingérable.

Il reste que les développements réalisés sur le plan technologique ont engendré des avancées intéressantes pour les approches cognitives, ce qui aide à surmonter certains inconvénients liés au temps de calcul grâce à la puissance actuelle des machines. Cependant, les environnements dynamiques sont synonymes de mise à jour régulière de leurs cartes et de replanification de la nouvelle tâche du robot dans le nouvel environnement. La replanification de chemins des robots dans les problèmes de navigation est un bon exemple [Fraichard 99]. Dans les environnements fortement dynamiques, et même si certains travaux réclament avoir nettement alléger le temps de calcul [Van den Berg 05], ce dernier reste directement lié à la fréquence de replanification.

Les architectures de contrôle réactives permettent d'avoir une réponse plus rapide grâce à un lien direct capteurs-actionneurs. Disposer des comportements élémentaires dans le cas des approches comportementales permet de les tester individuellement jusqu'à ce qu'ils soient adaptés à leurs tâches (trouver les meilleurs gains de commande, vérifier leur stabilité, etc.). La mise à jour se fait alors simplement en les ajoutant aux comportements déjà existants.

En contrepartie, l'absence d'une représentation explicite de l'environnement et de raisonnement de haut niveau limitent leurs applications dans des cas complexes.

La figure 1.16 [Arkin 98], [Adouane 05] compare les caractéristiques des deux types d'architectures de contrôle. La figure 1.13 peut alors être mise à jour en rajoutant un niveau déterminant la classification des architectures de contrôle selon leur degré d'intelligence (cf. Figure 1.17).

COGNITIVE	**REACTIVE**
Symbolique	**Réflexive**
Vitesse de la réponse	
Capacités prédictives	
Dépendance à la précision, modèle du monde complet	
Modélisation explicite de l'environnement	Absence d'une modélisation explicite
Vitesse de réponse relativement lente dans certains environnements	Fonctionne en temps réel
Haut niveau d'intelligence	Peu ou pas d'intelligence
Planification sophistiquée et préalable de la tâche	Fonctionnement en stimulus-réponse
Peut tenir compte de son passé	Pas de mémoire de son historique

FIGURE 1.16 – Contrôle cognitif versus réactif.

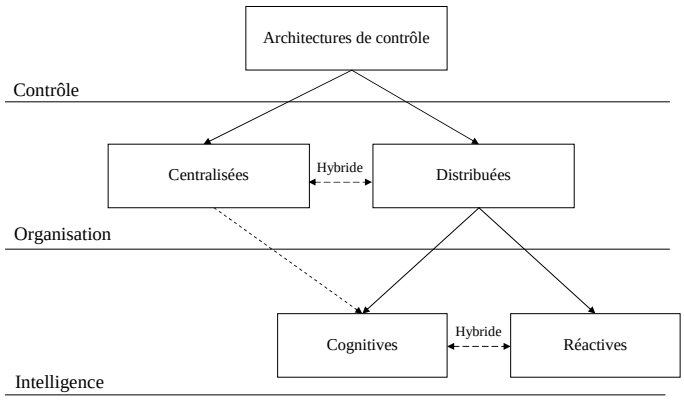


FIGURE 1.17 – Classification des architectures de contrôle selon leur degré d'intelligence.

Les avantages des unes et des autres ont poussé certains roboticiens à combiner les deux approches : une bonne illustration des approches hybrides peut-être montrée par les travaux réalisés dans [Ranganathan 03] pour un cas mono-robot, où une architecture de contrôle combinant navigation réactive et planification de trajectoire (cognitive) est présentée. Le but est de diminuer les inconvénients des deux approches. Ainsi, l'architecture tend à utiliser la planification de trajectoire le moins de temps possible, uniquement quand le robot est incapable d'atteindre sa destination en un temps déterminé (cf. Figure 1.18). Le fonctionnement de cette architecture se fait selon 3 modes :

1. d'abord le robot fonctionne avec une approche réactive : il se déplace vers un objectif en évitant les obstacles qu'il détecte par ses capteurs,
2. si le robot n'arrive pas à son objectif au bout d'un certain temps PERSISTENCE, cela veut dire qu'il ne progresse pas. Le planificateur intervient alors et met en place une trajectoire à suivre et dont le bout est un point de transition à atteindre (une sorte d'objectif intermédiaire). Le déplacement vers cet objectif virtuel se fait en mode réactif (mode 1),
3. si les deux modes précédents échouent dans l'acheminement du robot vers son objectif : un planificateur de trajectoire planifie alors au robot le chemin à suivre jusqu'à l'objectif final.

Cette méthode a donné de bons résultats : elle engage moins de moyens que l'utilisation d'une approche cognitive pure, et réussit là où l'approche réactive échoue. Si on reprend la figure (1.18), une navigation réactive pure laisse le robot bloqué dans le canyon. L'ouverture est assez limitée pour être correctement détectée par ses capteurs. L'utilisation d'une navigation purement cognitive est une solution plausible, cependant elle sera inutile en dehors du canyon vu qu'une simple navigation réactive suffirait pour atteindre l'objectif final. Une planification est alors entreprise uniquement pour faire sortir le robot du canyon, avant que la navigation réactive ne prenne le relais.

Beaucoup d'autres architectures ont été développées dans ce sens [Kim 03b], [Paquier 03]. Néanmoins, il est aussi utile de bien étudier le problème de savoir à quel moment il faut engager un contrôle réactif ou un contrôle cognitif [Firby 87], [Arkin 89], [Gat 97].

D'après ce qui précède, on a vu que les architectures de contrôle peuvent être centralisées, ou distribuées. En fonction du cahier des charges que nous nous sommes imposé (cf. Introduction générale), nous nous intéressons à la tâche de la navigation en formation de robots mobiles. Nous souhaitons une architecture de contrôle qui soit la plus distribuée possible. Aussi, le maintien de formation se fera de façon réactive. En effet, l'architecture proposée permettra d'accomplir cette tâche sans recourir à un contrôle cognitif gourmand en calcul. Ces choix sont justifiés plus en détail dans la section 1.7.

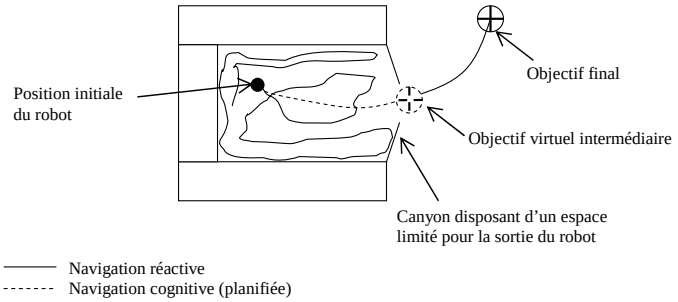


FIGURE 1.18 – Utilisation d'un contrôle hybride cognitif/réactif pour faire sortir le robot du canyon.

Dans ce qui suit, nous allons alors compléter notre état de l'art en nous intéressant plus précisément aux différentes stratégies existantes dans la littérature pour accomplir cette tâche de maintien de formation.

1.6 Tâche de navigation en formation : stratégies de contrôle

Avec un champ d'application large, la tâche de la navigation en formation attire depuis quelques années de plus en plus de roboticiens et fait l'objet de cette thèse.

Est appelée navigation en formation, la tâche accomplie par un ensemble de robots mobiles consistant à se déplacer en un seul groupe tout en maintenant une position et une orientation relative entre les entités robotiques. Ces positions et orientations définissent la formation géométrique souhaitée (voir figure 1.19 par exemple) selon l'application désirée. Nous nous intéressons ici aux différentes approches traitant le problème de la navigation en formation pour des robots mobiles à roues qui s'inscrivent dans le cadre réactif. Les travaux réalisés dans le domaine convergent principalement dans trois approches différentes : approche hiérarchique, approche comportementale, et méthode de la structure virtuelle.

1.6.1 Approche hiérarchique

Dans cette approche, un ou plusieurs robots de la formation sont considérés comme des robots guides qui mènent le reste des robots appelés suiveurs. Généralement, le guide évolue sur une trajectoire prédéfinie tandis que les autres robots suiveurs le suivent avec une certaine posture (distance et orientation) relative (cf.

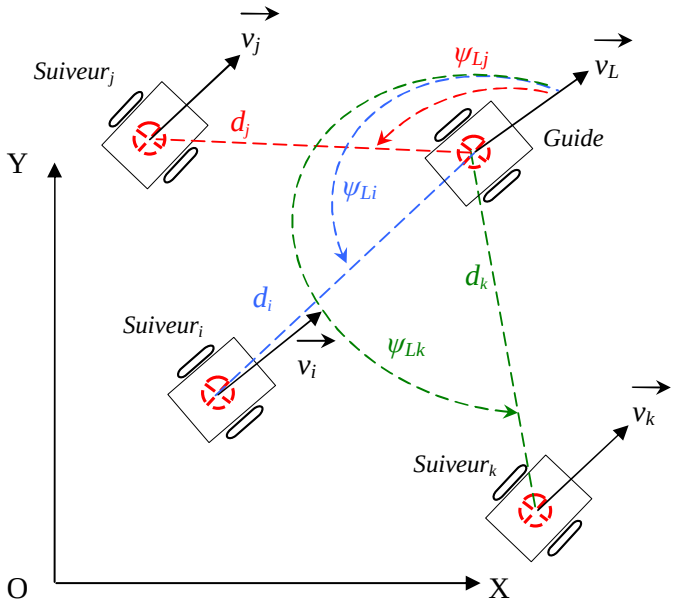


FIGURE 1.19 – Formation composée d'un guide L et de trois suiveurs.

Figure 1.19) [Das 02], [Tanner 04], [Léchevin 06], [Gustavi 08]. Ainsi, quand le guide suit sa trajectoire, les suiveurs utilisent sa position et son orientation pour calculer leurs propres consignes.

Les différentes techniques proposées dans cette approche pour des robots mobiles à roues, et qu'on trouve dans [Fierro 01], [Das 02], [Spletzer 01], sont :

1.6.1.1 Contrôle par séparation-orientation (CSO)

Dans ce cas, le robot suiveur j suit le guide i avec une distance (séparation) d_{ij} et une orientation ψ_{ij} désirées (cf. Figure 1.20).

1.6.1.2 Contrôle par séparation-séparation (CSS)

Le robot suiveur k suit deux robots guides i et j (ou plus), chacun avec des distances désirées d_{ik} et d_{jk} respectivement (cf. Figure 1.21).

1.6.1.3 Contrôle par séparation-séparation de l'obstacle (CSSO)

Le contrôle d'un suiveur j est défini en suivant le guide i avec une distance désirée d_{ij}. De plus, un robot virtuel évoluant sur la bordure de l'obstacle est

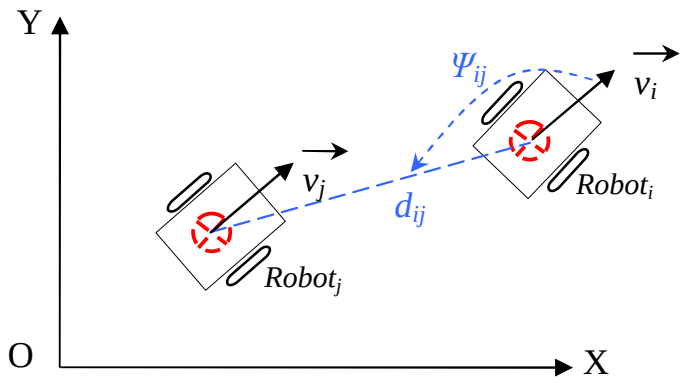


FIGURE 1.20 – Contrôle par séparation-orientation (CSO).

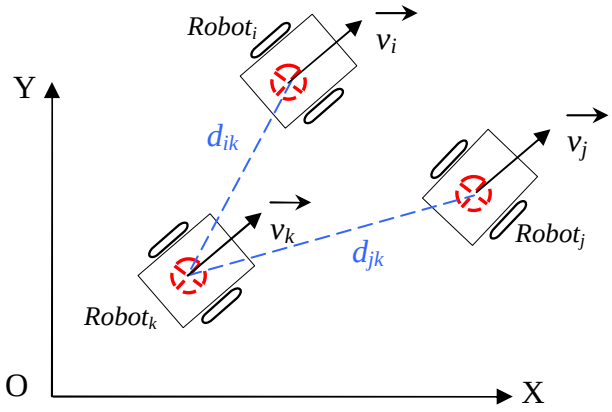


FIGURE 1.21 – Contrôle par séparation-séparation (CSS).

défini. Le suiveur est alors contrôlé à maintenir une distance d_{ij} du guide et une distance d_{ok} du robot virtuel. Il s'agit d'un cas particulier de CSS, où le deuxième robot guide est un robot virtuel (cf. Figure 1.22).

1.6.1.4 Contrôle par dilatation

Si on prend l'exemple de trois robots (cf. Figure 1.23), le robot j suit le robot i avec une distance et une orientation désirées d_{ij} et ψ_{ij} respectivement. Le robot k suit également le robot i avec une distance d_{ik} et une orientation ψ_{ik} mais suit aussi le robot j avec une distance d_{jk} et une orientation ψ_{jk}. Ceci signifie que i est un guide par rapport à j et k, j est un suiveur par rapport à i et un guide par rapport à k. Un coefficient de dilatation/contraction ρ permet de changer les

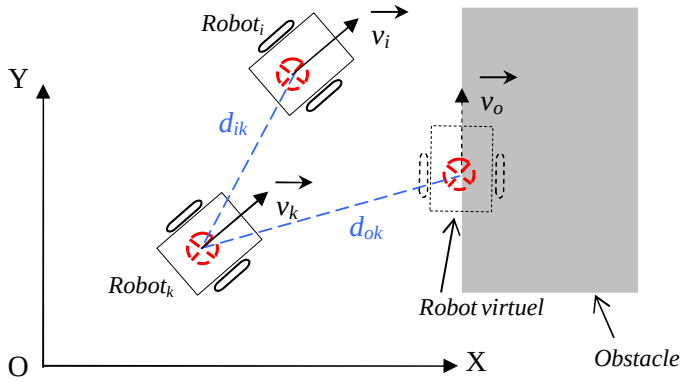


FIGURE 1.22 – Contrôle par séparation-séparation de l'obstacle (CSSO).

dimensions mais pas la forme géométrique (cf. Figure 1.23).

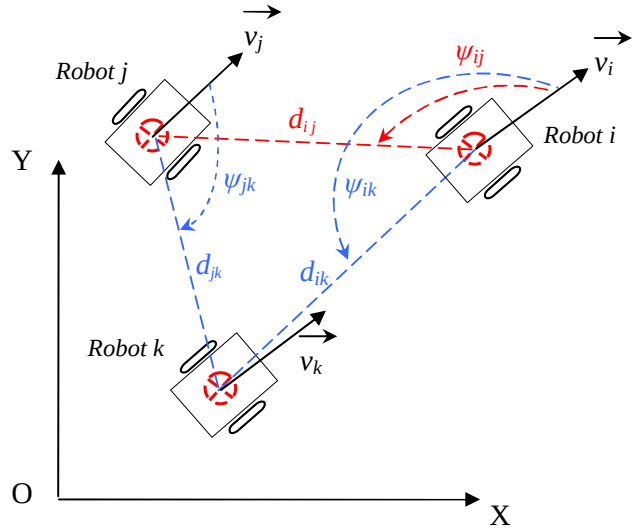


FIGURE 1.23 – Contrôle par dilatation.

Le choix de la technique de suivi du ou des guides pour un robot dépend des conditions de l'environnement et de la position du robot en question dans la formation. Par exemple, dans la figure 1.24, la réalisation de cette formation nécessite que les robots i et j suivent le robot l par la technique CSO, et que le robot k suit i et j par CSS. Un changement de situation peut aussi impliquer une commutation entre les techniques : ainsi, à la détection d'un obstacle, il devient impératif de commuter vers le mode CSSO quelque soit le mode dans lequel le robot évoluait avant [Fierro 01].

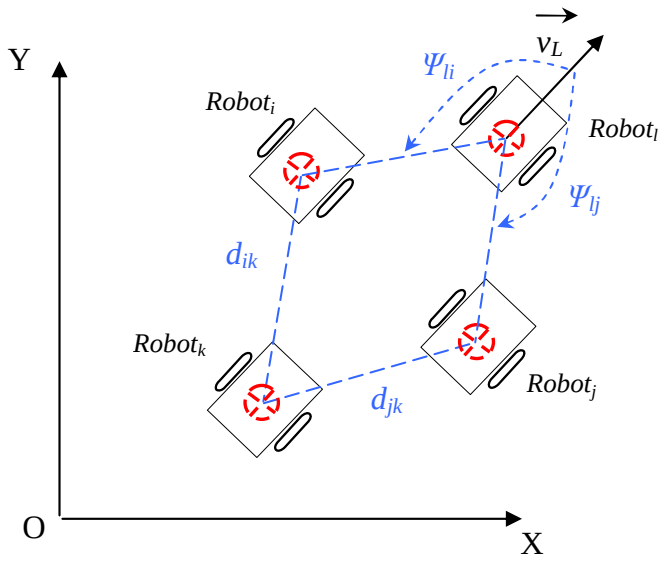


FIGURE 1.24 – Exemple de formation désirée.

L'approche hiérarchique est principalement appréciée pour la simplicité de sa mise en œuvre et sa nature réactive. En effet, il suffit simplement de définir le comportement désiré du robot guide. Les suiveurs auront uniquement à suivre son état. La limitation principale qui lui est reprochée est que la formation dépend impérativement du guide dont la panne implique un arrêt de toute la flotille. Ceci signifie que l'avantage de la distribution des ressources pour un SMR consistant à éviter la dépendance de la flottille d'une seule unité est perdu (cf. Section 1.4.3). Un autre inconvénient de cette approche est qu'il n'y a généralement pas un retour d'informations au guide sur l'état des suiveurs. Ainsi, si un ou plusieurs se perdent à cause d'une perturbation quelconque de l'environnement, le guide ne se rend pas compte et continue sa navigation.

Enfin, un autre inconvénient majeur peut apparaître avec l'augmentation du nombre des entités robotiques engagés dans le SMR. En effet, s'ils utilisent différentes techniques de suivi (CSO, CSS, etc.), le SMR devient non homogène car il faudra concevoir autant de règles de contrôle que de robots.

1.6.2 Approche comportementale

La structure de base de cette approche est de disposer de comportements élémentaires (cf. Section 1.5.2) dont les entrées sont les informations fournies par les capteurs du robot et les sorties sont les consignes de vitesses ou d'accélérations

envoyées aux actionneurs (cf. Figure 1.15, page 35). Un coordonnateur est alors nécessaire pour choisir le comportement à activer.

Cette méthode a été principalement inspirée de la nature : certains animaux se déplacent en groupe pour combiner leurs sens et maximiser ainsi leur chance de détecter leurs prédateurs ou de la nourriture [Vehrencamp 79]. L'étude de ces troupeaux d'animaux montre que ce déplacement en groupe émerge d'un désir individuel de rester proche des autres membres et de garder une certaines distance avec ses voisins [Cullen 65].

Les chercheurs en robotique et en intelligence artificielle ont voulu reproduire cette émergence de comportements sur de vrais robots ou des agents simulés : les premiers travaux de simulation sur des comportements de troupeaux animaux ont été réalisés par Reynolds [Reynolds 87] où un comportement global d'un ensemble d'agents simulés émerge de comportements d'agents individuels basés uniquement sur la perception locale de l'environnement et des agents voisins. On parle d'émergence d'un comportement lorsque plusieurs agents ou robots pouvant accomplir plusieurs tâches élémentaires interagissent créant ainsi un comportement inattendu du groupe global. On entend par inattendu que ce comportement n'est programmé explicitement nulle part. D'autres travaux d'amélioration sur des agents simulés ont suivi les travaux de Reynolds. Le but était essentiellement de rendre ces simulations plus proches de la réalité en introduisant de vraies contraintes (anatomie animale, non-holonomie pour les robots, etc) [Tu 94], [Brogan 97].

Les travaux de Mataric' montrent que l'émergence de comportements est tout aussi possible sur des robots mobiles à roues [Mataric' 92a], [Mataric' 92b]. L'application des approches comportementales à la navigation en formation a alors vite suivi ces travaux [Parker 94], [Parker 96], [Balch 99]. En effet, Parker propose ALLIANCE [Parker 94], [Parker 96]. Il s'agit d'une architecture de contrôle comportementale complètement distribuée, ce qui permet au SMR d'être plus tolérant aux défauts de fonctionnement, à certaines pannes, à l'ajout ou à l'élimination d'un ou plusieurs robots. Ces robots reproduisent des sentiments humains, comme l'impatience d'un robot à remplacer un autre qui accomplit maladroitement une tâche, ou alors le consentement du robot à abandonner sa tâche au profit d'un autre robot quand il réalise qu'il ne l'accomplit pas correctement. Mataric' dans [Mataric' 97] définit un comportement comme étant une loi de commande satisfaisant certaines contraintes afin d'accomplir un objectif (tâche) particulier. Ces comportements élémentaires sont testées pour satisfaire à des critères comme la répétabilité, la stabilité, la robustesse et leur aptitude à évoluer. Les vitesses appliquées aux actionneurs des robots sont le résultat d'un seul comportement choisi parmi ceux existant ou d'une combinaison de plusieurs comportements simultanément.

Balch [Balch 99] a exploré les approches comportementales pour la navigation en formation. Selon la forme géométrique désirée, il attribue à chaque robot une position relative dans le SMR. Il emprunte aussi la notion de guide à l'approche hiérarchique. Ce dernier n'est pas responsable de tenir la formation, c'est aux suiveurs de satisfaire cette exigence.

Ayant le point commun de décomposer la tâche globale en plusieurs comportements élémentaires, les approches comportementales diffèrent cependant dans la manière de gérer la coordination entre ces comportements afin de générer la commande de sortie à appliquer aux actionneurs. En effet, les informations fournies par les capteurs peuvent activer plusieurs comportements en même temps et chacun voudra appliquer sa commande calculée aux actionneurs. Ces derniers ne peuvent naturellement pas répondre à tous ces comportements au même instant d'où l'apparition des conflits.

Les solutions apportées dans la littérature se classent principalement dans deux catégories : les comportements compétitifs et les comportements coopératifs [Arkin 98]. Dans la méthode des comportements compétitifs, un seul comportement est activé et la commande appliquée au robot provient exclusivement de ce comportement. Dans la méthode des comportement coopératifs, deux ou plusieurs commandes issues de comportements différents sont fusionnées pour aboutir à une seule commande correspondant à une somme pondérée des comportements actifs. Le coefficient de pondération est calculé selon l'importance du comportement dans la tâche globale.

Les deux paragraphes suivants abordent ces deux stratégies de façon plus détaillée.

1.6.2.1 Approche compétitive (sélection d'action)

La résolution de conflits entre comportements se fait en considérant une compétition entre eux. Le comportement gagnant a le droit d'envoyer sa commande aux actionneurs. Généralement, une fonction servant de superviseur se base sur une forme de hiérarchisation des comportements ou de la sélection de l'action du comportement le plus important dans le contexte de la tâche à accomplir. Brooks [Brooks 85] est l'un des premiers à avoir défini une hiérarchie entre les comportements dans le cadre des approches compétitives avec la stratégie de subsomption. Il s'agit de permettre à ceux de rang supérieur de subsumer l'action de ceux de rang inférieur (cf. Figure 1.25).

Cependant, un passage aussi brusque entre les consignes de comportements différents engendre naturellement une discontinuité de commande pouvant affecter le comportement global du robot. Cela se traduit par des à-coups de commandes, voire amener le robot à un état instable.

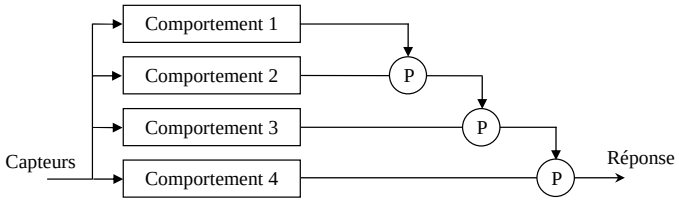


FIGURE 1.25 – Approches compétitive : hiérarchie entre les comportements.

La littérature sur les architectures comportementale est pauvre en travaux traitant ce problème. En effet, il est souvent difficile de prouver la stabilité lorsque plusieurs comportements différents coexistent. L'un des travaux traitant ce problème est celui de Harper [Harper 06] qui utilise des modules de protection. Quand un comportement de haute priorité entraîne l'état du système en dehors de la zone de stabilité d'un autre comportement, ce dernier peut alors rendre le contrôle global instable s'il est activé. L'utilité du module de protection relatif à ce comportement est donc de supprimer/inhiber l'action de celui pouvant entraîner le système vers sa zone instable.

Si cette suppression/inhibition est justifiée pour assurer la stabilité de l'architecture de contrôle, elle ne l'est pas pour l'exécution de la tâche. En effet, on peut supprimer un comportement important avant qu'il n'achève sa mission sans proposer un comportement alternatif.

D'autres mécanismes compétitifs peuvent être trouvés dans la littérature. On trouve alors la représentation par automates à états finis des comportements dont les transitions sont assurées grâce aux perceptions [Kube 97], la sélection d'action dynamique [Maes 89] produite par des liens activateurs/inhibiteurs entre les comportements, etc.

1.6.2.2 Approche coopérative (fusion d'actions)

Cette approche est venue dans le but de répondre aux situations de conflit qui peuvent apparaître entre les contrôleurs dans le cas de l'approche compétitive. Au lieu de choisir un seul contrôleur parmi les candidats pour l'envoyer aux actionneurs, l'idée est d'envoyer une commande comprenant la somme de toutes les commandes issues des contrôleurs actifs pondérés par un coefficient. Ce dernier peut être calculé et changé dynamiquement par un superviseur connaissant l'ordre d'importance des contrôleurs grâce aux informations capteurs. Les schémas moteurs [Arkin 86], principalement inspirés des sciences biologiques comme la neuroscience, sont l'exemple classique de la fusion d'actions. Un schéma moteur décrit l'interaction entre les perceptions et les actions. Il calcule une sortie

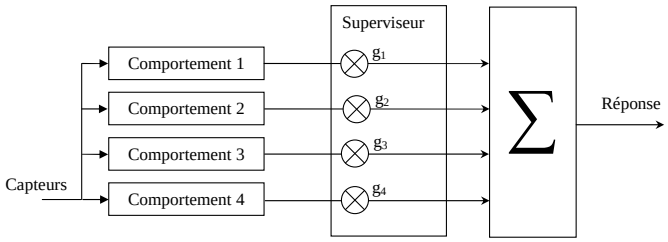


FIGURE 1.26 – Approches coopérative : fusion d'actions émanant de plusieurs comportements.

de commande qui sera multipliée par un gain avant d'être additionnée aux sorties des autres (cf. Figure 1.26).

La différence principale avec les approches compétitives est que la commande appliquée aux robots est une combinaison linéaire des commandes générées par tous les comportements élémentaires, ce qui signifie que du point de vue de chaque tâche élémentaire, elle n'est pas complètement accomplie et le système fonctionne en dégradé.

D'autres méthodes sont venues dans le but d'améliorer la fusion d'actions basée sur une simple combinaison linéaire. Certaines se définissent comme des stratégies de coordination des contrôleurs à part entière loin de la fusion d'actions [Egerstedt 00], [Antonelli 10]. Ainsi, les techniques des modèles cinématiques inverses utilisés dans le cas des robots manipulateurs ont été appliquées [Yang 03], [Bishop 03], [Antonelli 03]. Dans [Yang 03], les robots dits redondants (dans le cas où le SMR contient un nombre de robots supérieur aux robots nécessaires à la tâche) peuvent accomplir des tâches en parallèle à la tâche principale. Afin d'accomplir ces tâches subordonnées sans affecter les comportements prioritaires de la tâche principale ou entrer en conflit avec eux, l'idée est d'utiliser la projection orthogonale des comportements subordonnés dans le noyau de l'ensemble des comportements principaux. Cela est mieux illustré dans l'architecture NSB Control (Null-Space-based Behavioral Control) [Antonelli 10]. Elle reprend la notion de priorité entre comportements définie par un superviseur. Le comportement le plus prioritaire est accompli de façon complète (le gain associé à la vitesse correspondante est égal à 1). Les vitesses des autres comportements sont projetées sur la direction de celle de ce comportement principal. Ce sont les projections des vitesses qui viennent alors s'ajouter à celle du contrôleur le plus prioritaire.

Comparé à la stratégie hiérarchique où le guide ne connaît pas l'état des suiveurs et peut les perdre, l'élément important est qu'il y a un retour d' états de et vers tous les robots (ou du moins de et vers les robots voisins) grâce à leur homogénéité. Ceci implique que chacun prend en considération sa distance avec

les autres. De plus, cette homogénéité signifie qu'un robot défaillant n'affecte pas ou peu l'ensemble du SMR. Un autre avantage de cette stratégie est la distribution du contrôle sur les entités robotiques. Le contrôle et les moyens de perception de chaque robot lui sont embarqués, ce qui signifie que l'ajout de nouveaux robots dans le système n'affecte pas (ou très peu) la complexité du contrôle des entités robotiques.

L'inconvénient de ce type d'approches est la difficulté d'analyser l'architecture de contrôle afin de prouver la stabilité du système. La littérature conséquente qu'on trouve sur les approches comportementales aborde rarement des preuves de stabilité. En effet, celle-ci est souvent difficile à établir principalement à cause des transitions d'un comportement/contrôleur à un autre dans le cas des architectures de subsomption [Brooks 85] ou la difficulté d'analyser un contrôle global comprenant plusieurs comportements simultanément comme dans le cas des schémas moteur (fusion d'actions) [Arkin 86].

1.6.3 Structure virtuelle

Appelée aussi corps rigide virtuel voire guide virtuel [Ogren 02], [Lalish 06], la méthode de la structure virtuelle considère la formation comme un seul objet rigide et traduit la dynamique de ce corps en un contrôle adéquat [Do 07], [Li 05]. Chaque robot tient une position relative dans ce corps rigide mené par un guide virtuel. Ceci rappelle les propriétés de l'approche hiérarchique (maintien d'une posture par rapport au guide). La nuance entre les deux approches (structure virtuelle et approche hiérarchique) est que bien entendu tous les robots sont homogènes : il n'y a pas de guides (si ce n'est le corps virtuel), ni de suiveurs.

Les travaux de Leonard et Fiorelli [Leonard 01] présentent une illustration de la structure virtuelle. Chaque robot de la formation est soumis à des champs de potentiel (cf. Section 2.3.2) et donc des forces qui en dérivent. Ces forces sont créées par la présence des robots voisins ainsi que par les robots virtuels définis par la structure. Une force f appliquée à un robot est répulsive lorsqu'il s'agit d'un robot réel pour éviter la collision, et attractive lorsqu'elle provient d'un robot virtuel pour assurer la formation. En effet, si chaque robot suit un robot virtuel du corps dynamique, la formation globale est assurée. Ögren [Ogren 02] a repris ces travaux en ajoutant une contrainte sur la dynamique de la structure virtuelle. En effet, il est proposé que le maintien de la formation et le suivi des robots virtuels ne soient pas exclusivement à la charge des robots, et que la structure puisse décélérer afin d'attendre les robots éprouvant du mal à la rattraper. Il est cependant important de signaler que le modèle utilisé est un modèle dynamique qui ne prend pas en compte les contraintes effectives des robots (non-holonomie, vitesses et accélérations maximales, etc.).

L'avantage principal de cette stratégie est la souplesse relative de définir la dynamique (vitesse, dynamique interne) du corps virtuel selon l'application désirée. Elle est aussi simple à mettre en œuvre que l'approche hiérarchique et la détrône par l'homogénéité entre les robots utilisés. Ainsi, elle n'a pas besoin de concevoir un type de contrôle pour les guides et un autre pour les suiveurs. Cette stratégie permet à titre d'exemple de réaliser le suivi de chemin pour le SMR sans avoir recours à décomposer les robots en guides-suiveurs. Une fois le chemin de référence défini, il sera suivi par le centre de la structure virtuelle [Ghommam 08] tandis que les robots auront uniquement à suivre les sommets de la structure virtuelle.

L'inconvénient est que cette approche nécessite généralement le recours à une unité centrale traitant les robots comme un seul corps rigide. De plus, elle est généralement basée sur les méthodes de potentiel [Ogren 02], [Leonard 01](utilisées à l'origine dans l'évitement d'obstacles [Khatib 86]) et qui trouvent leurs limitations dans certains cas notamment si la formation désirée est fréquemment réorganisée en cours de navigation [Ghommam 08].

D'après ce qui précède, on voit que chaque stratégie présente ses avantages et ses inconvénients, ce qui rend difficile de se prononcer sur la meilleure approche à suivre. C'est pourquoi certains travaux ont développé des architectures de contrôle basées sur plusieurs approches. Ainsi, l'architecture proposée dans [Beard 01], contient un superviseur central qui collecte les informations sur l'environnement. Il permet de choisir la stratégie à suivre, en désignant un véhicule comme guide (approche hiérarchique), livrer à chaque robot les informations sur la position de ses voisins pour qu'il suivent une approche comportementale, ou donner l'état de la structure virtuelle si la troisième stratégie est retenue.

D'après ce qui précède, on peut mettre à jour la figure 1.17 avec les approches explorées précédemment. Le schéma final proposé est représenté dans la figure 1.27.

1.7 Positionnement de la stratégie du maintien de formation proposée

L'objectif de cette thèse est de réaliser une architecture de contrôle permettant le maintien de formation de robots mobiles dans un environnement encombré. Nous souhaitons que cette architecture reste la plus générique et flexible possible. En d'autres termes, nous souhaitons qu'elle soit simple d'utilisation, insensible au nombre de robots présents dans la formation et valable même dans des environnements encombrés et dynamiques. Elle doit être aussi ouverte et évolutive dans

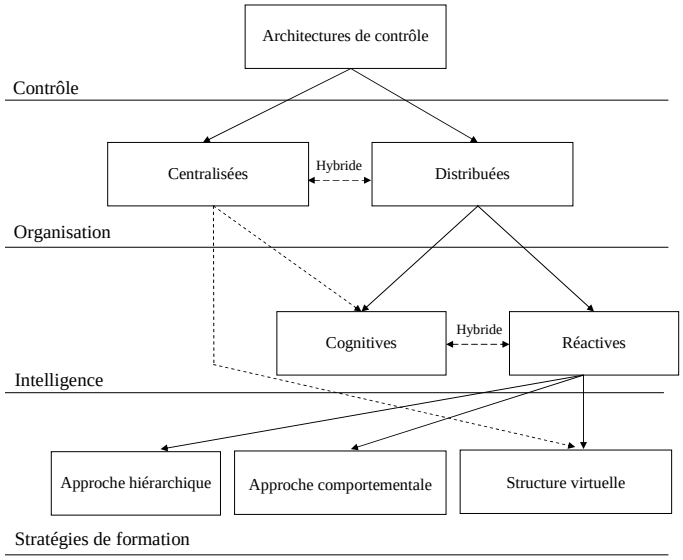


FIGURE 1.27 – Approches classiques pour la navigation en formation.

le but d'accomplir des tâches plus complexes sans pour autant être complètement changée.

Considérant les caractéristiques désirées citées ci-dessus, on s'aperçoit que confier le contrôle et la gestion du SMR à un superviseur central devient rapidement une tâche inextricable pour ce dernier avec l'augmentation du nombre de robots mobiles. Il est alors judicieux voire indispensable de doter chaque entité de ses propres outils décisionnels. Nous serons donc dans le cadre d'un contrôle distribué.

Par ailleurs, un contrôle cognitif ne peut être choisi. En effet, ce contrôle se base sur la planification de l'action à accomplir, ce qui nécessite de modéliser l'environnement et de minimiser un certain coût. Modéliser un environnement hautement dynamique est difficile voire impossible. Les coût minimisés prennent en considération plusieurs facteurs : distance parcourue, temps nécessaire, risque de collision, etc.. La tâche de quantifier ces critères est tout aussi difficile dans ce type d'environnement du fait de ses changements fréquents. En conséquence, la planification d'une action en prenant en compte tous ces éléments à l'instant (t), peut la rendre inutile ou irréalisable à l'instant $(t+\Delta t)$ car l'environnement à cet instant n'est plus celui de l'instant (t) (changement de la position des obstacles, leurs formes, de la position du point à atteindre, etc.). Même si des solutions ont été apportées par la communauté travaillant sur cette thématique et qui consiste à replanifier l'action en prenant en considération le nouvel environnement, voire en anticipant celle des éléments dynamiques [Latombe 91] [Berg 06], ces solu-

tions sont encore plus gourmandes en calcul que pour le cas d'un environnement statique. C'est pourquoi nous opterons pour un contrôle réactif où les capteurs propres aux robots sont les seules sources d'informations pour établir la stratégie du contrôle.

Enfin, pour le maintien de la formation, nous nous inspirons des stratégies présentées ci-dessus, à savoir : approche hiérarchique, approche comportementale et la structure virtuelle. L'objectif est évidemment de combler les lacunes de l'une grâce aux avantages de l'autre. Cependant, nous souhaitons que les entités du SMR aient toutes la même architecture de contrôle plutôt que de prévoir plusieurs contrôles.

L'architecture de contrôle proposée s'inspire alors principalement des deux stratégies que sont l'approche comportementale et la structure virtuelle. Nous proposons de reprendre la figure 1.27 classifiant les architectures de contrôle utilisées dans les SMR afin de positionner les travaux de cette thèse (cf. Figure 1.28) et de justifier notre choix dans le paragraphe suivant.

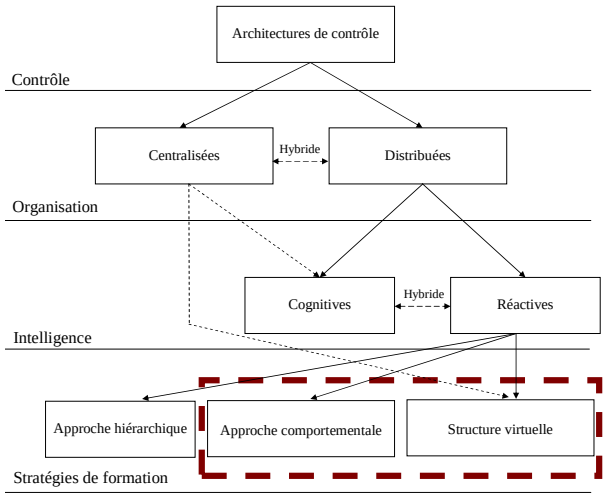


FIGURE 1.28 – Placement des travaux réalisés et l'architecture de contrôle proposée.

1.7.1 Justification des choix adoptés

Les avantages et les inconvénients des stratégies de maintien de formation réactives ont été présentés ci-dessus (cf. Section 1.6). L'architecture de contrôle proposée et inspirée des deux stratégies que sont l'approche comportementale et

la structure virtuelle, a pour objectif de pallier les inconvénients de ces différentes approches prises séparément.

En effet, la structure virtuelle est relativement simple à mettre en œuvre, et offre une homogénéité aux entités robotiques (un seul contrôle pour tous). Cependant, certains lui reprochent d'avoir recours à un niveau central car elle considère tous les robots comme un seul corps rigide [Beard 01], [Lalish 06]. Cela peut être perçu comme un handicap emprunté aux architectures de contrôle centralisées.

L'autre limitation que l'on reproche à la structure virtuelle est qu'elle est souvent associée aux méthodes des champs de potentiel (cf. Section 2.3.2) pour sa mise en œuvre [Ghommam 08]. Bien que cette méthode soit très intéressante par son élégance et la simplicité de sa mise en œuvre, elle peut trouver ses limitations dans l'application du maintien de formation. Hormis les minima locaux dus à la somme nulle des forces appliquées au robot et qui peuvent être résolues [Barnes 07], ces méthodes ne sont pas très adaptées aux cas où la formation varie dans le temps ou doit être fréquemment réorganisée [Nguyen 06].

Au vu de ces commentaires, et afin d'assurer un maximum de flexibilité à notre architecture de contrôle, il est proposé de coupler l'approche comportementale, à la structure virtuelle afin de pallier les inconvénients de cette dernière.

D'une part, l'aspect purement distribué de l'approche comportementale permet de briser l'utilité de recourir à un superviseur central, et d'autre part, nous pourrons se passer de la méthode des champs de potentiel. Ainsi, nous assurons le contrôle distribué désiré. De plus, nous souhaitons qu'à long terme, notre architecture puisse subir un changement de formation en cours de navigation. En d'autres termes, nous souhaitons que le corps virtuel ne soit pas forcément rigide et que l'architecture de contrôle soit ouverte à une possible dynamique interne de ce corps, d'où le besoin de se passer des champs de potentiel. Cependant, se priver de ces derniers pour la méthode de la structure virtuelle nécessite de trouver une alternative afin d'assurer la sécurité des robots face aux obstacles ou à la collision entre entités. Grâce à l'approche comportementale, nous introduisons un comportement/contrôleur élémentaire responsable de l'évitement d'obstacles.

Enfin, cette approche semble un choix pertinent pour donner un aspect d'ouverture à notre architecture de contrôle par une approche ascendante. Cette ouverture permet de mettre à jour l'architecture au fur et à mesure que l'on veut confier au SMR une tâche plus complexe. Cette tâche peut être décomposée en plusieurs tâches élémentaires. Il suffit alors d'incrémenter les nouveaux comportements/contrôleurs accomplissant les nouvelles tâches élémentaires dans la base de contrôleurs disponibles. La reconstruction complète d'une nouvelle architecture de contrôle est alors évitée.

1.8 Conclusion

Ce chapitre nous a permis en premier lieu d'avoir une introduction générale à la robotique mobile et à ses applications. Etant donné que l'objectif de la thèse est de contrôler un groupe de robots mobiles naviguant en formation, nous nous sommes d'abord intéressé aux éléments nécessaires au contrôle d'un seul robot mobile, à savoir la perception et la localisation, la décision et l'action. Avec la maîtrise du contrôle d'un robot mobile, l'idée d'en faire coopérer plusieurs devient attrayante. L'objectif est évidemment de tirer profit de la coopération du groupe. Cependant, il est indispensable de bien coordonner l'action des entités robotiques, et de prendre en compte certains paramètres afin de justifier le recours à plusieurs robots plutôt qu'à un seul. Une fois le choix du SMR retenu, il faut concevoir le contrôle adéquat. Nous avons pu voir les différentes architectures de contrôle existant dans la littérature. Elles ont été classifiées selon leur méthode d'implantation : centralisées au niveau d'un superviseur global, ou distribuées (délocalisées) sur les entités robotiques. Comme nous optons pour une architecture de contrôle plutôt distribuée afin de ne pas dépendre d'un superviseur central (dont la défaillance affecte le fonctionnement du SMR), nous avons vu les différents types d'architectures selon le degré de sophistication des entités robotiques utilisées : ainsi, l'architecture peut être cognitive ou réactive.

Enfin, en fonction du cahier des charges que nous nous sommes imposé, à savoir la réalisation de la navigation en formation d'un groupe de robots mobiles, nous avons vu les différentes stratégies disponibles dans la littérature, leurs avantages et inconvénients afin de nous aider à retenir le meilleur choix pour l'architecture de contrôle que nous proposons dans le chapitre suivant.

Nous avons retenu la solution de s'inspirer de l'approche « structure virtuelle » tout en profitant de l'approche comportementale. La première stratégie permettra aux robots de maintenir la formation géométrique souhaitée grâce au suivi d'un corps rigide, tandis que la deuxième offre la possibilité de se passer d'un superviseur global et d'introduire l'évitement d'obstacles comme une tâche à part entière.

A noter que dans la littérature, il y a peu d'architectures de contrôle se basant explicitement sur différentes stratégies simultanément afin de tirer le maximum d'avantages ou réduire les inconvénients. Nous pouvons citer les travaux de Balch [Balch 99] qui désigne un robot guide dans un SMR basée sur une approche comportementale, perdant ainsi l'avantage d'homogénéité de cette dernière (il doit prévoir un contrôle pour le guide et un autre pour les suiveurs). Beard [Beard 01] change de stratégie en fonction de l'environnement, mais il utilise une seule à la fois.

Dans les travaux que nous proposons, il y aura deux tâches élémentaires dé-

crites sous forme de deux comportements/contrôleurs avec une sélection d'action selon les informations fournies par les capteurs. Le chapitre suivant est dédié à cette architecture.

Chapitre 2

Architecture de contrôle proposée

L'architecture de contrôle proposée est introduite dans ce chapitre. Après une description générale du principe adopté, les blocs de l'architecture sont détaillés individuellement. S'inspirant d'une approche comportementale (cf. Section 1.7), elle sera composée principalement de deux contrôleurs accomplissant deux tâches élémentaires : attraction vers une cible dynamique et évitement d'obstacles. Ces deux contrôleurs sont minutieusement étudiés. Nous décrivons ensuite le principe de coopération entre les entités robotiques.

Chaque contribution apportée est illustrée par des résultats de simulation.

2.1 Principe général de la stratégie de maintien de formation

Considérons un SMR à N entités dont le but est d'atteindre et de maintenir une formation désirée dans un environnement encombré. En d'autres termes, les robots doivent aussi pouvoir éviter les obstacles gênant et la collision entre eux-mêmes. S'inspirant d'une approche comportementale (cf. Section 1.7.1), cette tâche peut alors être décomposée en deux tâches élémentaires : le maintien de formation et l'évitement d'obstacles.

Se mettant au niveau d'une entité robotique, la mission du maintien en formation revient alors à garder une position relative dans le groupe afin de satisfaire la formation générale désirée.

Utilisant l'approche de structure virtuelle, l'idée est de définir une forme dont les sommets constituent des cibles dynamiques pour les robots. Ainsi, si chaque robot atteint et suit continuellement une cible dynamique, la formation du SMR est assurée.

La structure dynamique virtuelle proposée est définie comme suit :

- définir un point où la dynamique désirée (vitesse v_C et direction θ_C) de la formation est appliquée. Ce point est appelé cible dynamique principale (cf. Figure 2.1).
- définir la structure virtuelle en fonction de la formation désirée : il s'agit de déterminer au moins autant de sommets N_c que de robots ($N_c \geq N$) avec des positions relatives à la cible principale. Chaque sommet constitue une cible dynamique secondaire qui sera suivie par un robot. Les positions relatives sont générées en fonction de la forme géométrique désirée. Pour des raisons de simplicité, nous choisissons d'exprimer ces positions en coordonnées polaires dans un repère parallèle au repère absolu (O_m, X_m, Y_m) et dont l'origine est la cible principale. Ainsi, une cible i est située à une distance D_i et un angle Φ_i par rapport à la cible principale. Les éléments permettant le passage à un repère relatif sont expliqués dans le chapitre 3, paragraphe 3.3.1.2.

La figure 2.1 montre un exemple d'une formation triangulaire d'un SMR à trois robots mobiles.

Nous nous intéressons au cas où la structure virtuelle est rigide et qu'il n'y a pas de changement de formation en fonction du temps (cf. Figure 2.2). Ceci se traduit sur la dynamique interne de la structure par

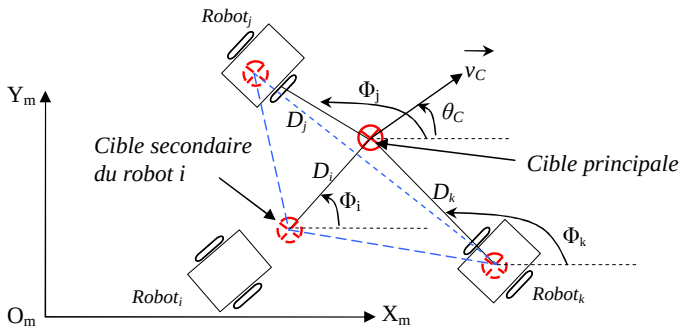


FIGURE 2.1 – Exemple d'une structure virtuelle.

$$\begin{cases} \dfrac{\partial D_i}{\partial t}|_{i=1..N_c} = 0 \\ \dfrac{\partial \Phi_i}{\partial t}|_{i=1..N_c} = 0 \end{cases} \tag{2.1}$$

Ces relations permettent de déduire que toutes les cibles dynamiques ont la même vitesse v_C et la direction θ_C. De même que pour la définition de la formation dans un repère relatif, les premiers éléments de réponse pour le passage à un corps à dynamique interne sont donnés dans le chapitre 3, (section 3.3.1.2, page 128).
Pour que les robots accomplissent cette tâche, il est impératif de savoir comment contrôler chaque élément du SMR. Nous nous intéresserons dans le paragraphe suivant à une seule entité robotique pour comprendre comment aborder son contrôle.

2.2 Modèle de l'entité robotique utilisé

Le problème du contrôle d'un robot mobile est grandement lié à ses caractéristiques cinématiques et dynamiques. En effet, la notion de robot mobile est vaste et inclut différents types : ainsi, les robots à roues, humanoïdes, véhicules sous marins autonomes, véhicules aériens autonomes, etc sont des robots mobiles. Cependant, ils diffèrent clairement par leurs caractéristiques ce qui génère une formulation différente du problème de contrôle pour chaque type.

Dans cette thèse, nous nous focaliserons sur les robots mobiles à roues qui nous serviront pour expérimenter l'architecture de contrôle proposée pour la navigation en formation. Plus précisément, nous utilisons pour illustrer nos propos des robots

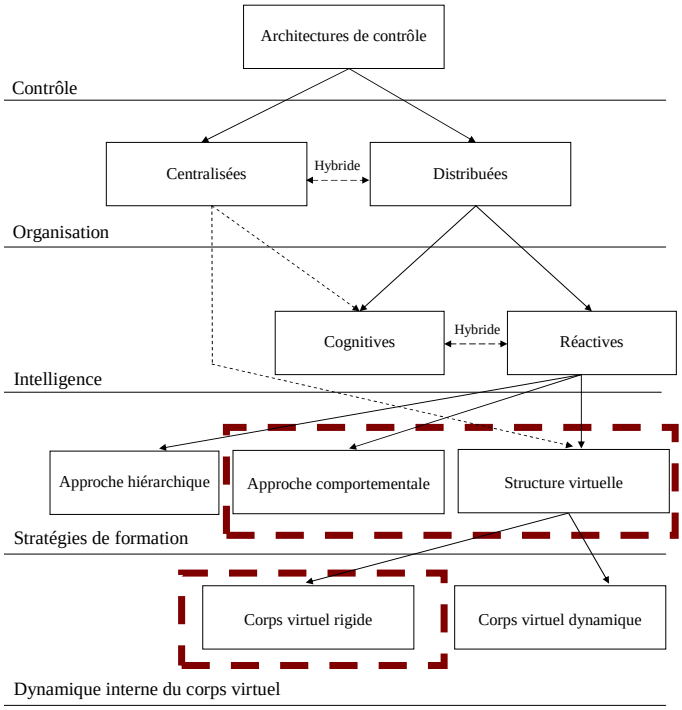


FIGURE 2.2 – L'utilisation d'un corps virtuel rigide dans l'architecture de contrôle proposée.

Khepera III (la troisième génération Khepera de la société K-Team) (cf. Figure 2.3).

2.2.1 Robots mobiles à roues

Les robots à roues sont les premiers robots mobiles et les plus étudiés, sans doute car la roue est l'un des premiers mécanismes de locomotion créée par l'homme et reste largement suffisante pour les déplacements des véhicules. Les caractéristiques des roues d'un robot mobile (forme, angle de braquage, rayon, etc.) définissent ses caractéristiques et les degrés de liberté de sa mobilité. Ainsi, on trouve les robots dits omnidirectionnels (ou plus connus chez les roboticiens sous le terme de robots mobiles holonomes). Ce type de robots peut se déplacer instantanément dans n'importe quelle direction indépendamment de son orientation. Dans cette catégorie s'inscrivent les robots à roues de formes sphériques ou à galets pouvant tourner librement (cf. Figure 2.4).

Par opposition à ce type de robots, on trouve les robots non-holonomes plus communs dans la communauté robotique. Contrairement aux autres, ils perdent

FIGURE 2.3 – Le robot Khepera III de K-Team.

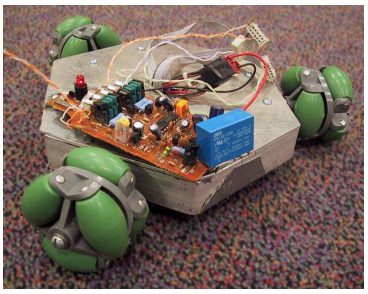


FIGURE 2.4 – Exemple de robot mobile utilisant une roue à galets.

un degré de liberté correspondant à la translation instantanée dans une direction latérale. Il y a principalement trois types de robots mobiles non-holonomes :

- le robot tricycle : il est constitué de deux roues fixes de même axe lui conférant la vitesse linéaire et d'une roue centrée orientable et placée sur son axe longitudinal. L'angle de braquage de cette roue définit le centre de rotation du robot.
- le robot de type voiture : similaire au tricycle, il diffère par le niveau avant qui contient deux roues plutôt qu'une seule assurant une meilleure stabilité. On parle de robot lorsqu'il s'agit d'une voiture autonome sans conducteur ni télépilotage. Les véhicules autonomes utilisés pour la navigation en convoi illustrent ce type de robots (cf. Figure 2.5).
- le robot unicycle : il est actionné par deux roues dont les moteurs sont indépendants. Il peut contenir d'autres roues folles dont le but est uniquement d'assurer sa stabilité horizontale. Les robots Khepera III (cf. Figure 2.3) sont des robots unicycles, c'est pourquoi nous allons nous intéresser au modèle cinématique de ces robots afin de décrire l'architecture de contrôle proposée.

FIGURE 2.5 – Véhicules autonomes de type cycab de la société Robosoft.

2.2.2 Modèle cinématique du robot mobile utilisé

Nous avons vu que les roues, leurs caractéristiques et leur disposition définissent la mobilité du robot au même temps qu'elles lui imposent des contraintes.

Pour modéliser le déplacement du robot mobile en tenant compte de ces contraintes, un repère convenable doit être judicieusement choisi. Ainsi, afin de spécifier sa position, il est possible de considérer le robot comme un objet rigide se déplaçant sur un plan horizontal avec deux degrés de liberté : le premier correspond à un mouvement longitudinal (déplacement à l'avant ou à l'arrière) tandis que le deuxième est celui de la rotation autour d'un axe vertical. Un repère relatif (O_r, X_r, Y_r) est associé à cet objet (cf. Figure 2.6). L'origine O_r correspond au centre du robot entre les des deux roues. La position du robot et son orientation sont définies en exprimant la relation entre le repère relatif et un repère absolu (O_m, X_m, Y_m). Ainsi la position d'un robot mobile i du SMR est spécifiée par (x_i, y_i) correspondant aux coordonnées du point O_r dans le repère(O_m, X_m, Y_m). Son orientation θ_i est l'angle compris entre l'axe X_r et X_m.

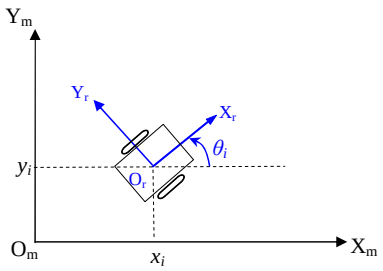


FIGURE 2.6 – Représentation de la pose du robot dans le repère global et relatif.

$$O_{r_i} = \begin{pmatrix} x_i \\ y_i \\ \theta_i \end{pmatrix} \tag{2.2}$$

Le déplacement du robot peut alors être exprimé par le système d'équations suivant

$$\dot{x}_i = v_i.\cos(\theta_i) - u_i.\sin(\theta_i) \tag{2.3a}$$
$$\dot{y}_i = v_i.\sin(\theta_i) + u_i.\cos(\theta_i) \tag{2.3b}$$
$$\dot{\theta}_i = \omega_i \tag{2.3c}$$

Ce qui peut être exprimé sous une forme matricielle :

$$\dot{O}_{r_i} = R(\theta) \begin{pmatrix} v_i \\ u_i \\ \omega_i \end{pmatrix} \tag{2.4}$$

Où v_i, u_i, sont les composantes de la vitesse linéaire du robot i exprimées dans le repère (O_r, X_r, Y_r) et ω_i est sa vitesse angulaire. La matrice $R(\theta)$ est la matrice de rotation permettant d'exprimer la variation des positions du robot dans le repère (O_m, X_m, Y_m) et est définie comme suit

$$R(\theta) = \begin{pmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 \\ \sin(\theta_i) & \cos(\theta_i) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.5}$$

Cependant, la contrainte non-holonome impose que le déplacement linéaire du robot se fait uniquement selon l'axe X_r relatif. Ainsi, la composante u de la vitesse linéaire correspondant au déplacement latéral sur l'axe Y_r est nulle. Les équations 2.3 se réduisent alors à

$$\dot{x}_i = v_i.\cos(\theta_i) \tag{2.6a}$$
$$\dot{y}_i = v_i.\sin(\theta_i) \tag{2.6b}$$
$$\dot{\theta}_i = \omega_i \tag{2.6c}$$

Dans le cas d'un robot dit unicycle, où les deux roues sont contrôlées par deux moteurs séparément, et en l'absence de glissement, les vitesses linéaire et angulaire se calculent par

$$\begin{pmatrix} v_i \\ \omega_i \end{pmatrix} = \begin{pmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{l} & \frac{-r}{l} \end{pmatrix} \begin{pmatrix} \omega_{d_i} \\ \omega_{g_i} \end{pmatrix} \tag{2.7}$$

où ω_{d_i} et ω_{g_i} sont les vitesses angulaires des roues droite et gauche respectivement et r est le rayon de la roue. l représente la distance entre les roues.

Généralement, on définit le contrôle nécessaire du robot en établissant les vitesses linéaire et angulaire v et ω. Le contrôle des roues se déduit alors grâce au modèle décrit par le système d'équations 2.7.

2.3 Structure de l'architecture de contrôle proposée

Comme précédemment mentionné, l'architecture de contrôle proposée contiendra deux contrôleurs, à savoir celui responsable du maintien en formation et celui de l'évitement d'obstacles. Nous avons vu qu'au niveau d'une seule entité (comme il s'agit d'une architecture distribuée), le maintien de la formation revient à tenir une position relative par rapport à la cible principale (ce qui revient à suivre l'une des cibles secondaires). Le contrôleur utilisé sera alors un contrôleur d'attraction vers une cible dynamique. L'architecture de contrôle proposée est représentée dans la figure 2.7.

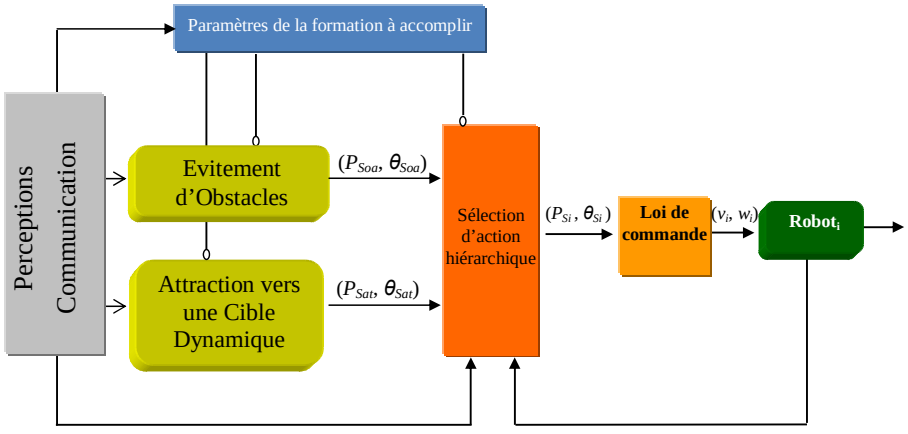


FIGURE 2.7 – Architecture de contrôle proposée.

Nous commençons par définir les deux blocs *Paramètres de la formation à accomplir*, et *Perceptions Communication*.

- Paramètres de la formation à accomplir : il attribue les grandeurs (D_i, Φ_i) (cf. Figure 2.1, page 57) définissant les positions des cibles virtuelles secondaires par rapport à la cible principale,
- Perceptions Communication : Il représente les capteurs délivrant les informations sur l'environnement local et toute la partie communication permettant l'échange entre les entités robotiques. Ce bloc interagit avec les contrôleurs pour informer des positions des obstacles, de la cible, etc. et avec le bloc de la *Sélection d'action hiérarchique* pour prendre des décisions de transitions entre contrôleurs.

A présent, nous allons décrire plus en détail les deux contrôleurs car c'est sans doute les deux blocs les plus importants de l'architecture.

2.3.1 Attraction vers une Cible Dynamique

L'attraction/suivi d'une cible dynamique est une problématique déjà étudiée dans la littérature et les application sont diverses : on trouve alors l'usage militaire comme le maintien d'un objet dynamique dans une zone de surveillance [Schulz 01], les robots footballeurs dont l'objectif est de suivre le ballon et donnant lieu même à des tournois comme RoboCup [1].

Pourtant, la cible dynamique n'a pas reçu autant d'attention que son analogue statique. En effet, le problème devient difficile, spécialement quand la dynamique de cette cible est inconnue ou difficile à prédire.

En conséquence, les méthodes basées sur des modèles analytiques ne sont pas nombreuses. Généralement, elles sont basées sur une fusion des méthodes des champs de potentiels et des fonctions de Lyapunov : dans [Zhu 09], ces deux approches sont combinées pour établir le contrôle d'un robot mobile aérien suivant une cible dynamique dont la vitesse est supposée constante. Le facteur vent, important dans la commande de ce type de véhicules, est aussi supposé constant. Les travaux de [Adams 99] s'inscrivent aussi dans cette approche de fusion. Dans [Lee 00], une loi basée sur une fonction de Lyapunov est établie. Elle prouve que le robot converge vers une distance désirée de la cible. Cependant, certaines singularités n'ont pas été traitées. Comme précédemment cité (cf. Section 1.7.1), bien que les méthodes des champs de potentiels soient intéressantes, elles ne sont pas convenables à notre application et à l'ouverture désirée de l'architecture de contrôle proposée (cf. Section 1.7). Une autre communauté établit des contrôles basés sur la logique floue [Zadeh 75] et s'intéresse à l'attraction et/ou le suivi de cibles dynamiques [Stonier 98], [Jeong 99], [Tong 98]. Cependant, la logique floue a l'inconvénient qu'il faut déterminer les règles de contrôle de façon appropriée,

1. www.robocup.org

sans oublier qu'elles doivent être régulièrement mises à jour, notamment dans le cas des changements fréquents de la dynamique de l'environnement (ce qui est généralement le cas).

Nous proposons dans ce qui suit une solution analytique simple permettant au robot de converger vers la cible et de continuer à la suivre dans la limite où le déplacement de celle-ci reste non-holonome : dans le cas contraire, la cible se trouve dans une position non atteignable pour le robot et il doit entamer des manoeuvres pour l'atteindre. Une étude détaillée des limites de la dynamique de la cible est fournie dans le chapitre 3 (cf. section 3.3.1.1, page 122).

Dans ce qui suit, nous nous intéresserons à un robot mobile i. Dans le souci de simplifier la notation, la cible que le robot s'est approprié sera notée i aussi. Elle aura la même dynamique que la structure virtuelle globale vu que celle-ci est rigide. Il s'agit donc d'une direction unique θ_C et d'une vitesse $v_C \geq 0$ pour toutes les cibles (cf. Figure 2.8).

La variation de la position de la cible dynamique peut alors être exprimée par la relation

$$\begin{cases} \dot{x}_{C_i} = v_C . \cos(\theta_C) \\ \dot{y}_{C_i} = v_C . \sin(\theta_C) \end{cases} \tag{2.8}$$

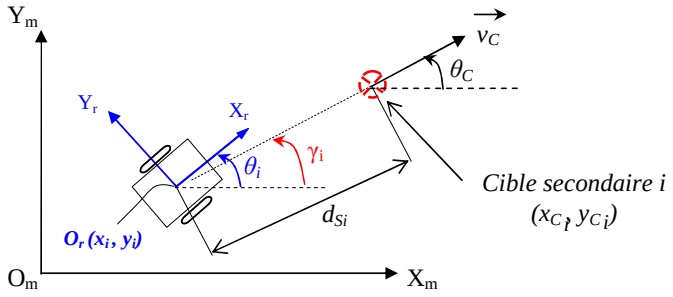


FIGURE 2.8 – Un robot mobile i et sa cible dynamique i.

D'après la figure 2.8, on définit les erreurs de position entre le robot et sa cible

$$\begin{cases} e_{x_i} = & (x_{C_i} - x_i) = d_{S_i} \cos(\gamma_i) \\ e_{y_i} = & (y_{C_i} - y_i) = d_{S_i} \sin(\gamma_i) \end{cases} \tag{2.9}$$

d_{S_i} étant la distance euclidienne qui sépare le robot (point O_m situé au milieu de l'axe passant par le centre des deux roues du robot (cf. Figure 2.8)) de sa cible. γ_i est l'angle que le robot aurait s'il se dirigeait tout droit vers la cible.

La distance d_{S_i} s'exprime

$$d_{S_i} = \sqrt{e_{x_i}^2 + e_{y_i}^2} \tag{2.10}$$

Pour connaitre la variation de la distance d_{S_i}, il faut calculer sa dérivée

$$\dot{d}_{S_i} = \frac{e_{x_i}\dot{e}_{x_i} + e_{y_i}\dot{e}_{y_i}}{d_{S_i}} \tag{2.11}$$

On remarque qu'elle est à son tour en fonction de la variation des erreurs de position e_x et e_y (cf. Equation 2.9) qui s'expriment

$$\begin{cases} \dot{e}_{x_i} = & (\dot{x}_{C_i} - \dot{x}_i) \\ \dot{e}_{y_i} = & (\dot{y}_{C_i} - \dot{y}_i) \end{cases} \tag{2.12}$$

D'après les relations 2.8 et 2.6, le système 2.12 devient

$$\begin{cases} \dot{e}_{x_i} = & v_C.\cos(\theta_C) - v_i.\cos(\theta_i) \\ \dot{e}_{y_i} = & v_C.\sin(\theta_C) - v_i.\sin(\theta_i) \end{cases} \tag{2.13}$$

En remplaçant les erreurs ainsi que leurs variations (systèmes 2.9 et 2.13) dans 2.11

$$\dot{d}_{S_i} = \frac{d_{S_i}\cos(\gamma_i)(v_C.\cos(\theta_C) - v_i.\cos(\theta_i)) + d_{S_i}\sin(\gamma_i)(v_C.sin(\theta_C) - v_i.sin(\theta_i))}{d_{S_i}} \tag{2.14}$$

Après simplification, on obtient

$$\dot{d}_{S_i} = \cos(\gamma_i)(v_C.\cos(\theta_C) - v_i.\cos(\theta_i)) + \sin(\gamma_i)(v_C.\sin(\theta_C) - v_i.\sin(\theta_i))$$

ce qui donne

$$\dot{d}_{S_i} = v_C.\cos(\gamma_i - \theta_C) - v_i.\cos(\gamma_i - \theta_i) \tag{2.15}$$

Pour que le robot atteigne sa cible, l'idée est de trouver une consigne d'angle qui permettrait à la distance d_{S_i} de toujours décroître. En d'autres termes, on doit avoir

$$\dot{d}_{S_i} < 0 \tag{2.16}$$

Pour cela, il est proposé que le robot garde un angle γ_i constant avec sa cible. Nous prouverons alors que sous cette contrainte, et si le robot est plus rapide que la cible, la relation 2.16 sera toujours satisfaite. La définition de l'angle γ_i permet d'établir son expression comme suit

$$\gamma_i = \arctan(\frac{e_{y_i}}{e_{x_i}}) \tag{2.17}$$

Garder un angle γ_i constant revient à mettre

$$\dot{\gamma}_i = 0 \tag{2.18}$$

Calculons d'abord l'expression de $\dot{\gamma}_i$

$$\dot{\gamma}_i = \frac{\frac{d}{dt}(e_{y_i}/e_{x_i})}{1+(e_{y_i}/e_{x_i})^2} \tag{2.19a}$$

$$= \frac{(v_C.s(\theta_C) - v_i.s(\theta_i))c(\gamma_i) - s(\gamma_i)(v_C.c(\theta_C) - v_i.c(\theta_i))}{d_{S_i}} \tag{2.19b}$$

(où $s(\theta) = \sin(\theta)$, $c(\theta) = \cos(\theta)$).

Sachant

$$\sin(\alpha+\beta) = \sin\alpha\cos\beta + \cos\alpha\sin\beta$$
$$\sin(\alpha-\beta) = \sin\alpha\cos\beta - \cos\alpha\sin\beta$$

On obtient après développement de 2.19b

$$\dot{\gamma}_i = \frac{v_C.\sin(\theta_C-\gamma_i)}{d_{S_i}} - \frac{v_i.\sin(\theta_i-\gamma_i)}{d_{S_i}} \tag{2.20}$$

Afin de garder γ_i constant, il suffit alors d'assurer

$$\frac{v_C\sin(\theta_C-\gamma_i)}{d_{S_i}} - \frac{v_i.\sin(\theta_i-\gamma_i)}{d_{S_i}} = 0 \tag{2.21}$$

ce qui permet de déduire l'angle consigne désirée $\theta_{S_{at}}$

$$\theta_{S_{at}} = \arcsin(b\sin(\theta_C-\gamma_i)) + \gamma_i \tag{2.22}$$

avec

$$b = \frac{v_C}{v_i}$$

Le résultat trouvé ici est proche des travaux de Belkhouche [Belkhouche 06]. Néanmoins, des différences non négligeables subsistent dans les deux approches :

- les travaux de Belkhouche reposent sur un observateur virtuel dont la position affecte directement l'angle consigne résultant $\theta_{S_{at}}$. Ici, le raisonnement est différent et l'angle consigne ne dépend que des vitesses linéaires du robot et de la cible, et de l'orientation de celle-ci. Aucun observateur n'est considéré,
- nous n'imposons pas une contrainte de vitesses constantes (de la cible et du robot) comme c'est le cas dans [Belkhouche 06], seule la non-holonomie est imposée à la cible afin de rester dans le domaine atteignable du robot et éviter les saturations au niveau des vitesses angulaires des robots,
- la loi de commande proposée (cf. Section 2.36) permettra au robot d'atteindre la cible en décélérant jusqu'à ce que sa vitesse tende vers celle de la cible tandis que dans [Belkhouche 06], il est seulement prouvé qu'il existe un point d'intersection entre la trajectoire du robot et celle de sa cible. En effet, comme les vitesses qu'ils considèrent sont constantes avec la supposition que $v_i \geq v_T$, stabiliser le robot sur cette cible n'est pas possible.

Pour prouver que le robot atteint sa cible s'il suit la consigne d'angle définie en équation 2.22, nous devons prouver que la distance d_{S_i} est toujours décroissante (cf. Equation 2.16).

Comme cité précédemment, la vitesse linéaire du robot sera élaborée (cf. Section 2.36) de telle sorte d'avoir toujours

$$v_i \geq v_C$$

Il en résulte alors

$$b = \frac{v_C}{v_i} \leq 1 \tag{2.23}$$

Pour prouver nos propos, les deux propriétés suivantes sont rappelées

$$\cos(-x) = cos(x), \forall x \in R$$

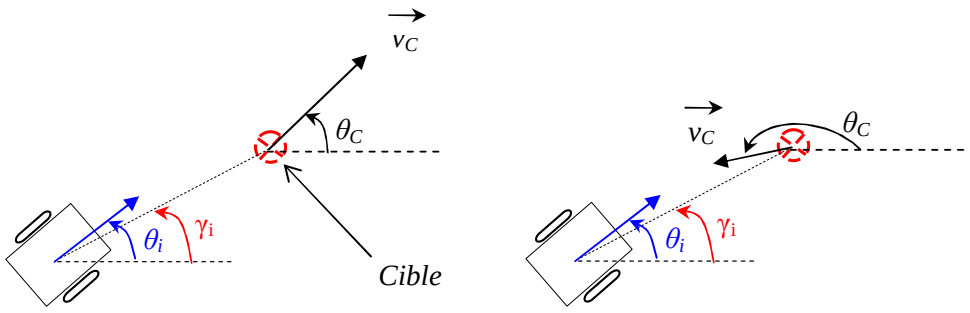
$$\arcsin(x) \in [-\frac{\pi}{2}, \frac{\pi}{2}], \forall x \in [-1, 1]$$

Considérant l'équation 2.15, on distingue deux cas possibles (cf. Figure 2.9)

1. $(\theta_C - \gamma_i) \in [\frac{-\pi}{2}, \frac{\pi}{2}]$ (cible s'éloignant) ;
   on peut écrire

$$\cos(\theta_C - \gamma_i) = \sqrt{1 - (\sin(\theta_C - \gamma_i))^2} \geq 0$$

   ce qui donne



(a) Cible s'éloignant du robot (b) Cible s'approchant du robot

FIGURE 2.9 – Configurations possibles des cibles : s'éloignant ou s'approchant du robot.

$$\begin{aligned} \dot{d}_{S_i} &= v_C\sqrt{1-(\sin(\theta_C-\gamma_i))^2} \\ &\quad -v_i\sqrt{1-(b\sin(\theta_C-\gamma_i))^2} \end{aligned} \tag{2.24}$$

Cependant, tant que le robot n'a pas atteint la cible, alors $b < 1$ (cf. Equation 2.23). Cela signifie

$$v_C\sqrt{1-(\sin(\theta_C-\gamma_i))^2} < v_i\sqrt{1-(b\sin(\theta_C-\gamma_i))^2}$$

(car $\sqrt{1-(\sin(\theta_C-\gamma_i))^2} < \sqrt{1-(b\sin(\theta_C-\gamma_i))^2}$ et $v_C < v_i$)
Il en résulte que

$$\dot{d}_{S_i} < 0$$

Une fois la cible atteinte, nous verrons que $v_i \approx v_C$ et donc

$$d_{S_i} = 0$$
$$\dot{d}_{S_i} = 0$$

2. $(\theta_C - \gamma_i) \in [\frac{\pi}{2}, \frac{3\pi}{2}]$ (cible s'approchant) :
cela signifie

$$\cos(\theta_T - \gamma_i) = -\sqrt{1-(\sin(\theta_T-\gamma_i))^2} \leq 0$$

La variation de d_{S_i} s'écrit alors

$$\begin{aligned} \dot{d}_{S_i} &= -v_C\sqrt{1-(\sin(\theta_C-\gamma_i))^2} \\ &\quad -v_i\sqrt{1-(b\sin(\theta_C-\gamma_i))^2} \end{aligned} \tag{2.25}$$

Il est immédiatement déduit que

$$\dot{d}_{S_i} < 0$$

Notons que dans le premier cas (cible s'éloignant), on observe que $\dot{d}_{S_i}$ est d'autant plus négative que la vitesse du robot est supérieure à celle de la cible. Nous prendrons en compte cette constatation dans l'élaboration de la loi de commande (cf. Section 2.36, page 86).

Le contrôleur d'attraction vers la cible proposé a un autre avantage. En effet, dans la plupart des travaux où le robot suit une cible dynamique, il se dirige d'abord vers cette cible sans prendre en compte sa dynamique tant qu'il ne l'a pas atteinte [Lalish 06], [Lee 00], [Balch 99]. En d'autres termes, il suit d'abord l'angle consigne noté ici γ_i avant d'atteindre la cible. Une fois atteinte, il suit alors la direction de celle-ci. Ceci peut engendrer une discontinuité de consigne et un comportement indésirable du robot. En revanche, l'angle consigne proposé (cf. Equation 2.22) prend en compte les deux grandeurs au même temps. De plus, la loi de commande sera conçue de telle sorte que $v_i \rightarrow v_C$ quand $d_{S_i} \rightarrow 0$ (cf. Section 2.36). Il en résulte dans ce cas que $b \rightarrow 1$ (cf. Equation 2.23). Ainsi, on retrouve les deux cas (au voisinage de $d_{S_i} \rightarrow 0$) :

1. $(\theta_C - \gamma_i) \in [\frac{-\pi}{2}, \frac{\pi}{2}]$ (cible s'éloignant) :

$$\begin{array}{rcl} \theta_{S_{at}} & \rightarrow & \arcsin(\sin(\theta_C - \gamma_i)) + \gamma_i \\ & = & \theta_C - \gamma_i + \gamma_i \\ & = & \theta_C \end{array} \tag{2.26}$$

   On remarque bien que l'angle consigne tend automatiquement vers la direction de la cible décrite par θ_C de façon lisse.

2. $(\theta_C - \gamma_i) \in [\frac{\pi}{2}, \frac{3\pi}{2}[$ (cible s'approchant) :

$$\begin{array}{rcl} \theta_{S_{at}} & \rightarrow & \pi - (\theta_C - \gamma_i) + \gamma_i \\ & = & \pi + 2\gamma_i - \theta_C \end{array} \tag{2.27}$$

Bien que la convergence du robot vers la cible soit prouvée ci-dessus pour le cas 2, on remarque qu'avec cette consigne, le robot dépassera certainement la cible. Dès qu'il la dépasse, le robot recalcule l'angle γ_i afin de la rejoindre de nouveau. Comme le mouvement de la structure virtuelle est non-holonome, le robot retombera dans le cas 1 (cible s'éloignant). Nous verrons ce cas dans la simulation qui suit.

Remarque

Il est à noter Afin d'éviter les singularités relatives au domaine de définition de l'angle γ_i, (le cas où $e_{x_i} = 0$ ou $(e_{x_i}, e_{y_i}) = (0, 0)$), la fonction $atan2$ constituant une variante de la fonction $arctan$ est exploitée. Elle utilise le signe de ses arguments pour déterminer le quadrant du résultat. Elle est définie comme suit

$$\text{atan2}(y,x) = \begin{cases} \arctan(\frac{y}{x}) & x > 0 \\ \pi + \arctan(\frac{y}{x}) & y \geq 0, x < 0 \\ -\pi + \arctan(\frac{y}{x}) & y < 0, x < 0 \\ \frac{\pi}{2} & y > 0, x = 0 \\ \frac{-\pi}{2} & y < 0, x = 0 \\ \text{indéfini} & y = 0, x = 0 \end{cases}$$

L'angle γ_i est alors défini comme suit :

$$\gamma_{\text{i}} = \begin{cases} atan2(e_{x_i}, e_{y_i}) \quad si \qquad e_{x_i} \neq 0 \\ 0 \qquad\qquad\qquad\qquad sinon \end{cases}$$

Nous proposons de simuler le contrôleur d'attraction vers la cible. La trajectoire du robot et de sa cible dynamique à différents instants $t_0, ..., t_5$ ainsi que la variation de la distance qui les sépare d_{S_i} sont représentées dans les figures 2.10 et 2.11 respectivement. Sur cette dernière sont également reportés les instants $t_0, ..., t_5$. La loi de commande (vitesse linéaire et angulaire) utilisée sera exposée dans la section 2.3.3.

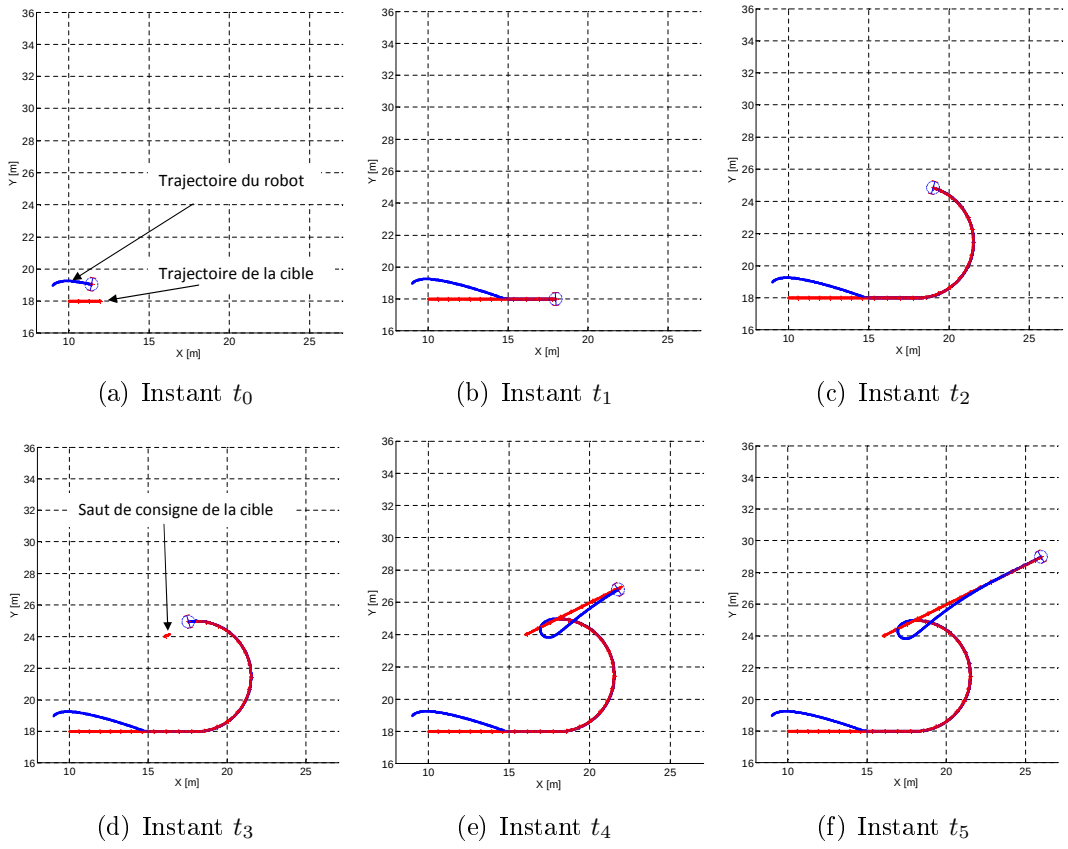
D'abord, le cas d'une cible s'éloignant du robot est illustré. On remarque que le robot s'approche de celle-ci (instant t_0) jusqu'à l'atteindre. Cela peut être confirmé par la distance d_{S_i} qui décroît (cf. Figure 2.11) (la légère croissance de d_{S_i} au début correspond à l'orientation initiale du robot et sa contrainte de non holonomie). Une fois atteinte (instant t_1), le robot continue à la suivre (instants t_1, t_2). On remarque que la distance d_{S_i} est nulle pendant cette évolution.

A partir de l'instant t_3, nous simulons un saut de la consigne. Ceci explique la croissance brusque de d_{S_i}.

On remarque que la nouvelle cible est une cible s'approchant du robot.

Ce dernier l'atteint bien. Toutefois, son orientation vers celle-ci impose qu'il la dépasse. Ceci explique que la distance d_{S_i} s'annule (instant $t = 100$)(cf. Figure 2.11) et croît de nouveau.

On remarque qu'une fois le robot dépasse la cible, celle-ci devient une cible s'éloignant. Le robot corrige alors son orientation et la rejoint de nouveau (instants t_4, t_5).



(a) Instant t_0 (b) Instant t_1 (c) Instant t_2

(d) Instant t_3 (e) Instant t_4 (f) Instant t_5

FIGURE 2.10 – Simulation du contrôleur d'*Attraction vers une Cible Dynamique*.

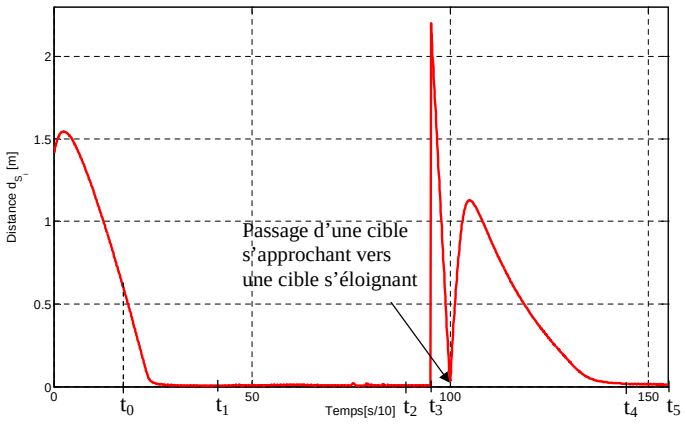


FIGURE 2.11 – Distance d_{S_i} séparant le robot de la cible dynamique.

2.3.2 Evitement d'Obstacles

L'évitement d'obstacles est un comportement de base devant être présent dans tous les robots mobiles. En effet, la caractéristique de mobilité impose que le robot doit pouvoir agir dans un environnement encombré voire dynamique. C'est pourquoi, tous les robots mobiles sont quasiment équipés de capteurs permettant de détecter les obstacles (télémètre laser, ultrasons, infrarouges, etc). La littérature regorge de travaux sur l'évitement d'obstacles dont l'objectif est de garantir le maximum de sécurité aux robots mobiles en cours de navigation.

Parmi les méthodes d'évitement d'obstacles réactives (basées sur la perception locale de l'environnement), la méthode des champs de potentiel est sans doute la plus répandue dans la littérature. L'idée est d'imaginer des forces virtuelles agissant sur le robot [Andrews 83], [Khatib 86] : les obstacles lui exercent des forces répulsives tandis que le point à atteindre ou le chemin à suivre exerce une force attractive. La somme résultante de ces forces virtuelles détermine la direction finale du robot ainsi que sa vitesse de navigation. Cette méthode est si simple à mettre en œuvre que certains travaux de planification de chemin (contrôle plutôt cognitif) s'en inspirent [Krogh 86], [Tanner 01]. Dans [Hussein 02], les équations de Maxwell ont été utilisées pour générer les fonctions de potentiel. De façon plus réactive, les travaux de Brooks [Brooks 85], ou les schémas moteurs de Arkin [Arkin 86] utilisent aussi les champs de potentiel.

Néanmoins, cette méthode présente deux inconvénients majeurs :

1. dans le cas où la somme des forces résultante est nulle, le robot se trouve bloqué dans un minimum local où sa vitesse est nulle. Pour pallier à cet inconvénient, des solutions ont été apportées comme l'introduction de bruits dans la somme des forces appliquées au robot [Slack 93], ou d'ajouter des champs de potentiels circulaires entourant les obstacles. Néanmoins, ces solutions diminuent le problème mais ne le règlent pas complètement. Clark dans [Clark 92] se base sur une mémoire pour signaler les points de minima locaux et les considérer comme des zones répulsives à leur tour mais qui peuvent engendrer de nouveaux minima locaux à leur tour.
2. Comme tout obstacle dans l'environnement génère des forces virtuelles, ces forces vont affecter tout robot se trouvant dans son voisinage. Ainsi, certains obstacles non gênants en réalité, affectent la direction et la vitesse du robot (cf. Figure 2.12).

D'autres comportements indésirables peuvent apparaître, comme des oscillations dues au passage entre plusieurs obstacles ou dans des zones étroites [Koren 91].

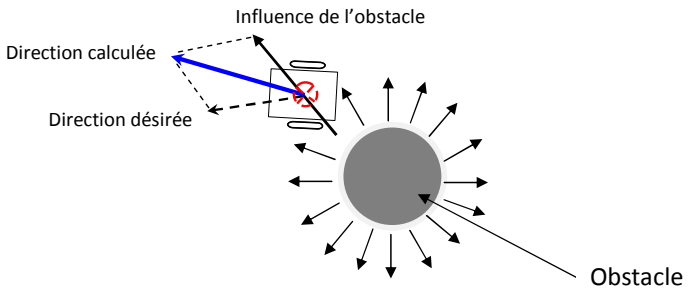
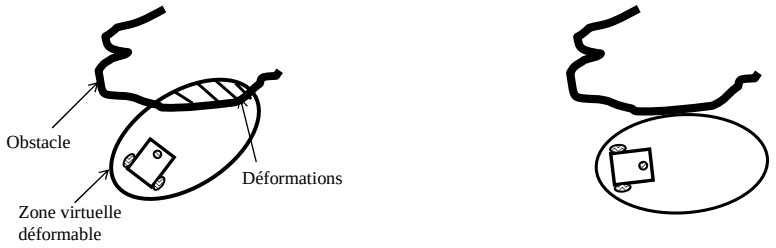


FIGURE 2.12 – Forces virtuelles d'un obstacle affectant la direction du robot.

La méthode *Vector Field Histogram* (VFH) [Koren 91] est venue remplacer celle appelée *Virtual Force Field* (VFF) [Borenstein 89] basée sur les champs de potentiel et rencontrant alors les mêmes problèmes. VFH utilise une grille d'occupation locale construite à partir des capteurs du robot. Chaque cellule de cette grille est associée à un nombre (valeur de certitude) d'autant plus élevé qu'elle a été perçue comme contenant un obstacle. Un histogramme représentant l'occupation de l'environnement autour du robot est reconstruit grâce à cette grille d'occupation. Pour cela, la grille est discrétisée en secteurs angulaires. La somme des valeurs des cellules de chaque secteur est calculée. Tous les secteurs dont la somme est inférieure à un seuil déterminé constituent des directions tolérées pour le robot. Cette méthode a été amélioré et appelée VFH+ [Ulrich 98] prenant en compte les dimensions du robot. Elle semble donner de meilleurs résultats car elle minimise les problèmes d'oscillations et de minima locaux générés par les champs de potentiel. Cependant, elle ne prend pas en compte la distance aux obstacles. Ainsi, un secteur avec un obstacle proche et un autre secteur contenant des obstacles plus loin peuvent afficher le même score (somme des valeurs associées aux cellules du secteur). De plus, cette méthode trouve ses limitations avec les obstacles de forme U, où tous les secteurs risquent d'avoir des scores identiques. Le calcul des valeurs de certitude associées aux cellules peut aussi s'avérer gourmand en mémoire ce qui peut altérer l'aspect réactif de la réponse en temps réel.

La méthode de la zone virtuelle déformable ZVD [Zapata 94] est aussi très efficace. De plus, elle convient à n'importe quelle forme d'obstacles. Il s'agit d'imaginer le robot enveloppé dans une zone déformable le protégeant grâce aux capteurs de proximité (cf. Figure 2.13). Pour un robot mobile, cette enveloppe virtuelle est paramétrable en fonction des vitesses du robot et surtout des informations fournies par ses capteurs sur l'environnement. Lorsqu'un obstacle se trouve dans cette zone, il provoque sa déformation. L'objectif du contrôle du robot est d'alors de minimiser cette déformation, ce qui revient à éviter l'obstacle gênant. Cette méthode est une façon de contrôle distribué assurant à chaque entité du SMR

une navigation sans risque de collision [Gil Pinto 05]. Cependant, elle a aussi ses minima locaux correspondant à la symétrie de la zone virtuelle [Zapata 99].



(a) Instant t (détection de l'obstacle)

(b) Instant t + Δt (évitement de l'obstacle en minimisant la déformation)

FIGURE 2.13 – Utilisation de la zone virtuelle déformable pour éviter les obstacles.

Il existe d'autres approches basées sur des optimisations contraintes. Nous en citons Curvature Velocity Method [Simmons 96], Lane Curvature Method [Ko 98], dynamic window [Fox 97], [Ogren 05] etc. Le principe général de ces méthodes est de sélectionner un couple de vitesses linéaire et angulaire (v, ω) qui répond à différentes contraintes dont celle de l'évitement d'obstacles. Un tel couple de vitesses produit une trajectoire pour laquelle la satisfaction des différentes contraintes est évaluée. A l'issue de l'évaluation de tous les couples de vitesses possibles, le plus pertinent est choisi. Cependant, comme les couples de vitesse prennent en considération plusieurs contraintes (éviter l'obstacle, atteindre le point désiré, etc.), il devient difficile de prédire le comportement global du robot et s'il accomplit tous les objectifs correctement.

Le contrôleur d'évitement d'obstacles utilisé dans nos travaux est inspiré des méthodes de trajectoires décrites par des équations différentielles [Stuart 96]. Parmi elles, on trouve celle du cycle-limite proposée par Kim [Kim 03a] et améliorée dans [Adouane 09]. En plus de l'évitement d'obstacles, elle permet de définir une distance que le robot garde avec l'obstacle durant cette phase, ce qui offre l'assurance de ne pas trop s'en approcher. Aussi, connaissant la position de sa cible, le robot peut choisir le coté d'évitement optimal lui permettant de l'atteindre plus rapidement.

Dans ce qui suit, et après une brève introduction des travaux réalisés dans [Adouane 09] (cf. Annexe A), nous proposons d'apporter une contribution relative à l'évitement d'obstacles dynamiques.

2.3.2.1 Méthode du cycle-limite

Pour accomplir la tâche d'évitement, il est supposé que le robot et l'obstacle à éviter sont entourés de contours cylindriques de rayons R_R et R_O respectivement.

Grâce à ses capteurs, le robot peut percevoir ou calculer plusieurs informations dont il a besoin (cf. Figure 2.14) :

- D_{RO_i} représente la distance entre le robot et le centre de l'obstacle i,
- D_{lO_i} est la distance séparant le centre de l'obstacle de la droite (l) qui passe par la cible et le centre du robot,
- D_{CO_i} est la distance entre la cible et le centre de l'obstacle.

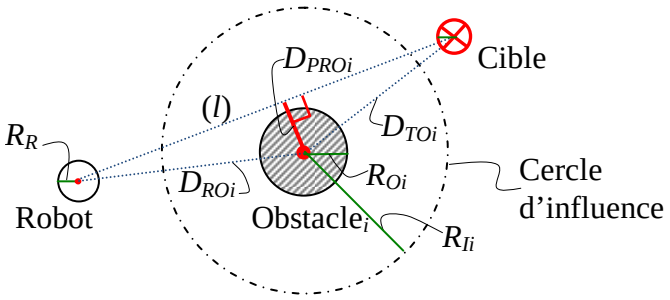


FIGURE 2.14 – Grandeurs nécessaires à l'évitement de l'obstacle i.

Chaque obstacle détecté est entouré d'un cercle d'influence (cf. Figure 2.14) avec un rayon $R_{I_i} = R_R + R_{O_i} + Marge$. La grandeur $Marge$ considère une tolérance relative aux incertitudes des capteurs, une sécurité supplémentaire, etc.

Pour accomplir la tâche d'évitement d'obstacles, le robot doit suivre le cycle-limite décrit par le système d'équations différentielles suivant :

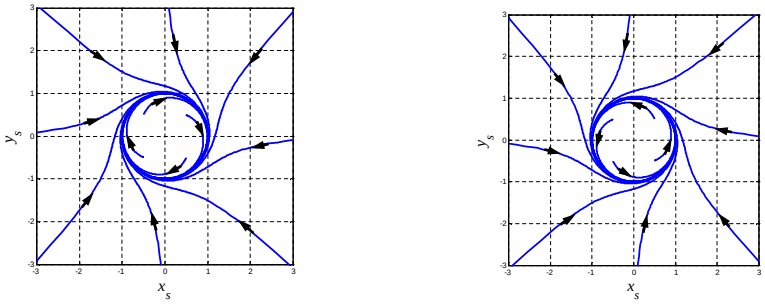
- pour un évitement dans le sens anti-trigonométrique (cf. Figure 2.15(a)) :

$$\begin{aligned} \dot{x}_s &= y_s + x_s(R_c^2 - x_s^2 - y_s^2) \\ \dot{y}_s &= -x_s + y_s(R_c^2 - x_s^2 - y_s^2) \end{aligned} \tag{2.28}$$

- un évitement dans le sens trigonométrique (cf. Figure 2.15(b)) :

$$\begin{aligned} \dot{x}_s &= -y_s + x_s(R_c^2 - x_s^2 - y_s^2) \\ \dot{y}_s &= x_s + y_s(R_c^2 - x_s^2 - y_s^2) \end{aligned} \tag{2.29}$$

Où (x_s, y_s) correspond à la position relative du robot par rapport au centre d'un obstacle de rayon R_c. La figure 2.15 montre que les systèmes 2.28 et 2.29 représentent des orbites. Ce sont ces orbites que l'on appelle les cycles-limites. On remarque alors que toutes les trajectoires convergent sur le cercle de rayon



(a) Anti-trigonométrique (b) Trigonométrique

FIGURE 2.15 – Sens possibles pour parcourir un cycle-limite

R_c indépendamment de la position initiale du robot (qu'elle soit à l'intérieur ou à l'extérieur du cercle). L'angle consigne que le robot doit suivre pour éviter l'obstacle est donné par

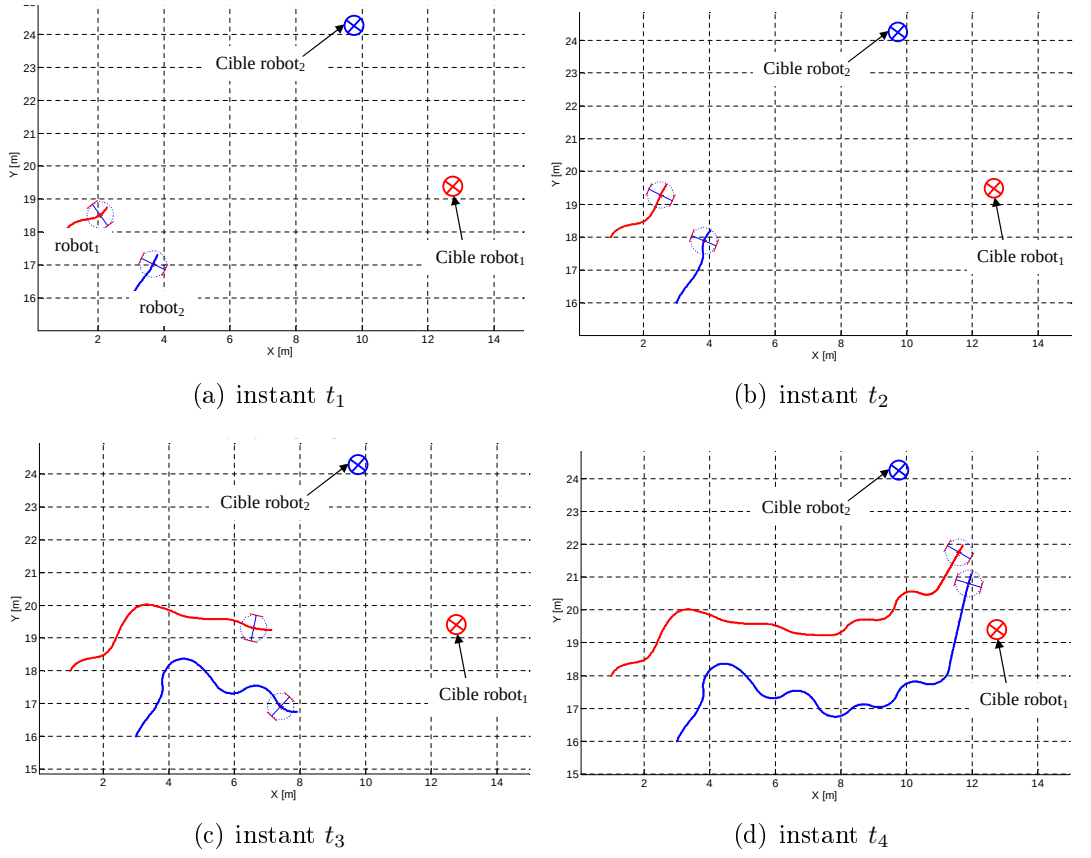
$$\theta_{S_{oa}} = \arctan(\frac{\dot{y}_s}{\dot{x}_s}) \tag{2.30}$$

Afin de se concentrer sur les contributions apportées dans cette thèse, les détails de l'algorithme d'évitement d'obstacles par la méthode du cycle-limite [Adouane 09] sont donnés dans l'annexe A. Les caractéristiques principales de cet algorithme sont la réduction du risque d'oscillations lors du contournement de l'obstacle, le choix automatique de l'obstacle le plus dangereux, et sa capacité à éviter les minima locaux (notamment les obstacles de type U). Il prend en compte la position relative du robot par rapport à l'obstacle et à la cible pour déterminer le sens d'évitement (trigonométrique ou anti-trigonométrique).

2.3.2.2 Et pour les obstacles dynamiques ?

Les entités robotiques de tout SMR doivent aussi être capables de gérer les risques de collision entre elles et avec les obstacles dynamiques de l'environnement (autres robots étrangers au SMR, etc.). Tout robot considère alors les autres comme étant potentiellement gênants et il sera souhaitable d'utiliser le même contrôleur d'évitement d'obstacles (basé sur les cycles-limites). Bien que l'algorithme présenté dans [Adouane 09] n'impose pas que l'obstacle à éviter soit statique et le robot peut suivre un cycle limite dynamique, il subsiste une situation de conflit propre aux obstacles en mouvement. Par exemple, si les deux robots se dirigent l'un vers l'autre, et que leurs cibles respectives font que l'un évite l'autre dans le sens trigonométrique tandis que le deuxième l'évite dans le sens anti-trigonométrique, un conflit va apparaître. En effet, l'algorithme 5 donné

en annexe A impose que si le robot commence l'évitement dans un sens, il continue dans ce sens tant que le contrôleur d'évitement d'obstacles est actif. Les deux robots chercheront alors à s'éviter indéfiniment voire vainement sans rejoindre leurs cibles. La figure 2.16 illustre un exemple de deux robots voulant atteindre leurs cibles (statiques) respectives. On remarque qu'ils n'y arrivent pas car ils cherchent à s'éviter dans deux sens opposés.



(a) instant t_1 (b) instant t_2

(c) instant t_3 (d) instant t_4

FIGURE 2.16 – Evitement d'obstacles dynamiques utilisant l'algorithme d'évitement d'obstacles présenté dans [Adouane 09].

Nous proposons dans ce qui suit de résoudre ce conflit. Utilisant le minimum d'information afin de garder l'aspect réactif du contrôle, l'idée est de projeter le vecteur vitesse de l'obstacle sur l'axe (Y_{O_i}) du repère qui lui est associé (O_i, X_{O_i}, Y_{O_i}) (cf. Figure 2.17). Ainsi, si la projection est positive, l'obstacle est évité dans le sens trigonométrique, sinon il est évité dans le sens inverse (anti-trigonométrique).

Pour cela, nous considèrons le vecteur vitesse de l'obstacle dont les composantes dans son repère (O_i, X_{O_i}, Y_{O_i}) sont telles que $\vec{v}_{O_i}\left(v_{O_{ix}}, v_{O_{iy}}\right)$. $\vec{v}_{O_i}\left(v_{O_{ix}}, v_{O_{iy}}\right)$

s'exprime par

$$v_{O_{ix}} = v_{O_i}.\cos(\varphi_i - \alpha_i) \quad (2.31a)$$
$$v_{O_{iy}} = v_{O_i}.\sin(\varphi_i - \alpha_i) \quad (2.31b)$$

L'algorithme que nous proposons prend alors en compte le signe de $v_{O_{iy}}$ plutôt que celui de l'ordonnée du robot dans le repère de l'obstacle y_O (détail sur l'incidence du signe de y_O sur le sens d'évitement en annexe A).

avec φ_i est l'orientation de l'obstacle (cf. Figure 2.17).

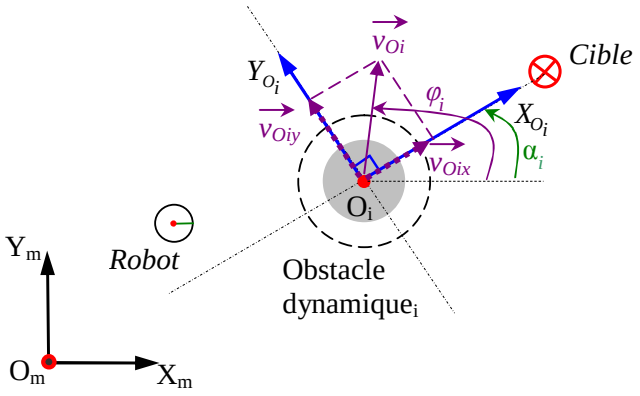


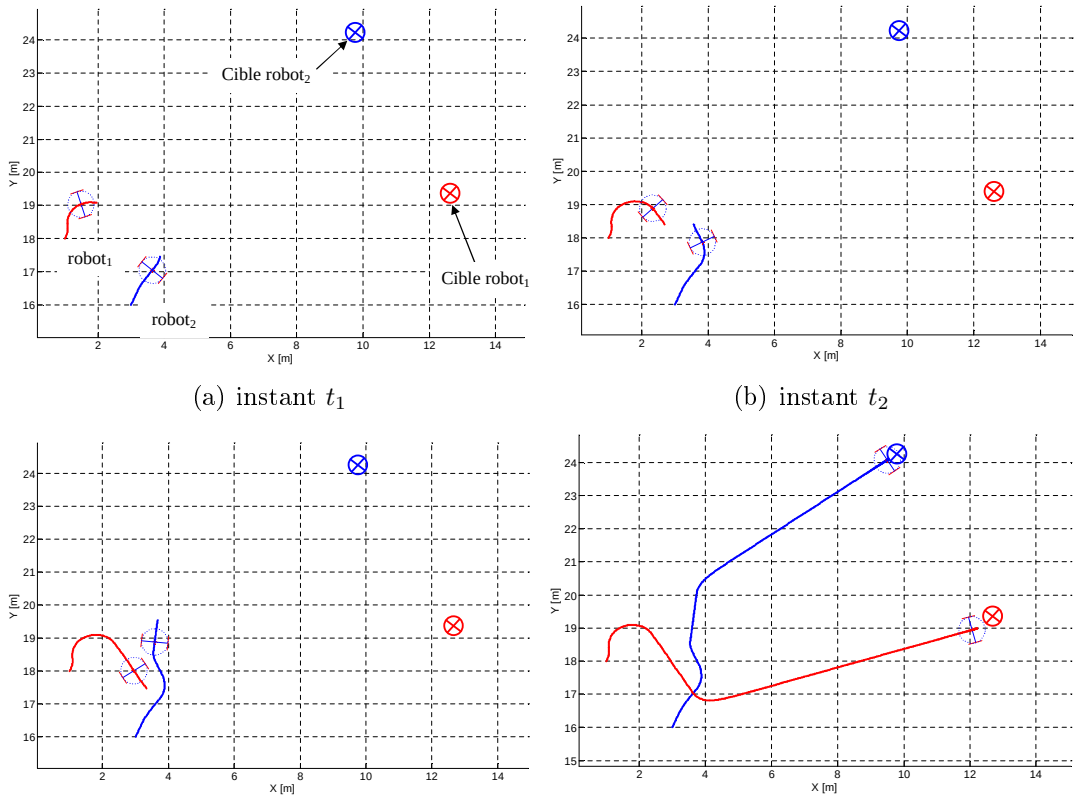
FIGURE 2.17 – Projection du vecteur vitesse de l'obstacle dans le repère relatif (O_i, X_{O_i}, Y_{O_i}).

Le signe de $v_{O_{iy}}$ est pris en compte dans les équations du cycle limite décrites dans l'algorithme 5 (ligne 9) qui deviennent

$$\dot{x}_s = signe(v_{O_{iy}})y_s + x_s(R_c^2 - x_s^2 - y_s^2) \quad (2.32a)$$
$$\dot{y}_s = -signe(v_{O_{iy}})x_s + y_s(R_c^2 - x_s^2 - y_s^2) \quad (2.32b)$$

avec $signe(v_{O_{iy}}) = 1$ si $v_{O_{iy}} > 0$ et $signe(v_{O_{iy}}) = -1$ si $v_{O_{iy}} < 0$. Si $v_{O_{iy}} = 0$, alors, l'algorithme proposé dans [Adouane 09] est alors appliqué (raisonnement sur le signe de y_O) et atteindre rapidement la cible peut redevenir la priorité. Ainsi, l'idée est que le robot évite l'obstacle dynamique sans couper sa trajectoire. Cette dernière est prédite sans complexité de calcul, et uniquement grâce à son vecteur vitesse.

La figure 2.18 montre que grâce à l'utilisation des équations 2.32 dans l'algorithme 5, la situation de conflit est bien évitée.



(a) instant t_1 (b) instant t_2

(c) instant t_3 (d) instant t_4

FIGURE 2.18 – Évitement de collision entre deux robots mobiles.

En effet, plutôt que de s'intéresser uniquement à la position de sa cible, la projection de la vitesse linéaire de l'obstacle $v_{O_{iy}}$ permet aux deux robots de déduire le sens d'évitement optimal qui résout le conflit pouvant être créé dans ce genre de situations. Notons que les mêmes conditions initiales qu'en figure 2.16 sont considérées.

On peut aussi observer le problème où les deux robots (ou plus) cherchent à s'éviter indéfiniment (cf. Figure 2.16) même avec l'idée de projection des vecteurs vitesses proposée. En effet, comme les entités d'un même système utilisent le même algorithme, chacun va projeter le vecteur vitesse de l'autre. Dans le cas où les signes de projection de leurs vecteurs vitesses respectifs sont opposés, ils vont s'éviter dans deux sens différents et risquent de tomber dans ce conflit. Pour résoudre ce problème, nous proposons que les entités robotiques s'évitent entre elles toujours dans un seul sens (prendre le sens trigonométrique par exemple) et reproduire alors le principe du rond-point connu dans le code de la route humain.

Par ailleurs, la projection du vecteur vitesse sera toujours très utile et utilisée

pour des obstacles dynamiques extérieurs au SMR.

Contrairement aux obstacles statiques dont la vitesse est nulle, il se peut que les deux robots s'approchent trop vite l'un de l'autre (lors de la phase d'attraction par exemple, ou alors s'ils s'approchent avec des directions opposées en se déplaçant à vitesses élevées, etc.). C'est pourquoi, nous proposons dans le paragraphe suivant que la vitesse linéaire de chaque robot soit affectée par les robots susceptibles de le gêner.

2.3.2.3 Vers un risque de collision nul

Pour éloigner tout risque de collision entre les robots, nous proposons de pénaliser la vitesse linéaire de chacun en prenant en compte la distance qui le sépare des robots susceptibles de le gêner dans sa progression.

Convention 1. *On considère qu'un robot j est susceptible d'être gênant pour i si j est tel que $D_{lR_iR_j} \leq R_{I_j}$. (La grandeur $D_{lR_iR_j}$ correspond à D_{lRO_j} (cf. Figure 2.14) si l'obstacle considéré est un robot j).*

Ainsi, nous définissons une fonction que nous appellons une fonction de pénalité. Elle est relative à j, dépendant de la distance d_{ij} séparant i et j, et affecte la vitesse linéaire de i. Nous la notons $\psi_j(d_{ij})$. Cette fonction a un impact sur la vitesse v_i à condition que j se trouve dans un cercle virtuel extérieur centré sur i et de rayon R_{ext} entourant le robot i (cf. Figure 2.19). Elle l'affecte d'autant plus que j s'enfonce dans ce cercle si bien que s'il s'approche de façon critique et que la distance d_{ij} est inférieure à un rayon virtuel intérieur R_{int}, le robot i s'arrête.

D'après ces contraintes, la fonction de pénalité $\psi_j(d_{ij})$ affectant la vitesse du robot i aura pour expression

$$\psi_{\mathrm{j}}(d_{ij}) = \begin{cases} 1 & d_{ij} \geq R_{ext} \\ (d_{ij} - R_{int_i})/(R_{ext} - R_{int_i}) & R_{int_i} < d_{ij} < R_{ext} \\ 0 & d_{ij} \leq R_{int_i} \end{cases} \tag{2.33}$$

La vitesse linéaire du robot (cf. Section 2.3.3) devient donc

$$v_i^{'} = v_i \psi_j(d_{ij}) \tag{2.34}$$

En généralisant, et s'il y a M robots susceptibles d'être gênants pour le robot i, sa vitesse linéaire s'exprime alors

$$v_i^{'} = v_i \prod_{j=1, j\neq i}^{M} \psi_j(d_{ij}) \tag{2.35}$$

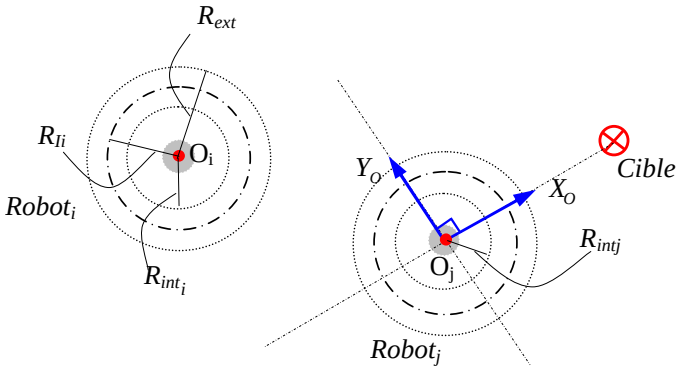


FIGURE 2.19 – Distances déterminant la variation de la fonction de pénalité.

Il est alors possible d'ajouter un bloc à l'architecture de contrôle proposée (cf. Figure 2.7) venant après celui de la loi de commande et correspond à cette fonction de pénalité (cf. Figure 2.20).

FIGURE 2.20 – Introduction de la fonction de pénalité à l'architecture de contrôle.

Il est important de noter que le robot j affecte le robot i et vice versa. Ainsi, une fonction de pénalité ψ_i peut aussi affecter j. Un minimum local correspondant à

$$\psi_i = \psi_j = 0$$

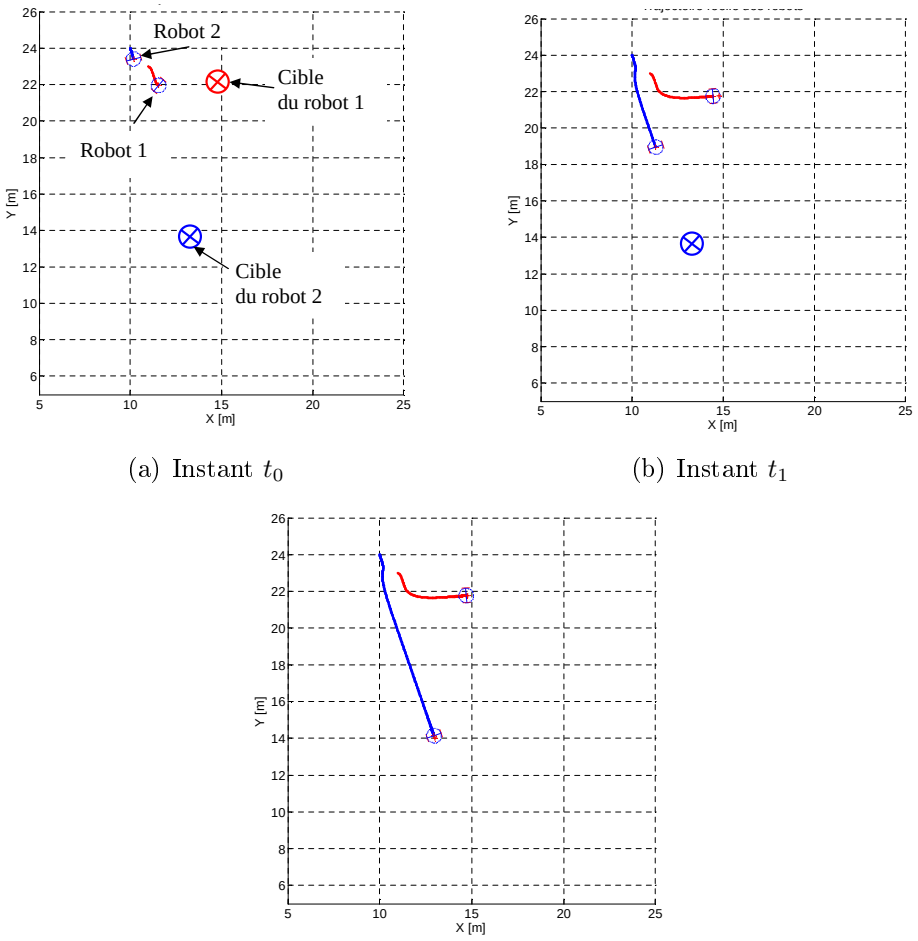
entraînera alors l'arrêt simultané des deux robots. Afin de pallier à cet inconvénient, l'idée est de définir les distances critiques $R_{int_1}, R_{int_2}, ..., R_{int_N}$ telles que

$$R_{int_1} \neq R_{int_2} \neq ... \neq R_{int_N}$$

Avec cette contrainte, l'ensemble des robots ne s'arrêtent jamais simultanément. Aussi, si l'un des deux s'arrête, il devient un obstacle statique et est évité selon l'algorithme 5. Il va de soi que la différence entre ces distances critiques doit prendre en considération l'incertitude des capteurs. Ainsi, pour deux robots i et j, il est fondamental d'avoir $|R_{int_i} - R_{int_j}| \geq \xi$ où ξ est l'incertitude des capteurs des robots. Grâce à la définition de la fonction de pénalité, il redémarre dès que

le robot l'affectant s'éloigne de lui. Par contre, tous les robots peuvent avoir le même rayon R_{ext} sans créer de minima locaux.

La simulation de deux robots mobiles devant se déplacer vers leurs cibles respectives est considérée dans la figure (cf. Figure 2.21). Les conditions initiales sont telles que le robot 1 est mis devant le robot 2. De cette façon, le robot 1 n'est pas gêné pour atteindre sa cible. A l'instant t_0, la distance qui sépare les deux robots d_{12} est déjà telle que $d_{12} < R_{ext}$ (cf. Figure 2.22). Cependant, comme le robot 1 n'est pas gêné, il ne décélère pas inutilement. Seul le robot 2 décélère en évitant le robot 1. Ce dernier est arrivé à sa cible à l'instant t_1 tandis que le robot 2 accélère de nouveau. A l'instant t_2, il arrive aussi en décélérant jusqu'à l'arrêt.



(a) Instant t_0

(b) Instant t_1

(c) Instant t_2

FIGURE 2.21 – Introduction de la fonction de pénalité : seul le robot 2 est gêné par le robot 1.

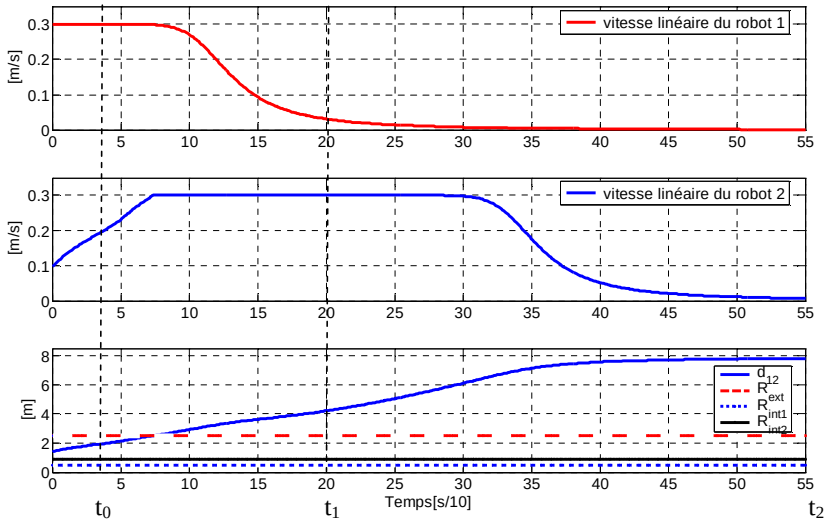
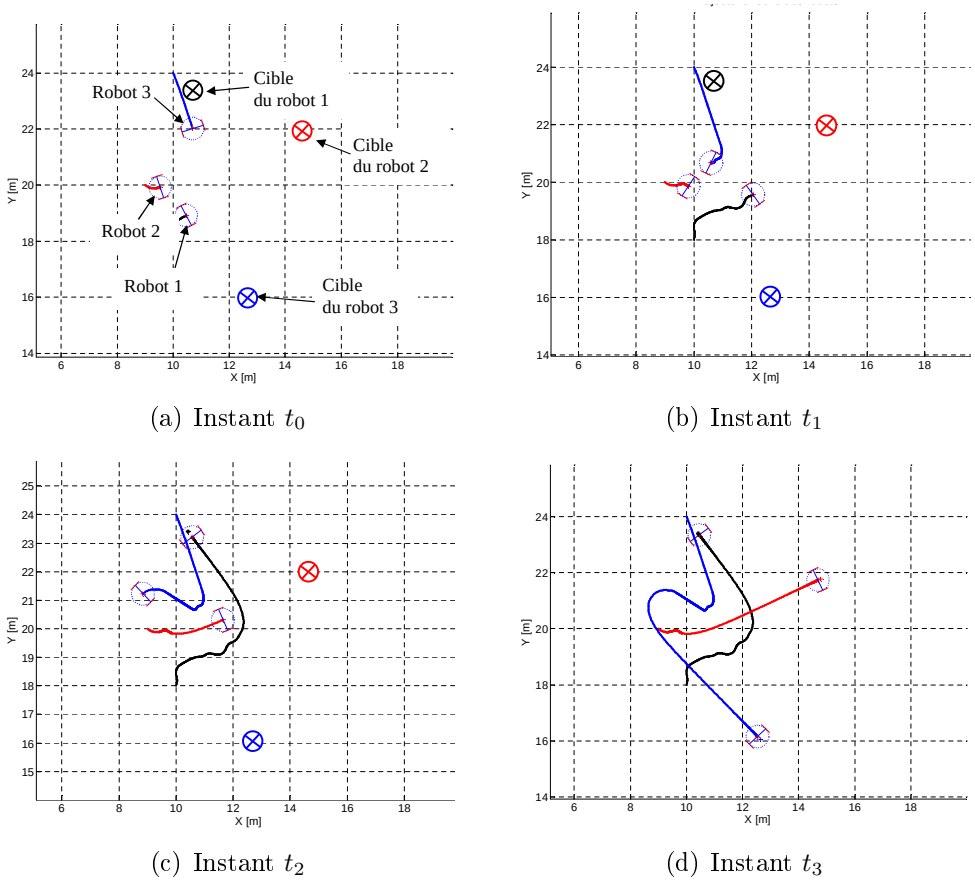


FIGURE 2.22 – Évolution des vitesses linéaires des robots et de la distance d_{12} qui les sépare : seul le robot 2 est gêné par le robot 1.

Maintenant, nous proposons de généraliser la simulation à un système à trois robots. Chacun doit atteindre une cible statique. Les conditions initiales sont choisies de telle sorte que chaque robot est gêné par les deux autres avant d'atteindre sa cible. Les trajectoires des robots sont représentées dans la figure 2.23. Nous représentons également les trois vitesses linéaires ainsi que les distances séparant chaque entité robotique des deux autres dans les figures 2.24(a) et 2.24(b) respectivement.

On remarque que chaque robot évite les deux autres et atteint sa cible avec succès. Ainsi, le robot 1 décélère d'abord car il est gêné par le robot 2. La distance d_{12} comprise entre R_{int1} et R_{ext} le confirme (cf. Figure 2.24(b)). Sa vitesse est ensuite affectée par le robot 3 ce qui explique qu'il continue de décélérer (la distance d_{13} est comprise entre R_{int1} et R_{ext}). Une fois qu'il a évité les deux robots et qu'ils ne sont plus gênants pour lui, il accélère de nouveau et s'arrête progressivement en s'approchant de sa cible jusqu'à ce qu'il s'arrête sur celle-ci.

De même, affecté d'abord par le robot 1, le robot 2 décélère aussi (voir distance d_{12} dans la figure 2.24(b)). Sa vitesse est ensuite affectée par le robot 3 (distance d_{23}). Cette distance diminue jusqu'à ce qu'elle soit $d_{23} < R_{int2}$ ce qui explique que la vitesse du robot 2 s'annule. Comme $R_{int1} \neq R_{int2} \neq R_{int3}$, les robots 1 et 3 ne s'arrêtent pas et continuent d'éviter le robot 2 en tant qu'obstacle statique. Dès que les deux robots s'éloignent, le robot 2 redémarre progressivement (l'accélération linéaire des robots est fixée à $1.2m/s^2$). Il décélère de nouveau en s'approchant de sa cible.



(a) Instant t_0 (b) Instant t_1

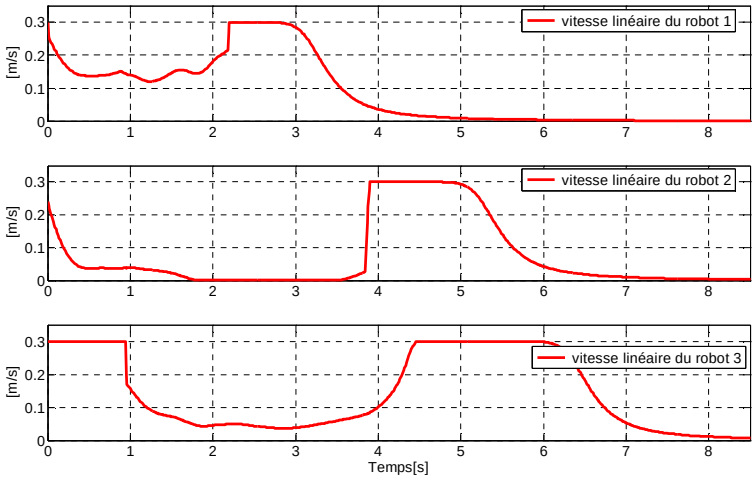
(c) Instant t_2 (d) Instant t_3

FIGURE 2.23 – Introduction de la fonction de pénalité : cas de trois robots.

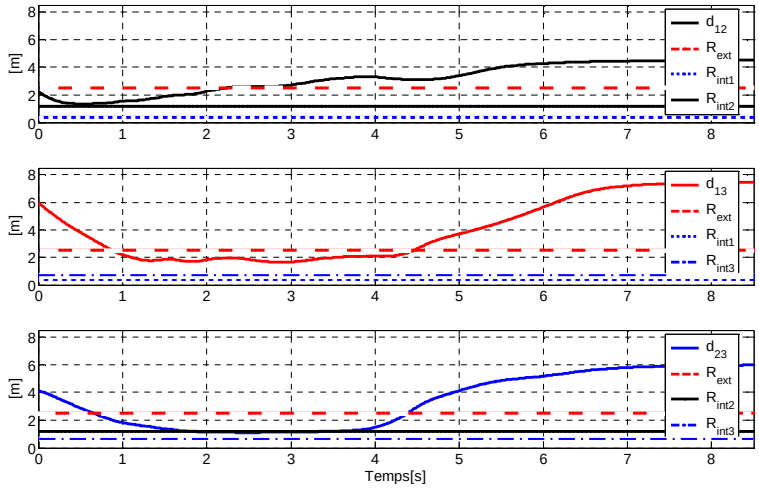
Le même raisonnement peut être mené sur le robot 3. Affecté d'abord par le robot 2, et ensuite le robot 1, il décélère. Il n'accélère qu'une fois loin des deux robots. Il réussit aussi à atteindre sa cible.

Grâce à l'idée des fonctions de pénalité, nous permettons d'augmenter la robustesse du contrôle du SMR face aux risques de collision entre entités robotiques. Il est aussi possible de généraliser à d'autres obstacles dynamiques relatifs à l'environnement sans aucune augmentation de complexité de calcul comme il est souvent le cas dans la littérature. En effet, dans [Reif 99], [Gulec 05], [MacArthur 07], le système ressort-amortisseur virtuel est imaginé entre les entités robotiques. Cependant, il n'y a pas d'adaptation aux environnements encombrés et seulement la collision entre entités est évitée. Ceci laisse à penser que la généralisation serait difficile. Dans [Ha 05], les robots voisins et présentant des risques de collision deviennent des robots guides (comme dans l'approche hiérarchique vue dans le chapitre 1). Cependant, seul le cas de deux robots voisins est considéré, et

le robot ne suit plus le véritable guide de la formation, ce qui peut détourner sa trajectoire. D'autres travaux utilisent les fonctions de potentiel (cf. Section 2.3.2) en fusionnant maintien de formation et évitement de collision [Leonard 01], [Mastellone 07], [Fazenda 07], [Urcola 08].



(a) Évolution des vitesses linéaires des robots



(b) Évolution des distances séparant les trois robots

FIGURE 2.24 – Évolution des vitesses linéaires et des distances séparant les robots.

Nous nous intéressons dans ce qui suit à un autre bloc important de l'architecture de contrôle proposée (cf. Figure 2.7, page 62). Il s'agit du bloc *Loi de commande*.

2.3.3 Loi de commande proposée

La loi de commande proposée (bloc Loi de commande) permet au robot de converger vers la consigne $(P_{S_i}, \tilde{\theta}_{S_i})$ calculée par les contrôleurs. Elle est exprimée comme suit

$$v_i = v_{max} - (v_{max} - v_C)e^{-(d_{S_i}^2/\sigma^2)} \tag{2.36a}$$

$$\omega_i = \omega_{S_i} + k\tilde{\theta}_i \tag{2.36b}$$

où
- v_i et ω_i sont les vitesses linéaire et angulaire respectivement,
- d_{S_i} est la distance séparant le robot de la cible (cf. Figure 2.8, page 64),
- v_{max} est la vitesse linéaire maximale du robot,
- v_C est la vitesse linéaire de la cible,
- $\omega_{S_i} = \dot{\theta}_{S_i}$, est la variation de l'angle consigne (qu'il provienne de l'attraction vers la cible ou de l'évitement d'obstacles),
- σ, k sont des constantes positives.
- $\tilde{\theta}_i$ représente l'erreur d'orientation par rapport à l'angle consigne θ_{S_i} (ce dernier est calculé par le contrôleur actif : celui d'attraction vers la cible ou de l'évitement d'obstacles). Nous avons donc $\tilde{\theta}_i = \theta_{S_i} - \theta_i$.

La loi de commande est élaborée de telle sorte que la vitesse linéaire proposée prend en compte la vitesse de la cible ainsi que la distance qui la sépare du robot. Ainsi, le robot doit accélérer sans pour autant dépasser la vitesse maximale autorisée (définie par ses contraintes structurelles) afin d'atteindre la cible. En convergeant vers cette dernière, il décélère jusqu'à ce que sa vitesse tende vers celle de la cible. Naturellement, afin de pouvoir atteindre et suivre sa cible, il est imposé

$$v_C < v_{max}$$

En effet, dans le cas d'une cible s'éloignant du robot (cf. Figure 2.9(a), page 68), ce dernier doit toujours pouvoir aller plus vite qu'elle afin de la rejoindre. La variation de la vitesse linéaire du robot en fonction de la distance qui le sépare de sa cible d_{S_i} est représentée dans la figure 2.25.

On remarque qu'un grand d_{S_i} entraîne la croissance de la vitesse jusqu'à la vitesse maximale. Cependant, quand $d_{S_i} \to 0$, alors $v_i \to v_C$. On remarque aussi que la vitesse croît plus ou moins rapidement en fonction du gain σ. Le choix de ce dernier est donc important. En effet, plus σ est grand, moins la vitesse est sensible à la variation de la distance d_{S_i}.

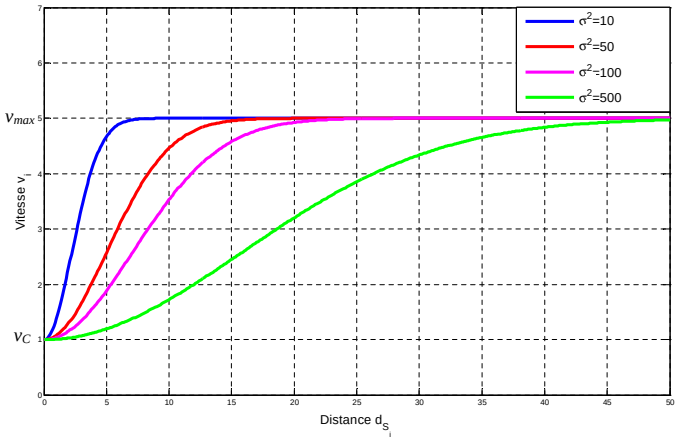


FIGURE 2.25 – Variation de la vitesse linéaire en fonction de la distance d_{S_i} et du gain σ.

Pour la vitesse angulaire, ce choix permet d'avoir une décroissance exponentielle de l'erreur $\tilde{\theta}_i$. En effet, cette erreur s'exprime par

$$\tilde{\theta}_i = \theta_{S_i} - \theta_i \tag{2.37}$$

(avec θ_{S_i} et θ_i sont l'angle consigne (provenant de l'un des contrôleurs), et le cap du robot respectivement).

En dérivant

$$\dot{\tilde{\theta}}_i = \omega_{S_i} - \omega_i \tag{2.38}$$

En remplaçant dans la loi de commande (cf. Equation 2.36b), on retrouve l'équation du premier ordre

$$\dot{\tilde{\theta}}_i = -k\tilde{\theta}_i$$

dont la solution est

$$\tilde{\theta}_i = \tilde{\theta}_i(t_0)e^{-kt}$$

où $\tilde{\theta}_i(t_0)$ représente l'erreur initiale à l'instant t_0. De même que σ, le choix du gain k affecte la vitesse de convergence de l'erreur d'orientation. L'idéal est donc d'augmenter le gain k au maximum afin que l'erreur d'orientation converge rapidement vers 0. En pratique, le gain k doit être borné du fait des contraintes

structurelles du robot et sera discuté dans le chapitre 3.
De même, ω_{S_i} sera discuté dans le même chapitre pour les deux contrôleurs utilisés (*Attraction vers une Cible Dynamique* et *Évitement d'Obstacles*). En effet, pour le contrôleur d'attraction vers la cible par exemple, il ne suffit pas d'imposer des contraintes sur la vitesse linéaire de celle-ci, il faudra aussi contraindre sa vitesse angulaire.

Le domaine de stabilité de cette loi de commande sera alors évoquée plus en détail dans le chapitre 3. La valeur du gain k sera aussi discutée.

En prenant en compte les fonctions de pénalité ψ_j traitant le risque de collision (cf. Section 2.3.2.3), la loi de commande proposée (cf. Equation 2.36) devient alors

$$v_i = (v_{max} - (v_{max} - v_C)e^{-(d_{S_i}^2/\sigma^2)}) \prod_{j=1, j\neq i}^{M} \psi_j(d_{ij}) \tag{2.39a}$$

$$\omega_i = \omega_{S_i} + k\tilde{\theta}_i \tag{2.39b}$$

Maintenant que la loi de commande est expliquée, le paragraphe suivant est dédié au bloc *Sélection d'Action Hiérarchique* (cf. Figure 2.7, page 62).

2.3.4 Sélection d'action hiérarchique

Ce bloc est responsable de la commutation entre le contrôleur d'*Attraction vers une Cible Dynamique*, et celui d'*Evitement d'Obstacles*. L'activation du contrôleur d'évitement survient si le robot entre dans le cercle d'influence de l'obstacle à éviter (cf. Figure 2.14, page 75) ou en d'autres termes, si $D_{RO_i} \leq R_{I_i}$.

Ce bloc exécute alors l'algorithme 1. Dans la plupart des architectures de contrôle comportementales, chaque contrôleur calcule les vitesses correspondantes, et il existe autant de lois de commandes que de tâches élémentaires [Balch 99], [Brooks 85], etc. Dans l'architecture de contrôle proposée, activer un contrôleur signifie que le bloc *Sélection d'action hiérarchique* choisit les consignes relatives à ce contrôleur. En fonction du contrôleur choisi, la consigne (P_{S_i}, θ_{S_i}) correspondante est attribuée à un seul bloc *Loi de commande* qui calcule les vitesses adéquates (cf. Section 2.3.3). Ainsi, selon les contrôleurs

1. *Attraction vers une Cible Dynamique*
   - $(P_{S_{at}} = d_{S_i})$ est donc la distance qui sépare le robot de la cible dynamique,
   - $(\theta_{S_i} = \theta_{S_{at}})$ (cf. Equation 2.22).
2. *Évitement d'Obstacles*

- $(P_{S_{oa}} = 0)$,
- $(\theta_{S_i} = \theta_{S_{oa}})$ (cf. Equation 2.30).

Le choix de $(P_{S_{oa}} = 0)$ impose que $v_i = v_C$ (cf. Equation 2.39a). En effet, nous proposons que le robot se déplace au plus à la même vitesse linéaire que sa cible pendant l'évitement de l'obstacles (les fonctions de pénalité peuvent aussi l'affecter). Comme le robot doit contourner ce dernier, on suppose qu'il parcourt plus de distance que sa cible dynamique et, à vitesses égales, il ne la dépasse pas. En effet, dans le cas contraire (où il la dépasse), il est obligé de revenir vers celle-ci une fois l'évitement achevé. Bien que le contrôleur d'attraction vers la cible proposée permette cette tâche, il est préférable d' enlever -ou du moins diminuer- ce cas de figure. Une fois l'évitement achevé, il accélère de nouveau grâce au contrôleur d'*Attraction vers une Cible Dynamique*.

1: **Si** $D_{RO_i} \leq R_{I_i}$ **alors**
2: Activer le contrôleur d'*Evitement d'Obstacles*
3: **Sinon**
4: Activer le contrôleur d'*Attraction vers une Cible Dynamique*
5: **finSi**

Algorithm 1: Sélection d'action hiérarchique.

2.3.5 Principe de coopération entre les robots (allocation dynamique des cibles)

D'après ce qui précède, chaque entité robotique aura « simplement » à suivre une cible dynamique virtuelle (ou leader virtuel dans certains travaux [Ogren 02], [Lalish 06]) en évitant des obstacles. On pourrait dire qu'en accomplissant cette tâche, il y a bien un maintien de formation, mais il n'y aucune coopération explicite entre les robots. En effet, chaque robot garde une position relative dans la structure, et la formation émerge du fait que chacun accomplit sa mission alors qu'il n'y a aucun échange entre les éléments. C'est le cas des travaux proposés par Balch et al [Balch 99] : le robot a une position attribuée dans la formation basée sur un numéro d'identification (ID). Il peut seulement détecter les autres robots et les éviter afin d'arriver à la position relative qui lui est attribuée. Le même raisonnement est retrouvé dans [Lewis 97] où la dynamique de la formation est contrainte afin de prendre en compte la non-holonomie des robots. Même dans des travaux plus récents, les robots suivent chacun une cible prédéterminée dans la formation [Lalish 06], [Ren 04], [Skjetne 02], [Ghommam 08] et la question d'allocation dynamique des cibles n'est pas abordée ce qui laisse à croire qu'elle se fait toujours de façon déterministe et figée.

Nous proposons dans ce qui suit d'optimiser l'allocation des cibles aux éléments du SMR dans l'objectif d'atteindre la formation le plus rapidement possible. Ainsi, plutôt que de se baser uniquement sur une émergence d'un comportement de formation empruntée aux sociétés animales (non humaines), nous proposons d'exploiter les moyens de communication entre les entités robotiques. Il s'agit de rendre chaque robot capable de négocier, avec les autres, la position relative dans la formation (cible dynamique) qu'il juge la plus convenable. L'idée est tirée de la capacité des humains à échanger et interagir en fonction de nos besoins, émotions, etc.

S'inspirer d'un modèle de coopération humain existe déjà dans la littérature. Dans son architecture ALLIANCE, Parker [Parker 98] introduit deux notions inspirées de sentiments humains appelées comportements motivationnels : l'impatience qu'un robot peut afficher pour remplacer un autre qui exécuterait maladroitement une tâche, et le consentement de l'autre robot à abandonner l'exécution quand il estime qu'il n'a pas atteint l'objectif attendu. Dans [Dahl 08], un robot remplace un autre robot dont les performances ne lui assurent pas de continuer l'exécution de la tâche imitant ainsi le départ en retraite d'un agent et son remplacement par un autre. D'autres travaux reprennent la méthode des ventes aux enchères et aux marchés où le robot disposant des ressources nécessaires obtient l'exécution de la tâche [Brian 02], [Goldberg 03].

2.3.5.1 Attribution des cibles basée sur une hiérarchie

Pour notre SMR, la première idée intuitive permettant d'optimiser le temps nécessaire pour atteindre la formation serait que chaque robot cherche et choisisse la cible la plus proche. Pour cela, le robot est supposé connaitre la position de la cible principale ainsi que les coordonnées polaires des cibles secondaires (D_i, Φ_i) définies ci-dessus (cf. Section 2.1) (cf. Figure 2.1, page 57). Chaque robot exécute alors l'algorithme 2 pour choisir sa cible.

Entrées: Distances $d_{S_{ij}}$ séparant le robot i des cibles j ($j = 1..N_c$) (cf. Figure 2.26)

Sorties: numéro cible choisie

1: classer par ordre croissant les distances $d_{S_{ij}}$ séparant le robot i de toutes les cibles de la structure dans un vecteur D_{S_i} ;
2: choisir la cible la plus proche ;
3: numéro cible choisie=j ;

Algorithm 2: Attribution des positions (cibles) aux robots dans la formation

Cette idée est intéressante. En effet, plutôt que d'avoir une cible préalablement définie pour chaque robot, dont la position n'est pas forcément la plus optimale

par rapport à la position initiale du robot en question, il est intéressant que les éléments du SMR choisissent leurs cibles de cette façon dynamique. Ainsi, on épargne aux entités robotiques de parcourir des distances et gérer des collisions inutiles. Un simple exemple de comparaison peut être vu dans la figure 2.27 où les états initiaux des robots et de la formation sont illustrés. Dans le cas où chaque cible est préalablement définie, le numéro de la cible correspond à celui du robot. On voit bien que pour que le robot 1 atteigne la sienne, il doit parcourir une plus grande distance, et gérer la collision avec les robots 2 et 3 avant d'arriver. Il en est de même pour les deux autres robots. En revanche, avec une attribution dynamique basée sur l'algorithme 2, le robot 1 prend directement la cible 2 qui est la plus proche. Le robot 2 prend la cible 3 et le robot 3 prend la cible 1. la formation sera atteinte en parcourant moins de distance (et donc moins de temps), sans qu'il y ait besoin de gérer un risque de collision.

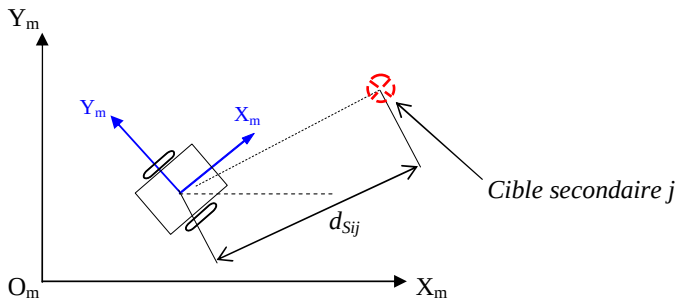


FIGURE 2.26 – Distance $d_{S_{ij}}$ séparant le centre de l'essieu du robot i de la cible dynamique j.

L'exemple de la figure 2.27 est évidemment élémentaire. Il est certain qu'avec l'algorithme 2, des situations de conflit entre les entités robotiques peuvent vite apparaître empêchant la formation. En effet, qu'en est il quand deux ou plusieurs robots choisissent la même cible la plus proche au même temps ?

Des situations de conflit, où plusieurs robots veulent partager la même ressource ou accomplir la même tâche, ont été rencontrées dans la littérature et des solutions ont été apportées. Comme précédemment cité, Brian et al [Brian 02] se sont inspirés des méthodes d'intelligence artificielle distribuée et ont repris une activité humaine assez ancienne : la mise aux enchères des ressources insuffisantes. Ainsi, lorsqu'une tâche doit être accomplie, un agent responsable d'attribuer les tâches le fait savoir en publiant les ressources nécessaires (capteur, mobilité, etc.). Comme plusieurs robots peuvent disposer des ressources, il faudra les départager en choisissant celui qui, *a priori*, pourrait mieux l'accomplir. Seuls les robots disposant de ces ressources communiquent un score déterminant leurs aptitudes à acquérir cette tâche (ressources disponibles, performances en fonction de ces

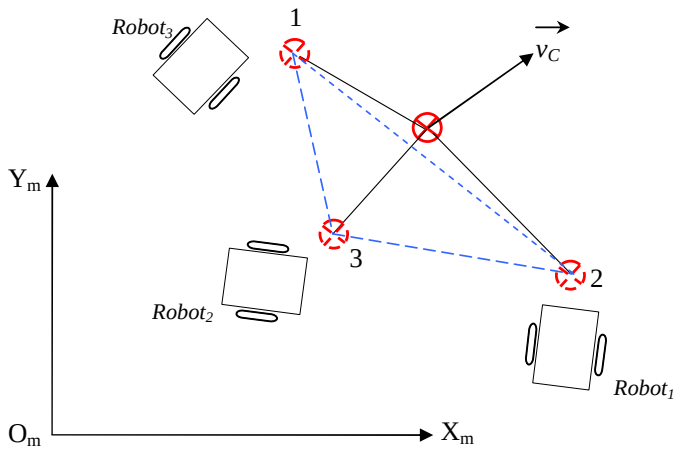


FIGURE 2.27 – Allocation des cibles pour un SMR à 3 robots.

ressources, autonomie énergétique, etc.). Le robot le plus apte (affichant le score le plus élevé) obtient alors son exécution.

Dans [Benzerrouk 10a], nous avons proposé la méthode de la distribution hiérarchique des cibles où une hiérarchie entre les robots est considérée. Ainsi, si plusieurs robots réclament la même cible, celui dont le rang hiérarchique est le plus élevé l'obtient. L'attribution des cibles est alors optimisée en se basant sur les distances mais aussi de résoudre les problèmes de conflits grâce à une hiérarchie entre les robots. Pour que les robots puissent se partager les cibles de cette façon, notons qu'il est nécessaire d'établir une communication entre eux. Comme le critère de choix de la cible est la distance et l'ordre dans la hiérarchie de chaque robot en cas de conflit, chaque entité doit alors être capable d'envoyer aux autres éléments un vecteur composé de la distance qui la sépare de chaque cible, ainsi que son ordre dans la hiérarchie. Ainsi, et pour N_c cibles composant la structure virtuelle, chaque élément d'ordre i du SMR doit envoyer un vecteur D_{S_i} de taille $N_c + 1$. Ce vecteur est composé des distances qui le séparent de chaque cible (dans un ordre connu de tous) et de l'ordre dans la hiérarchie.

Pour éviter des conflits de communication, nous proposons que ce même ordre d'hiérarchie définit l'ordre d'envoi. Ainsi, si nous considèrons l'ordre 1 comme étant le plus élevé, le robot correspondant envoie alors son vecteur D_{S_1} à tout le SMR. Une fois que le robot d'ordre 2 a reçu D_{S_1}, il envoie D_{S_2} et ainsi de suite jusqu'au robot d'ordre N. A noter qu'il est possible d'imposer que chaque entité i attend un temps δt avant d'envoyer son vecteur D_{S_i} pour prendre en considération un retard éventuel dans la réception des trames.

De cette manière, lorsqu'un robot i choisit une cible (la plus proche), il est

en mesure de vérifier (en analysant les vecteurs reçus des autres entités) s'il peut se l'approprier ou si elle sera prise par un autre robot de rang supérieur et qu'il doit donc vérifier la cible suivante. L'algorithme précédent est alors amélioré pour obtenir l'algorithme 3.

ENTRÉES: Distances $d_{S_{ij}}$ séparant le robot i des cibles j ($j = 1..N_c$) (cf. Figure 2.26)
SORTIES: numéro cible choisie
1: Recevoir les vecteurs D_{S_k} provenant des entités robotiques pour $k = 1, ..., i-1$;
2: Envoyer le vecteur D_{S_i} après un temps δt ;
3: Recevoir les vecteurs D_{S_k} provenant des entités robotiques pour $k = i+1, ..., N$;
4: définir le booléen : CibleChoisie = faux ;
5: **Tantque** CibleChoisie == faux **Faire**
6: classer par ordre croissant les distances $d_{S_{ij}}$ séparant le robot de rang i de toutes les cibles j de la structure dans un vecteur D_{S_i} ;
7: choisir la cible la plus proche j correspondant à $D_{S_i}(1)$;
8: **Si** la cible choisie j est réclamée par un robot de rang k tel que $k > i$ **alors**
9: Enlever du vecteur D_{S_i} la distance $d_{S_{ij}}$ séparant le robot i de la cible j ;
10: **Sinon**
11: CibleChoisie = vrai ;
12: numéro cible choisie=j ;
13: **finSi**
14: **fin Tantque**

Algorithm 3: Attribution des positions (cibles) aux robots dans la formation en définissant une hiérarchie entre les robots.

Cependant, nous pouvons reprocher à cette hiérarchie de ne pas être justifiée : pourquoi un robot i serait-il hiérarchiquement supérieur à un robot k alors qu'ils accomplissent tous la même tâche de la même façon ? nous proposons alors un nouveau critère d'attribution de telle sorte que la cible sujet à un conflit soit attribuée au robot dont il a le plus besoin plutôt qu'au robot hiérarchiquement supérieur. Par exemple, si deux entités réclament la même cible, il serait préférable que celui qui peut en trouver une autre pour un coût peu élevé, « se sacrifie » pour celui qui n'a pas cette opportunité. En effet, ce dernier peut être trop loin des autres cibles. Chaque robot passe alors l'intérêt du groupe avant le sien imitant le comportement d'altruisme.

2.3.5.2 Attribution des cibles par la méthode des coefficients de coût relatif

Nous rappelons que l'objectif que nous nous sommes imposé est d'atteindre la formation le plus rapidement possible. Cela revient à ce que tous les robots atteignent leurs cibles le plus rapidement possible (toujours de façon réactive).

Pour cela, nous proposons que chaque entité robotique calcule un coefficient pour chaque cible de la structure à intervalle de temps régulier ΔT. Ce coefficient détermine le coût relatif de la cible en question par rapport aux autres cibles. En d'autres termes, il aide à savoir si elle est proche ou loin du robot en comparant aux autres sommets de la structure virtuelle.

Ce coefficient conduit donc le robot à connaître la cible la plus adéquate mais aussi à décider de façon complètement distribuée s'il peut se l'approprier ou doit la laisser pour un autre robot plus à même de l'atteindre avec un coût moindre pour le groupe de robots.

D'après ce qui précède, nous proposons d'appeler cette grandeur *coefficient de coût relatif* que nous noterons δ.

Un coefficient de coût relatif du robot i pour la cible j est noté δ_{ij}. Il se calcule par

$$\delta_{ij} = \frac{d_{S_{ij}}}{\sum_{k=1}^{N_c} d_{S_{ik}}} \tag{2.40}$$

où N_c est le nombre de cibles de la structure tel que $N_c \geq N$ (il doit toujours être prévu au moins une cible par robot). L'ensemble des coefficients est stocké dans un vecteur Δ_i.

La définition du coefficient de coût relatif (cf. Equation 2.40) permet d'écrire

$$\delta_{ij} = \frac{d_{S_{ij}}}{d_{S_{ij}} + \sum_{k=1,k\neq j}^{N_c} d_{S_{ik}}} \tag{2.41}$$

On déduit alors que l'équation 2.41 impose clairement que

$$0 \leq \delta_{ij} \leq 1$$

δ_{ij} est d'autant plus proche de 0 que

$$d_{S_{ij}} \ll \sum_{k=1,k\neq j}^{N_c} d_{S_{ik}} \tag{2.42}$$

Ainsi, et d'après la relation 2.42, la cible qu'un robot préfère, sera celle dont le coefficient de coût est le plus petit voire proche de 0, car cela signifie que la distance qui le sépare de cette cible est négligeable par rapport aux autres cibles.

En effet, pour un robot i choisissant entre deux cibles j et l tel que

$$\delta_{ij} < \delta_{il} \tag{2.43}$$

c'est la cible j qui sera privilégiée.

Pour un seul robot, on pourrait croire que le même résultat aurait été obtenu en comparant simplement les distances $d_{S_{ij}}$ et $d_{S_{il}}$. D'ailleurs, la relation 2.43 implique

$$\frac{d_{S_{ij}}}{\sum_{k=1}^{N_c} d_{S_{ik}}} < \frac{d_{S_{il}}}{\sum_{k=1}^{N_c} d_{S_{ik}}} \tag{2.44}$$

il résulte

$$d_{S_{ij}} < d_{S_{il}}$$

Cependant, nous rappelons que son but sera également de gérer des situations de conflit avec les autres entités robotiques qui souhaitent s'attribuer la même cible. Dans ce cas de figure, la simple distance à la cible des robots n'apporte pas une contribution d'aide à la décision afin d'attribuer la cible en conflit. Par exemple, si deux robots i et k souhaitent s'attribuer la même cible j car elle leur est la plus proche, on ne peut pas justifier son attribution à l'un plutôt qu'à l'autre en analysant que les distances $d_{S_{ij}}$ et $d_{S_{ik}}$.

Avant d'approfondir notre stratégie d'attribution basée sur le coefficient de coût relatif, nous proposons d'établir les deux définitions suivantes :

Définition 1. *Une cible j est la moins coûteuse pour un robot i si son coefficient de coût relatif est le plus petit des coefficients de toutes les cibles formant la structure virtuelle.*

Définition 2. *Plusieurs robots mobiles sont en conflit pour la cible j s'ils n'ont choisi aucune cible et si j est la cible la moins coûteuse pour eux.*

Ces deux définitions permettent d'établir une convention assurant aux robots en conflit la possibilité de décider individuellement s'ils peuvent s'approprier une cible ou la laisser au profit d'un autre.

Convention 2. *Si plusieurs robots mobiles sont en conflit pour une cible, alors la cible sera prise par celui dont le coefficient de coût relatif pour cette cible est le plus petit.*

En effet, d'après la définition du coefficient de coût relatif (cf. Equation 2.40), même si une cible donnée est la moins coûteuse pour plusieurs robots à la fois, elle est attribuée à celui dont le coût est minimal. En effet, d'après la relation 2.42, cela signifie que c'est ce robot qui est le moins à même de trouver une autre cible relativement proche de lui.

2.3.5.3 Discussion

D'après ce qui précède, les coefficients de coût relatif proposés permettent de concilier distance du robot par rapport à la cible désirée et besoin des autres robots à cette cible afin de décider s'il peut s'emparer d'elle. L'algorithme 4 résume cette démarche que chaque robot doit suivre. Naturellement, les robots doivent communiquer entre eux pour échanger les vecteurs des coefficients de coût relatif respectifs Δ. De la même façon que l'algorithme utilisant une hiérarchie entre les robots (cf. Algorithme 3), le protocole de communication adopté consiste à envoyer un vecteur de $(N_c + 1)$ valeurs. Il contient les N_c éléments du vecteur de coût Δ pour chaque cible dans un ordre connu de tous les robots. La valeur (N_c+1) comporte un descripteur de robot. Il s'agit d'un numéro propre à chaque entité pour qu'elle soit reconnue par les autres. Ainsi, et à intervalle de temps δt, le robot numéro 1 commence par envoyer son vecteur Δ_1, ensuite vient le tour du robot 2 qui envoie Δ_2 et ainsi de suite.

Pour simplifier au maximum l'algorithme 4, l'étape d'échange de vecteurs Δ_i $(i = 1, ..., N)$ par communication et que nous venons d'expliquer n'y figure pas. Elle se fait de la même manière que dans l'algorithme 3 en remplaçant D_{S_i} par Δ_i.

Même si ce coefficient diminue considérablement les conflits et optimise l'attribution des cibles aux robots sans hiérarchie, certains cas particuliers peuvent subsister et nécessitent d'être traités :

- le cas où les robots ont le même coefficient pour la même cible j en conflit. En classant les valeurs des vecteurs Δ_i par ordre croissant, il sera intéressant de voir la cible au coefficient de coût relatif immédiatement supérieur à celui de j (deuxièmes valeurs des Δ_i). Ainsi, la cible en conflit j est obtenue par

celui dont la deuxième cible disponible lui coûte plus cher (et donc au SMR global aussi) car cela signifie qu'il est le moins capable d'atteindre cette deuxième cible qui s'offre à lui.

– s'ils partagent le même coût relatif pour la deuxième cible aussi, l'attribution de la première se fait alors aléatoirement. En effet, donner la cible à l'un ou l'autre a, a priori, le même coût pour le SMR. Cependant, comme l'attribution se fait de façon distribuée, il faut que l'entité sache si elle prend la cible j. Elle teste alors son descripteur par rapport à ceux avec lesquels elle est en compétition. Si elle a le descripteur le plus petit, elle prend la cible j. On pourrait penser que c'est une forme d'hiérarchie comme pour l'algorithme 3 pour cette cible. Néanmoins, utiliser un descripteur est purement symbolique dans le sens où il intervient uniquement afin d'éviter que les robots choisissent ou délaissent la même cible au même temps. Que l'un ou l'autre se l'approprie ne change rien car cela génère le même coût pour le SMR.

De cette manière, chaque robot est apte à décider de la cible qu'il peut prendre sans conflits. Si une cible est jugée non convenable (coefficient de coût relatif ou descripteur plus grand par rapport à ceux d'autres entités), il l'efface automatiquement de sa liste de cibles disponibles car elle sera prise par une autre entité. L'algorithme 4 résume l'attribution de cibles basée sur les coefficients de coût relatif proposés. A noter que le descripteur d'un robot i est égal à i.

Entrées: Vecteurs Δ_i $(i = 1, ..., N)$ classés dans un ordre croissant
Sorties: numéro cible choisie

définir le booléen : CibleChoisie = faux ;
Tantque CibleChoisie == faux **Faire**
  choisir la cible la plus proche correspondant à $\Delta_i(1)$;
  Si la cible choisie j est réclamée par d'autres robots p **alors**
    définir $E = \{\Delta_k / \Delta_k(1) = \min \Delta_p(1)\}$;
    Si cardinal E > 1 **alors**
      définir $I = \{\Delta_k \in E / \Delta_k(2) = \min \Delta_p(2)\}$;
    finSi ;
    Si cardinal I > 1 **alors**
      choisir n'importe quel élément de I noté Δ_k ;
    finSi
    Si $([\Delta_i(1), \Delta_i(2)] \neq [\Delta_k(1), \Delta_k(2)])$ **alors**
      Si $(\Delta_i(1) < \Delta_k(1))$ **alors**
        numéro cible choisie = numéro de cible correspondant à $\Delta_i(1)$;
        CibleChoisie = vrai ;
      SinonSi $\Delta_i(1) == \Delta_k(1)$ **alors**
        Si $\Delta_i(2) > \Delta_k(2)$ **alors**
          numéro cible choisie = numéro cible correspondant à $\Delta_i(1)$;
          {changer de cible coûte plus cher pour le robot i que pour le robot k}
          CibleChoisie = vrai ;
        Sinon
          {pour tous les autres cas}
          supprimer la cible la plus proche du vecteur Δ_i car elle sera prise par un autre robot ;
          aller à l'étape 3 ;
        finSi ;
      Sinon
        supprimer la cible la plus proche du vecteur Δ_i car elle sera prise par un autre robot ;
        aller à l'étape 3 ;
      finSi ;
    Sinon
      Si $(i < k)$ **alors**
        CibleChoisie = vrai ;
      finSi
    finSi ;
  Sinon
    CibleChoisie = vrai ;
  finSi ;
fin Tantque ;

Algorithm 4: Algorithme d'attribution des cibles basé sur les coefficients de coût relatif δ et exécuté dans le robot i.

L'attribution des cibles basée sur les coefficients de coût relatif est simulée dans la figure 2.28 avec un exemple à trois robots $R1, R2$ et $R3$. La structure virtuelle comporte trois cibles $C1, C2$ et $C3$. Sa position initiale ainsi que celles des robots sont représentées à l'instant t_0. Les distances séparant les robots des cibles et les coefficients de coût relatif à cet instant sont donnés dans les tableaux 2.1 et 2.2 respectivement. On remarque que le robot $R1$ est le plus proche de la cible $C3$. Il a aussi le coefficient de coût relatif le plus petit pour cette cible. Le robot $R1$ obtient donc cette cible d'autant plus que si on observe les coefficients de coût des autres robots, on remarque qu'elle ne leur est pas convenable (ils ont des coefficients plus petits pour d'autres cibles). Par exemple, $R2$ voudra obtenir $C1$ car son coefficient de coût pour cette cible est le plus petit. Il en est de même pour le robot $R3$. $R2$ et $R3$ sont donc en conflit pour $C1$. D'ailleurs, on remarque qu'ils sont équidistants de cette cible (cf. tableau2.1). Comme le robot $R2$ a le coefficient le plus petit, il l'obtient. Le robot R3 prendra la cible $C2$ correspondant au coefficient immédiatement plus grand et encore libre.

A l'instant t_1, la formation est atteinte avec le robot $R3$ qui a laissé (à t_0) la cible le plus proche au robot $R2$ grâce à la méthode des coefficients de coût relatif (altruisme). Le conflit est ainsi évité grâce à l'altruisme du robot $R3$.

TABLE 2.1 – Distances séparant les robots des cibles à l'instant t_0.

	C1	C2	C3
R1	7.61	3.56	0.88
R2	2.00	5.93	9.36
R3	2.00	3.56	7.12

TABLE 2.2 – Coefficients de coût à l'instant t_0.

	C1	C2	C3
R1	0.63	0.29	0.07
R2	0.11	0.34	0.54
R3	0.15	0.28	0.56

A l'instant t_2, nous simulons un saut de consigne de la structure virtuelle et un changement de sa dynamique. Les nouveaux coefficients de coût relatif et les distances des cibles sont donnés dans les tableaux 2.4 et 2.3 respectivement. On observe que la cible $C1$ est la cible désirée par tous les robots. Cependant, le robot $R1$ l'obtient grâce à son coefficient de coût relatif qui est le plus petit. Si on

observe le tableau des distances (cf. tableau 2.3), on remarque que la cible $C2$ est plus proche du robot $R1$ que les autres. Cependant, il s'est obstiné pour garder la cible $C1$. Le coefficient de coût relatif permet donc de faire un compromis : grâce à ce coefficient, le robot peut garder la cible la plus proche et ne pas avoir d'altruisme si la cible est trop proche de lui. En effet, c'est plus intéressant pour lui et pour le groupe de ne pas quitter une cible sur laquelle il se trouve déjà. Chacun des robots $R2$ et $R3$ considère que la cible $C1$ sera prise car il y a au moins un robot ($R1$) qui a un coefficient relatif pour cette cible plus petit que le sien. Uniquement les deux cibles restant sont retenues. En observant leurs coefficients pour ces deux cibles, le robot $R2$ obtient la cible $C2$ et le robot $R3$ obtient $C3$. On voit qu'aux instants t_3 et t_4, les robots atteignent les cibles choisies et atteignent la formation souhaitée.

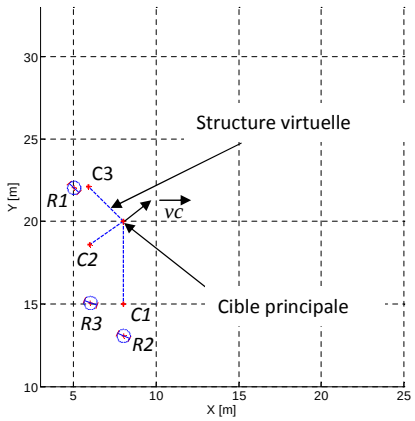
TABLE 2.3 – Distances séparant les robots des cibles à l'instant t_2.

	C1	C2	C3
R1	0.67	3.46	6.76
R2	6.95	9.40	13.04
R3	2.86	6.85	9.87

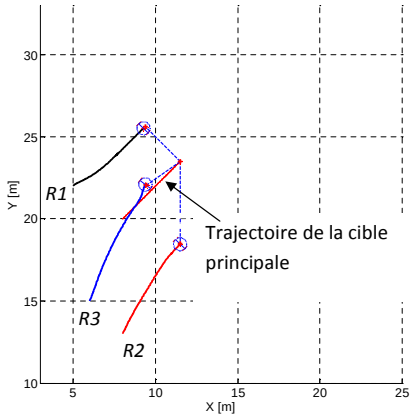
TABLE 2.4 – Coefficients de coût à l'instant t_2

	C1	C2	C3
R1	0.06	0.31	0.62
R2	0.22	0.32	0.42
R3	0.14	0.35	0.50

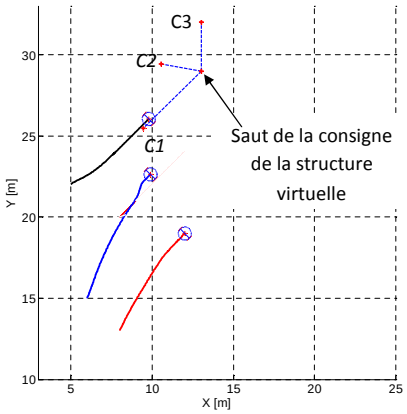
Grâce au compromis offert par les coefficients de coût relatif, il est possible que la méthode d'attribution de la cible soit appliquée à un instant où les robots ont déjà atteint la formation. A ce moment, chacun aura un coefficient de coût proche de 0 pour la cible qu'il est déjà en train de suivre. Chaque entité reste donc sur sa cible et aucun changement ne se produit.



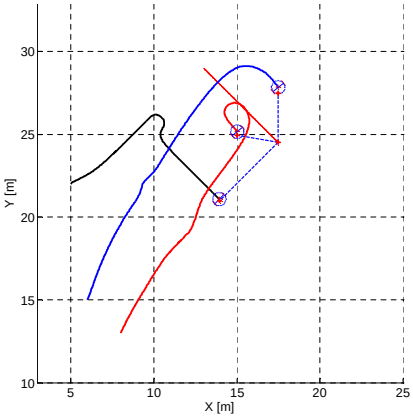
(a) Instant t_0



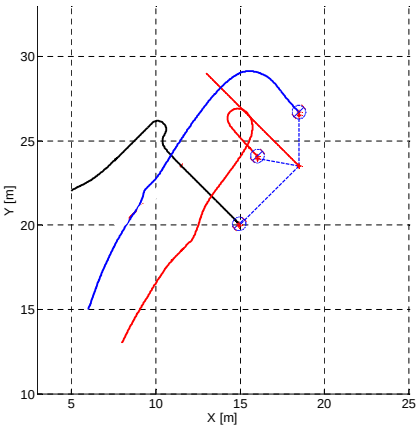
(b) Instant t_1



(c) Instant t_2



(d) Instant t_3



(e) Instant t_4

FIGURE 2.28 – Exemple d'attribution des cibles par la méthode des coefficients de coût relatif.

2.4 Conclusion

L'architecture de contrôle proposée est inspirée des différentes stratégies proposées dans la littérature, à savoir : l'approche comportementale et la structure virtuelle. En effet, la méthode de l'approche hiérarchique n'est pas convenable à notre cahier des charges. Elle a donc été écartée.

L'idée principale est que chaque robot suit une cible dynamique appartenant à un corps rigide virtuel dont la dynamique de déplacement traduit le mouvement souhaité de la formation. L'approche comportementale a permis de se passer de l'aspect centralisé utilisé dans les structures virtuelles, mais aussi d'ajouter un contrôleur permettant aux entités robotiques d'éviter les obstacles pouvant gêner leur navigation.

Le contrôleur d'attraction vers une cible dynamique offre au robot des consignes permettant d'atteindre la position de la cible mais aussi de tendre vers son orientation de façon plus lisse que ce qui est proposé dans la littérature. L'évitement d'obstacles délivre des consignes assurant la réalisation de la tâche tout en traitant les différentes situations de conflits : impasses, obstacles dynamiques, etc.

Ces situations de conflit sont traitées en gardant un aspect réactif.

Une seule loi de commande délivre les vitesses linéaire et angulaire permettant de converger vers les consignes indépendamment de la provenance de ces consignes (du contrôleur d'évitement d'obstacles, ou d'attraction et suivi de cible). Elle est basée sur des gains affectant la vitesse de convergence. Cependant, ces gains théoriques doivent prendre en compte la stabilité de la commande et les contraintes structurelles du robot. Ils sont discutés et bornés dans le chapitre suivant.

Enfin, dans la construction de l'architecture de contrôle, l'aspect distribué est toujours pris en compte pour étudier l'affectation dynamique des cibles entre les entités. L'objectif est d'optimiser le temps pour atteindre la formation finale. L'idée est que chaque robot s'empare de la cible la plus proche. Les cas de conflit sont gérés en utilisant des coefficients de coût relatif. Pour cela, nous nous sommes inspiré des comportements humains comme l'altruisme. Ainsi, si plusieurs robots souhaitent s'attibuer la même cible, ils se sacrifient au profit du groupe et la laissent au robot le plus à même de l'atteindre avec un coût moindre pour le groupe de robots.

Conçue de cette manière, l'architecture de contrôle proposée répond parfaitement aux contraintes que nous nous sommes imposé (cf. Section 1.7).

Dans ce qui suit, une étude complète de la stabilité de notre architecture de contrôle est détaillée.

Chapitre 3

Stabilité de l'architecture de contrôle proposée

Nous proposons dans ce chapitre d'étudier la stabilité de l'architecture de contrôle proposée. Pour cela, nous utiliserons les systèmes hybrides. Nous commençons alors par une introduction à ces systèmes. Ensuite, nous nous intéressons à la stabilité de l'architecture tout en introduisant les contraintes cinématiques des robots (vitesses et accélérations maximales). Les contributions apportées sont appuyées par des résultats de simulation.

La complexité inhérente à la coopération (coordination) des mouvements entre les entités autonomes est traitée dans les chapitres précédents en investiguant plus avant les potentialités des architectures de contrôle comportementales dont le but premier est de briser la complexité de la tâche globale de maintien de formation d'un groupe de robots mobiles dans un environnement encombré.

Ainsi, pour maîtriser cette complexité, le contrôle global dédié à la réalisation de la tâche complexe est décomposé en un ensemble de comportements/contrôleurs élémentaires (évitement d'obstacles, attraction vers une cible) qui lient les différentes informations capteurs (provenant de caméras, des capteurs locaux du robot, etc.) aux actions des différentes entités robotiques (vitesses calculées par la loi de commande) (cf. Figure 2.7, page 62). Nous avons vu que ces contrôleurs délivrent des consignes fiables, qui permettent d'accomplir la tâche (maintien de formation, évitement d'obstacles) si elles sont suivies par les entités robotiques.

La spécificité de l'approche théorique consiste à allier les avantages des architectures de contrôle comportementales à la méthode de la structure virtuelle où le groupe de robots mobiles suit un corps virtuel avec une dynamique (vitesse, direction) donnée. Ainsi, l'activation d'un comportement élémentaire en faveur d'un autre (commutation de l'attraction vers la cible vers l'évitement d'obstacles et vice versa) se fait en fonction de l'environnement dans lequel l'entité évolue : si un obstacle est rencontré, la priorité du robot est de l'éviter plutôt que de continuer à suivre sa cible. Cependant, ces contraintes imposées par l'environnement extérieur ne sont pas les seules à prendre en considération lors des commutations entre contrôleurs. En effet, il faut aussi respecter les contraintes structurelles propres aux robots (non holonomie, contraintes cinématiques/dynamiques, etc.) afin de s'assurer que les consignes imposées sont atteignables par les entités robotiques. De plus, la commutation entre contrôleurs implique une discontinuité de contrôle/commande pouvant engendrer des effets indésirables voire l'instabilité de la commande appliquée au robot.

Dans la littérature, il est reproché à la plupart des architectures de contrôle comportementales l'absence d'une étude analytique rigoureuse de la stabilité. En effet, dans le cas des architectures de subsomption [Brooks 85], celle-ci est souvent difficile à établir principalement à cause des commutations d'un contrôleur à un autre et des discontinuités de consigne qui en découlent. Le cas des schémas moteur (fusion d'actions) [Arkin 86] est tout aussi difficile à gérer à cause de la difficulté d'analyser un contrôle global comprenant la fusion de plusieurs comportements simultanément.

Pour pallier ces inconvénients, nous proposons dans ce chapitre d'exploiter les potentialités des systèmes hybrides permettant de commander des systèmes continus en présence d'évènements discrets. La coordination de l'activité (partie discrète) des différents comportements (partie continue) disponibles au niveau de

l'architecture de contrôle se fait alors en appliquant une analyse rigoureuse des performances de l'architecture de contrôle : stabilité, vitesse d'exécution, etc.

Dans ce qui suit, nous allons brièvement présenter les systèmes hybrides et certains de leurs théorèmes, que nous utilisons dans le but de prouver la stabilité de l'architecture de contrôle proposée.

3.1 Introduction aux systèmes hybrides

Beaucoup de systèmes rencontrés en pratique impliquent l'utilisation des systèmes continus couplés à des évènements discrets. Ces systèmes où dynamiques continue et discrète coexistent sont appelés *systèmes hybrides*. Une valve commandée pour s'ouvrir et se fermer alternativement, les roues d'une voiture qui tantôt glissent, tantôt roulent sur une route verglacée, sont des exemples typiques de systèmes hybrides.

Un système continu peut être représenté par une équation d'état $\dot{x} = F(x, u)$, où $x \in \mathbb{R}^n$ est l'état du système et $u \in \mathbb{R}^m$ est sa commande. Pour illustrer les évènements discrets, nous pouvons considérer un automate à nombre d'états finis q. La commutation d'un état à un autre est enclenchée suivant les valeurs d'une variable v. Ainsi, la commande du système continu u est fonction de l'état discret où se trouve le système, tandis que la variable v est déterminée selon les valeurs de l'état x. Elle permet de décider dans quel nœud de l'automate doit se trouver le système. Si ces conditions sont réunies, on est alors en présence d'un système hybride (cf. Figure 3.1).

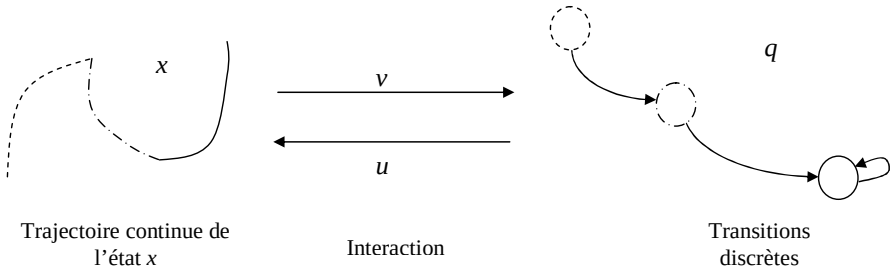


FIGURE 3.1 – Un système hybride.

L'un des problèmes les plus courants dans le cadre des systèmes hybrides sont l'étude de leur stabilité globale. En effet, ils sont généralement considérés comme des systèmes continus qui constituent l'objet principal à étude, et les

évènements discrets sont vus comme des perturbations susceptibles de rendre le système instable [Liberzon 03], [Bourgeot 04].

Si les commutations entre les systèmes continus se font de façon aléatoire, le système global peut diverger ou alors des effets indésirables peuvent apparaître. L'effet indésirable le plus connu est le phénomène Zeno [1]. Ce phénomène apparaît s'il y a plusieurs commutations non maîtrisées et répétées entre plusieurs systèmes en un temps relativement court [Johansson 99].

Ce phénomène se traduit en robotique par des comportements indésirables des robots : dans le chapitre 2, nous avons vu que si le robot navigue au voisinage du cercle d'influence de l'obstacle, le contrôle va osciller entre évitement d'obstacles et attraction vers la cible. Ceci se traduit par des oscillations au niveau de la trajectoire du robot, d'où la solution d'introduire des zones attractives et répulsives dans le contrôleur d'évitement d'obstacles (cf. Chapitre 2). Ces oscillations ont été évitées dans les travaux proposés dans [Benzerrouk 08], [Benzerrouk 09] en introduisant un troisième noeud (contenant un contrôleur) dans l'automate de contrôle afin les oscillations entre les deux noeuds/contrôleurs principaux (suivi de trajectoire et évitement d'obstacles).

Deux autres propriétés des systèmes hybrides peuvent apporter des solutions pour éviter les effets indésirables de ce phénomène. Elles ont même été reprises dans des travaux de contrôle de robots mobiles afin d'assurer la stabilité du contrôle. Ces deux propriétés sont les modes glissants et les commutations hystérétiques.

3.1.1 Modes glissants

Dans [Egerstedt 99], l'oscillation entre les deux contrôleurs utilisés pour la commande d'un robot mobile (attraction vers la cible et évitement d'obstacles) est enlevée grâce aux modes glissants introduits par Filippov [Filippov 88]. Pour expliquer brièvement ces modes, il faut imaginer une surface de commutation entre deux systèmes dont les évolutions sont assimilées à des champs de potentiel pointant vers cette surface. L'état du système global sera alors obligatoirement mené vers cette surface. Une fois dessus, il sera alors bloqué et soumis à la force résultante des deux champs pointant dans des directions opposées (cf. Figure 3.2). Ainsi, l'évolution résultante de l'état le mènera à glisser le long de cette surface sans oscillations (ou avec des oscillations négligeables et sans incidence).

1. En référence au philosophe grec Zénon d'Elée dont la philosophie aborde des paradoxes selon lesquels le mouvement est impossible et ces concepts d'évolution conduisent à des contradictions.

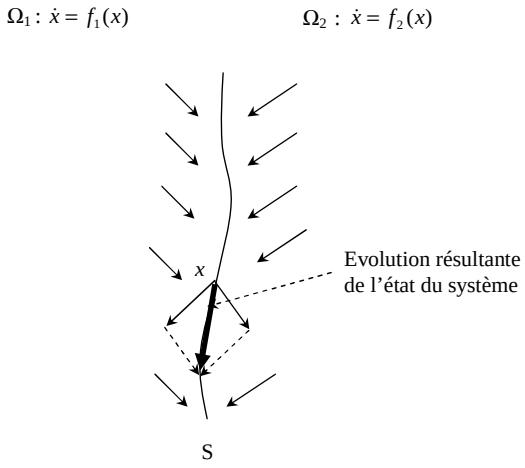


FIGURE 3.2 – Mode glissant au sens de Filippov.

3.1.2 Commutation hystérétique

Pour expliquer cette propriété [Liberzon 03], considérons deux régions Ω_1 et Ω_2 en excluant la surface de commutation S (cf. Figure 3.3(a)). Dans cette figure, cette surface originale S est présentée en ligne discontinue. On définit deux nouvelles surfaces (en lignes continues) S_1 et S_2. L'intersection entre les deux régions est la bande délimitée par S_1 et S_2.

On veut que l'état du système global x suive la dynamique $\dot{x} = f_1(x)$ dans la région Ω_1 et $\dot{x} = f_2(x)$ dans la région Ω_2. La commutation entre f_1 et f_2 survient alors quand la trajectoire de l'état touche la surface S_1 ou S_2. Ceci est formalisé en introduisant une fonction temporelle discrète $\sigma(t)$ dont l'évolution est décrite à l'instant initial t_0 comme suit :

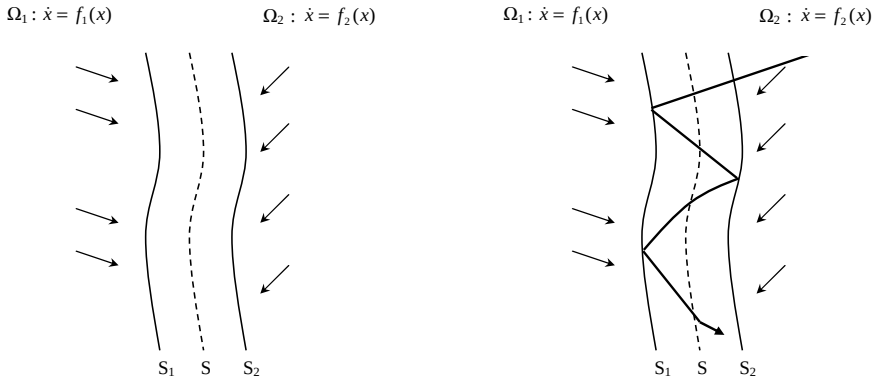
$$\sigma(t_0) = \begin{cases} 1 & si \quad x(t_0) \in \Omega_1 \\ 2 & sinon \end{cases}$$

Ainsi, pour chaque $t > 0$, si $\sigma(t^-) = i \in \{1, 2\}$, et $x(t) \in \Omega_i$, alors garder $\sigma(t) = i$.

D'un autre coté, si $\sigma(t^-) = i$, mais $x(t) \notin \Omega_i$ (le système se trouve alors dans l'autre état), alors changer la valeur de $\sigma(t)$ (passer de i à l'autre valeur de l'ensemble $\{1, 2\}$).

De cette façon, on génère une fonction temporelle constante par morceaux σ toujours continue à droite. Définie de cette manière, la fonction σ ne peut changer sa valeur que lorsque la trajectoire passe par la bande délimitée par les deux nouvelles surfaces de transition S_1, S_2 . Les oscillations sont alors évitées.

Les commutations observées dans la figure 3.3(b) arrivent alors à des instants assez espacés.



(a) Surfaces de transition entre Ω_1 et Ω_2,

(b) Trajectoire de l'état.

FIGURE 3.3 – Commutation hystérétique.

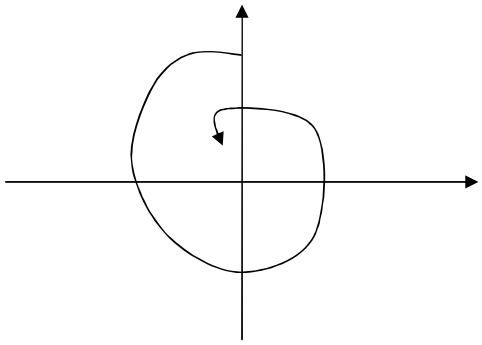
Notons qu'ici, la partie discrète du système a une sorte de « mémoire » : en effet, la valeur de σ n'est pas déterminée uniquement par la valeur de x, mais dépend aussi de la valeur précédente de σ (d'où la notion d'Hystérésis). Ainsi, dans la figure 3.3(b), l'état du système est d'abord mené par la fonction f_2 jusqu'à la surface S_1 (et non S). Au delà de S_1, il se trouve dans la région Ω_1. Il est alors mené par f_1 jusqu'à S_2 (et non S).

Cette propriété a été reprise dans les travaux d'Egerestedt [Egerstedt 00] pour déterminer les conditions de commutations entre les contrôleurs du robot. Cependant, les automates proposés dans les travaux d'Egerstedt présentent au moins un noeud où plusieurs contrôleurs sont simultanément actifs. Cela rejoint alors l'idée de la fusion d'actions qui empêche de pouvoir toujours étudier chaque contrôleur séparément.

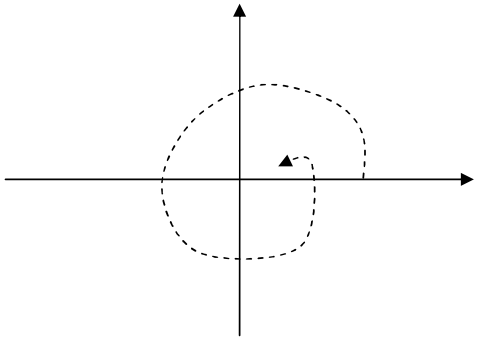
Ces propriétés ne sont donc pas convenables à l'étude de la stabilité de l'architecture de contrôle proposée, car elles imposent d'utiliser une fusion d'actions entre au moins deux contrôleurs. Afin d'accomplir cette étude, ainsi que l'analyse du système global, nous proposons donc d'explorer les théorèmes de stabilité au sens de Lyapunov appliqués aux systèmes hybrides. Dans ce qui suit, les définitions de base de la stabilité, la stabilité asymptotique, la stabilité au sens de Lyapunov, etc. sont supposés familières au lecteur. L'essentiel de ces définitions est toutefois fourni dans l'annexe B.

3.2 Stabilité des systèmes hybrides

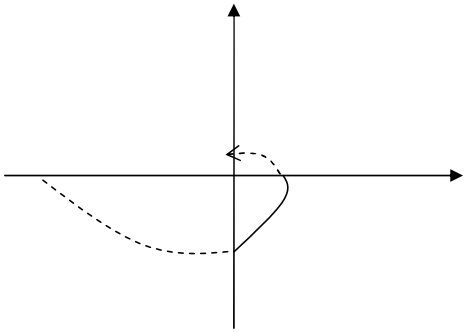
Pour illustrer le problème de la stabilité des systèmes hybrides, nous empruntons l'exemple donné dans [Liberzon 03] où la commutation entre deux systèmes 1 et 2 est considérée. L'état global est tel que $x \in \mathbb{R}^2$. La commutation se fait alors dans le plan. Supposons que les deux systèmes élémentaires sont asymptotiquement stables. Chaque système appliqué séparément mène alors l'état x au point d'équilibre comme illustré dans les figures 3.4(a), 3.4(b). Cependant, la commutation entre ces deux systèmes ne mène pas forcément à la stabilité. Ainsi, dans la figure 3.4(c), la stratégie de commutation adoptée mène bien à la stabilité contrairement à celle de la figure 3.4(d).



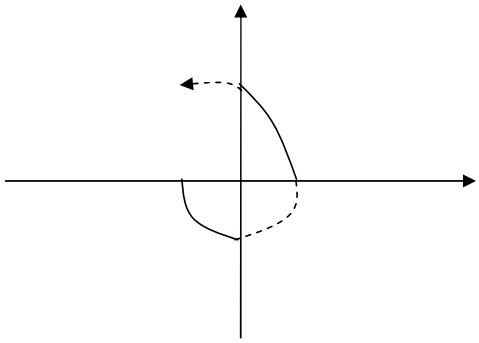
(a) Système 1



(b) Système 2



(c) Commutation entre systèmes menant à la stabilité



(d) Commutation entre systèmes menant à l'instabilité

FIGURE 3.4 – Commutations entres systèmes stables.

On en déduit alors que la commutation entre plusieurs systèmes élémentaires stables n'entraîne pas automatiquement la stabilité globale du système hybride [Branicky 93], [Liberzon 03].

Pour illustrer cela en pratique, nous proposons de prendre l'exemple d'un robinet à débit entrant Q_e devant alimenter deux réservoirs R_1, R_2 ayant chacun

un débit sortant Q_{s1} et Q_{s2} respectivement [Johansson 99]. Le robinet doit garder en permanence les deux réservoirs au dessus d'un seuil critique N_{sc1}, N_{sc2} en commutant entre les deux (cf. Figure 3.5). Il est supposé que sans commutations, le robinet peut remplir correctement et garder à un niveau désiré N_{d1}, N_{d2} chacun des deux réservoirs. Le système robinet-réservoir R_1 pris séparément est alors asymptotiquement stable (de même que le système robinet-reservoir R_2). Maintenant, supposons qu'on soumet la commutation du robinet entre les réservoirs au seul critère de niveau. Un cas d'instabilité possible est alors celui où R_1 et R_2 se trouvent au même temps à N_{sc1}, et N_{sc2}. En effet, le robinet va devoir commuter pour remplir l'un sans même avoir fini de sortir l'autre de son niveau critique. Il va alors osciller infiniment et les deux réservoirs vont se trouver vides. Par contre, si on contraint les commutations en imposant, à titre d'exemple, un temps d'attente τ_1 où le robinet doit rester sur le réservoir R_1, il va alors l'éloigner suffisamment de son seuil critique N_{sc1} (voire le ramener à N_{d1}) pour pouvoir remplir R_2 en toute sécurité pendant un temps τ_2. La stabilité globale du système est alors atteinte.

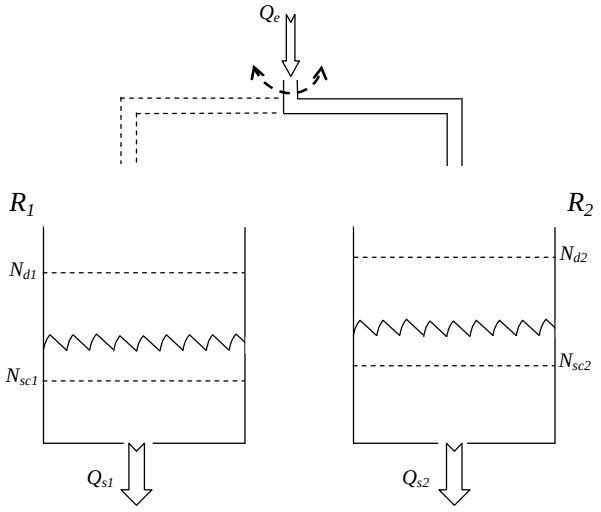


FIGURE 3.5 – Deux réservoirs à deux débits sortants alimentés par un seul débit entrant commutant entre eux.

La stabilité des systèmes hybrides peut être étudiée sous deux angles différents :

– Le premier correspond au cas où le mécanisme de commutation n'est pas connu ou trop compliqué pour être utilisé dans l'étude de la stabilité. L'idée générale est alors de trouver une fonction de Lyapunov qui soit décroissante tout au long de l'évolution du système global même aux instants de com-

mutations. Trouver cette fonction peut être une tâche très fastidieuse, voire impossible.
- Sous le deuxième angle, si la commutation se fait entre plusieurs systèmes élémentaires stables, l'idée est de trouver un mécanisme adéquat pour cette transition afin de conserver la stabilité globale. Il s'agit alors d'imposer des contraintes sur ce mécanisme afin de ne pas conduire à l'instabilité du système.

Le deuxième cas semble plus proche des configurations liées au contrôle des robots mobiles. En effet, pour notre architecture de contrôle/commande, il s'agit d'imposer des contraintes analytiques en plus des contraintes géométriques pour la commutation entre contrôleurs afin de garantir la stabilité. Dans ce qui suit, nous allons donc explorer plus en détail les théorèmes de stabilité relatifs aux commutations sous contraintes qui serviront à démontrer la stabilité de l'architecture de contrôle proposée.

Le théorème le plus connu dans cette catégorie est celui des *Fonctions de Lyapunov Multiples.*

3.2.1 Fonctions de Lyapunov Multiples (FLM)

Pour énoncer ce théorème, nous allons nous intéresser au cas d'une commutation entre deux systèmes asymptotiquement stables Σ_1 et Σ_2 définis sur le même espace d'état X et soumis à deux équations d'évolution différentes $\dot{x} = f_1(x)$ et $\dot{x} = f_2(x)$. Soit V_1 et V_2 leurs fonctions de Lyapunov respectives. Nous nous intéressons à la situation où il n'existe pas une fonction de Lyapunov unique qui soit décroissante tout au long de l'évolution de l'état du système malgré des transitions entre f_1 et f_2. Ainsi, V_1 et V_2 sont les seuls outils à notre disposition pour démontrer la stabilité.

Soit la fonction constante par morceaux $\sigma : [0, \infty[\rightarrow P = \{1, 2\}$. Les points de discontinuité de cette fonction constituent les temps des commutations. Le rôle de σ est alors de spécifier à chaque instant l'indice $\sigma(t) \in P$ du contrôleur actif. Un exemple de la fonction σ est présenté sur la figure 3.6. Cette fonction peut être généralisée à n systèmes avec $P \in \{1, 2, \ldots, n\}$. En l'absence d'une seule fonction de Lyapunov, la stabilité du système dépendra alors de ce signal.

Soit t_i, $i = 1, 2, ..$ les instants de transition. On vérifie aisément la propriété suivante.

Propriété 1. *Si les valeurs de V_1 et V_2 sont égales à chaque instant de transition, alors V_σ devient une fonction de Lyapunov continue pour tout le système hybride et le système global est asymptotiquement stable.*

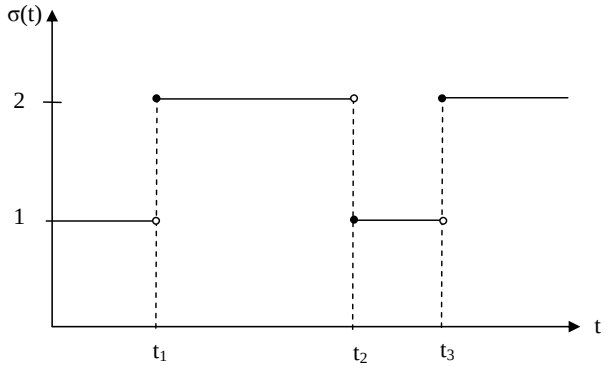


FIGURE 3.6 – Un exemple de signal de transition σ entre deux systèmes. Mise en évidence de la continuité à droite de σ.

En effet, si les valeurs de V_1 et V_2 sont telle que $(V_{\sigma(t_{i-1})}(x(t_i)) = V_{\sigma(t_i)}(x(t_i)))$ pour tout i, et comme les systèmes f_1 et f_2 sont asymptotiquement stables, V_1 et V_2 sont décroissantes. Ainsi, V_1 et V_2 forment une seule fonction décroissante pendant toute l'évolution du système global. La stabilité asymptotique en découle alors naturellement. Cette situation est montrée sur la figure 3.7.

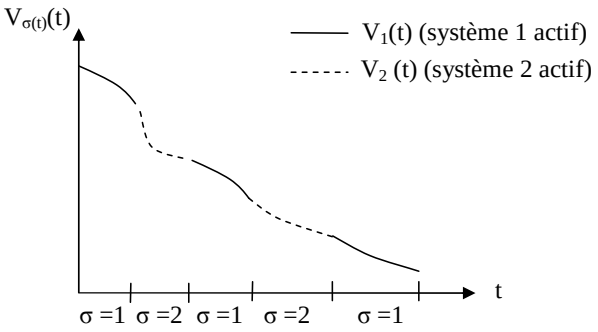


FIGURE 3.7 – Un exemple avec deux fonctions de Lyapunov : cas où les deux fonctions forment une seule fonction V_σ décroissante.

Cependant, il ne s'agit que d'un cas très particulier et V_σ pourra bien être discontinue et croissante au moment des transitions. En effet, si V_1 (respectivement V_2) décroît quand le système correspondant est actif, elle peut tout à fait croître quand il est inactif (cf. Figure 3.8).

Pour montrer la stabilité asymptotique du système global, nous nous intéressons aux valeurs des fonctions de Lyapunov V_σ à la fin de chaque intervalle où le système σ est actif ($\sigma \in \{1, 2\}$). Ces valeurs notées $V_\sigma(t_s)_i$ doivent former une

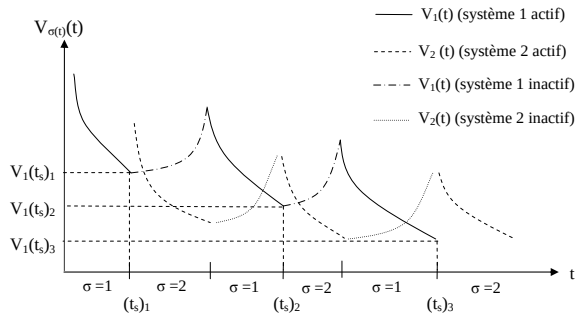


FIGURE 3.8 – Un exemple avec deux fonctions de Lyapunov discontinues.

séquence décroissante pour chaque système i. Ces valeurs sont illustrées sur la figure 3.8 pour le cas $\sigma = 1$.

L'énoncé du théorème est donné dans [Liberzon 03] comme suit

Théorème 1. *Soit un ensemble fini de n systèmes asymptotiquement stables. Soit V_p les fonctions de Lyapunov correspondantes, $p \in P = \{1, 2, ..., n\}$. S'il existe une fonction définie positive W_p pour chaque couple de fins d'intervalles $((t_s)_i, (t_s)_j)$ $(i < j)$, tel que $\sigma(t_s)_i = \sigma(t_s)_j = p \in P$, et qu'on a*

$$V_p(x(t_j)) - V_p(x(t_i)) \leq -W_p(x(t_i))$$

alors le système global est asymptotiquement stable.

Ce théorème stipule que lorsque le système est inactif, sa fonction de Lyapunov peut croître. Cependant, pour garder la stabilité asymptotique, il suffit de contraindre les commutations de telle façon que les valeurs des fonctions de Lyapunov de chaque système, à la fin de l'intervalle où ce système est actif, forment une séquence décroissante.

Note : Il est aussi possible de raisonner sur les valeurs des fonctions de Lyapunov correspondant au début de l'intervalle où le système est actif [Liberzon 03].

Sachant que chaque système p pris séparément est asymptotiquement stable, la fonction de Lyapunov correspondante V_p est décroissante tant qu'il est actif. Afin de garantir la stabilité globale sous la contrainte imposée par le théorème des *FLM*, il est parfois nécessaire d'empêcher des commutations. L'objectif est d'attendre que V_p décroisse assez afin d'assurer la séquence monotone de la fonction de Lyapunov. Ceci mène à la notion de la stabilité par des transitions lentes exposée dans le paragraphe suivant.

3.2.2 Stabilité sous les transitions lentes

Nous avons vu que les systèmes hybrides sont souvent perçus comme des systèmes continus où la transition discrète constitue une perturbation. Les transitions lentes permettent alors la dissipation des effets indésirables provoqués par la dernière perturbation avant la prochaine commutation.

Cela signifie que l'instabilité peut être le résultat de transitions rapides entre les systèmes (comme le phénomène Zeno (cf. Section 3.1)) d'où l'idée de contrôler le taux de commutation [Guo 95], [Zheng 93], [Shorten 05].

La façon la plus simple de définir des commutations lentes est de définir une constante de temps $\tau_d > 0$. Les signaux de commutation admissibles sont ceux dont les instants de commutations $t_1, t_2, \ldots$ satisfont à l'inéquation $t_{i+1} - t_i \geq \tau_d$ quelque soit i. Nous appelons cette constante « *le Temps de Tenue* » TT.

Le résultat suivant est établi dans [Morse 96].

Corollaire *Si un système hybride est constitué d'une famille de systèmes linéaires i de la forme $\dot{x} = A_i x$ et si tous ces systèmes sont asymptotiquement stables, alors le système global est asymptotiquement stable si le « temps de tenue » τ_d est choisi suffisamment large.*

Nous verrons dans la section 3.3 comment s'inspirer de ce résultat pour renforcer la stabilité de l'architecture de contrôle proposée.

Bien que les commutations lentes soient une question fondamentale car elles assurent une stabilité aux systèmes hybrides, peu de résultats ont été obtenus dans ce domaine. Par exemple, si une borne supérieure de τ_d peut être calculée [Morse 96], il n'en est pas de même pour la borne inférieure [Shorten 05]. Pourtant, c'est cette dernière qui sera la plus utile en pratique car il est plus intéressant de ne pas imposer un temps τ_d plus grand que le temps minimal $\tau_{d_{min}}$ nécessaire pour assurer la stabilité. En effet, imposer un temps τ_d trop grand devant la durée d'activation souhaitée d'un système peut être trop contraignant et inexploitable en pratique. Par exemple, si notre architecture de contrôle commute de l'évitement d'obstacles à l'attraction vers la cible, on ne peut pas imposer que le contrôleur d'attraction vers la cible reste toujours actif au moins pendant un temps τ_d. Effectivement, il y a une contrainte plus forte : qu'en est il si le robot rencontre un autre obstacle ? s'il n'active pas le contrôleur d'évitement d'obstacles à temps, il risque la collision. Un contrôle stable au prix de détériorer la sûreté des robots n'est sans doute pas intéressant.

Le résultat du TT a alors été étendu à la notion du *« Temps de Tenue Moyen »* TTM dans [Hespanha 99]. L'idée est que si une commutation rapide s'impose

inévitablement (le système actuel détériore le système global de façon inacceptable), elle est autorisée sous réserve de commuter suffisamment lentement plus tard. Ainsi, cette notion de *« TTM »* se base sur une constante N_0. Il s'agit du nombre de commutations successives possibles sans respecter le TT τ_d et sans pour autant provoquer des effets indésirables [Hespanha 99].

Dans le cas de l'architecture de contrôle proposée, il est très difficile, voire impossible de quantifier cette constante. La nature réactive de l'architecture de contrôle proposée (cf. Section 1.7, page 49) impose que le robot ne connaisse pas l'environnement dans lequel il évolue. Imposer ce nombre sera trop restrictif, car on imposera le nombre de commutations entre les contrôleurs ce qui revient par exemple à imposer le nombre d'obstacles que le robot peut éviter, voire leurs dispositions.

Dans le souci de garder une navigation réactive, et de n'imposer aucune contrainte sur la nature des obstacles à éviter, ces théorèmes peuvent s'avérer trop restrictifs. Ainsi, une architecture de contrôle basée sur *Fonctions de Lyapunov Multiples* est proposée [Benzerrouk 08], [Benzerrouk 09]. Bien qu'elle soit asymptotiquement stable, elle n'est pas adaptée à tous les types d'environnements et impose des restrictions sur la disposition des obstacles. Afin de lever ces restrictions, un autre théorème est exploré. Il s'agit du théorème de la stabilité faible.

3.2.3 Stabilité faible

Comme précédemment discuté, et pour prouver la stabilité asymptotique souhaitée, il est nécessaire d'imposer certaines contraintes incompatibles avec la navigation réactive : ainsi, pour former la séquence décroissante des fonctions de Lyapunov et assurer les conditions du théorème FLM (cf. Section 3.2.1), il faut laisser chaque contrôleur actif pendant un temps suffisamment large pour que la fonction en question atteigne les valeurs souhaitées. Cela se traduit à titre d'exemple par l'espacement des obstacles pour laisser le contrôleur d'attraction vers la cible actif assez longtemps. Nous verrons que le théorème de la stabilité faible nous permet de prouver la stabilité de l'architecture de contrôle sans contraintes sur l'environnement.

Ce théorème est le plus souvent utilisé dans les preuves de stabilité des systèmes mécaniques [Brogliato 97]. Dans [Bourgeot 04], le problème de la poursuite de trajectoires pour un système mécanique soumis à des contraintes unilatérales est étudié. Pour expliquer et illustrer ces systèmes, on imagine une balle rigide soumise à la force de son poids. La contrainte correspondra alors à une surface verticale coupant la trajectoire de cette balle. Ceci provoquera des rebonds de la

balle dissipant ainsi son énergie jusqu'à sa stabilisation sur la surface. Le théorème de la stabilité faible est alors valable durant cette phase de rebonds.

Proposition Soit un système évoluant sur un intervalle temporel $I = [t_i, t_f]$ avec plusieurs impacts (commutations) aux instants $t_k \in [t_i, t_f]$. Soit V la fonction de Lyapunov du système pendant toute son évolution. Si :

1. $V(x(t_f)) \leq V(x(t_i))$,
2. et $\dot{V}(x(t)) \leq 0$ pendant toute l'évolution du système en dehors des instants de commutations.

Alors le système est faiblement stable pendant toute son évolution. Si de plus $\dot{V}(x(t)) < 0$ quelque soit $x(t) \neq 0$; alors le système est asymptotiquement faiblement stable sur tout son intervalle d'évolution.

La stabilité faible impose que la fonction de Lyapunov soit toujours décroissante pendant l'évolution du système (point 2). Cependant, elle peut croître temporairement sur les phases d'impacts (commutations) sous réserve de ne pas dépasser sa valeur initiale (point 1) (cf. Figure 3.9). Ainsi, pour le cas de la balle en phase de rebonds, et si la fonction de Lyapunov considérée est son énergie potentielle, celle-ci augmente au rebond, car la balle va regagner de l'altitude par rapport à la surface de contrainte.

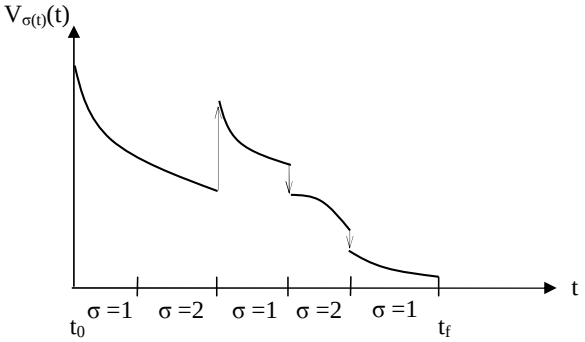


FIGURE 3.9 – Un exemple d'évolution de la fonction de Lyapunov d'un système faiblement stable ($V(x(t_f)) \leq V(x(t_0))$).

3.2.4 Discussion

Nous avons vu les trois types de stabilité des systèmes hybrides qui peuvent nous servir à prouver celle de l'architecture de contrôle proposée. Ainsi, en l'ab-

sence d'une seule fonction de Lyapunov décroissante tout au long de l'évolution du système, il est possible d'utiliser les fonctions de Lyapunov des systèmes élémentaires qui le forment.

Ainsi, le théorème FLM permet de prouver la stabilité asymptotique sous réserve que chaque fonction de Lyapunov des systèmes élémentaires forme une séquence de fins d'intervalles décroissante. Cela revient donc à laisser le système actif afin que sa fonction décroisse assez pour que sa valeur entre dans la séquence monotone. De cette idée est né le *« Temps de Tenue » TT*. Il s'agit du temps nécessaire à attendre avant la prochaine commutation. Ainsi, l'effet de la dernière commutation disparaît et même la séquence monotone imposée par le FLM peut être retrouvée. Le TT peut être très contraignant en pratique car le système actif peut détériorer le fonctionnement du système global si on empêche une commutation pour des raisons de stabilité (empêcher la commutation au contrôleur d'évitement d'obstacles peut entraîner la collision du robot). Le TTM contraint aussi le nombre de commutation ce qui dans notre application se traduit par de fortes contraintes sur l'environnement.

Nous prouverons dans ce qui suit que l'architecture de contrôle proposée peut être asymptotiquement stable. L'idée est d'essayer de diminuer le TT en jouant sur la vitesse de convergence de la fonction de Lyapunov. Celle-ci décroît alors plus rapidement afin d'atteindre la séquence monotone imposée par le théorème FLM. Cependant, afin de concilier stabilité et navigation réactive dans un environnement pouvant être très encombré, la stabilité faible formera un bon compromis avec la sécurité des robots. L'idée est qu'au moment des commutations, la fonction de Lyapunov du système global peut augmenter (ceci arrive inévitablement avec la discontinuité de la consigne due au changement de contrôleur), néanmoins tant que l'environnement permet au robot d'atteindre sa cible (le nombre des obstacles n'est pas contraint mais reste fini), il finit par se stabiliser sur cette cible.

3.3 Stabilité de l'architecture de contrôle proposée

L'architecture de contrôle proposée impose une transition entre les deux contrôleurs : le passage du contrôleur d'attraction vers la cible à l'évitement d'obstacles et vice versa se traduit par une discontinuité de consigne, notamment la consigne d'angle θ_S calculée selon le contrôleur actif (cf. Section 2.3, page 62). Même si nous allons prouver que la loi de commande proposée est stable utilisant un seul contrôleur, le changement de contrôleur pourrait altérer cette stabilité. Comme chaque robot est découplé des autres entités au niveau de son contrôle, qui rappelons-le est distribué, seule la stabilité de l'architecture de contrôle au niveau d'un seul

robot est étudiée. Nous étudierons aussi les domaines d'évolution de la dynamique des consignes (angles provenant des contrôleurs) afin qu'elles restent atteignables par les robots en fonction des contraintes structurelles (vitesses maximales, accélérations maximales) de ceux-ci. La stabilité sera donc assurée dans ces domaines.

3.3.1 Stabilité de la loi de commande utilisant un seul contrôleur

Pour étudier la stabilité de la loi de commande proposée, rappelons d'abord celle-ci

$$v_i = v_{max} - (v_{max} - v_C)e^{-(d_{S_i}^2/\sigma^2)} \tag{3.1a}$$

$$\omega_i = \omega_{S_i} + k\tilde{\theta}_i \tag{3.1b}$$

où
- v_i et ω_i sont les vitesses linéaire et angulaire respectivement,
- v_{max} est la vitesse linéaire maximale,
- σ, k sont des constantes positives,
- v_C est la vitesse linéaire de la cible,
- $\omega_{S_i} = \dot{\theta}_{S_i}$, est la variation de l'angle consigne (qu'il provienne de l'attraction vers la cible ou de l'évitement d'obstacles).

Par mesure de simplicité, nous n'introduisons pas la fonction de pénalité (cf. Section 2.3.2.3) car nous verrons qu'elle n'interviendra pas. Rappelons que l'erreur $\tilde{\theta}_i$ s'écrit

$$\tilde{\theta}_i = \theta_{S_i} - \theta_i \tag{3.2}$$

où θ_{S_i} est la consigne d'angle provenant de l'un des deux contrôleurs (cf. Figure 2.7, page 62). En dérivant

$$\dot{\tilde{\theta}}_i = \omega_{S_i} - \omega_i \tag{3.3}$$

Dans le chapitre 2, nous avons vu que chaque contrôleur délivre deux consignes (cf. Section 2.3.4) :
- une consigne de distance pour le contrôleur d'attraction vers la cible. Cette distance est considérée nulle pour l'évitement d'obstacles,
- une consigne d'angle θ_{S_i}.

On remarque que les deux contrôleurs partagent une consigne de la même nature : il s'agit de la consigne d'angle θ_{S_i}. Dans le cas du contrôleur d'attraction vers la cible, nous avons prouvé que le suivi de cette consigne conduit à la

convergence du robot vers sa cible (cf. Section 2.3.1). Pour prouver la stabilité de la loi de commande, la fonction de Lyapunov choisie est alors une fonction de l'erreur d'orientation du robot.

Une fonction de Lyapunov candidate est alors de la forme

$$V = \frac{1}{2}\tilde{\theta_i}^2 \tag{3.4}$$

La loi de commande est asymptotiquement stable si $\dot{V} < 0$.

En remplaçant l'équation (3.3) dans la loi de commande relative à la vitesse angulaire (3.1b), on obtient

$$\dot{\tilde{\theta}}_i = -k\tilde{\theta}_i \tag{3.5}$$

Ce qui implique que $\dot{V}$ devient

$$\dot{V} = -k\tilde{\theta}_i^2 < 0 \tag{3.6}$$

quelque soit $\tilde{\theta}_i \neq 0$ (rappelons que $k > 0$).

Ce résultat est obtenu grâce à la connaissance et la prise en compte de la variation de la consigne ω_{S_i} dans la loi de commande. Cela donne la relation 3.5 dont la solution permet une décroissance exponentielle de l'erreur d'orientation (cf. Section 2.36).

Cependant, nous rappellons que cette loi de commande sera appliquée à des robots mobiles non holonomes possédant des contraintes structurelles (vitesses maximales, accélérations maximales). Pour obtenir la relation 3.6, Il faut alors que le robot soit capable d'atteindre les vitesses linéaire et angulaire imposées par la loi de commande dans la limite de ses contraintes.

Nous avons vu que la loi régissant la vitesse linéaire du robot prend en compte sa vitesse maximale (cf. Equation 3.1a) de telle sorte qu'il ne la dépasse jamais. Sous cette contrainte, nous avons prouvé que le robot atteint sa cible sous réserve que la vitesse de celle-ci soit inférieure à sa vitesse maximale (cf. Section 2.3.1, page 63). De même, le robot mobile a une vitesse angulaire maximale que nous notons ω_{max}. La contrainte sur la vitesse linéaire étant déjà déterminée, nous nous intéressons à présent à contraindre la variation angulaire de la consigne (provenant des deux contrôleurs) afin qu'elle reste atteignable par le robot.

En effet, la loi de commande proposée est théoriquement stable. Remarquons d'après l'étude qui précède que cela provient de la relation 3.5. Celle-ci décrit un système linéaire du premier ordre (pour l'erreur d'orientation). Il en découle alors que cette étude n'est valable que dans la zone où cette erreur est effectivement

linéaire. Cela revient à ce que la vitesse angulaire nécessaire calculée doit être admissible. Une vitesse angulaire est dite admissible si elle satisfait

$$|\omega_i| \leq \omega_{max} \tag{3.7}$$

Où $\omega_{max} > 0$ est la vitesse angulaire maximale du robot.

On distingue deux cas : l'activation du contrôleur d'attraction vers la cible ou l'activation de l'évitement d'obstacles. Dans les deux cas, il faut trouver la variation de la consigne telle qu'elle reste atteignable par le robot malgré ses contraintes de non-holonomie et de vitesse angulaire maximale. Cette variation de la consigne est $\dot{\theta}_S = \omega_{S_i}$ (cf. Equation 3.3). L'idée est alors de trouver des conditions permettant de borner ω_{S_i} en fonction des contraintes du robot pour que la relation 3.7 soit toujours vérifiée. De plus, ω_{S_i} doit prendre en considération certains paramètres découlant de la nature réactive de la navigation imposée : ainsi, dans le cas du contrôleur d'attraction vers la cible dynamique, la borne de $\omega_{S_{at}}$ doit prendre en considération tous les robots de la formation. En effet, elle doit être admissible pour tous les robots. De même, dans le cas de l'évitement d'obstacles, $\omega_{S_{oa}}$ doit être indépendante des tailles et des dispositions des obstacles. Il s'agit donc de trouver ω_{S_i} qui vérifie toutes ces conditions.

En remplaçant l'expression de la vitesse angulaire (cf. Equation 3.1b) dans la relation 3.7, on obtient :

$$\left|\omega_{S_i} + k\tilde{\theta}_i\right| \leq \omega_{max} \tag{3.8}$$

or, on sait que

$$\left|\omega_{S_i} + k\tilde{\theta}_i\right| \leq |\omega_{S_i}| + \left|k\tilde{\theta}_i\right|$$

donc, pour trouver l'ensemble des valeurs de ω_{S_i} vérifiant 3.8, on propose d'utiliser la relation

$$|\omega_{S_i}| + k\left|\tilde{\theta}_i\right| \leq \omega_{max} \tag{3.9}$$

En effet, les valeurs vérifiant la relation 3.9, vérifient forcément 3.8.

Pour que la consigne ω_{S_i} soit toujours admissible par tous les robots de la formation, il suffit donc d'avoir

$$|\omega_{S_i}| \leq \omega_{max} - k\left|\tilde{\theta}_i\right| \tag{3.10}$$

Or, il faut vérifier la relation 3.10 pour toutes les entités robotiques de la formation en fonction des paramètres cités précédemment (prise en compte de toutes les entités robotiques et des variations possibles de l'environnement). Cela signifie que 3.10 doit être indépendante de $\tilde{\theta}_i$ relative à chaque robot. La relation 3.10 devient alors

$$|\omega_{S_i}| \leq \min_{i=1..N}(\omega_{max} - k\left|\tilde{\theta}_i\right|) \tag{3.11}$$

le minimum $\min_{i=1..N}(\omega_{max} - k\left|\tilde{\theta}_i\right|)$ est tel que

$$\min_{i=1..N}(\omega_{max} - k\left|\tilde{\theta}_i\right|) = \omega_{max} - \max_{i=1..N}(k\left|\tilde{\theta}_i\right|)$$

L'erreur d'orientation maximale possible correspond à un robot dont le cap réel a une orientation opposée à la consigne, ce qui signifie

$$\max_{i=1..N}(\left|\tilde{\theta}_i\right|) = \pi$$

On propose alors d'imposer à la variation de la consigne, la contrainte suivante :

$$\min_{i=1..N}(\omega_{max} - k\left|\tilde{\theta}_i\right|) = \omega_{max} - \lambda\pi \tag{3.12}$$

avec λ est un réel positif. La relation 3.11 devient donc

$$|\omega_{S_i}| \leq \omega_{max} - \lambda\pi \tag{3.13}$$

Maintenant que cette relation est obtenue, il faut déterminer les valeurs possibles de λ. Pour cela, on propose de prendre en considération l'accélération angulaire maximale des robots. Ainsi, pour assurer une continuité de la consigne en dehors des moments de commutations, on impose que la variation maximale de la vitesse de la consigne ω_{S_i}, en n période d'échantillonnage T, ne dépasse pas l'accélération angulaire maximale des robots notée $\dot{\omega}_{max}$.

La variation maximale de ω_{S_i} notée $\Delta\omega_{S_i}$ correspond au passage de la consigne maximale $\max(\omega_{S_i})$ à son opposée. Cela se traduit par la relation

$$\Delta\omega_{S_i} = \max(\omega_{S_i}) - (-\max(\omega_{S_i})) \tag{3.14}$$

On doit alors avoir

$$\frac{\max(\omega_{S_i}) + \max(\omega_{S_i})}{nT} \leq \dot{\omega}_{max} \tag{3.15}$$

or, d'après la relation 3.13

$$\max(\omega_{S_i}) = \omega_{max} - \lambda\pi$$

En remplaçant dans la relation 3.15, on obtient

$$\frac{2(\omega_{max} - \lambda\pi)}{nT} \leq \dot{\omega}_{max} \tag{3.16}$$

Ainsi la valeur de λ doit être telle que

$$\lambda \geq \frac{2\omega_{max} - \dot{\omega}_{max} nT}{2\pi} \tag{3.17}$$

De plus, d'après la relation 3.13, on doit avoir

$$\omega_{max} - \lambda\pi > 0$$

Il en résulte

$$\lambda < \frac{\omega_{max}}{\pi}$$

Les valeurs possibles de λ sont alors telles que

$$\frac{2\omega_{max} - \dot{\omega}_{max} nT}{2\pi} \leq \lambda < \frac{\omega_{max}}{\pi} \tag{3.18}$$

Dans ce qui suit, on s'intéresse à chaque contrôleur séparément. En effet, on voudra trouver les conditions permettant à chacun de délivrer une consigne d'angle dont la variation ω_{S_i} reste admissible (cf. Equation 3.7).

3.3.1.1 Contrôleur d'attraction vers une cible dynamique

Pour que la cible reste atteignable, il faut que la variation de la consigne d'orientation appliquée au robot soit dans le domaine admissible (cf. Equation 3.13). On a alors :

$$|\omega_{S_{at}}| \leq \omega_{max} - \lambda\pi \tag{3.19}$$

Et donc

$$\left|\dot{\theta}_{S_{at}}\right| \leq \omega_{max} - \lambda\pi \tag{3.20}$$

Calculons alors $\dot{\theta}_{S_{at}}$ en reprenant la formule de $\theta_{S_{at}}$ (cf. Equation 2.22, page 66) :

$$\theta_{S_{at}} = \arcsin(b\sin(\theta_C - \gamma_i)) + \gamma_i \tag{3.21}$$

avec $b = \frac{v_C}{v_i}$ et $b \in [0, 1]$.

En dérivant

$$\dot{\theta}_{S_{at}} = \frac{\frac{d}{dt}\left[b\sin(\theta_C - \gamma_i)\right]}{\sqrt{1 - (b\sin(\theta_C - \gamma_i))^2}} + \dot{\gamma}_i \tag{3.22}$$

Sachant que $\theta_{S_{at}}$ est obtenue en imposant $\dot{\gamma}_i = 0$ (cf. Section 2.3.1), $\dot{\theta}_{S_{at}}$ est en fait

$$\dot{\theta}_{S_{at}} = \frac{\frac{d}{dt}\left[b\sin(\theta_C - \gamma_i)\right]}{\sqrt{1 - (b\sin(\theta_C - \gamma_i))^2}} \tag{3.23}$$

On remarque que la variation de la consigne n'est pas définie si

$$(b\sin(\theta_C - \gamma_i))^2 = 1 \tag{3.24}$$

l'équation (3.24) est alors vérifiée si on a simultanément

$$\begin{cases} b = & 1 \\ \sin(\theta_C - \gamma_i) = & \pm 1 \end{cases} \tag{3.25}$$

Or, nous avons $b = 1$ uniquement quand d_{S_i} (distance séparant le robot i de sa cible) tend vers 0 (d'après la loi régissant la vitesse linéaire du robot (cf. Equation 3.1a, page 118)). En d'autres termes, cette singularité ne peut arriver que lorsque le robot a atteint la cible. En reprenant l'expression de la consigne (cf. Equation 3.21), avec $b = \frac{v_C}{v_i} = 1$ (au voisinage de $d_{S_i} \approx 0$), on obtient

$$\theta_{S_{at}} \approx \arcsin(\sin(\theta_C - \gamma_i)) + \gamma_i \tag{3.26}$$

On retrouve alors les deux cas (cf. Section 2.3.1, page 63)

1. $(\theta_C - \gamma_i) \in [\frac{-\pi}{2}, \frac{\pi}{2}]$ (cible s'éloignant) :
   la relation 3.21 donne
   $$\theta_{S_{at}} \approx \theta_C, \forall \gamma_i$$
2. $(\theta_C - \gamma_i) \in]\frac{\pi}{2}, \frac{3\pi}{2}[$ (cible s'approchant) :

$$\begin{array}{rcl} \theta_{S_{at}} & \approx & \pi - (\theta_C - \gamma_i) + \gamma_i \\ \theta_{S_{at}} & \approx & \pi + 2\gamma_i - \theta_C \end{array} \tag{3.27}$$

   On a vu que ce deuxième cas rejoint le premier une fois la cible dépassée et sous réserve que le mouvement de celle-ci reste non-holonome (cf. Section 2.3.1), ce qui est le cas ici grâce à la contrainte décrite par la relation 3.12.

Pour éviter la singularité en pratique, on propose alors de définir un cercle virtuel de rayon ρ au voisinage de $d_{S_i} \approx 0$. Ainsi, la consigne $\theta_{S_{at}}$ est définie comme suit

$$\theta_{\mathrm{S_{at}}} = \begin{cases} \arcsin(b\sin(\theta_C - \gamma_i)) + \gamma_i & si \quad d_{S_i} \geq \rho \\ \theta_C & si \quad d_{S_i} < \rho \; et \; (\theta_C - \gamma_i) \in [\frac{-\pi}{2}, \frac{\pi}{2}] \\ \pi + 2\gamma_i - \theta_C & si \quad d_{S_i} < \rho \; et \; (\theta_C - \gamma_i) \in]\frac{\pi}{2}, \frac{3\pi}{2}[\end{cases} \tag{3.28}$$

Comme $\theta_{S_{at}}$ est redéfinie (cf. Equation 3.29), sa dérivée $\dot{\theta}_{S_{at}}$ doit l'être aussi. On a alors (γ_i constant)

$$\dot{\theta}_{\mathrm{S_{at}}} = \begin{cases} \frac{[b\sin(\theta_C - \gamma_i)]'}{\sqrt{1-(b\sin(\theta_C-\gamma_i))^2}} & si \quad d_{S_i} \geq \rho \\ \dot{\theta}_C & si \quad d_{S_i} < \rho \; et \; (\theta_C - \gamma_i) \in [\frac{-\pi}{2}, \frac{\pi}{2}] \\ -\dot{\theta}_C & si \quad d_{S_i} < \rho \; et \; (\theta_C - \gamma_i) \in]\frac{\pi}{2}, \frac{3\pi}{2}[\end{cases} \tag{3.29}$$

Pour que tous les robots de la formation soient capables d'atteindre leurs cibles respectives, il faut que la variation de consigne de chaque robot $\dot{\theta}_{S_{at}}$ vérifie la relation 3.20, quelque soit d_{S_i}.

Dans le cas où $d_{S_i} < \rho$, on a (cf. Equation 3.29)

$$\left|\dot{\theta}_{S_{at}}\right| = \left|\dot{\theta}_C\right|$$

Ainsi, pour les entités robotiques situées au voisinage de leurs cibles, on doit avoir

$$\left|\dot{\theta}_C\right| \leq \omega_{max} - \lambda\pi \tag{3.30}$$

Pour les autres robots, il s'agit d'étudier le premier cas (où $d_{S_i} \geq \rho$) (cf. Figure 3.29). On note

$$\dot{\theta}_{S_{at}} = \frac{Num(\dot{\theta}_{S_{at}})}{Den(\dot{\theta}_{S_{at}})} =$$

En développant $Num(\dot{\theta}_{S_{at}})$ et en remplaçant b par $\frac{v_C}{v_i}$:

$$\begin{aligned} Num(\dot{\theta}_{S_{at}}) &= \frac{db}{dt}\sin(\theta_C - \gamma_i) + b\frac{d}{dt}(\sin(\theta_C - \gamma_i)) \\ &= \frac{\dot{v}_C v_i - v_C \dot{v}_i}{v_i^2}\sin(\theta_C - \gamma_i) + (\frac{v_C}{v_i})(\dot{\theta}_C - \dot{\gamma}_i)\cos(\theta_C - \gamma_i) \end{aligned}$$

γ_i étant constant, donc $\dot{\gamma}_i = 0$. Ainsi, on a

$$Num(\dot{\theta}_{S_{at}}) = \frac{\dot{v}_C v_i - v_C \dot{v}_i}{v_i^2}\sin(\theta_C - \gamma_i) + (\frac{v_C}{v_i})(\dot{\theta}_C)\cos(\theta_C - \gamma_i)$$

Comme on doit avoir

$$\dot{\theta}_{S_{at}} = \frac{Num(\dot{\theta}_{S_{at}})}{Den(\dot{\theta}_{S_{at}})} \leq \omega_{max} - \lambda\pi \tag{3.33}$$

alors

$$Num(\dot{\theta}_{S_{at}}) \leq (\omega_{max} - \lambda\pi)Den(\dot{\theta}_{S_{at}}) \tag{3.34}$$

L'objectif étant de contraindre la dynamique angulaire de la cible pour qu'elle reste atteignable par tous les robots, il s'agit alors de trouver l'ensemble des valeurs de $\dot{\theta}_C$ vérifiant la relation 3.34.

L'expression du $Num(\dot{\theta}_{S_{at}})$ étant fastidieuse à développer, on cherche alors sa borne supérieure. Sachant que

$$|a\cos(\alpha) + b\sin(\alpha)| \leq \sqrt{a^2 + b^2} \quad \forall a, b, \alpha \in \mathbb{R}$$

on a alors

$$Num(\dot{\theta}_{S_{at}}) \leq \sqrt{(\frac{\dot{v}_C v_i - v_C \dot{v}_i}{v_i^2})^2 + ((\frac{v_C}{v_i})(\dot{\theta}_C))^2} \tag{3.35}$$

Ainsi, et d'après le résultat décrit par la relation 3.35, pour trouver l'ensemble des valeurs de $\dot{\theta}_C$ vérifiant 3.34, il suffit de trouver celles vérifiant la relation

$$\sqrt{(\frac{\dot{v}_C v_i - v_C \dot{v}_i}{v_i^2})^2 + ((\frac{v_C}{v_i})(\dot{\theta}_C))^2} \leq (\omega_{max} - \lambda\pi) Den(\dot{\theta}_{S_{at}}) \tag{3.36}$$

En effet, celles vérifiant la relation 3.36, vérifient aussi 3.33. En remplaçant $Den(\dot{\theta}_{S_{at}})$

$$\sqrt{(\frac{\dot{v}_C v_i - v_C \dot{v}_i}{v_i^2})^2 + ((\frac{v_C}{v_i})(\dot{\theta}_C))^2} \leq (\omega_{max} - \lambda\pi)\sqrt{1 - ((\frac{v_C}{v_i})\sin(\theta_C - \gamma_i))^2}$$

En élevant les deux membres de l'inéquation au carré, on obtient

$$(\frac{\dot{v}_C v_i - v_C \dot{v}_i}{v_i^2})^2 + ((\frac{v_C}{v_i})(\dot{\theta}_C))^2 \leq (\omega_{max} - \lambda\pi)^2(1 - ((\frac{v_C}{v_i})\sin(\theta_C - \gamma_i))^2)$$

on a alors

$$((\frac{v_C}{v_i})\dot{\theta}_C)^2 \leq (\omega_{max} - \lambda\pi)^2(1 - ((\frac{v_C}{v_i})\sin(\theta_C - \gamma_i))^2) - (\frac{\dot{v}_C v_i - v_C \dot{v}_i}{v_i^2})^2$$

ce qui revient à

$$(\dot{\theta}_C)^2 \leq [(\omega_{max} - \lambda\pi)^2(1 - ((\frac{v_C}{v_i})\sin(\theta_C - \gamma_i))^2) - (\frac{\dot{v}_C v_i - v_C \dot{v}_i}{v_i^2})^2](\frac{v_i}{v_C})^2 \tag{3.37}$$

Pour que toutes les cibles du corps virtuel restent atteignables par tous les robots, il faut que la variation de leurs vitesses angulaires $\dot{\theta}_C$ soit inférieure au minimum du membre droit de cette inéquation. On remarque que ce minimum peut être atteint lorsque $b = \frac{v_C}{v_i} \rightarrow 1$. Or, cette condition n'est vraie qu'au voisinage de $d_{S_i} \approx 0$.

Rappelons que le cas où $v_i \rightarrow v_C$ correspond à $d_{S_i} < \rho$ et est déjà traité par la relation 3.30. Cependant, étudions cette relation à ce voisinage afin de déduire le comportement du robot à $d_{S_i} \approx \rho$.

D'abord, calculons $\dot{v}_i$ en dérivant son expression (cf. Equation 3.1a)

$$\dot{v}_i = \dot{v}_C e^{-\frac{d_{S_i}^2}{\sigma^2}} - (v_{max} - v_C)\frac{-2d_{S_i}\dot{d}_{S_i}}{\sigma^2}e^{-\frac{d_{S_i}^2}{\sigma^2}} \tag{3.38}$$

Lorsque $d_{S_i} \rightarrow 0$, et sachant que $\dot{d}_{S_i}$ est bornée (cf. Equations 2.24, 2.25, pages 68, 68), on déduit de 3.38 que

$$\dot{v}_i = \dot{v}_C$$

En remplaçant dans 3.37, la contrainte sur la variation de l'orientation devient

$$(\dot{\theta}_C)^2 \leq [(\omega_{max} - \lambda\pi)^2(1 - (\sin^2(\theta_C - \gamma_i)))$$

On remarque que cette relation dépend de $\sin(\theta_C - \gamma_i)$ qui peut s'annuler si $(\theta_C - \gamma_i) = \frac{\pi}{2} + k\pi$, avec $k \in 0, 1$ (rappelons que $\theta_C \in]-\pi, \pi]$ (resp. γ_i)). Ainsi, au voisinage de $d_{S_i} \approx \rho$, on aura (cf. Inéquation 3.37)

$$(\dot{\theta}_C)^2 \leq [(\omega_{max} - \lambda\pi)^2(1 - (\frac{v_C}{v_i(\rho)}\sin^2(\theta_C - \gamma_i)))\frac{v_i(\rho)}{v_C}$$

Où $v_i(\rho)$ est la vitesse linéaire du robot au voisinage de ρ (on négligera (par abus) le terme $\frac{\dot{v}_C v_i - v_C \dot{v}_i}{v_i^2}$ en considérant que $v_i(\rho) \rightarrow v_C$ (ρ étant choisi au voisinage de $d_{S_i} \approx 0$) (cf. Figure 3.1a).

D'après cela, on aura alors

$$(\dot{\theta}_C)^2 \leq [(\omega_{max} - \lambda\pi)^2(1 - (\frac{v_C}{v_i(\rho)}\sin^2(\theta_C - \gamma_i)))\frac{v_i(\rho)}{v_C}$$

avec $(1 - (\frac{v_C}{v_i(\rho)}\sin^2(\theta_C - \gamma_i)))$ pouvant être nul.

Pour les robots dont les distances d_{S_i} sont telles que $d_{S_i} \leq \rho$, on a (cf. Figure 3.29)

$$(\dot{\theta}_C)^2 \leq (\omega_{max} - \lambda\pi)^2 \tag{3.39}$$

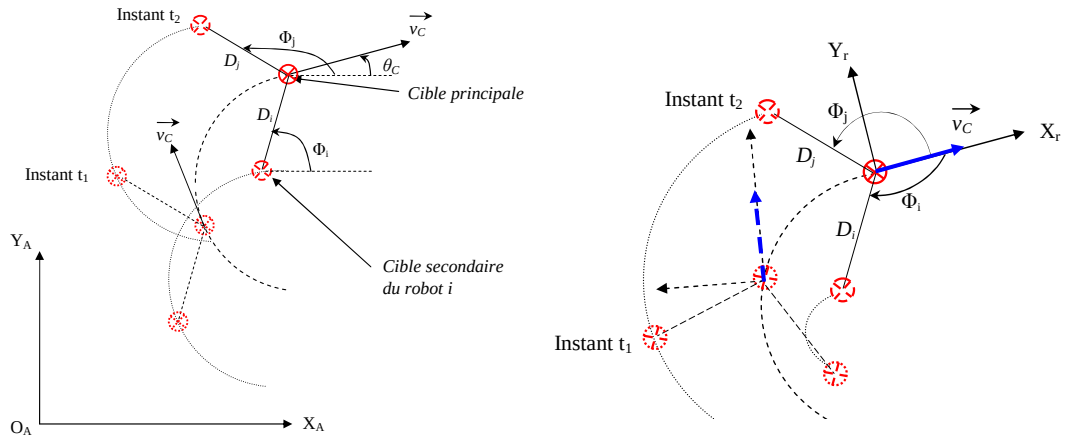
D'après ce qui précède, on déduit donc que la dynamique du corps virtuel doit passer par deux phases :

1. une phase transitoire où les robots n'ont pas atteint la formation : dans cette phase, la variation de sa vitesse angulaire doit obéir à la relation $\dot{\theta}_C = 0$ afin de respecter les contraintes de tous les robots de la formation,
2. une fois la formation atteinte par tous les robot, plus de flexibilité peut lui être offerte en obéissant à la relation 3.39.

Cette étude nous a donc permis de définir la variation de l'orientation de la cible dynamique autorisée afin qu'elle reste atteignable par les robots. Avant de réaliser la même étude avec l'angle consigne provenant de l'évitement d'obstacles, nous abordons brièvement le cas où la formation est définie dans un repère relatif.

3.3.1.2 Cas d'un repère relatif

On note que la forme de notre structure virtuelle est définie dans un repère absolu. Ceci entraîne alors une dynamique unique des cibles (vitesses linéaire et angulaire v_C et $\dot{\theta}_C$). Toutefois, si la formation est définie dans un repère relatif, chaque cible aura une dynamique propre. Cette dernière se décrit en fonction de celle de la cible principale, mais aussi de sa position relative dans la formation (et donc en fonction de (D_i, Φ_i))(cf. Equation 3.40). La figure 3.10 montre cette différence de dynamique en fonction du repère de définition, mais aussi sa répercussion sur la formation globale. On remarque alors que si les trajectoires de toutes les cibles sont les mêmes que celle de la cible principale (translatées en fonction de (D_i, Φ_i)) dans le cas d'une définition dans un repère absolu, ce n'est pas le cas pour une définition dans un repère relatif.



(a) Repère absolu (b) Repère relatif

FIGURE 3.10 – Définition de la structure virtuelle.

Notons que pour un repère absolu, on remarque des changements au niveau de la forme globale de la structure en fonction de la dynamique de la cible principale (cf. Figure 3.10(a)). Ainsi, a l'instant t_1, les deux cibles secondaires se trouvent à gauche de la cible principale et de sa direction. A l'instant t_2, elles se trouvent derrière celle-ci. Même si les calculs du cas d'un repère relatif ne seront pas détaillés dans cette thèse, nous donnons les variations de la position de chaque cible i notées $(\dot{x}_{C_i}, \dot{y}_{C_i})$ par rapport à une cible principale de variations de position $(\dot{x}_C, \dot{y}_C)$ et d'orientation θ_C.

La position de chaque cible dans la formation s'écrit

$$\begin{cases} x_{C_i} = x_C + D_i \cos(\Phi_i + \theta_C) \\ y_{C_i} = y_C + D_i \sin(\Phi_i + \theta_C) \end{cases} \tag{3.40}$$

En dérivant

$$\begin{cases} \dot{x}_{C_i} = \dot{x}_C - D_i \dot{\theta}_C \sin(\Phi_i + \theta_C) \\ \dot{y}_{C_i} = \dot{y}_C + D_i \dot{\theta}_C \cos(\Phi_i + \theta_C) \end{cases} \tag{3.41}$$

Pour déterminer la dynamique de la cible admissible, il suffit alors de reprendre les résultats précédents développés, en considérant les variations de la positions de la cible i décrites par le système d'équations 3.41. Cependant, il faudra prendre en considération la dynamique de toutes les cibles dans la définition des contraintes sur v_C et notamment $\dot{\theta}_C$ (cf. Equation 3.39). En effet, celle-ci a des conséquences directes sur v_{C_i} et $\dot{\theta}_{C_i}$. Il faudra que chacune de ces dernières reste atteignable pour l'entité correspondante.

On note aussi qu'une définition dans un repère relatif ouvre une possibilité de dynamique interne à la structure virtuelle afin de pouvoir changer la forme désirée. Il faudra alors considérer les variations $(\dot{D}_i, \dot{\Phi}_i)$ dans l'équation 3.41 en fonction de cette dynamique.

Dans le paragraphe suivant, on note

$$P = |\omega_{max} - \lambda\pi| \tag{3.42}$$

3.3.1.3 Simulation et discussion

Nous proposons de montrer l'utilité de borner la variation de l'orientation de la cible $\dot{\theta}_C$ pour qu'elle soit atteignable par le robot. Pour cela, nous simulons un robot mobile qui doit suivre une cible dynamique. Nous posons : $\omega_{max} = 3rd/s$. Nous ne voulons pas que le robot dépasse une accélération angulaire maximale

$\dot{\omega}_{S_i} = 1.2rd/s^{-2}$. Nous voulons aussi que la variation maximale de la vitesse angulaire de la consigne $\Delta\omega_{S_i}$ (cf. Equation 3.14) ne dépasse pas l'accélération maximale $\dot{\omega}_{S_i}$ en 20 périodes d'échantillonnage (nous choisissons $n = 20$). D'après cela, il est facile de trouver les bornes de λ (cf. Equation 3.18). Nous choisissons ainsi $\lambda = 0.6s^{-1}$. Le calcul de P (cf. Equation 3.42) donne $P = 1.1$ Nous simulons le contrôleur d'attraction vers la cible avec une cible dynamique dont la variation est d'abord comprise dans le domaine délimité par P (cf. Figure 3.11). La trajectoire du robot (cf. Figure 3.12) montre qu'il suit la cible dynamique tant que $\dot{\theta}_C \leq P$ (jusqu'à l'instant t_1). L'évolution de la distance séparant le robot de la cible (cf. Figure 3.11) le confirme. De même, la fonction de Lyapunov (cf. Equation 3.4, page 119) est toujours décroissante jusqu'à l'instant t_1. A partir de t_1, nous simulons une autre dynamique de la cible telle que $\dot{\theta}_C$ dépasse P. Le saut observé dans la distance d_{S_i} est traduit par une discontinuité (saut) dans la dynamique de la cible à t_1. La distance d_{S_i} décroît de nouveau mais elle n'est pas stabilisée, et on observe des fluctuations. La même évolution est remarquée au niveau de la fonction de Lyapunov. En effet, quand $\dot{\theta}_C$ dépasse P, la consigne générée est inatteignable pour le robot. La trajectoire du robot (instant t_2) illustre son comportement face aux changements trop importants de la dynamique de la cible.

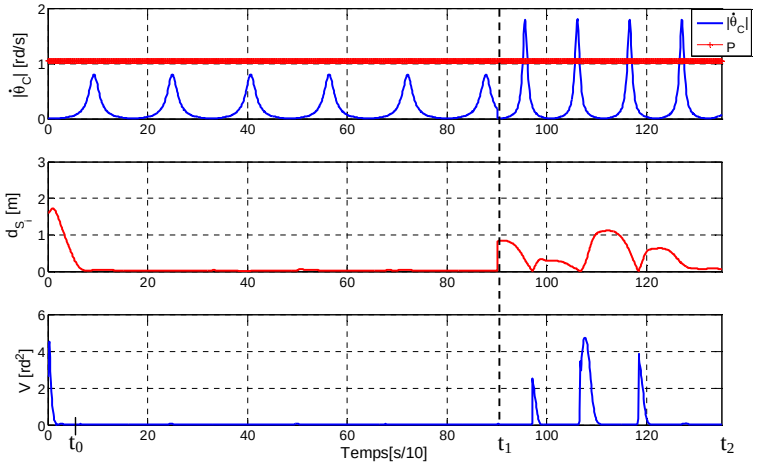
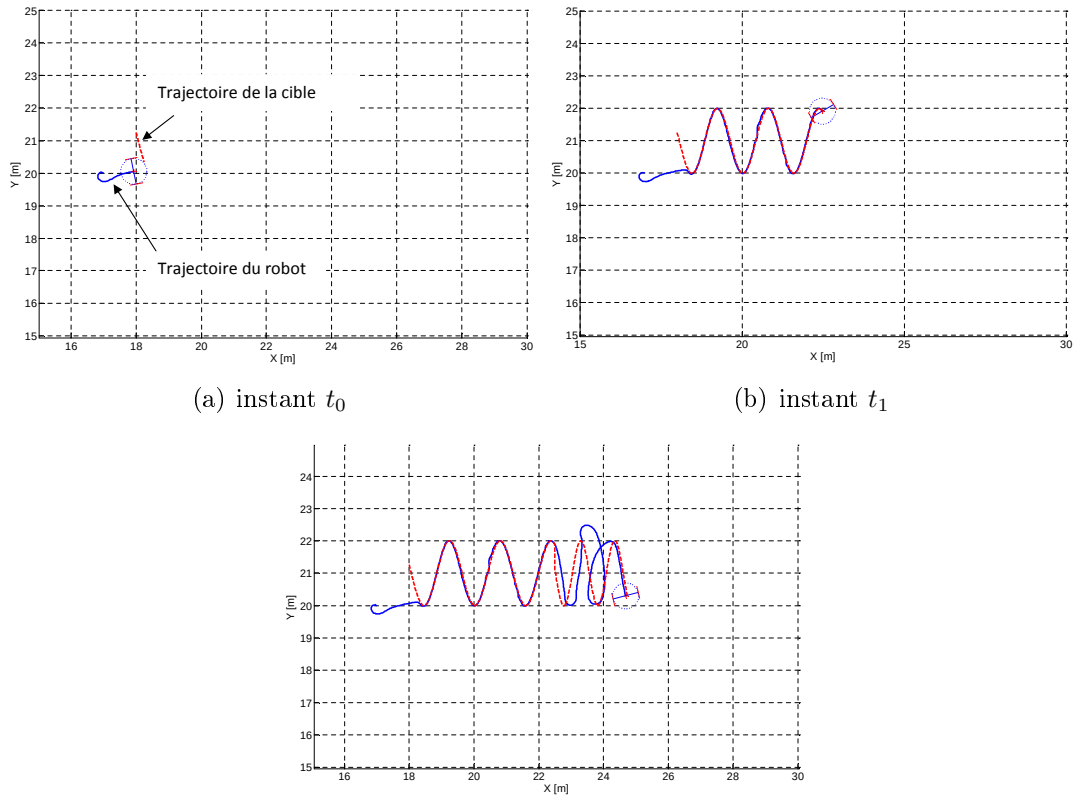


FIGURE 3.11 – Evolution de la variation de l'angle d'orientation $\dot{\theta}_C$, de la distance d_{S_i}, et de la fonction de Lyapunov V.



(a) instant t_0

(b) instant t_1

(c) instant t_2

FIGURE 3.12 – Trajectoire du robot et de la cible dynamique.

3.3.2 Contrôleur d'évitement d'obstacles

Nous rappellons que tout obstacle est évité en utilisant les équations différentielles du cycle limite (cf. Section 2.3.2, page 72) :

$$\begin{aligned} \dot{x}_s &= a y_s + x_s(R_c^2 - x_s^2 - y_s^2) \\ \dot{y}_s &= -a x_s + y_s(R_c^2 - x_s^2 - y_s^2) \end{aligned} \tag{3.43}$$

où $a = 1$ pour un évitement dans le sens anti-trigonométrique, et $a = -1$ dans le sens trigonométrique (cf. Section 2.3.2.1). Le couple (x_s, y_s) correspond à la position relative du robot par rapport au centre d'un obstacle de rayon R_c.

L'angle consigne que le robot doit suivre pour éviter l'obstacle est donné par

$$\theta_{S_{oa}} = \arctan(\frac{\dot{y}_s}{\dot{x}_s}) \tag{3.44}$$

De la même manière que pour le contrôleur d'attraction vers une cible dynamique, et en utilisant la même loi de commande (cf. Equation 3.1b), vérifions que cette consigne est toujours atteignable $\theta_{S_{oa}}$ par le robot en tenant compte de sa contrainte ω_{max}. En effet, d'après 3.12 il faut avoir

$$|\omega_{S_{oa}}| \leq \omega_{max} - \lambda\pi \tag{3.45}$$

et donc

$$\left|\dot{\theta}_{S_{oa}}\right| \leq \omega_{max} - \lambda\pi \tag{3.46}$$

Dans les équations du cycle limite, une constante μ peut être introduite. Cette constante affecte la trajectoire décrite par les équations différentielles 3.47 la rendant plus ou moins lisse (cf. Figure 3.13). Le système d'équations devient alors

$$\begin{array}{rcl} \dot{x}_s & = & a y_s + \mu x_s (R_c^2 - x_s^2 - y_s^2) \\ \dot{y}_s & = & -a x_s + \mu y_s (R_c^2 - x_s^2 - y_s^2) \end{array} \tag{3.47}$$

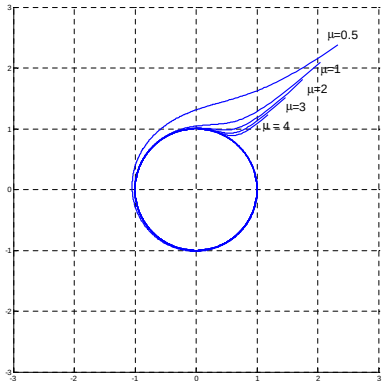


FIGURE 3.13 – Influence du paramètre μ sur le cycle limite.

Dans [Kim 03a], cette constante a été utilisé, mais son choix n'a pas été justifié. Nous proposons dans ce qui suit d'utiliser la constante μ et trouver l'ensemble de ses valeurs pour rendre la consigne $\dot{\theta}_{S_{oa}}$ toujours atteignable.

Calculons maintenant $\dot{\theta}_{S_{oa}}$ à partir de l'équation 3.44

$$\dot{\theta}_{S_{oa}} = \frac{\frac{d}{dt}\left(\frac{\dot{y}_s}{\dot{x}_s}\right)}{(1 + (\frac{\dot{y}_s}{\dot{x}_s})^2)} \tag{3.48a}$$

On remarque que cette expression est définie dans tout l'ensemble des réels $\mathbb{R}$. Dans les calculs qui suivent, nous notons

$$A = R_c^2 - x_s^2 - y_s^2 \tag{3.49}$$

Utilisant le système d'équations 3.47, nous avons alors

$$\dot{\theta}_{S_{oa}} = -a - 2a\mu^2 A(x_s^2 + y_s^2)/(a^2 + \mu^2 A^2)$$

Comme $a = \pm 1$ alors $a^2 = 1$ Ce qui revient à

$$\dot{\theta}_{S_{oa}} = -a - 2a\mu^2 A(x_s^2 + y_s^2)/(1 + \mu^2 A^2) \tag{3.51}$$

Sachant que nous voulons contraindre la variation de la consigne (cf. Equation 3.46), nous avons alors

$$|-a| \left| 1 + 2\mu^2 A \frac{(x_s^2 + y_s^2)}{(1 + \mu^2 A^2)} \right| \leq \omega_{max} - \lambda\pi$$

comme $|-a| = |\pm 1| = 1$, donc

$$\left| 1 + 2\mu^2 A \frac{(x_s^2 + y_s^2)}{(1 + \mu^2 A^2)} \right| \leq \omega_{max} - \lambda\pi \tag{3.53}$$

Or, on sait que

$$\left| 1 + 2\mu^2 A \frac{(x_s^2 + y_s^2)}{(1 + \mu^2 A^2)} \right| \leq 1 + \left| 2\mu^2 A \frac{(x_s^2 + y_s^2)}{(1 + \mu^2 A^2)} \right|$$

On peut alors, et pour des raisons de simplification, trouver l'ensemble des valeurs de μ vérifiant

$$1 + \left| 2\mu^2 A \frac{(x_s^2 + y_s^2)}{(1 + \mu^2 A^2)} \right| \leq \omega_{max} - \lambda\pi \tag{3.55}$$

En effet, les valeurs vérifiant 3.55, vérifient aussi 3.53.

Ce qui revient à

$$0 \leq \left| 2\mu^2 A \frac{(x_s^2 + y_s^2)}{(1 + \mu^2 A^2)} \right| \leq \omega_{max} - \lambda\pi - 1 \tag{3.56}$$

Dans ce qui suit, nous proposons de noter

$$P_{oa} = \omega_{max} - \lambda\pi - 1$$

Le choix de λ doit donc prendre en considération que

$$P_{oa} \geq 0$$

$$\omega_{max} - \lambda\pi \geq 1 \tag{3.57}$$

En effet, le membre droit de l'inéquation 3.56 doit toujours être positif. En rajoutant cette contrainte, les nouvelles valeurs possibles de λ (cf. relation 3.18, page 122)

$$\frac{2\omega_{max} - \dot{\omega}_{max} nT}{2\pi} \leq \lambda < \frac{\omega_{max} - 1}{\pi} \tag{3.58}$$

Comme

$$1 + \mu^2 A^2 \geq 1$$

Nous pouvons écrire (basée sur la relation 3.56)

$$2\mu^2 |A| (x_s^2 + y_s^2) \leq P_{oa} \tag{3.59}$$

Pour que la variation de la consigne $\dot{\theta}_{S_{oa}}$ reste admissible, il faut que la valeur μ soit telle que

$$\mu^2 \leq \frac{P_{oa}}{2 |A| (x_s^2 + y_s^2)} \tag{3.60}$$

En introduisant d_{RO} comme la distance du robot par rapport à l'obstacle, nous avons

$$x_s^2 + y_s^2 = d_{RO}^2$$

A (cf. Equation 3.49) peut aussi s'écrire

$$A = R_c^2 - d_{RO}^2$$

et la relation 3.60 s'écrit alors

$$\mu^2 \leq \frac{P_{oa}}{2\left|R_c^2 - d_{RO}^2\right| d_{RO}^2} \tag{3.61}$$

On remarque que l'ensemble des valeurs de μ dépend de la distance du robot à l'obstacle. Ainsi, pour trouver l'ensemble des valeurs μ permettant à la consigne $\omega_{S_{oa}}$ d'être toujours atteignable, il faut trouver le minimum du membre droit de l'inéquation qui soit indépendant de la distance du robot à l'obstacle. Ce minimum correspond au maximum de son dénominateur. Étudions ce dénominateur $Den = \left|R_c^2 - d_{RO}^2\right| d_{RO}^2$ (Nous omettons volontairement le coefficient 2). Nous distinguons deux cas

1. $R_c \geq d_{RO}$ (le robot est à l'intérieur du rayon du cycle-limite) : cela donne

   $$Den = (R_c^2 - d_{RO}^2)d_{RO}^2$$

   Le calcul de la dérivée de Den par rapport à la distance d_{RO} dans le domaine de définition ($R_c \geq d_{RO}$) permet de trouver la valeur de d_{RO} pour laquelle il est maximum. En effet,

   $$\begin{aligned} \frac{\partial Den}{\partial d_{RO}} &= -2d_{RO}d_{RO}^2 + 2d_{RO}(R_c^2 - d_{RO}^2) \\ &= 2d_{RO}(R_c^2 - 2d_{RO}^2) \end{aligned}$$

   Les deux racines correspondant à un maximum du dénominateur et compris dans le domaine de définition sont $\pm\frac{R_c}{\sqrt{2}}$ (la racine $d_{RO} = 0$ n'est pas retenue car elle correspond à une distance nulle entre le centre de l'obstacle et celui du robot, ce qui est absurde). En remplaçant dans la relation 3.60, la valeur de μ doit alors satisfaire

   $$\mu \leq \sqrt{\frac{P_{oa}}{2(R_c^2/2)^2}} \tag{3.63a}$$

   $$\mu \leq \frac{1}{R_c^2}\sqrt{2P_{oa}} \tag{3.63b}$$

2. $R_c \leq d_{RO}$ (le contrôleur d'évitement d'obstacles est activé à l'extérieur du cycle-limite) : le dénominateur devient alors

   $$Den = -(R_c^2 - d_{RO}^2)d_{RO}^2$$

   sa dérivée par rapport à d_{RO} est

$$\frac{\partial Den}{\partial d_{RO}} = -2d_{RO}(R_c^2 - 2d_{RO}^2)$$

Aucune des racines de $\frac{\partial Den}{\partial d_{RO}}$ n'est admissible car aucune ne satisfait $R_c \leq d_{RO}$. De plus sur ce domaine de définition, nous pouvons facilement remarquer que $\frac{\partial Den}{\partial d_{RO}} > 0$, ce qui signifie que Den est toujours croissante pour d_{RO} croissant. En théorie, le maximum de Den correspond à $d_{RO} = \infty$. En pratique, la distance maximale pouvant séparer le robot de l'obstacle correspond à l'instant où le contrôleur d'évitement d'obstacles est activé (toujours dans le cas $R_c \leq d_{RO}$). En effet, en suivant la consigne donnée par le système d'équation 3.47, le robot va s'approcher de l'obstacle (cf. Figure 2.15, page 76).

En remplaçant dans la relation 3.60, la valeur de μ doit alors satisfaire

$$\mu \leq \sqrt{\frac{P_{oa}}{2(R_c^2 - d_{RO_0}^2)d_{RO_0}^2}} \tag{3.64}$$

Où d_{RO_0} est la distance séparant le robot de l'obstacle à l'instant d'activation du contrôleur d'évitement d'obstacles (cf. Section 2.3.2, page 72).

Dans le paragraphe suivant, nous noterons $P_{S_{oa}} = \omega_{max} - \lambda\pi = P_{oa} + 1$.

3.3.2.1 Simulation et discussion

Nous proposons de montrer l'utilité du coefficient μ permettant à la variation de la consigne d'angle donné par le contrôleur d'évitement d'obstacles $\dot{\theta}_{S_{oa}}$ de satisfaire la relation 3.46. En d'autres termes, il faut que cette variation soit admissible par les contraintes de vitesse angulaire maximale du robot.

Pour cela, nous simulons la navigation d'un robot mobile vers une cible statique ($v_C = 0$) en évitant des obstacles. Nous posons : $\omega_{max} = 3rd/s$, $\lambda = 0.6s^{-1}$ (les mêmes paramètres que pour le contrôleur d'attraction vers une cible ont été repris). La contrainte décrite par la relation 3.57 dans le choix de λ a été prise en considération. La simulation est d'abord réalisée sans le coefficient μ (ce qui revient à $\mu = 1$). Le robot évite bien le premier obstacle. Cependant, lorsqu'il rencontre le deuxième, il ne peut l'éviter (cf. Figure 3.14). En effet, la variation de la consigne $\dot{\theta}_{S_{oa}}$ délivrée par le contrôleur d'évitement d'obstacles se trouve en dehors de la contrainte imposée (cf. Equation 3.46) (cf. Figure 3.15).

Maintenant, nous reprenons les simulations avec un coefficient μ calculé selon les cas cités ci-dessus. Le contrôleur d'évitement d'obstacles est activé pendant que le robot est en dehors du cycle limite. μ est alors calculé par

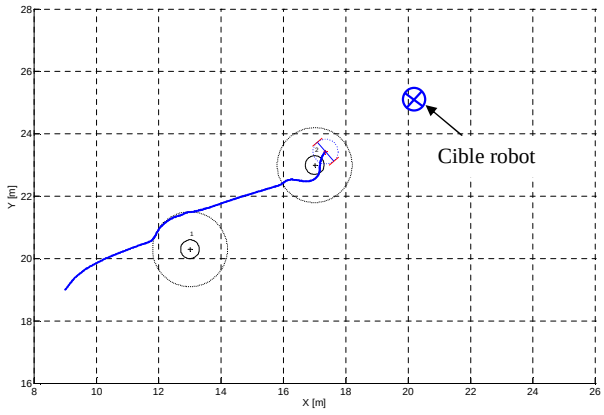


FIGURE 3.14 – Trajectoire du robot avec un coefficient μ constant ($\mu = 1$).

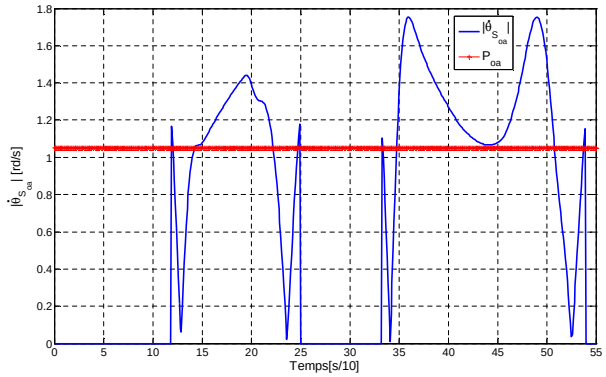


FIGURE 3.15 – Evolution de la variation de la consigne d'angle $\dot{\theta}_{S_{oa}}$.

$$\mu = \sqrt{\frac{(\omega_{max} - \lambda\pi) - 1}{2(R_c^2 - d_{RO_0}^2)d_{RO_0}^2}} \tag{3.65}$$

On remarque que le robot arrive à éviter les deux obstacles (cf. Figure 3.16). De plus, $\dot{\theta}_{S_{oa}}$ se trouve bien dans la zone imposée par la relation 3.46 (cf. Figure 3.17).

On déduit alors que l'introduction du coefficient μ permet bel et bien de garder une variation admissible de la consigne d'angle délivrée par le contrôleur d'évitement d'obstacles.

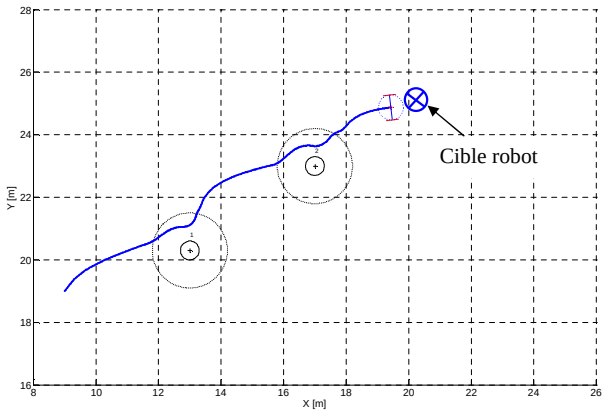


FIGURE 3.16 – Trajectoire du robot avec un coefficient μ variable.

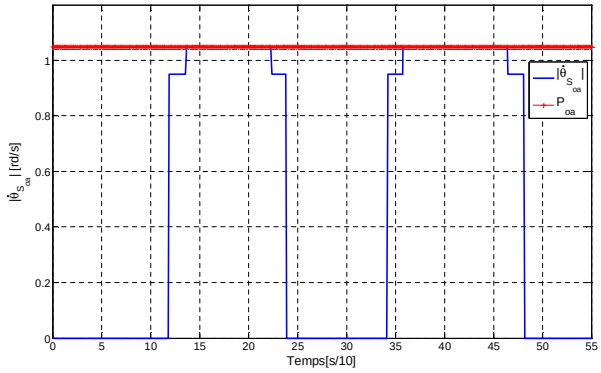


FIGURE 3.17 – Evolution de la variation de la consigne d'angle $\dot{\theta}_{S_{oa}}$.

3.3.3 Stabilité globale de l'architecture de contrôle

Nous avons vu que la loi de commande proposée est asymptotiquement stable tant qu'un seul contrôleur est actif (les consignes proviennent du même contrôleur). Ces consignes vérifient les conditions 3.20 et 3.46 imposées par les contraintes structurelles des robots afin qu'elles restent atteignables. Il s'agit de la zone linéaire de la variation de l'erreur d'orientation (cf. Figure 3.5, page 119). Cela signifie que la fonction de Lyapunov sera toujours décroissante au moment de l'exécution de chaque contrôleur.

Cependant, le problème se pose (comme pour tous les systèmes hybrides) au moment des commutations où il y aura une discontinuité de l'erreur $\tilde{\theta}$, avec le risque de sauts brusques au niveau des valeurs de la fonction de Lyapunov V. Si ces sauts font que V augmente au moment de la transition, on ne peut conclure

sur la stabilité globale du système.

Pour pallier cet inconvénient, nous proposons de s'inspirer du théorème Fonctions de Lyapunov Multiples (FLM) et de la propriété des transitions lentes présentés ci-dessus pour prouver la stabilité globale de l'architecture de contrôle. Ainsi, pour assurer une stabilité asymptotique à cette architecture au sens du théorème FLM, il faut contraindre les commutations de telle sorte que pour tout contrôleur i, la fonction de Lyapunov correspondante V_i satisfait

$$V_i(t_s + \tau) < V_i(t_{avs}) \tag{3.66}$$

$V_i(t_{avs})$ étant sa valeur à l'instant précédant celui de la dernière commutation de i vers un autre contrôleur (cf. Figure 3.18). t_s correspond à l'instant de la commutation. On pourra alors dire que l'instant t_{avs} pour le dernier contrôleur actif correspond à t_s pour le nouveau contrôleur. Nous noterons τ le temps mis par la fonction de Lyapunov pour satisfaire à la relation 3.66. En effet, d'après le théorème FLM et la propriété des transitions lentes, une fois un contrôleur est actif, il doit le rester au moins pendant un temps τ pour obtenir la stabilité asymptotique.

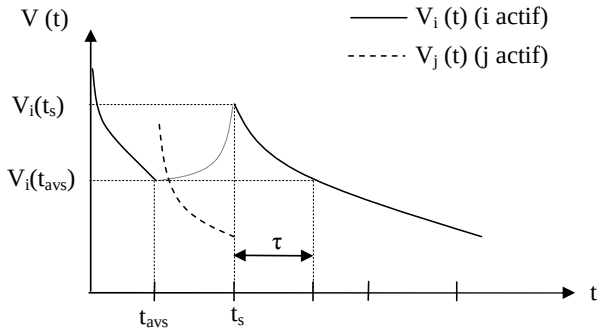


FIGURE 3.18 – Valeurs $V_i(t_{avs})$ et $V_i(t_s)$ correspondant au système i.

Nous avons vu qu'il est difficile de déterminer la borne inférieure de ce temps et que peu de résultats ont été obtenus dans ce domaine (cf. Section 3.2.2). Nous avons vu aussi qu'à cause de la nature inconnue de l'environnement, on ne peut pas laisser un contrôleur actif plus longtemps qu'il en faut afin d'assurer la sécurité des robots. Nous proposons alors de trouver le temps minimal τ (il s'agira d'une borne inférieure) nécessaire pour satisfaire à l'inéquation 3.66 et qui assure ainsi une stabilité asymptotique à l'architecture de contrôle. Ce temps dépendra des valeurs des fonctions de Lyapunov et sera donc calculé à chaque commutation en fonction de ces valeurs. Il s'agira d'un TT « *Temps de Tenue* » relative à

chaque phase continue permettant à la fonction de Lyapunov (et donc l'erreur d'orientation) de décroître assez pour dissiper l'effet de la dernière commutation avant l'arrivée d'une autre. En effet, nous avons vu que plusieurs commutations se succédant rapidement peuvent engendrer des effets indésirables voire l'instabilité du système (cf. Section 3.2.2, page 114).

L'évolution de la fonction de Lyapunov choisie (cf. Equation 3.4, page 119) peut être déduite à partir de celle de l'erreur d'orientation $\tilde{\theta}$. En effet, la relation 3.5 permet de remonter à l'évolution de $\tilde{\theta}$ grâce à la résolution de l'équation différentielle. Elle s'exprime alors

$$\tilde{\theta}(t) = \tilde{\theta}(t_s)e^{-k(t-t_s)} \tag{3.67}$$

$\tilde{\theta}(t_s)$ est la valeur de l'erreur d'orientation à l'instant de commutation t_s (instant initial de la phase où le nouveau contrôleur est actif).

L'évolution de la fonction de Lyapunov est alors

$$V(t) = (\tilde{\theta}^2(t_s)/2)e^{-2k(t-t_s)} \tag{3.68a}$$
$$V(t) = V(t_s)e^{-2k(t-t_s)} \tag{3.68b}$$

Pour trouver le temps minimal τ satisfaisant la relation 3.66, il suffit de calculer $V(t_s + \tau)$.

D'après l'équation 3.68b

$$V(t_s + \tau) = V(t_s)e^{-2k\tau} \tag{3.69}$$

Pour trouver le temps minimal τ pendant lequel le contrôleur doit rester actif avant la prochaine commutation, ce qui assure une stabilité asymptotique à l'architecture de contrôle, nous remplaçons $V(t_s + \tau)$ par sa valeur (cf. Equation 3.69) dans la relation 3.66, ce qui donne

$$V(t_s)e^{-2k\tau} < V(t_{avs}) \tag{3.70}$$

Le temps τ est alors tel que

$$\tau > \frac{ln(V(t_{avs})/V(t_s))}{-2k} \tag{3.71}$$

On remarque clairement que le temps τ dépend du gain k de la loi de commande. En effet, ce dernier affecte la vitesse de convergence de l'erreur d'orientation, et donc celle de la fonction de Lyapunov (cf. Equation 3.68).

Cela veut dire que si on augmente le gain k, on pourra alors diminuer le temps τ pendant lequel le système ne peut commuter dans un autre contrôleur pour assurer la stabilité asymptotique [Benzerrouk 10b].

Cependant, le gain k doit toujours respecter les conditions imposées par la contrainte structurelle des entités robotiques relative à leur vitesse angulaire maximale, à savoir celle décrite par l'équation 3.7 (page 120). L'expression de la vitesse angulaire (cf. Equation 3.1b) comprend aussi la variation de la consigne qui est à son tour bornée pour que la structure virtuelle reste atteignable par les robots (cf. Equations 3.20, 3.46).

En tenant compte de ces remarques, nous proposons de trouver la valeur maximale du gain k. Pour trouver cette valeur, nous remplaçons l'expression de la vitesse angulaire (cf. Equation 3.1b) dans la relation (cf. Equation 3.7), ce qui donne

$$\left|\omega_{S_i} + k\tilde{\theta}_i\right| \leq \omega_{max} \tag{3.72}$$

Or

$$\left|\omega_{S_i} + k\tilde{\theta}_i\right| \leq |\omega_{S_i}| + \left|k\tilde{\theta}_i\right| \tag{3.73}$$

La relation 3.73 permet alors d'écrire

$$|\omega_{S_i}| + k\left|\tilde{\theta}_i\right| \leq \omega_{max} \tag{3.74}$$

ce qui mène à

$$k \leq \frac{\omega_{max} - |\omega_{S_i}|}{\left|\tilde{\theta}_i\right|} \tag{3.75}$$

La valeur maximale de k autorisée doit alors être telle que

$$k_{max} \leq min(\frac{\omega_{max} - |\omega_{S_i}|}{\left|\tilde{\theta}_i\right|}) \tag{3.76}$$

Pour obtenir le minimum du membre droit de l'inéquation, il faut minimiser le numérateur et maximiser le dénominateur. L'utilisation des contraintes sur les variations de la consignes précédemment imposées (cf. Equation 3.12, page 121) nous permet de trouver le minimum du numérateur. De même, sachant que l'erreur d'orientation est décroissante sur une phase continue (un contrôleur actif en dehors des commutations) (cf. Equation 3.67) sa valeur maximale correspond alors à l'instant de commutation t_s.

Nous déduisons donc

$$k_{max} = \frac{\omega_{max} - |\omega_{max} - \lambda\pi|}{\left|\tilde{\theta}(t_s)\right|} \tag{3.77}$$

or

$$|\omega_{max} - \lambda\pi| = \omega_{max} - \lambda\pi$$

car

$$\omega_{max} - \lambda\pi \geq 0$$

La valeur de k_{max} s'écrit alors (en enlevant la valeur absolue) :

$$k_{max} = \frac{\omega_{max} - \omega_{max} + \lambda\pi}{\left|\tilde{\theta}(t_s)\right|} \tag{3.78a}$$

$$= \frac{\lambda\pi}{\left|\tilde{\theta}(t_s)\right|} \tag{3.78b}$$

On remarque que le gain k_{max} dépend de l'instant de commutation t_s et de l'erreur de transition à cet instant.

Ainsi, il est possible de diminuer le temps τ que le contrôle doit attendre avant la prochaine commutation (cf. Equation 3.71) grâce à l'augmentation du gain k. La valeur minimale possible est alors

$$\tau_{min} > \frac{ln(V(t_{avs})/V(t_s))}{-2k_{max}} \tag{3.79}$$

sous réserve que $V(t_{avs}) \neq 0$.

Une fois le temps τ écoulé, s'il n'y a pas de commutation, nous voudrons que le gain k retrouve sa valeur initiale k_{ini}. Nous proposons alors que le gain k devienne dynamique sur cette phase. Il serait intéressant qu'il décroisse progressivement, afin qu'il n y ait pas un changement brusque dans sa valeur.

Nous proposons alors qu'il soit dépendant de l'erreur d'orientation $\tilde{\theta}_i$ dont l'évolution est décroissante (cf. Equation 3.67).

La relation suivante est retenue comme règle d'évolution. Elle est simplement inspirée de l'évolution de la vitesse linéaire (cf. Equation 3.1a) [Benzerrouk 10c].

$$k = k_{ini} - (k_{ini} - k_{max}) \tanh^2(\tilde{\theta}) \tag{3.80}$$

Nous venons de voir que les commutations entre les contrôleurs en respectant un certain temps TT permet d'obtenir la stabilité asymptotique. Cependant, imposer qu'un contrôleur reste actif pendant ce temps peut être très restrictif ou dangereux pour le robot. Par exemple, si le contrôleur d'évitement d'obstacles n'est pas activé malgré la détection d'un obstacle, car il faut attendre un temps τ (si minimal soit-il), une collision peut avoir lieu.

Dans cette situation, on ne peut favoriser la stabilité asymptotique au détriment de la sécurité des robots. Cependant, grâce au théorème de la stabilité faible (cf. Section 3.2.3, page 115), il est possible de relâcher cette contrainte d'attendre un temps τ. En effet, la fonction de Lyapunov choisie est fonction de l'erreur d'orientation (cf. Equation 3.4, page 119). Nous avons vu qu'elle est décroissante en dehors des instants de commutations. Cependant, elle peut croître à ces instants à cause de la discontinuité de consigne due à la transition entre les contrôleurs. En remarquant que la valeur maximale de la fonction de Lyapunov est bornée et correspond à $max(\tilde{\theta}_i) = \pi$, elle vaut

$$max(V) = \frac{\pi^2}{2}$$

De plus, en supposant que l'environnement permet toujours au robot de converger vers sa cible (le nombre d'obstacles que le robot rencontre n'est pas infini), l'erreur d'orientation par rapport à la consigne provenant du contrôleur d'attraction vers la cible va décroître de façon exponentielle et tendre vers 0 en un temps $(t_f - t_i)$ fini (cf. Equation 3.1b, page 118). Il en est de même pour la fonction de Lyapunov. La première condition de la proposition 3.2.3 est alors satisfaite. En effet, si $V(t_f) \approx 0$, alors $\forall V(t_i), V(t_f) \leq V(t_i)$. La deuxième condition est aussi satisfaite, car la fonction de Lyapunov est déjà prouvée décroissante. On peut alors dire que l'architecture de contrôle est toujours faiblement stable. De plus, $\dot{V} < 0$ en dehors des instants de commutations. L'architecture de contrôle est donc asymptotiquement faiblement stable.

Enfin, en pratique, le saut de consigne (d'orientation dans ce cas) n'est pas provoqué uniquement par la commutation entre contrôleurs. En effet, au sein d'un seul contrôleur, on peut observer une discontinuité de consigne. Nous avons vu que le contrôleur d'attraction vers une cible dynamique offre au robot la possibilité d'atteindre et de suivre une cible dynamique pourvu qu'elle reste atteignable par le robot. La contrainte imposée sur la variation de la consigne $\omega_{S_{at}}$ et donc sur celle de la cible (cf. Equation 3.37, page 126) a pour objectif de permettre aux robots d'atteindre et de maintenir la formation. Cependant, si un changement brutal de la dynamique de la cible a lieu (une cible s'éloignant peut devenir une cible s'approchant et vice versa), cela signifie que chaque entité robotique doit atteindre de nouveau sa cible. Ce changement brutal peut se répercuter sur

$\theta_{S_{at}}$ et entraîner une discontinuité de consigne ce qui engendre une augmentation brusque de l'erreur d'orientation (et donc la fonction de Lyapunov) comme dans le cas d'une commutation d'un contrôleur à un autre. Il en est de même pour le contrôleur d'évitement d'obstacles. En effet, quand le robot évite un obstacle, il peut rencontrer un autre. Cela signifie qu'il doit calculer un nouvel angle consigne basé sur un nouveau cycle-limite. Nous proposons alors que le gain k soit réévalué à ces instants aussi qui correspondent à des évènements discrets.

Nous pouvons alors résumer les cas de réévaluation du gain k comme suit :

1. commutation d'un contrôleur à un autre,
2. changement brutal de la dynamique de la cible au sein du contrôleur d'attraction vers une cible dynamique (un changement est considéré brutal s'il est en dehors de la plage décrite par la relation 3.37 (page 126)),
3. changement de l'obstacle évité au sein du contrôleur d'évitement d'obstacles.

3.4 Simulation et discussion

Nous proposons ici de simuler un robot mobile utilisant l'architecture de contrôle proposée pour suivre une cible dynamique en évitant des obstacles. Grâce à l'aspect distribué de l'architecture de contrôle proposée, le résultat est généralisé à toutes les entités du SMR.

Afin de montrer l'utilité d'un gain dynamique k, nous proposons de comparer les résultats obtenus avec ceux que donne l'utilisation d'un gain constant k_{ini}. Le robot mobile doit atteindre une cible dynamique en évitant deux obstacles (cf. Figure 3.19). On remarque que le robot parvient à atteindre la cible en évitant les obstacles. Cependant, remarquons que le robot s'enfonce dans la marge de sécurité des obstacles et s'approche dangereusement d'eux. En effet, un gain k constant ne permet pas au robot d'augmenter la rapidité de réaction aux obstacles en fonction de la taille de ceux-ci. En d'autre termes, le robot ne réagit pas en fonction de l'importance de l'erreur d'orientation.

Dans la figure 3.20, la grandeur indicateur (2^e courbe) indique quel contrôleur est actif. Ainsi, la valeur 0 correspond au contrôleur d'attraction vers une cible dynamique, tandis que la valeur 1 correspond à celui d'évitement d'obstacles. Dans la même figure, l'évolution de la fonction de Lyapunov V montre qu'à l'instant t_{s1}, il y a commutation vers le contrôleur d'évitement d'obstacles ce qui provoque une augmentation brutale de celle-ci. τ_1 est le temps mis pour que elle décroisse et s'annule. Cependant, elle croît immédiatement après (instant t_{s2}) suite à l'évitement du deuxième obstacles avant de décroître à nouveau

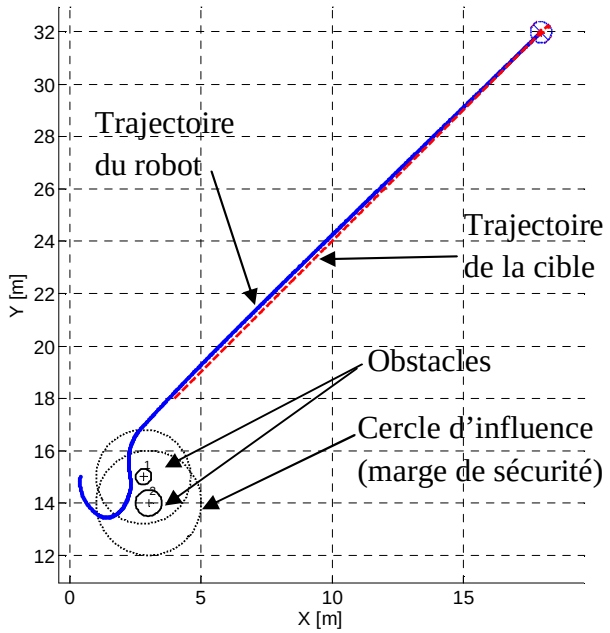


FIGURE 3.19 – Robot mobile utilisant l'architecture de contrôle avec un gain k constant.

en un temps τ_2. La nouvelle commutation vers le contrôleur d'attraction vers la cible dynamique provoque un saut assez petit comparé à la dernière valeur de V quand ce contrôleur était actif (instant immédiatement précédant t_{s1}). D'après le théorème FLM (cf. Section 3.2.1), le système est alors asymptotiquement stable. On remarque néanmoins que cette propriété est obtenue à une période d'échantillonnage près. En effet, le deuxième pic de la fonction de Lyapunov survient juste après τ_1.

Enfin, la vitesse angulaire ω reste proche de 0 et est loin d'être saturée. Nous reprenons les mêmes conditions de navigation (condition initiale du robot, position des obstacles). Cette fois, un gain k variable est utilisé. La trajectoire du robot est donnée dans la figure 3.21.

Dans la figure 3.22, on observe l'évolution de la fonction de Lyapunov des contrôleurs. Le premier saut indique que le contrôleur d'évitement d'obstacles est activé (l'indicateur passant à la valeur 1 le confirme). Suite à ce saut, on remarque que le gain k est réévalué. Il augmente permettant alors d'augmenter la vitesse de convergence de l'erreur d'orientation. On remarque alors que le temps de convergence τ_1' est nettement inférieur à τ_1 obtenu avec un gain constant (cf. Figure 3.20). Au niveau de la trajectoire du robot (cf. Figure 3.21), cela se traduit par un robot plus réactif aux obstacles. Ainsi, dès qu'un obstacle est détecté, il

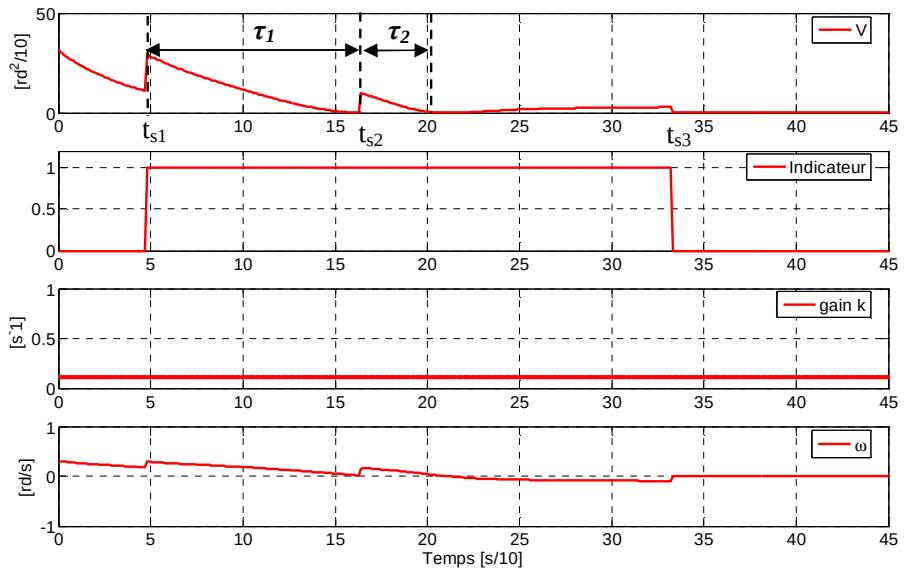


FIGURE 3.20 – Variation de la fonction de Lyapunov et de la vitesse angulaire (gain k constant).

est rapidement évité sans trop s'enfoncer dans la zone de sécurité. On remarque que le deuxième saut correspondant à l'obstacle suivant est nettement plus petit que celui obtenu avec un gain constant. Il est vite atténué grâce à un nouveau gain k. De plus, on remarque qu'il y a un temps non négligeable entre les deux sauts $(t_{s2} - (t_{s1} + \tau' 1))$. La stabilité asymptotique est obtenue d'autant plus que la deuxième commutation dans le contrôleur d'attraction vers la cible passe sans saut important. Remarquons que le gain k retrouve sa valeur k_{ini} chaque fois que les effets de commutations sont dissipés. Il la retrouve aussi rapidement que la vitesse de convergence (on observe un retour quasi instantané après le deuxième saut de la fonction de Lyapunov pendant l'évitement).

Enfin, notons que la vitesse angulaire du robot ω varie de façon plus importante que dans le cas d'un gain constant (avec une accélération angulaire choisie de $1rd/s^2$).

Nous avons vu aussi que la stabilité asymptotique n'est pas toujours garantie. En effet, si l'environnement est trop encombré, ou s'il y a plusieurs sauts dans la dynamique de la cible, la fonction de Lyapunov peut connaître plusieurs sauts consécutifs.

Nous proposons alors de reprendre la simulation dans un environnement en présence d'obstacles avec un gain variable et un saut de la dynamique de la cible. La trajectoire du robot ainsi que celle de la cible sont représentées à trois

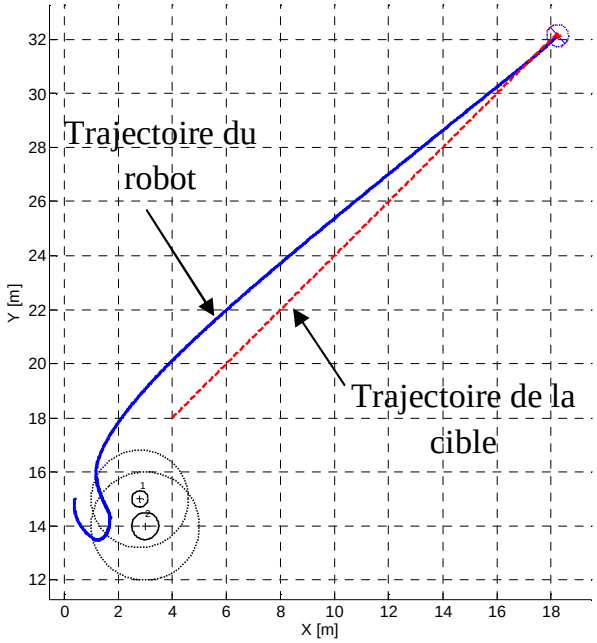


FIGURE 3.21 – Robot mobile utilisant l'architecture de contrôle avec un gain k dynamique.

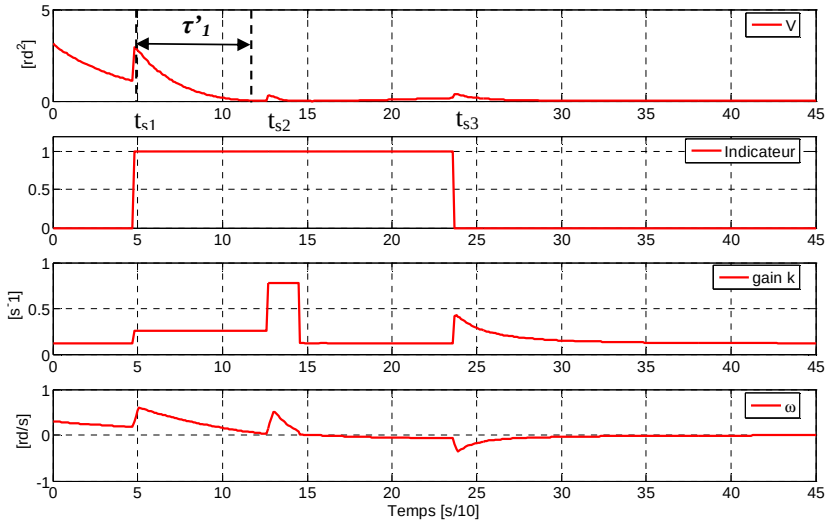
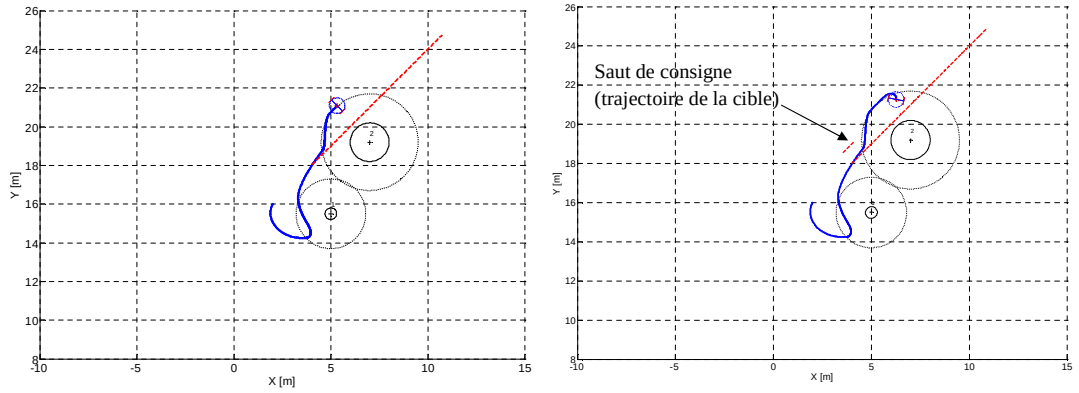


FIGURE 3.22 – Variation de la fonction de Lyapunov et de la vitesse angulaire (gain k dynamique).

instants différents (cf. Figure 3.23). Le robot évite bien les obstacles, et à l'instant t_s (cf. Figure 3.24), un saut de la cible survient. Le robot corrige de nouveau son orientation pour suivre la nouvelle cible dynamique. On remarque deux sauts consécutifs dans la fonction de Lyapunov : le premier consiste au saut observé dans l'orientation de la cible (l'indicateur est toujours à 0 montrant le maintien du contrôleur d'attraction vers la cible), et le deuxième correspond à l'activation du contrôleur d'évitement d'obstacles (indicateur passant à 1). Bien qu'avant ces deux sauts, la fonction de Lyapunov décroît avant chaque prochain saut, on ne peut conclure sur la stabilité asymptotique. En effet, à cause des deux derniers sauts consécutifs, seule une faible stabilité asymptotique est prouvée (cf. Section 3.2.3).



(a) instant t_0 (b) instant t_1

(c) instant t_2

FIGURE 3.23 – Robot mobile utilisant l'architecture de contrôle avec un saut de consigne (gain k dynamique).

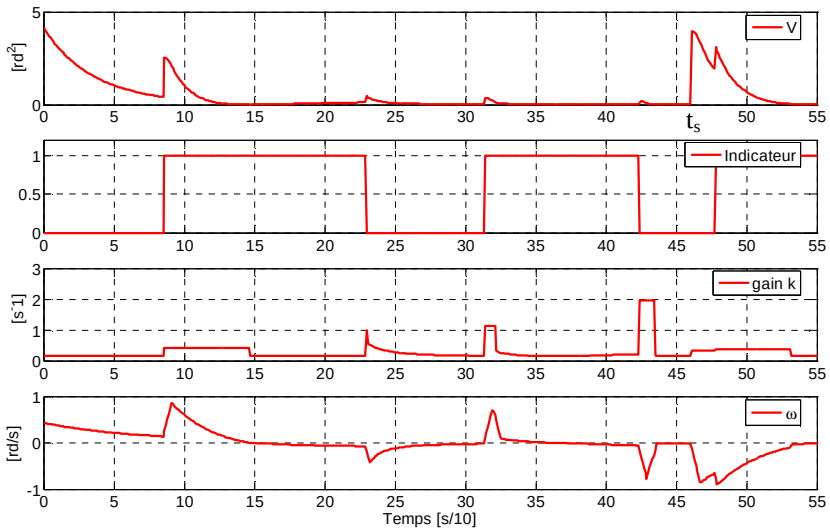


FIGURE 3.24 – Variation de la fonction de Lyapunov et de la vitesse angulaire ω : présence d'obstacles et changement brutal de la dynamique de la cible.

3.5 Conclusion

La plupart des architectures de contrôle comportementales souffrent du manque d'études rigoureuses de leur stabilité. En effet, l'existence de plusieurs tâches élémentaires et la commutation entre ces tâches (méthode de la sélection d'action), ou leur exécution simultanée (fusion d'action), rend difficile cette étude. Dans ce chapitre, nous avons pu mettre en exergue et traiter cette difficulté en explorant les propriétés des systèmes hybrides. Ainsi, il est prouvé que la loi de commande proposée assure une stabilité globale à l'architecture de contrôle tout en respectant les contraintes structurelles du robot (vitesse linéaire maximale, vitesse angulaire maximale). La stabilité asymptotique peut être prouvée. Cependant, elle peut être altérée par la nature de l'environnement notamment si celui-ci est très encombré. Nous avons toutefois prouvé que l'architecture de contrôle proposée est toujours asymptotiquement faiblement stable.

Chapitre 4

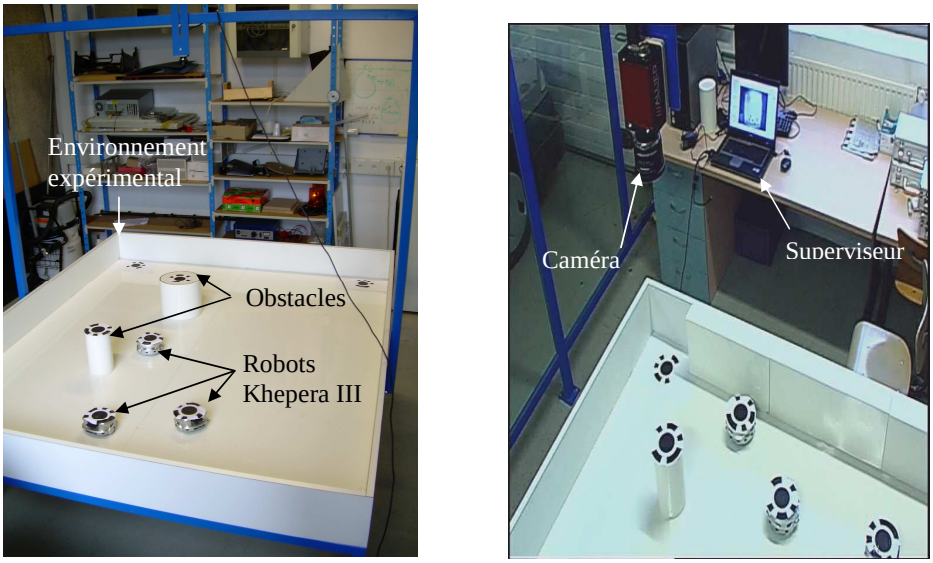
Phase expérimentale

Le principal objectif de ce chapitre est d'implémenter l'architecture de contrôle proposée, testée jusqu'à présent en simulation, sur des robots mobiles effectifs. Ceci permettra alors de prendre en compte et de traiter certaines contraintes matérielles et logicielles rencontrées en pratique. L'implantation est faite sur des robots Khepera III de la société K-team (cf. Figure 4.7).

Nous commençons d'abord par une description de la plate-forme disponible au LASMEA. Ensuite, nous verrons les implémentations effectives de l'architecture de contrôle proposée pour la navigation en formation d'un groupe de robots mobiles ainsi que les difficultés rencontrées et leur résolution.

4.1 Plate-forme expérimentale

Afin de mener à bien nos phases expérimentales, une plate-forme dédiée aux robots utilisés a été mise en place au LASMEA. Nous décrivons dans ce qui suit la plate-forme ainsi que les éléments nécessaires à l'expérimentation.



(a) (b)

FIGURE 4.1 – Plate-forme expérimentale.

4.1.1 L'environnement

La plate-forme est constituée principalement d'une enceinte de dimension $220cm \times 180cm$ (cf. Figure 4.1(a)). La nécessité de disposer d'un outil permettant de suivre le déroulement de l'expérimentation de manière précise nous a poussés à l'équiper d'une caméra placée en haut de l'enceinte (cf. Figure 4.1(b)). Ceci permet d'avoir une vue sur l'ensemble de l'espace offert à la navigation des robots Khepera III.

4.1.2 Supervision de l'expérimentation

4.1.2.1 La caméra

Dans la perspective d'avoir une architecture de contrôle complètement distribuée, les robots doivent avoir une autonomie perceptuelle complète. Cependant,

comme premières expérimentations, c'est la caméra de la plate-forme (cf. Figure 4.1(b)) qui délivrera certaines informations sur l'environnement et son évolution. Elle est de marque MARLIN et permet une acquisition des images à une fréquence $15fps$ [1].

La première image acquise par la caméra au début de l'expérimentation permet aussi de calculer ses paramètres extrinsèques. Il s'agit des éléments de la matrice de transformation permettant le passage d'un repère associé à la plate-forme au repère relatif à la caméra et vice versa. Le repère de la plate-forme est défini pour la caméra par quatre points (codes barres) (cf. Section 4.1.2.2).

Ainsi, pour calculer les coordonnées d'un point M dans l'espace (repère absolu) $(X, Y, Z, 1)$, il faut passer par celles données dans le repère caméra $(x, y, z, 1)$. Utilisant la matrice des paramètres extrinsèques, elles se calculent comme suit :

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{pmatrix} & R_{3\times3} & & t_x \\ & & & t_y \\ & & & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}^{-1} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{4.1}$$

avec $R_{3\times3}$ et (t_x, t_y, t_z) sont respectivement la matrice de rotation et le vecteur de translation permettant le passage du repère lié à la plate-forme au repère lié à la caméra (cf. Figure 4.2).

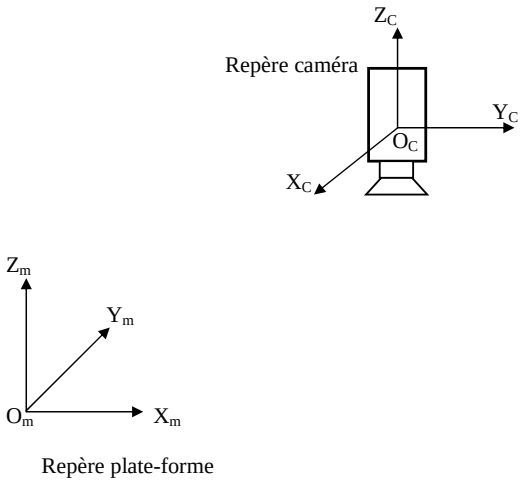


FIGURE 4.2 – Codes barres identifiant les robots et les obstacles.

En calculant ces paramètres à chaque début d'expérimentation, nous rendons cette dernière insensible à la position et l'orientation initiale de la caméra.

1. 15 *Frames Per Second*, ce qui correspond à une image acquise chaque $66.66ms$.

4.1.2.2 L'outil de détection et de suivi

Le besoin d'identifier les éléments de la plate-forme et de suivre l'évolution de l'expérimentation impose de développer un outil robuste et fiable pour traiter le flux d'images fourni par la caméra.

Grâce à la librairie OpenCV [2], nous avons pu associer un outil de détection robuste. Ce dernier permet, pour chaque image acquise par la caméra, de détecter toutes les ellipses présentes sur l'image. Les ellipses de nos éléments (robots, obstacles) seront des cercles. Chacun est entouré d'un code barre comportant un message unique qui identifie l'élément associé à ce message (cf. Figure 4.3). Pour le robot, le point de début de son code barre permet aussi de calculer son orientation. Développé au LASMEA pour l'étalonnage automatique des caméras [Lébraly 10], cette méthode s'avère très robuste pour l'application désirée.

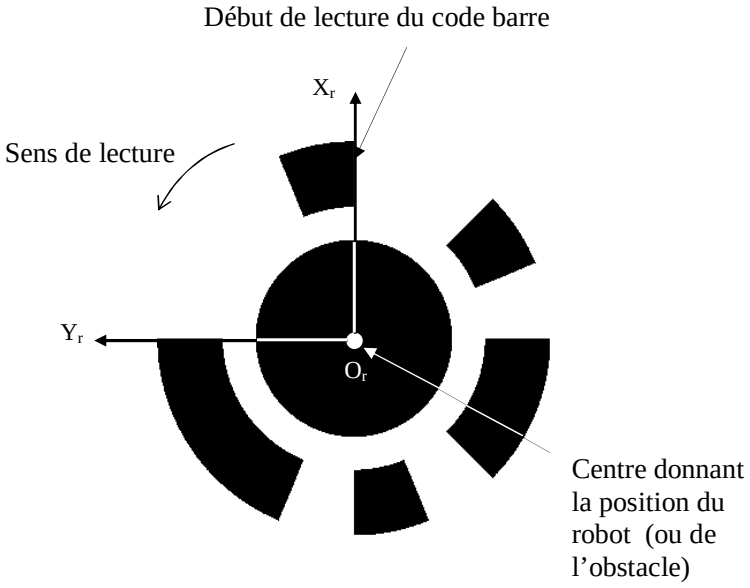


FIGURE 4.3 – Codes barres identifiant les robots et les obstacles.

4.1.2.3 Suivi et contrôle de l'expérimentation

Le lancement et le contrôle de l'expérimentation se fait à l'aide d'un ordinateur appelé superviseur (cf. Figure 4.1(b)). La caméra est reliée à celui-ci via une

2. Il s'agit d'une librairie développée en C++ et spécialisée dans le traitement d'images en temps réel.

carte IEEE 1394-800. Une interface développée grâce à Qt [3] permet à l'utilisateur d'interagir avec la plate-forme (lancer ou arrêter l'expérimentation, choisir la formation désirée, etc.) (cf. Figure 4.4).

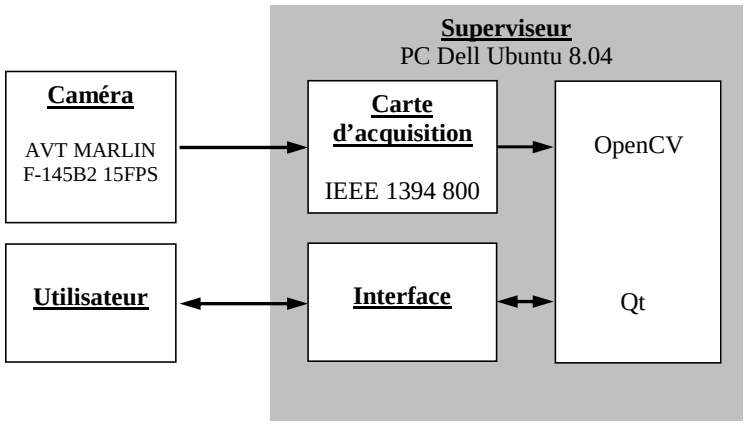


FIGURE 4.4 – Chaîne d'acquisition et de traitement des images acquises par la caméra.

La figure 4.5 permet d'illustrer l'interface utilisée et la vue de l'environnement depuis la caméra centrale.

Afin de recevoir les informations (positions des robots, des obstacles, etc.) perçues par la caméra, chaque Khepera est connecté au superviseur par Wi-Fi via un socket [4]. Le protocole de communication retenu est le TCP [5]. Même si on lui reproche d'être plus lent que son homologue UDP [6], l'émission/réception des données, la gestion des collisions des paquets échangés entre les robots et le superviseur sont assurées.

Nous avons mesurer le temps mis par une trame envoyée par le superviseur à un robot. La valeur **maximale** obtenue est de $5ms$. Comparée à la période d'acquisition d'images imposée par la caméra ($66.66ms$), cette valeur peut être négligée au profit d'un protocole plus sûr.

Un diagramme de classes résumant le principe de fonctionnement et d'échange du superviseur (ordinateur) avec les entités robotiques est donné par la figure 4.6.

3. Il s'agit d'un ensemble de librairies programmées en C++ destinées principalement à développer des applications graphiques.

4. Il s'agit d'un point de communication par lequel le robot ou le superviseur émet ou reçoit des données.

5. Transmission Control Protocol

6. User Datagram Protocol

FIGURE 4.5 – Vue de l'environnement expérimental depuis la caméra centrale.

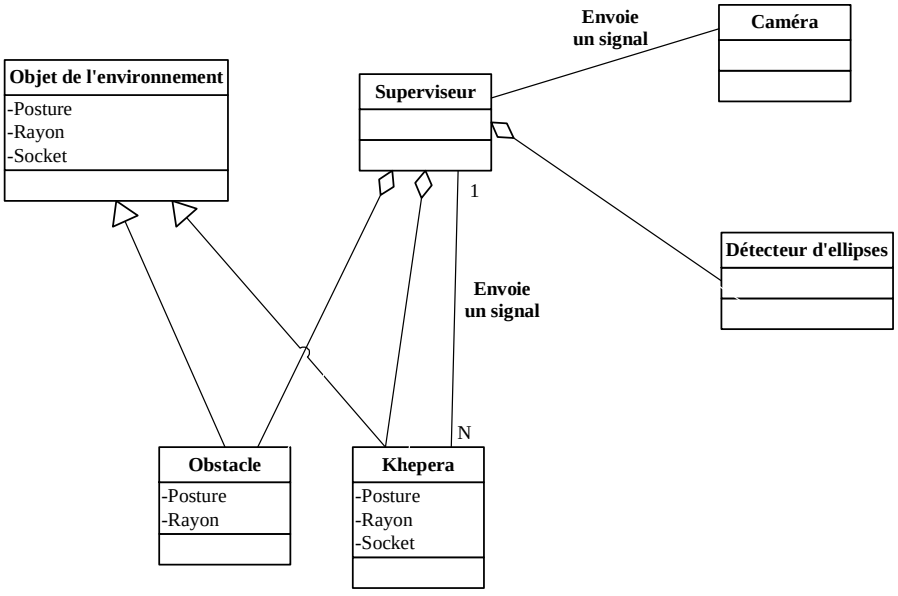


FIGURE 4.6 – Diagramme de classe représentant la gestion de l'expérimentation par le superviseur.

4.1.3 Les robots Khepera III

Khepera III (cf. Figure 4.7) est produit par la société K-Team. Il est de type unicycle avec deux roues entraînées séparément à l'aide de deux servomoteurs

indépendants.

Grâce à la carte KoreBot qui vient s'ajouter à la structure du robot (cf. Figure 4.7), le Khepera III dispose d'un système Linux embarqué. KoreBot lui permet aussi d'augmenter considérablement ses performances (cf. Tableau 4.1). De plus, elle lui offre la possibilité d'exécuter un programme initialement développé en C sur une autre machine (ordinateur) grâce à la compilation croisée. A noter que la plupart des librairies standards C sont supportées. Le robot est également capable d'accueillir des cartes d'extension Compact Flash. Il accepte le Bluetooth, et le Wi-Fi. Les principales caractéristiques [7] du Khepera III dotée d'une carte Korebot sont résumées dans le tableau 4.1.

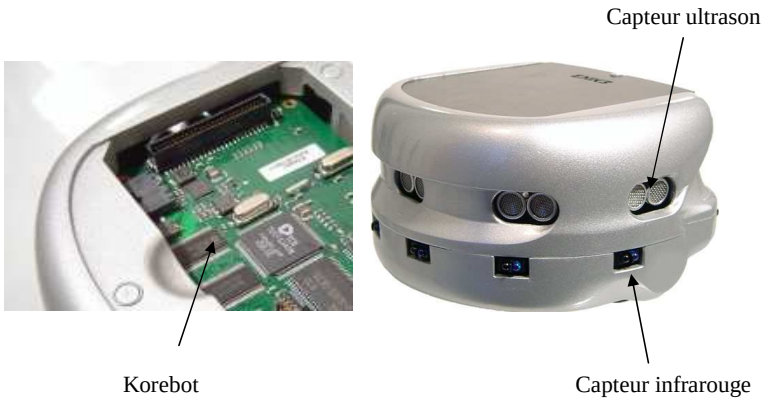


FIGURE 4.7 – Le robot Khepera III.

4.2 Implémentation de l'architecture de contrôle proposée

Après avoir présenté les éléments matériels nécessaires aux expérimentations envisagées sur la plate-forme, nous nous intéressons à présent à l'implantation de l'architecture de contrôle proposée pour la tester à une situation réelle.

L'implémentation de l'architecture de contrôle proposée exige de tenir compte de certains aspects matériels. Nous proposons de détailler ces aspects dans les paragraphes suivants.

7. Les spécifications sont disponibles sur http ://www.k-team.com/mobile-robotics-products/khepera-iii/specifications

Eléments	Informations techniques
Processeur	DsPIC 30F5011 at 60MHz, Extension 400MHz avec XScale de la carte KorBot
Taille	diamètre : 130 mm, hauteur : 70 mm
Poids	690g
Capteurs	- 9 capteurs infrarouge de proximité et de lumière ambiante de petite portée ($10cm$ environ), - 2 capteurs infrarouge de proximité (en bas) pour les applications de suivi de lignes, - 5 capteurs ultra sons de moyenne portée ($1m$ environ)
Vitesse maximale	0.298 m/s
Alimentation	batterie Lithium-Polymère
Autonomie énergétique	8 heures environ, en mouvement permanent (sans Korebot), diminue avec KoreBot et surtout Wi-Fi (moins de 2 heures).

TABLE 4.1 – Caractéristiques techniques du robot Khepera III.

Aspects matériels

Dans la perspective d'obtenir le contrôle distribué souhaité, les robots doivent disposer d'une autonomie perceptuelle complète (utilisation de capteurs ultrasons, infrarouges, caméra embarquée). Cependant, comme nous l'avons déjà précisé, et dans le cadre des expérimentations menées dans cette thèse, c'est la caméra de la plate-forme qui délivre les informations de l'environnement aux robots à travers le superviseur. L'aspect commande de l'architecture (choix du contrôleur à activer, calcul des commandes) est complètement implanté sur les robots. Une importante caractéristique de la caméra est sa fréquence d'acquisition d'images qui est de $15fps$. La caméra donne alors une image chaque $\frac{1}{15}s = 66.66ms$. Une fois acquise, l'image est utilisée par l'outil de détection du superviseur. La détection des codes barre, le calcul de toutes les postures des robots et l'envoi des données par Wi-Fi nécessitent dans l'ensemble entre $20ms$ et $30ms$.

Nous estimons alors le temps entre deux trames reçues par chaque Khepera III à $100ms$.

Coté entité robotique, et grâce à la performance du processeur de la KoreBot (cf. Tableau 4.1), le temps mis par les calculs nécessaires (choix du contrôleur actif, calcul des consignes, vitesses, etc.) (cf. Figure 2.7, page 62) est évalué au maximum à $5ms$.

On remarque que le superviseur est relativement lent comparé aux entités robotiques, principalement à cause de la caméra utilisée. En effet, il envoie les données à une fréquence de $10Hz$, alors que la fréquence de calcul des vitesses au niveau des robots peut aller jusqu'à $200Hz$.

En pratique, cela se répercute sur les vitesses maximales autorisées pour les entités robotiques. En effet, si elles sont élevées, les robots risquent de parcourir des distances importantes avant la réception de la prochaine observation de la caméra. Ainsi, leurs états (positions et orientations) évoluent de façon significative sans qu'ils en soient informées. Chaque robot se déplace alors aux vitesses (v_i, ω_i) calculées à l'instant $(t_i + 5)ms$ jusqu'à l'obtention d'une autre observation de la caméra (cf. Figure 4.8).

Pour pallier cette contrainte matérielle, et pouvoir augmenter la vitesse maximale autorisée, il est proposé d'exploiter un capteur proprioceptif capable de fournir aux robots leurs localisations à une fréquence plus élevée. Il s'agit de l'odomètre. Le recours à ce capteur est très répandu en robotique mobile. Il est appréciée pour avoir une cadence d'acquisition des mesures très élevée. L'odomètre des Khepera III (comme la plupart des odomètres) permet de fournir une quantification des déplacements curvilignes du robot en mesurant la rotation de ses roues. Ces mesures sont réalisées par des codeurs incrémentaux. Le change-

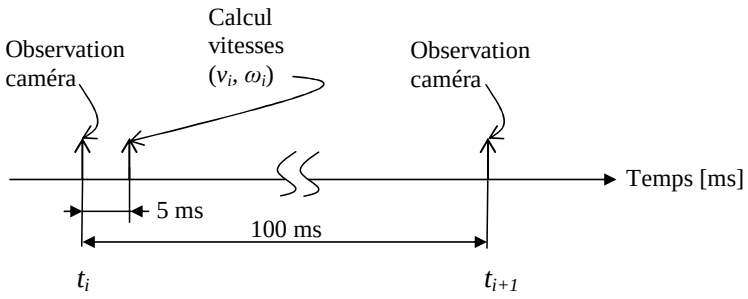


FIGURE 4.8 – Séquentialité entre la réception des observations issues de la caméra et le calcul des vitesses au niveau d'un Khepera III.

ment d'orientation est déduit de la différence de distance parcourue entre la roue droite et gauche.

Le calcul de la position relative et de l'orientation du robot est réalisé par intégrations successives de ses déplacements élémentaires. Les résultats sont donc donnés par rapport à son état initial. En pratique, cela signifie aussi que la position calculée dérive au cours du temps à cause, en partie, du bruit de mesure. Nous proposons de réduire cette dérive grâce à la fusion des données provenant de l'odomètre (données proprioceptives) et les observations fournies par la caméra (données extéroceptives) à travers un filtre de Kalman [Kalman 60].

Le filtre de Kalman opère en deux étapes : *Prédiction* et *Mise à jour*. La phase de prédiction utilise l'état estimé de l'instant précédent pour produire une estimation de l'état courant. Dans l'étape de mise à jour, les observations de l'instant courant (issues de la caméra centrale dans notre cas) sont utilisées pour corriger l'état prédit dans le but d'obtenir une estimation plus précise.

Pour expliquer le filtre de Kalman implémenté au sein de chaque robot i, nous considèrons le vecteur d'état de celui-ci comme étant $\underline{x_i} = (x_i, y_i, \theta_i)$. Il s'agit de sa position et de son orientation dans le repère absolu de l'environnement.

Ce vecteur d'état est associé à une matrice de covariance P_i (cf. Equation 4.2) décrivant les erreurs liées à ses paramètres.

$$P_i = \begin{bmatrix} \sigma_{x_i}^2 & \sigma_{x_i y_i} & \sigma_{x_i \theta_i} \\ \sigma_{y_i x_i} & \sigma_{y_i}^2 & \sigma_{y_i \theta_i} \\ \sigma_{\theta_i x_i} & \sigma_{\theta_i y_i} & \sigma_{\theta_i}^2 \end{bmatrix} \tag{4.2}$$

Nous proposons dans ce qui suit d'expliquer le principe des deux phases du filtre de Kalman (la prédiction et la mise à jour) et la façon dont elles sont implémentées dans les robots.

4.2.0.1 La prédiction

Considérons un traitement au niveau d'un robot i. Ce dernier fait évoluer sa posture (le vecteur d'état $\underline{x}_i$ et sa matrice de covariance P_i) selon un modèle qui tient compte des informations délivrées par son odomètre.

Le vecteur d'état $\underline{x}_i$ évolue selon la relation

$$\underline{x}^-_{i_{k+1}} = f_i(\underline{x}_{i_k}, u_{i_k}) \tag{4.3}$$

où $\underline{x}^-_{i_{k+1}}$ est la prédiction de l'état du robot i. k représente le k^{eme} échantillon de temps et u_{i_k} les informations proprioceptives. $\underline{x}_{i_k}$ est l'état calculé à l'étape de mise à jour à l'instant k (cf. Equation 4.10).

La fonction d'évolution f_i est donnée pour chaque élément de $\underline{x}_i$ selon le modèle d'évolution suivant :

$$\begin{cases} x^-_{i_{k+1}} = x_{i_k} + v_i \cos(\theta_{i_k})\Delta t \\ y^-_{i_{k+1}} = y_{i_k} + v_i \sin(\theta_{i_k})\Delta t \\ \theta^-_{i_{k+1}} = \theta_{i_k} + \omega_i \Delta t \end{cases} \tag{4.4}$$

Les vitesses linéaire et angulaire (v_i, ω_i) utilisées dans le modèle d'évolution sont celles issues des odomètres. Δt est la durée entre l'instant du dernier état du robot $\underline{x}_{i_k}$ et l'instant de la perception de l'odomètre.

L'équation d'évolution de la matrice de covariance $P^-_{i_{k+1}}$ associée au vecteur d'état prédit $x^-_{i_{k+1}}$, s'exprime par [Siegwart 04]

$$P^-_{i_{k+1}} = F_{x_{i_k}} P_{i_k} F^T_{x_{i_k}} + F_{u_{i_k}} Q_{i_k} F^T_{u_{i_k}} \tag{4.5}$$

où

- $F_{x_{i_k}}$ et $F_{u_{i_k}}$ sont respectivement les jacobiennes de la fonction f_i par rapport à $\underline{x}_i$ et u_i,
- Q_{i_k} est la matrice de covariance du bruit qui affecte la mesure de l'odomètre (mesures de la vitesses linéaire et angulaire du robot),
- P_{i_k} est la matrice de covariance liée au vecteur d'état $\underline{x}_{i_k}$ et calculée à l'étape de mise à jour à l'instant k (cf. Equation 4.11). A l'instant initial $t = 0$, nous la réinitialisons à une valeur quelconque (une matrice à 0 par exemple).

Des essais sur un Khepera III utilisant l'odométrie nous ont permis de connaître la matrice Q_{i_k}. Ainsi, pour déterminer la variance de la vitesse linéaire fournie par l'odomètre, l'expérimentation consiste à donner une vitesse linéaire constante

au Khepera III ($0.1m/s$) et une vitesse angulaire nulle. Nous relevons la distance donnée par l'odomètre au bout de $10s$. Nous pouvons alors calculer la vitesse linéaire du robot estimée par l'odomètre. L'opération est renouvelée une dizaine de fois afin de calculer l'écart type.

Nous reprenons le même principe pour la vitesse angulaire. Cette fois, la vitesse linéaire est nulle et le robot tourne autour de lui-même avec une vitesse angulaire constante ($0.2rd/s$).

En supposant que la vitesse linéaire et angulaire sont des variables indépendantes, la matrice Q_{i_k} devient

$$Q_{i_k} = \begin{bmatrix} 10^{-4} & 0 \\ 0 & 7.6 \times 10^{-3} \end{bmatrix} \tag{4.6}$$

les unités des variances des vitesses linéaire et angulaire sont $(m/s)^2$ et $(rd/s)^2$ respectivement.

Chaque robot fait donc évoluer son état selon le modèle décrit dans l'étape de prédiction jusqu'à l'arrivée de l'observation extéroceptive permettant de mettre à jour son dernier état prédit.

4.2.0.2 La mise à jour de l'état du robot

A l'arrivée d'une information extéroceptive mesurée z^m_{k+1} (mesure de l'état du robot issue de la caméra liée à une matrice de covariance B_k), le robot i met à jour son état selon les relations suivantes.

$$z_{k+1} = (z^m_{k+1} - I\underline{x}^-_{i_{k+1}}) \tag{4.7}$$

$$S_{k+1} = P^-_{i_{k+1}} + B_{k+1} \tag{4.8}$$

$$K_{k+1} = P^-_{i_{k+1}} S^{-1}_{k+1} \tag{4.9}$$

$$\underline{x}_{i_{k+1}} = \underline{x}^-_{i_{k+1}} + K_{k+1} z_{k+1} \tag{4.10}$$

$$P_{i_{k+1}} = (I - K_{k+1}) P^-_{i_{k+1}} \tag{4.11}$$

Où
- z_{k+1} représente l'innovation de l'observation,
- $\underline{x}^-_{i_{k+1}}$ la prédiction du vecteur d'état $\underline{x}_i$,
- I est la matrice identité,
- S_{k+1} la matrice de covariance de z_{k+1},
- K_{k+1} représente le gain de Kalman,
- $\underline{x}_{i_{k+1}}$ le vecteur d'état mis à jour,

– $P_{i_{k+1}}$ la matrice de covariance de $\underline{x}_{i_{k+1}}$.

Pour déterminer la matrice de variance liée à la mesure provenant de la caméra B_k, nous avons mis un Khepera III dans plusieurs positions de la plate-forme. Dans chacune de ces positions, le robot est laissé pendant $30s$ environ. Le taux d'acquisition d'images ($15fps$) donne alors 450 mesures en moyenne. En supposant que les éléments de z^m_{k+1} $(x^m_{k+1}, y^m_{k+1}, \theta^m_{k+1})$ sont des variables indépendantes, la matrice B_k s'exprime alors

$$B_k = \begin{bmatrix} 9 \times 10^{-6} & 0 & 0 \\ 0 & 9 \times 10^{-6} & 0 \\ 0 & 0 & 1.2 \times 10^{-3} \end{bmatrix} \tag{4.12}$$

L'unité des variances des positions mesurées (x^m, y^m) est m^2. La variance de l'orientation mesurée θ^m est exprimée en rd^2.

La séquentialité des étapes du filtre de Kalman et le calcul des vitesses des robots est donnée dans la figure 4.9. En fonction des vitesses maximales, il faudra adapter la fréquence de l'étape de prédiction entre deux étapes de mise à jour : plus le robot se déplace à vitesse importante, plus sa posture change rapidement. Il faudra donc la prédire plus souvent.

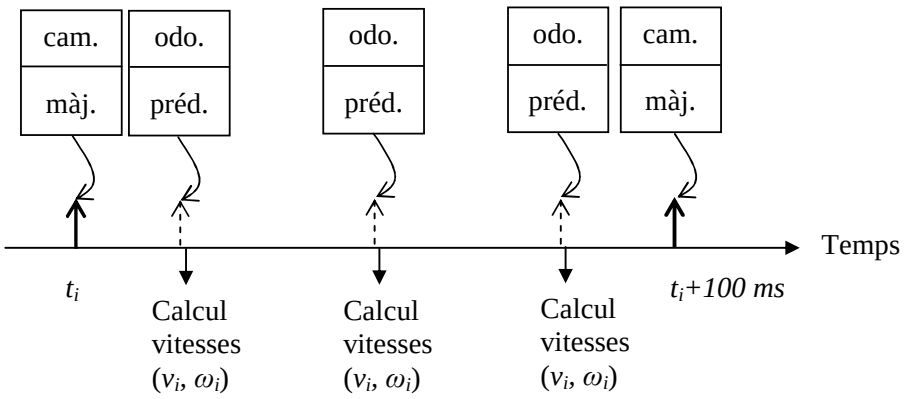


FIGURE 4.9 – Séquentialité des étapes de prédiction, de mise à jour et de calcul des vitesses du robot i.

4.2.0.3 Prédiction des mouvements des autres robots

Pour s'assurer que les robots calculent leurs commandes à partir de données viables et relativement à jour, chaque robot i doit tenir compte de l'évolution des autres afin de pouvoir les éviter et gérer la collision avec eux. En effet, il ne doit pas attendre la prochaine observation de la caméra pour mettre à jour leurs positions car l'information sur l'environnement fournie par la caméra à l'instant t_i peut devenir peu fiables au voisinage et avant t_{i+1} (instant de l'observation suivante). Ceci est particulièrement vrai si les robots se déplacent à vitesses élevées. Pour cela, nous proposons que chaque entité robotique fasse évoluer sa propre posture grâce au filtre de Kalman expliqué précédemment, mais doit aussi faire évoluer celles des autres.

Comme le robot n'a pas accès aux odomètres des autres, nous proposons qu'il les fasse évoluer selon un modèle cinématique décrit par le système d'équations 4.13 où chacun évolue à vitesse et orientation constantes. Pour connaître ou plutôt estimer les vitesses des autres, il est proposé que chaque robot utilise les deux dernières observations de la caméra reçues. Connaissant le temps Δt_{obs} entre ces deux observations, il peut estimer des vitesses moyennes (v_j, ω_j) pour chaque entité j.

Du point de vue d'un robot i, chaque robot j évolue alors selon les équations suivantes

$$\begin{cases} x^-_{j_{k+1}} = x_{j_k} + v_j \cos(\theta_{j_k})\delta t \\ y^-_{j_{k+1}} = y_{j_k} + v_j \sin(\theta_{j_k})\delta t \\ \theta^-_{j_{k+1}} = \theta_{j_k} + \omega_j \delta t \end{cases} \tag{4.13}$$

où $(x_{j_k}, y_{j_k}, \theta_{j_k})$ correspondent à la posture du robot j fournie par la dernière observation de la caméra. δt est le temps entre l'instant de cette observation $(x_{j_k}, y_{j_k}, \theta_{j_k})$ et l'instant où se fait le calcul de prédiction (cf. Equation 4.13).

Il s'agit évidemment d'une prédiction donnant des résultats moins précis que ceux obtenus avec un filtre de Kalman collectif qui traiterait un vecteur d'état englobant tous les robots. Néanmoins, l'utilisation d'un tel filtre nécessite que les robots échangent leurs données odométriques. Il faudra alors prendre en compte certains éléments comme les temps de latence liés à la communication entre entités autonomes. Ce problème constitue une thématique de recherche à part entière [Karam 09], et qui n'est pas l'objet de cette thèse.

Nous allons nous intéresser dans les paragraphes qui suivent aux expérimentations effectuées et aux résultats obtenus.

4.3 Expérimentations

Les contributions menées illustrent d'abord la contribution relative à l'évitement de collision entre les entités robotiques. Nous nous intéressons ensuite à la possibilité d'ajout de nouveaux robots à la formation en cours de navigation. Enfin, nous abordons la stabilité de l'architecture de contrôle proposée en présence de commutations entre les contrôleurs d'attraction vers la cible et d'évitement d'obstacles.

4.3.1 Évitement de collision entre les robots

La tâche de l'évitement de collision entre les robots ainsi que la fonction de pénalité ont été testées sur deux Khepera III (cf. Section 2.3.2, page 72). Afin de pouvoir observer leurs comportements, les deux robots sont initialement disposés de telle manière qu'ils aient à se croiser et s'éviter entre eux avant de rejoindre leurs cibles respectives. On remarque que les deux robots s'évitent bien dans le sens trigonométrique (cf. Figure 4.10) et finissent par atteindre leurs cibles choisies ici statiques.

La figure 4.11 représente les vitesses linéaires des deux robots ainsi que la distance qui les sépare d_{12}. On remarque que cette distance affecte la vitesse des deux robots grâce à la fonction de pénalité. Ainsi, se déplaçant à une vitesse maximale de $6cm/s$, les deux robots décelèrent pour s'éviter. Ils accélèrent de nouveau une fois s'éloignant l'un de l'autre (avec l'augmentation de d_{12}). La deuxième décélération de chacun correspond à l'atteinte des cibles finales respectives.

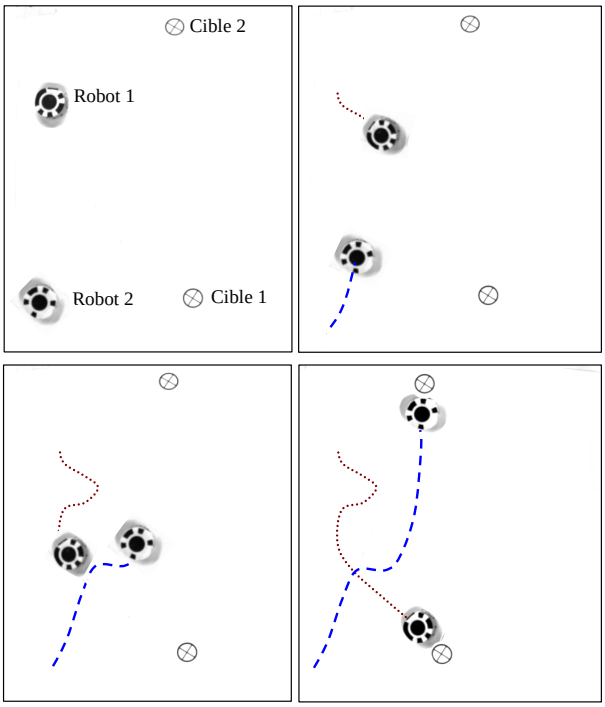


FIGURE 4.10 – Trajectoires des deux robots mobiles allant vers leurs cibles en s'évitant.

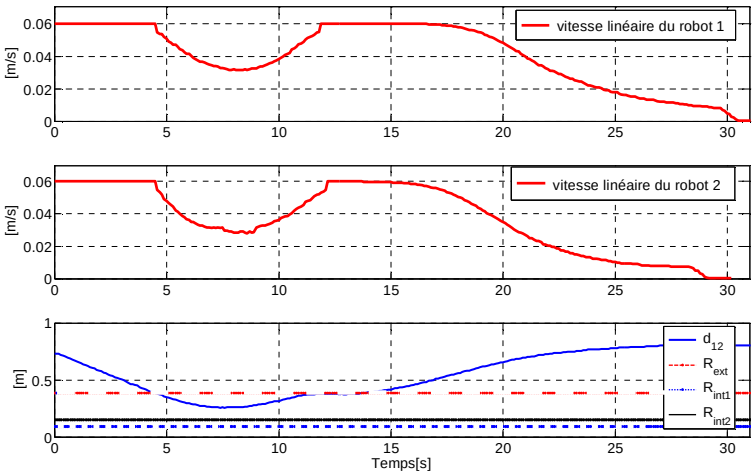


FIGURE 4.11 – Vitesses linéaires des deux robots dépendant de la distance qui les sépare.

4.3.2 Navigation en formation et flexibilité de l'architecture de contrôle proposée

Pour voir la tâche de la navigation en formation, nous proposons d'utiliser un groupe de robots Khepera III contrôlés par l'architecture de contrôle proposée (cf. Section 2.3, page 62). Ainsi, une structure virtuelle initialement de type triangle est demandée par le superviseur aux entités robotiques. Pour se focaliser sur la formation du groupe, l'environnement est d'abord sans obstacles. Pour chaque entité, seul le contrôleur d'attraction vers la cible dynamique sera alors actif. On remarque que les robots atteignent et suivent bien leurs cibles (cf. Figure 4.12). Les distances d_{S_i} séparant les robots de leurs cibles respectives sont données dans la figure 4.13. On remarque que toutes ces distances tendent vers 0 ce qui confirme le comportement des robots observé dans la figure 4.12.

Pour rajouter d'autres robots à la formation, il suffit d'ajouter des cibles secondaires à la structure virtuelle existante comme nous l'avons déjà expliqué (section 2.1, page 56). Ainsi, une quatrième cible est accrochée à la structure virtuelle. En rajoutant un robot, on remarque que celui-ci la rejoint et la suit. La formation a alors évolué d'un triangle à un losange. L'évolution de la distance d_{S_4} de ce robot à la cible est donnée dans la figure 4.14. Elle confirme la trajectoire observée.

L'architecture de contrôle proposée peut être qualifiée de flexible. En effet, il est possible d'ajouter autant de robots qu'on souhaite. Il suffit simplement de s'assurer que le nombre de robots N et de cibles N_C vérifient $N_C \geq N$.

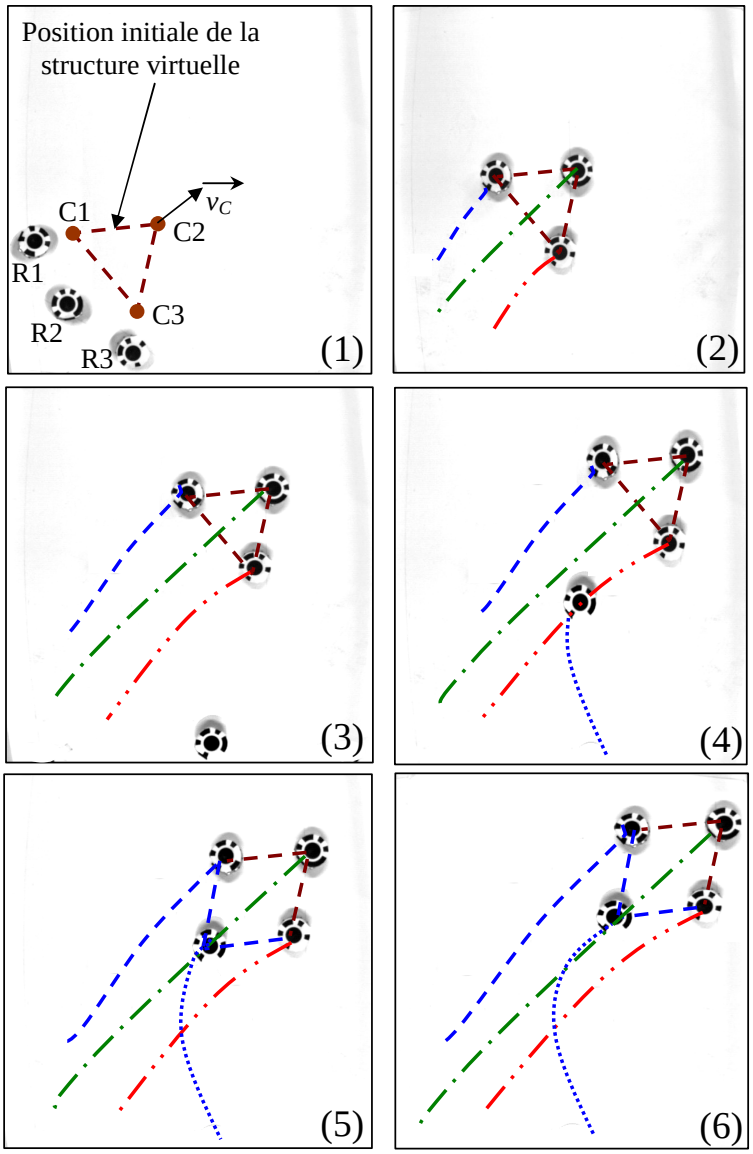


FIGURE 4.12 – Trajectoires des robots mobiles navigant en formation sur la plateforme.

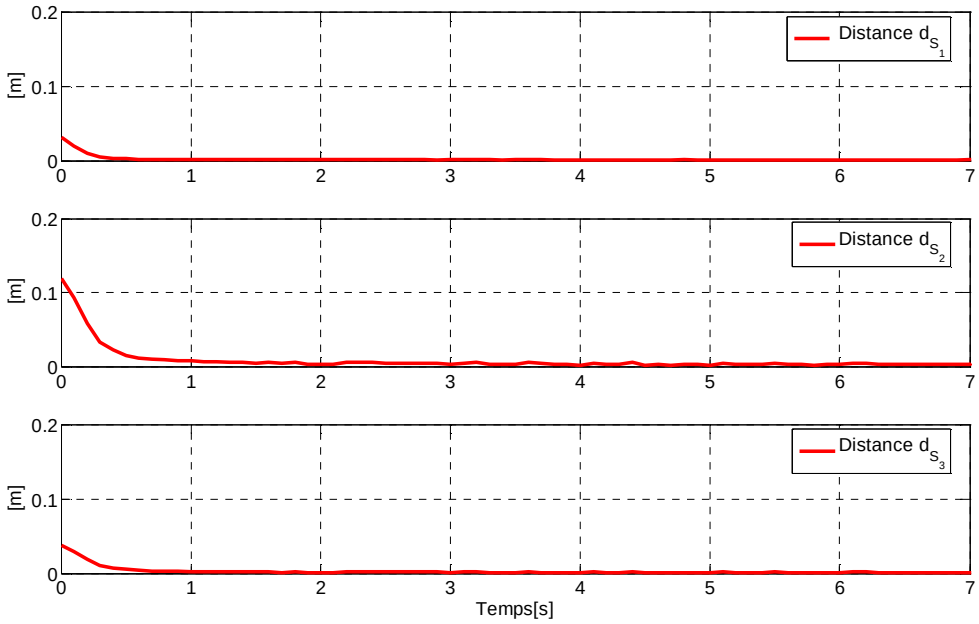


FIGURE 4.13 – Evolution des distances séparant les robots de leurs cibles.

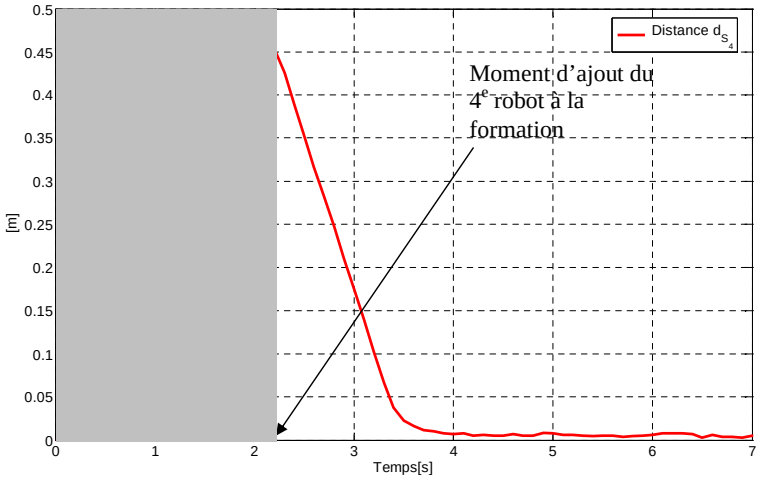


FIGURE 4.14 – Evolution de la distance séparant le dernier robot ajouté de sa cible.

4.3.3 Stabilité de l'architecture de contrôle proposée en présence d'obstacles

Dans le chapitre 3, il est proposé de changer le gain de commande k relatif à l'expression de la loi de commande régissant la vitesse angulaire du robot (cf. Equation 3.1b, page 118). Celui-ci devient alors dynamique et évolue en fonction de l'erreur d'orientation $\tilde{\theta}$. L'objectif est d'augmenter la rapidité de la convergence de cette erreur (et donc celle de la fonction de Lyapunov) tout en respectant les contraintes des vitesses maximales des robots (cf. Section 3.3.3, page 138).

Nous proposons ici d'étudier la stabilité de l'architecture de contrôle proposée et de valider les résultats théoriques et de simulation du chapitre 3 (cf. Section 3.3.3, page 138). Pour cela, une structure virtuelle triangulaire est donnée aux robots. Deux des trois robots doivent éviter chacun un obstacle avant de rejoindre la formation (cf. Figure 4.15).

Pour voir l'utilité du gain dynamique proposé, l'expérimentation est d'abord menée avec un gain constant. Nous nous intéressons plus particulièrement aux deux robots devant commuter entre les deux contrôleurs. En effet, l'autre robot (robot 1) a un comportement identique à ceux vus précédemment dans la formation triangle/losange (cf. Figure 4.13).

On remarque que les deux robots en question accomplissent bien leurs tâches. En effet, ils évitent les obstacles et rejoignent les cibles dynamiques (cf. Figure 4.15). Les distances qui les séparent de leurs cibles le confirment (cf. Figure 4.16). A noter que ces distances ne sont pas calculées au moment où le contrôleur d'évitement d'obstacles est actif (car elles ne sont pas significatives dans ce cas).

La figure 4.16 illustre également l'évolution des fonctions de Lyapunov. Lorsque le contrôleur d'attraction vers la cible dynamique est actif, les fonctions de Lyapunov sont décroissantes voire nulles. Toutefois, on remarque un saut de la fonction à l'instant $t = 15s$ pour le robot 2. Ceci s'explique par la nature de la cible correspondante à ce robot. Il s'agit d'une cible s'approchant : ainsi, le robot la dépasse d'abord avant de corriger son orientation pour la rejoindre de nouveau (cf. Section 2.3.1, page 63). Une légère augmentation de la distance au même instant le confirme. On constate la même situation pour le robot 3 (instant $20.3s$) avec une augmentation moins importante de la fonction de Lyapunov.

A l'instant de la commutation vers le contrôleur d'évitement d'obstacles, on remarque un saut important dans la fonction de Lyapunov pour les deux robots. Toutefois, les fonctions décroissent de nouveau après. Un saut relativement important est remarqué quand le contrôleur d'évitement d'obstacles est actif pour le robot 3. Il correspond au passage de la phase d'attraction à la phase de répulsion du contrôleur (cf. Section 2.3.2, page 72).

D'après ce qui précède, l'architecture de contrôle proposée utilisant un gain constant pour la commande de la vitesse angulaire est faiblement stable.

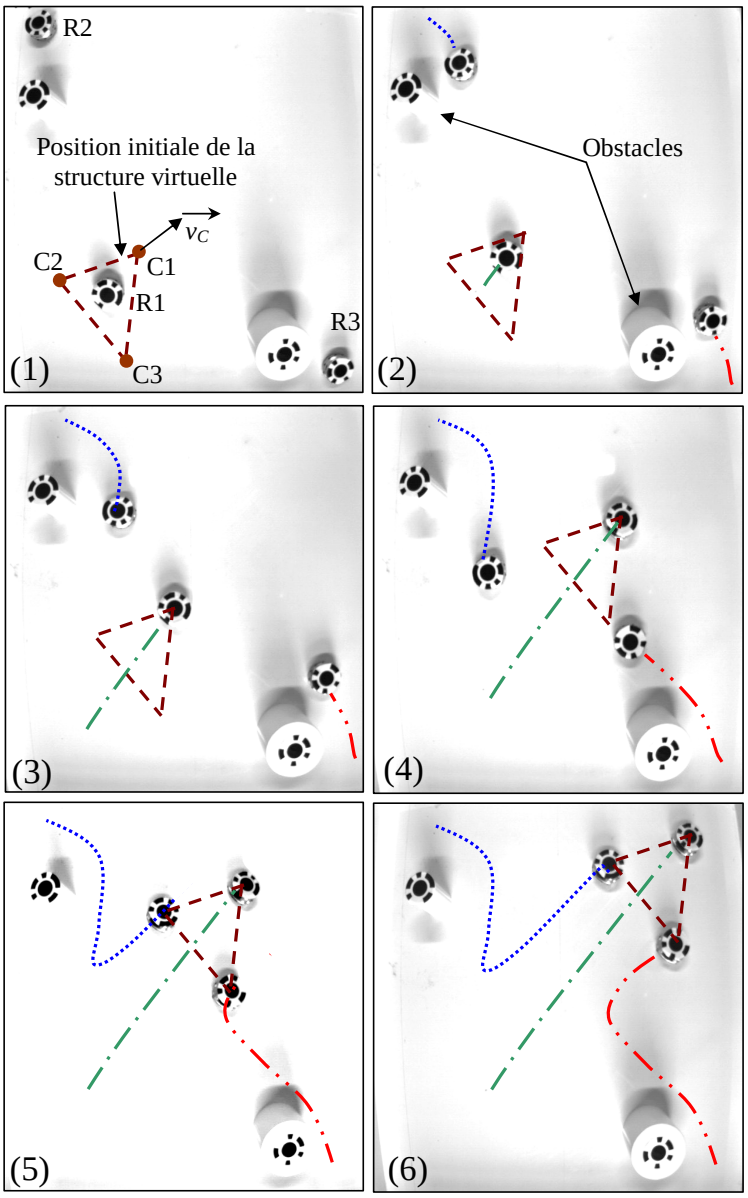
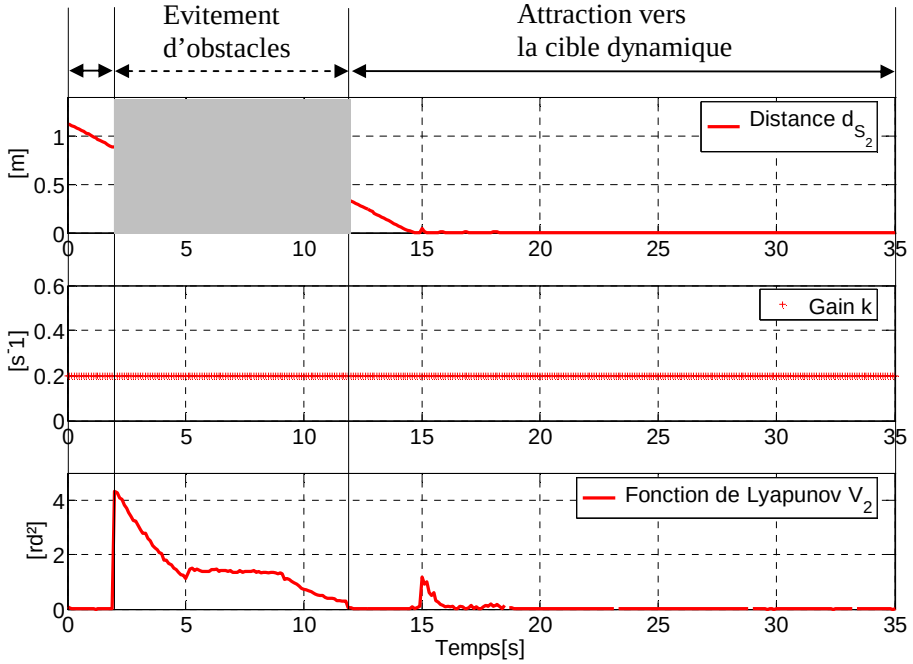
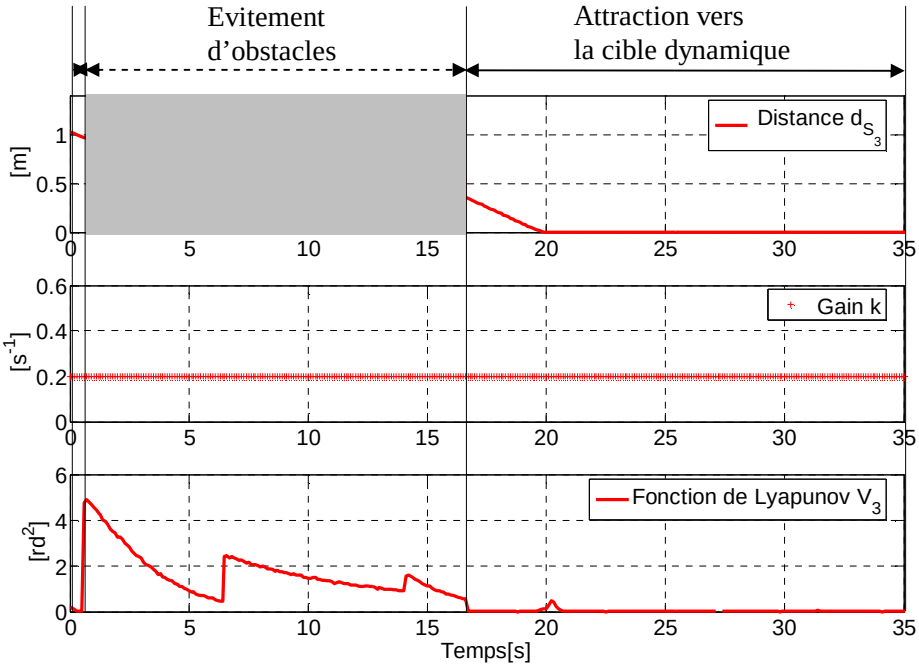


FIGURE 4.15 – Navigation en formation en présence d'obstacles.



(a) Robot 2



(b) Robot 3

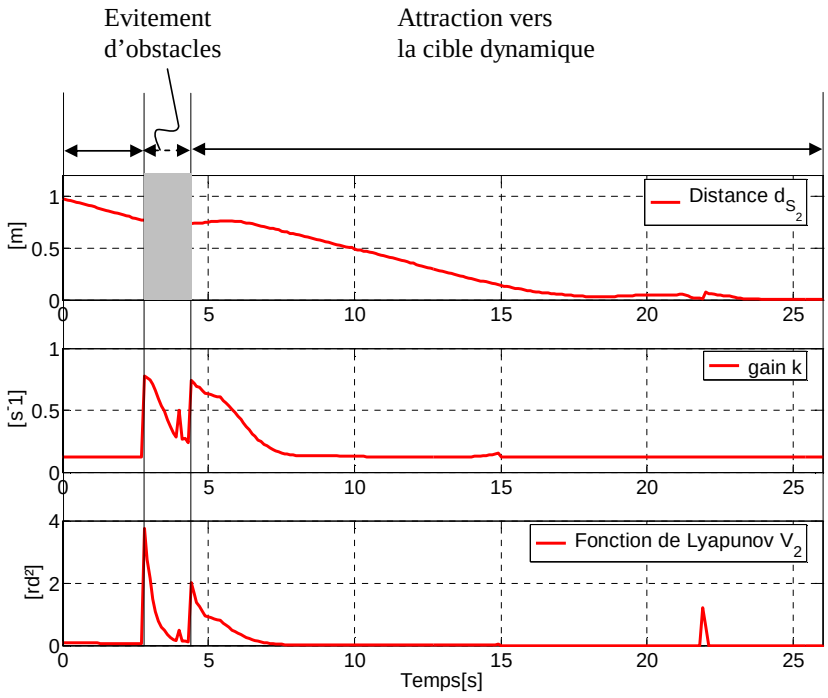
FIGURE 4.16 – Evolution des fonctions de Lyapunov et des distances séparant les robots de leurs cibles (gain k constant).

Nous répétons la même l'expérimentation avec les mêmes conditions initiales. Cette fois, le gain k est dynamique. Il est réévalué à chaque évènement discret (cf. Section 3.3.3, page 138). Les distances séparant les robots 2 et 3 de leurs cibles, les variations du gain k et la fonction de Lyapunov de chaque robot sont représentées dans la figure 4.17.

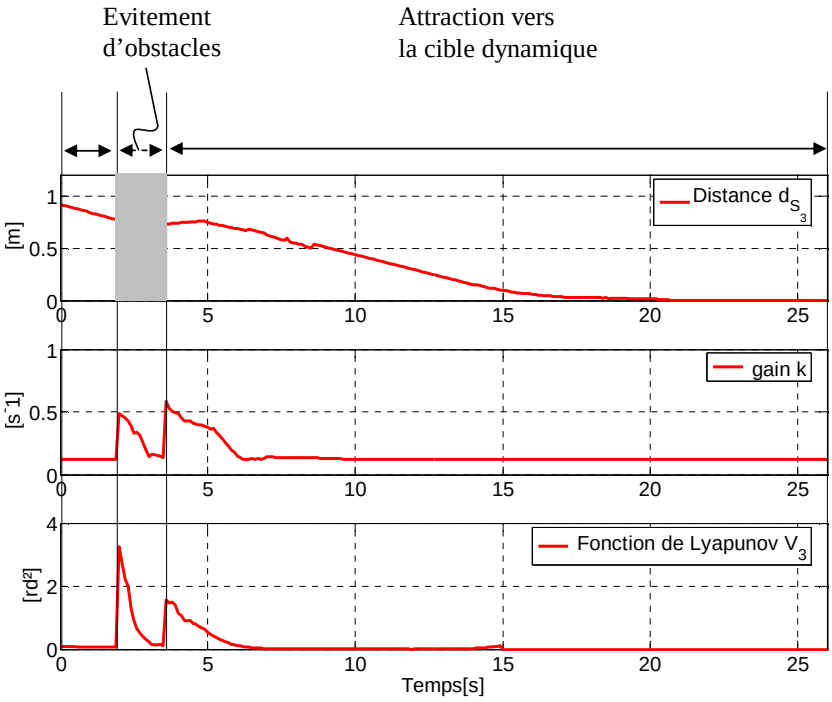
On remarque que grâce au gain dynamique, le robot répond plus rapidement à l'angle consigne. Pour les deux robots, ceci est particulièrement remarquable dans le contrôleur d'évitement d'obstacles où la fonction de Lyapunov converge de façon plus rapide que lors de l'utilisation d'un gain constant (cf. Figure 4.16). Le gain k augmente à la commutation d'un contrôleur à un autre. On remarque qu'il diminue avec la diminution de l'erreur d'orientation et donc de la fonction de Lyapunov. En dehors des instants de commutations et lorsque l'erreur d'orientation se stabilise à 0, le gain retrouve sa valeur initiale.

Pour le robot 2, on observe un saut dans la fonction de Lyapunov alors qu'il n'y a pas d'évènements discrets (instant $22s$). En effet, comme dans le cas d'un gain constant, la cible passe d'une cible s'approchant à une cible s'éloignant. L'augmentation de la distance d_{S_2} au même moment le confirme.

De la même façon que précédemment, nous pouvons conclure que l'architecture de contrôle est faiblement stable.



(a) Robot 2



(b) Robot 3

FIGURE 4.17 – Evolution des fonctions de Lyapunov et des distances séparant les robots de leurs cibles (gain k dynamique).

4.4 Conclusion

Afin de mener à bien nos expérimentations sur la plate-forme dédiée aux Khepera III, nous avons commencé par l'équiper d'un superviseur avec un outil de détection robuste connecté à une caméra centrale. Les expérimentations nous ont permis de tester l'architecture de contrôle proposée, de valider ses contrôleurs élémentaires et de vérifier sa stabilité de façon effective.

Les contraintes matérielles liées principalement à la caméra utilisée ont imposé des vitesses maximales relativement basses pour les robots. Pour pallier cela, nous avons équipé chacun d'un filtre de Kalman assurant une fusion de données entre des capteurs proprioceptifs travaillant à une fréquence élevée, et des données extéroceptives arrivant à une fréquence relativement basse. Grâce à ce filtre, nous avons pu augmenter les vitesses maximales autorisées pour les robots. Cependant, nous avons remarqué que la fréquence trop élevée de l'odomètre n'est pas systématiquement un avantage. En effet, si le robot est contrôlé à cette fréquence, il peut devenir très sensible aux moindres variations de sa position et celle de sa cible ce qui engendre des oscillations. Au même temps, une fréquence d'échantillonnage trop basse reproduit l'effet observé à l'utilisation de la caméra centrale, celui où les entités parcourent des distances importantes en boucle ouverte. Il faudra alors trouver la meilleure fréquence de contrôle assurant un bon compromis. Cette fréquence change naturellement avec la vitesse maximale autorisée pour les entités mobiles.

Ces expérimentations constituent les premiers essais. En effet, afin de valider l'aspect distribué de notre architecture de contrôle, il est impératif que les robots disposent d'une autonomie perceptuelle complète. Il faut alors utiliser les capteurs ultrasons et infrarouges du robot. Des caméras embarquées peuvent aussi être associées aux Khepera III. Ces capteurs permettent de percevoir les obstacles et de connaître leurs tailles indispensable pour la méthode du cycle-limite [Baudoin 10].

Notons que la caméra centrale et le superviseur permettent de définir un repère absolu et donc commun à toutes les entités robotiques, ce qui simplifie considérablement leur localisation. L'utilisation exclusive des capteurs embarqués des robots impose que chacun se localise uniquement par rapport à ses voisins et la cible principale.

Chapitre 5

Conclusion générale et perspectives

Conclusion générale

Les travaux réalisés dans cette thèse correspondent à la volonté d'investiguer les architectures de contrôle pour les systèmes multi-robots mobiles. L'intérêt est porté plus particulièrement au contrôle distribué et réactif des entités robotiques autonomes. La tâche à laquelle nous nous sommes intéressé est la navigation en formation en présence d'obstacles.

Au chapitre 1, nous avons d'abord abordé la robotique mobile d'une manière générale et les enjeux liés à la coopération d'un groupe d'entités autonomes. Au vu du contexte des travaux de thèse relatifs au contrôle/commande d'un Système Multi-Robots (SMR), l'état de l'art réalisé s'est focalisé particulièrement sur les architectures de contrôle/commande dédiées à ces systèmes. L'objectif est de s'inspirer des approches traitées dans la littérature afin de proposer une architecture la plus appropriée pour le contrôle du SMR. L'aspect distribué souhaité, notamment pour l'autonomie décisionnelle qu'il offre aux entités, impose de s'éloigner des architectures centralisées où le contrôle de tout le système est géré au niveau d'une unité de calcul centrale. Plusieurs travaux exposés au chapitre 1 montrent aussi l'intérêt grandissant porté par la communauté scientifique pour le contrôle réactif. Se plaçant dans ce cadre, nous nous sommes focalisé sur les approches réactives et distribuées pour la réalisation de la tâche de la navigation en formation en présence d'obstacles. L'approche hiérarchique, comportementale, et la structure virtuelle constituent les principales méthodes utilisées. Chacune de ces méthodes présente ses avantages et ses inconvénients. L'idée de nos travaux est de combiner ces approches afin de pallier les inconvénients de chacune et d'obtenir l'ouverture et la flexibilité désirées de notre architecture de contrôle.

Ainsi, l'étude bibliographique réalisée nous a motivé pour choisir l'approche

comportementale afin de distribuer le contrôle sur les entités robotiques et satisfaire au critère d'ouverture. Il s'agit d'offrir la possibilité de mettre à jour le contrôle des robots au fur et à mesure que leur tâche est amenée à être plus complexe, plutôt que de concevoir un nouveau contrôle à chaque mise à jour de cette tâche. Cette approche consiste à décomposer la tâche globale en un ensemble de tâches élémentaires. L'architecture de contrôle dispose alors d'un ensemble de comportements/contrôleurs. Chacun permet d'accomplir une tâche élémentaire correspondante. Pour garantir une navigation en formation et une flexibilité pour le nombre des entités robotiques utilisées, la méthode de la structure virtuelle est combinée à l'approche comportementale. L'idée est que les robots suivent un corps virtuel de la forme géométrique souhaitée et dont la dynamique (vitesse, orientation, changement de la structure interne) détermine le contrôle à appliquer aux robots. L'utilisation de la structure virtuelle offre la flexibilité souhaitée. En effet, l'ajout d'autres robots à la formation se fait simplement en ajoutant de nouvelles cibles à la structure virtuelle.

Le chapitre 2 nous a permis d'introduire notre architecture de contrôle proposée. Elle comprend principalement deux comportements/contrôleurs élémentaires. Le contrôleur d'attraction vers une cible dynamique permet à chaque entité de suivre une cible qui est un sommet du corps virtuel souhaité. Nous avons prouvé qu'il permet au robot de converger vers la cible (en position et en orientation). A la détection d'un obstacle, l'architecture commute vers le deuxième contrôleur : celui de l'évitement d'obstacles. Ce dernier a été choisi après une étude bibliographique sur les méthodes utilisées pour cette tâche. Il est basé sur les cycles-limites et permet à chaque entité d'éviter les obstacles statiques. Dans cette thèse, ce contrôleur est amélioré de telle sorte qu'il assure l'évitement d'obstacles dynamiques et la collision entre les entités robotiques. En plus du contrôleur d'évitement d'obstacles, et pour assurer un risque de collision nul entre robots, une fonction de pénalité affecte la vitesse linéaire de toute entité s'approchant dangereusement des autres. Afin de suivre les consignes désirées (position, angle) par le contrôleur actif, une loi de commande unique pour les deux contrôleurs a été proposée. Elle permet de traduire ces consignes en vitesses (linéaire et angulaire) adéquates pour le robot.

Le chapitre 2 traite également le mécanisme de coopération entre entités autonomes. Nous proposons que l'attribution des cibles du corps virtuel soit soumise à un critère de coût relatif pour que chaque robot soit en mesure de choisir la cible la plus appropriée pour lui, mais aussi pour le groupe. Ainsi, chacun est en mesure de connaître la cible la plus convenable, de la garder ou de se sacrifier au profit d'une autre entité qui en a le plus besoin.

La cohabitation de plusieurs comportements/contrôleurs dans toute architec-

ture de contrôle comportementale ne se fait pas toujours sans encombre. En effet, la commutation entre ces contrôleurs est souvent synonyme de changement brutal et d'une discontinuité de commande pouvant occasionner l'instabilité du système. Ceci est un inconvénient souvent reproché à l'approche comportementale et peu traité dans la littérature à cause de la difficulté de formaliser et d'étudier la stabilité de ces architectures. Le chapitre 3 a été dédié à cette étude. Grâce aux systèmes hybrides permettant le contrôle des systèmes continus (contrôleurs) en présence d'évènements discrets (conditions de commutations), une stabilité au moins faible de l'architecture de contrôle a été prouvée. Cette stabilité est déterminée dans un domaine exprimé en fonction des contraintes structurelles des robots (vitesses maximales et accélérations maximales). Ceci a permis alors de déterminer la limite de la dynamique (vitesse linéaire, angulaire) autorisée au corps virtuel en fonction de ces contraintes. Pour obtenir une stabilité asymptotique, nous proposons de satisfaire le théorème des Fonctions de Lyapunov Multiples FLM. Cependant, plutôt que d'empêcher les commutations en attendant la convergence des fonctions de Lyapunov en un Temps de Tenue TT, nous proposons de minimiser ce TT. L'idée est d'utiliser un gain dynamique dans la loi de commande régissant la vitesse angulaire du robot. Ce gain est réévalué à chaque commutation (changement brusque de consigne). Sa valeur maximale est déterminée pour respecter les contraintes structurelles des entités robotiques. Ces contraintes imposent alors au TT une borne inférieure qu'on ne peut pas réduire. Si une commutation s'impose en un temps inférieur à cette borne, seul la stabilité faible reste alors garantie.

La structure virtuelle a été définie dans un repère absolu. Dans le chapitre 3, nous avons également donné des éléments de réponse permettant le passage à une définition de la structure virtuelle dans un repère plus adapté et relatif à la cible principale et sa direction.

Tout au long des chapitres 2 et 3, des résultats de simulation ont permis de valider les contributions théoriques apportées dans cette thèse. Enfin, nous avons clôturé ce manuscrit par un chapitre sur la plate-forme expérimentale développée ainsi que les expérimentations réalisées. Ces expérimentations nous ont permis de tester l'architecture de contrôle proposée et de valider sa stabilité sur des robots mobiles réels.

Perspectives

Ces travaux de thèse ont permis d'ouvrir le champ à des perspectives variées et étendues à l'image du thème de recherche abordé. Ces perspectives peuvent être classées en deux catégories principales : perspectives liées aux contributions

théoriques, et celles relatives aux expérimentations.

Perspectives théoriques

L'architecture de contrôle proposée est basée sur une approche comportementale lui assurant une ouverture afin d'accomplir de nouvelles tâches et une structure virtuelle favorisant une flexibilité du nombre de robots utilisés. Même si l'approche comportementale a permis de distribuer le contrôle sur les entités robotiques, il faudra recourir à une unité de calcul pour la définition de la forme et de la dynamique de la structure virtuelle. Utiliser une unité indépendante du SMR est commun dans les travaux basés sur cette approche. Néanmoins, ceci revient toujours à centraliser une partie du contrôle. C'est la méthode utilisée dans le chapitre expérimental pour réaliser les premières expérimentations. Pour garder l'aspect complètement distribué souhaité, il serait possible que cette tâche soit attribuée à l'un des robots mobiles du groupe, ce qui rendrait le SMR complètement indépendant d'un niveau central. Ceci rejoindrait alors l'approche hiérarchique où un robot est désigné comme un guide et qui semble indispensable pour une distribution complète du contrôle/commande. Il faudrait alors choisir entre une architecture de contrôle hybride distribuée/centralisée, où le niveau central sert uniquement à définir la structure virtuelle, et les entités robotiques doivent seulement suivre leurs cibles (et éviter des obstacles s'ils existent), et une architecture complètement distribuée avec l'un des robots comme guide prenant en charge la mission de l'unité centrale. Dans ce dernier cas, il y aura alors un guide et des suiveurs.

L'attribution des cibles basée sur la méthode du coût relatif proposée permet à chaque entité de négocier la cible la plus convenable avec les autres. Nous avons proposé que les coefficients de coût relatif soient calculés à intervalles de temps ΔT. Il serait nettement souhaitable de quantifier et surtout de justifier ce temps en fonction de la dynamique du système.

Le passage à un repère relatif par rapport à la cible principale pour la définition de la structure virtuelle est une étape importante. Dans ce cas, chaque cible secondaire aura une dynamique (vitesse linéaire et angulaire) qui lui est propre. Elle est en fonction de la dynamique de la cible principale, et de sa position relative à celle-ci désirée dans la formation. Il faudrait alors réétudier la dynamique autorisée de la cible principale (vitesses linéaire et angulaire), en prenant en considération que toutes les cibles secondaires doivent rester atteignables par leurs robots malgré les contraintes structurelles de ces derniers.

Ce passage à un repère relatif ouvre également la perspective de considérer un corps virtuel dynamique où les positions désirées des cibles secondaires changent

au cours du temps. Ce changement pourrait être motivé par la nature de l'environnement (par exemple, passer d'une formation en polygone à une formation en ligne pour traverser un passage étroit). Cette dynamique interne doit aussi considérer les contraintes structurelles des robots pour qu'ils puissent suivre leurs cibles.

Perspectives expérimentales

Le recours à la caméra centrale de la plate-forme comme élément de perception nous a permis de mener les premières expérimentations afin de tester l'architecture de contrôle proposée. Pour atteindre l'objectif d'une architecture de contrôle complètement distribuée, il est envisagé pour chaque entité d'utiliser exclusivement ses capteurs embarqués afin de se localiser et de percevoir l'environnement (obstacles, autres robots, etc.). Les capteurs ultrasons servirait à mesurer les distances des obstacles et des entités robotiques tandis que la caméra embarquée permettrait au robot d'avoir plus d'information sur l'environnement : taille des obstacles, nature de l'obstacle, position et orientation de la cible principale, etc.

Afin de distribuer complètement le contrôle, il faudrait s'inspirer de l'approche hiérarchique en attribuant la définition de la cible principale à l'un des robots plutôt que le superviseur central. Ce robot doit alors être identifiable pour les autres. Ces derniers doivent aussi connaître la formation géométrique souhaitée afin que chacun puisse trouver la position relative (cible secondaire) la plus convenable.

Annexes

Annexe A

Algorithme d'évitement d'obstacles utilisé

Cet annexe reprend l'algorithme d'évitement d'obstacles basé sur la méthode du cycle-limite [Adouane 09]. Pour accomplir la tâche d'évitement, il est impératif de définir trois étapes essentielles :

- détecter l'obstacle à éviter,
- choisir une direction d'évitement (trigonométrique ou anti-trigonométrique),
- définir un critère pour évaluer si l'obstacle est considéré comme étant évité.

Naturellement, ces étapes doivent prendre en considération les situations indésirables comme les minima locaux, d'inter blocage entre plusieurs obstacles, les oscillations dues aux multiples transitions entre le contrôleur d'attraction vers la cible et celui de l'évitement d'obstacles, etc.

Avant d'expliquer ces situations indésirables et les solutions apportées, on propose d'énumérer les points essentiels avant d'entamer l'algorithme d'évitement :

1. A chaque instant d'échantillonnage, obtenir les distances D_{RO_i} et D_{lRO_i} (cf. Section 2.3.2.1) (cf. Figure 2.14) pour tout obstacle gênant i (un obstacle est considéré gênant si $D_{lRO_i} \leq R_{I_i}$),
2. Parmi les obstacles gênants, choisir l'obstacle le plus proche (ayant la plus petite valeur D_{RO_i}). Il possède un rayon R_{O_i} et la position (x_{obst}, y_{obst}),
3. Définir quatre zones déterminant le comportement du robot (sens d'évitement, phase d'attraction vers l'obstacle, phase de répulsion) (cf. Figure A.1), (cf. Algorithme 5). Ces zones sont distinguées grâce à un repère orthonormé direct (O_i, X_{O_i}, Y_{O_i}) dont l'axe $(O_i X_{O_i})$ a le centre de l'obstacle O_i pour origine et passe par la cible du robot.

4. Passer de la position du robot $(x, y)_A$ donné initialement dans le repère absolu à la position $(x, y)_O$ dans le repère (O_i, X_{O_i}, Y_{O_i}) grâce à une matrice de transformation comme suit :

$$\begin{pmatrix} x \\ y \\ 0 \\ 1 \end{pmatrix}_O = \begin{bmatrix} \cos\alpha_i & -\sin\alpha_i & 0 & x_{obst} \\ \sin\alpha_i & \cos\alpha_i & 0 & y_{obst} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{pmatrix} x \\ y \\ 0 \\ 1 \end{pmatrix}_A \tag{A.1}$$

où α_i est l'angle que fait l'axe $(O_i X_{O_i})$ avec $(O_m X_m)$ (cf. Figure A.1).

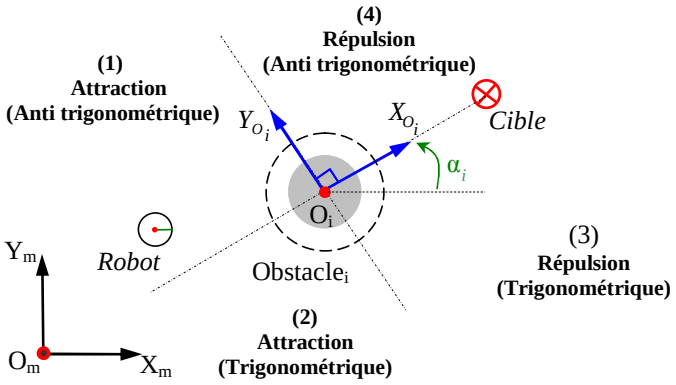


FIGURE A.1 – Zones entourant l'obstacle et déterminant la manière d'évitement de l'obstacle.

Le repère relatif à l'obstacle donne de nombreux outils d'aide à la décision au robot : direction d'évitement, phase d'attraction ou de répulsion, etc.

L'utilité de ces deux dernières phases est d'éviter des oscillations indésirables que l'on illustre ci-après (cf. Section A). Le signe de l'ordonnée du robot dans ce repère y_O permettra de savoir s'il faut éviter le robot dans le sens trigonométrique ou anti-trigonométrique :

- si $y_O \geq 0$ alors l'obstacle est évité dans le sens anti-trigonométrique,
- si $y_O < 0$, il est évité dans le sens trigonométrique.

Ceci permet alors d'optimiser la trajectoire du robot pour éviter l'obstacle et rejoindre plus rapidement sa cible. De même, grâce au signe de l'abscisse du robot x_O, il est possible de savoir s'il est en phase de répulsion ou d'attraction :

- $x_O \leq 0$ correspond à la phase d'attraction,
- $x_O > 0$ correspond à la phase de répulsion.

Ces étapes de prise de décision sont résumées dans l'algorithme 5.

Entrées: Caractéristiques de l'obstacle le plus proche : position (x_{obst}, y_{obst}), rayon R_{I_i}
Sorties: l'angle consigne $\theta_{S_{oa}}$

1: **Si** $x_O \leq 0$ **alors**
2: $R_c = R_{I_i} - \xi$ (phase d'attraction) ; {ξ est une constante telle que $\xi \ll Marge$ (cf. Section 2.3.2.1). Cette phase permet au robot de ne pas se trouver au voisinage de R_{I_i} ce qui risque de provoquer des oscillations (cf. Section A)}
3: **Sinon**
4: $R_c = R_c + \xi$ (phase de répulsion) ;{Critère de sortie de la zone de l'obstacle en gardant une trajecoitre lisse}
5: **finSi**
6: **Si** le contrôleur d'évitement d'obstacles est actif à l'instant $(t-1)$ **alors**
7: Appliquer le même sens d'évitement qu'à l'instant t (anti-trigonométrique (cf. Equation 2.28) ou trigonométrique (cf. Equation 2.29)) ;{voir le paragraphe A}
8: **Sinon**
9: Le système d'équation du cycle-limite :
$\dot{x}_s = signe(y_O)y_s + x_s(R_c^2 - x_s^2 - y_s^2)$
$\dot{y}_s = -signe(y_O)x_s + y_s(R_c^2 - x_s^2 - y_s^2)$
{avec $signe(y_O) = 1$ si $y_O \geq 0$ et $signe(y_O) = -1$ sinon.}
10: **finSi**
11: $\theta_{S_{oa}} = \arctan(\frac{\dot{y}_s}{\dot{x}_s})$

Algorithm 5: Algorithme d'évitement d'obstacles [Adouane 09].

Gestion des situations de conflit

La performance de l'algorithme 5 réside dans son aptitude à gérer certaines situations de conflits ou de comportements indésirables.

Risque d'oscillation au voisinage de R_{I_i} Les phases d'attraction et de répulsion décrites précédemment, dont les zones sont illustrées dans la figure A.1, et exprimées dans l'algorithme 5 par les lignes 2 et 4 servent à éviter ces oscillations entre $D_{ROi} \leq R_{Ii}$ (activation du contrôleur d'évitement d'obstacles) et $D_{ROi} \geq R_{Ii}$ (activation du contrôleur d'attraction vers la cible). En effet, la zone d'attraction permet d'entrer progressivement dans la zone de l'obstacle en décrémentant le rayon du cycle-limite de ξ à chaque itération, ce qui limite le risque de se trouver au voisinage de R_{Ii}. La zone de répulsion permet de préparer le robot à sortir de la zone de l'obstacle en gardant une trajectoire lisse et d'attraper sa

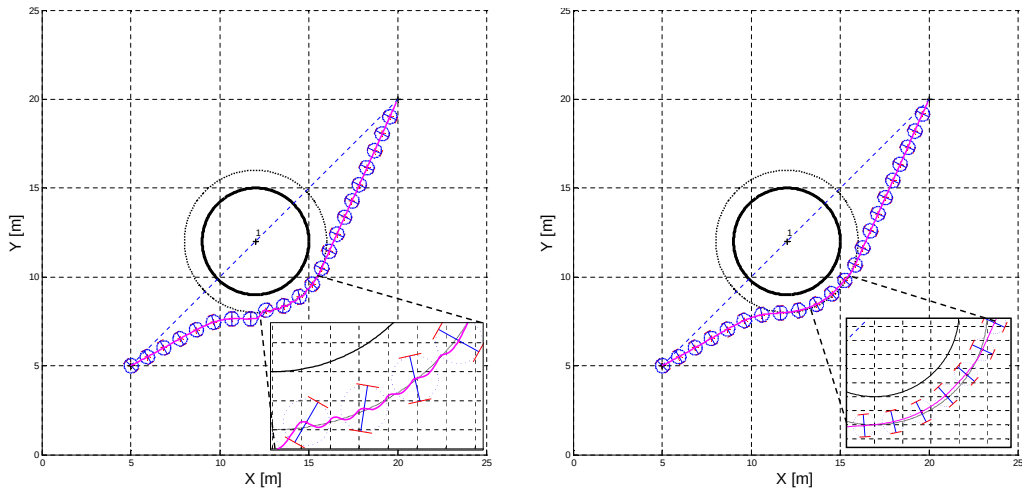
cible. A noter qu'afin d'éviter que le rayon du cycle-limite ne soit trop décrémentée, ce qui peut provoquer la collision, une limite peut être imposée telle que la ligne 4 de l'algorithme 5 n'est plus exécutée si

$$R_c \leq R_l$$

où R_l est un rayon limite à ne pas franchir.

Les oscillations causées par l'utilisation directe du rayon R_{I_i} comme rayon du cycle limite est illustrée dans la figure A.2(a). La figure A.2 montre que ces oscillations sont éliminées grâce aux phases d'attraction et de répulsion de l'algorithme 5.

Choix de l'obstacle à éviter S'il arrive qu'il y ait deux obstacles ou plus qui ont la même distance la plus proche du robot, alors il y a un conflit et le robot risque de ne pouvoir choisir l'obstacle à éviter. l'algorithme 6 permet de régler ce conflit.



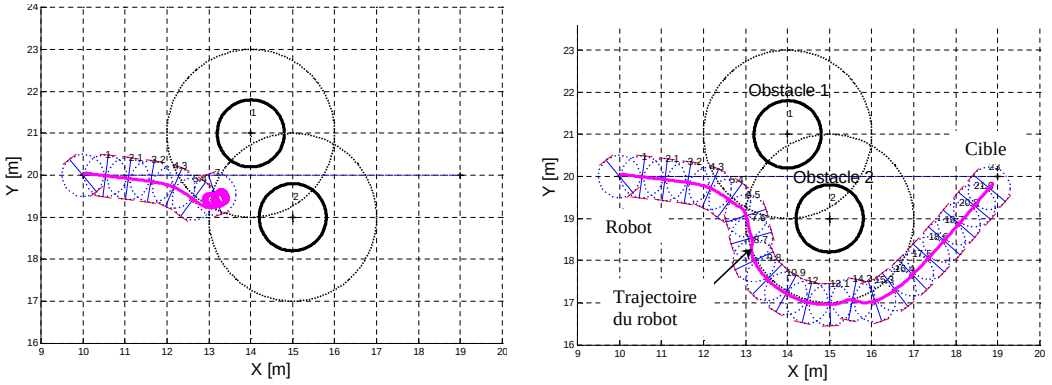
(a) Avec oscillations (b) Sans oscillations

FIGURE A.2 – Evitement des oscillations de la trajectoire du robot grâce à l'algorithme 5 [Adouane 09].

1: **Si** deux ou plusieurs obstacles ont la même distance D_{RO_i} **alors**
2: le robot choisira celui avec le plus petit D_{lRO_i} ;
3: **finSi**
4: **Si** ils ont le même D_{lRO_i} **alors**
5: le robot choisira celui avec le plus petit D_{CO_i} ;
6: **finSi**
7: **Si** ils ont le même D_{CO_i} **alors**
8: choisir l'un des obstacles arbitrairement ;
9: **finSi**

Algorithm 6: Choix de l'obstacle à éviter [Adouane 09].

Problèmes des minima locaux et des obstacles de type U (impasses) La ligne 7 de l'algorithme 5 permet de forcer la direction du cycle-limite afin d'éviter les minima locaux. En effet, un minimum local peut apparaître si le robot doit éviter deux obstacles qui se chevauchent. S'il éviter le premier dans le sens trigonométrique, et le deuxième dans le sens inverse en suivant uniquement les quatre zones définies par le repère relatif à l'obstacle (cf. Figure A.1), il se trouvera bloqué entre les deux comme illustré dans la figure A.3(a). Dans [Kim 03a], ce problème est géré en imaginant un seul obstacle virtuel englobant tous les obstacles chevauchés. Cependant, ceci suppose que lorsque le robot affronte le premier obstacle, il connaît déjà la totalité des obstacles chevauchés. De plus, plusieurs obstacles peuvent générer un obstacle virtuel avec un rayon important qui n'est pas toujours justifié. Vu toutes ces informations nécessaire à l'avance, la méthode risque de devenir moins réactive que l'on souhaite. Quant à l'algorithme 5, il se base sur une information mémoire minimale gardant uniquement le sens d'évitement à l'instant d'échantillonnage précédent. Grâce à cette information minimale, le robot peut aussi sortir des impasses imposées par les obstacles de type U (cf. Figure A.4).



(a) Sans imposer la direction, le robot tombe dans un minimum local

(b) En imposant la direction

FIGURE A.3 – Gestion du minimum local [Adouane 09].

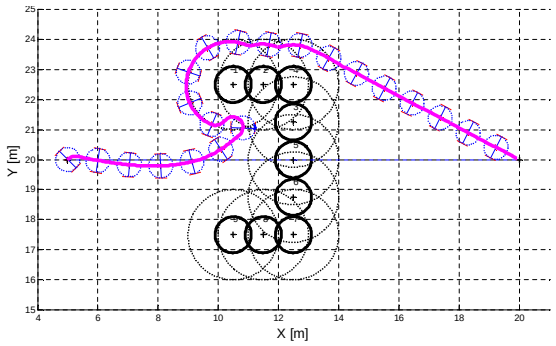


FIGURE A.4 – Gestion des impasses provoquées par les obstacles de forme U [Adouane 09].

Annexe B

Définitions de base de la stabilité

Pour introduire ces définitions, nous nous intéressons à un système dynamique représenté par une équation d'état

$$\dot{x} = f(x) \tag{B.1}$$

Où $x \in \mathbb{R}^n$ représente l'état du système. Nous supposons que $x = 0$ est son point d'équilibre.

Stabilité simple, asymptotique et exponentielle

Stabilité simple

Soit l'instant initial du système $t_0 = 0$. On dit que le point d'origine est stable si

$$\forall \epsilon > 0, \exists \delta > 0 : |x(0)| \leq \delta \Rightarrow |x(t)| \leq \epsilon \qquad \forall t \geq 0$$

Stabilité asymptotique

Le système est dit *asymptotiquement stable* s'il est stable et δ peut être choisie telle que :

$$|x(0)| \leq \delta \Rightarrow x(t) \overset{t \to \infty}{\to} 0.$$

Si cette condition est vraie pour tout δ, alors le système est globalement asymptotiquement stable.

Stabilité exponentielle

Le système est exponentiellement stable si

$$\exists \delta > 0, c > 0, \lambda > 0 : |x(0)| \leq \delta \Rightarrow |x(t)| \leq c\,|x(0)|\,e^{-\lambda t} \qquad \forall t \geq 0$$

Stabilité au sens de Lyapunov

Première méthode : méthode indirecte

La première méthode de Lyapunov est basée sur l'examen de la linéarisation du système $f(x)$ autour de l'état d'équilibre. Il s'agit d'examiner les valeurs propres $\lambda_i(J)$ de la matrice jacobienne évaluée à l'équilibre :

$$J = \frac{\partial f}{\partial x}(0) \tag{B.2}$$

Les propriétés de la stabilité du système s'expriment alors comme suit

Théorème 2. ***Première méthode de Lyapunov***

1. *Si toutes les valeurs propres de la matrice J ont une partie réelle strictement négative, le système est exponentiellement stable.*
2. *Si la matrice Jacobienne possède au moins une valeur propre de partie réelle strictement positive, alors le système est instable.*

Si le système a au moins une valeur propre à partie réelle nulle et aucune valeur propre à partie réelle strictement positive, nous ne pouvons conclure sur sa stabilité. Le système peut alors être étudié par la seconde méthode.

Deuxième méthode : méthode directe

Il s'agit d'une traduction mathématique d'une observation élémentaire : si l'énergie totale d'un système décroît (se dissipe) avec le temps de façon continue, alors ce système tend vers un état d'équilibre (il est donc stable). L'idée est alors de trouver une fonction définie positive temporelle qui admet une dérivée négative. La méthode directe peut être résumée dans le théorème suivant

Théorème 3. *L'état d'équilibre est stable s'il existe une fonction V continument dérivable et dont la dérivée est notée $\dot{V}$, telle que*

1. $V(0) = 0$,
2. $V(x) > 0 \quad \forall x \neq 0$,
3. $\dot{V}(x) \leq 0 \quad \forall x \neq 0$,

Si de plus (3) est remplacée par $\dot{V}(x) < 0$ *alors le système est asymptotiquement stable.*

Bibliographie

[Adams 99] M. Adams. *High Speed Target Pursuit and Asymptotic stability in Mobile Robotics.* IEEE Transactions on Robotics and Automation, vol. 15, pp. 230 – 237, 1999.

[Adouane 04] L. Adouane & N. LeFort-Piat. *Hybrid Behavioral Control Architecture for the Cooperation of Minimalist Mobile Robots.* International Conference on Robotics and Automation, pp. 3735–3740, 2004.

[Adouane 05] L. Adouane. *Architectures de Contrôle Comportementales et Réactives pour la Coopération d'un Groupe de Robots Mobiles.* Thèse de doctorat, Laboratoire d'Automatique de Besançon (UMR CNRS 5696), 2005.

[Adouane 09] L. Adouane. *Orbital Obstacle Avoidance Algorithm for Reliable and On-Line Mobile Robot Navigation.* 9th Conference on Autonomous Robot Systems and Competitions, May 2009.

[Alami 98a] R. Alami, R. Chatila, S. Fleury, M. Ghallab & F. Ingrand. *An architecture for autonomy.* International Journal of Robotics Research,Special Issue on Integrated Architectures for Robot Control and Programming, vol. 17(4), pp. 315–337, 1998.

[Alami 98b] R. Alami, S. Fleury, M. Herrb, F. Ingrand & F. Robert. *Multi-Robot Cooperation in the MARTHA Project.* IEEE Robotics and Automation Magazine, vol. 5, pp. 36–47, 1998.

[Anderson 90] T. Anderson & M. Donath. *Animal Behavior as a Paradigm for Developing Robot Autonomy.* Robotics and Autonomous Systems, vol. 6, pp. 145–168, 1990.

[Andrews 83] J. R. Andrews & N. Hogan. *ASME Winter Conference Boston.* Control of Manufacturing Processes and Robotic Systems, pp. 243–251, 1983.

[Antonelli 03] G. Antonelli & S. Chiaverini. *Kinematic Control of a Platoon of Autonomous Vehicles.* IEEE International Conference on Robotics and Automation, pp. 1464–1469, 2003.

[Antonelli 10] G. Antonelli, F. Arrichiello & S. Chiaverini. *The NSB control : a Behavior-based Approach for Multi-robot Systems.* PALADYN Journal of Behavioral Robotics, vol. 1, pp. 48–56, 2010.

[Arkin 86] R. C. Arkin. *Motor schema-based mobile robot navigation.* International Journal of Robotics Research, vol. 8(4), pp. 92–112, 1986.

[Arkin 89] R. C. Arkin. *Towards the Unification of Navigational Planning and Reactive Control.* In AAAI Spring Symposium on Robot Navigation, pp. 1–5, 1989.

[Arkin 98] C Arkin R. *Behavior-Based Robotics.* The MIT Press, 1998.

[Bahr 09] A. Bahr. *Cooperative Localization for Autonomous Underwater Vehicles.* Thèse de doctorat, Massachusetts Institute of Technology/Woods Hole Oceanographic Institution, 2009.

[Balch 99] T. Balch & R.C. Arkin. *Behavior-based Formation Control for Multi-robot Teams.* IEEE Transactions on Robotics and Autmation, 1999.

[Barnes 07] L. Barnes, M. A. Fields & K. Valavanis. *Unmanned Ground Vehicle Swarm Formation Control Using Potential Fields.* 15th Mediterranean Conference on Control and Automation, pp. 1–8, 2007.

[Baudoin 10] Léo Baudoin. *Perception pour l'évitement d'obstacles d'un robot mobile de type Khepera III.* Rapport technique, LASMEA, Université Blaise Pascal, 2010.

[Beard 01] R.W. Beard, J. Lawton & F. Y. Hadaegh. *A Coordination Architecture for Spacecraft Formation Control.* IEEE Transactions on Control Systems Technology, vol. 9, pp. 777–790, 2001.

[Belkhouche 06] F. Belkhouche, B. Belkhouche & P. Rastgoufard. *Line of Sight Robot Navigation Toward a Moving Goal.* IEEE Transactions on Systems, Man, and Cybernetics, Part B, vol. 36(2), pp. 255–267, 2006.

[Benzerrouk 08] A. Benzerrouk, L. Adouane, P. Martinet & N. Andreff. *Toward an hybrid architecture control for mobile robots.* Control Architecture of Robots, 2008.

[Benzerrouk 09] A. Benzerrouk, L. Adouane, P. Martinet & Andreff N. *Multi Lyapunov Function Theorem Applied to a Mobile Robot Tracking a Trajectory in Presence of Obstacles.* European Conference on Mobile Robots, 2009.

[Benzerrouk 10a] A. Benzerrouk, L. Adouane, L. Lequievre & P. Martinet. *Navigation of Multi-Robot Formation in Unstructured Environment Using Dynamical Virtual Structures.* IEEE/RSJ International Conference on Intelligent Robots and Systems, 2010.

[Benzerrouk 10b] A. Benzerrouk, L. Adouane & P. Martinet. *Lyapunov Global Stability for a Reactive Mobile Robot Navigation in Presence of Obstacles.* ICRA'10 International Workshop on Robotics and Intelligent Transportation System, 2010.

[Benzerrouk 10c] A. Benzerrouk, L. Adouane & P. Martinet. *Stabilite globale pour la navigation reactive d'un robot mobile en presence d'obstacles.* Sixieme Conference Internationale Francophone d'Automatique, 2010.

[Berg 06] J. V. D. Berg, D. Ferguson & J. Kuffner. *Anytime Path Planning and Replanning in Dynamic Environments.* IEEE International Conference on Robotics and Automation, pp. 2366 – 2371, 2006.

[Bishop 03] B.E. Bishop. *On the Use of Redundant Manipulator Techniques for Control of Platoons of Cooperating Robotic Vehicles.* IEEE Transactions on Systems, Man and Cybernetics, vol. 33, pp. 608–615, 2003.

[Bom 05] J. Bom, B. Thuilot, F. Marmoiton & P. Martinet. *Nonlinear control for urban vehicles platooning, relying upon a unique kinematic GPS.* International Conference on Robotics and Automation, pp. 4149–4154, 2005.

[Borenstein 89] J. Borenstein & Y. Koren. *Real-time Obstacle Avoidance for Fast Mobile Robots.* IEEE Transactions on Systems, Man, and Cybernetics, vol. 19, pp. 1179–1187, 1989.

[Bourgeot 04] J.M Bourgeot. *Contribution à la commande de systèmes mécaniques non-réguliers.* Thèse de doctorat, Institut National Polytechnique de Grenoble- France, 2004.

[Branicky 93] M. S. Branicky. *Stability of Switched and Hybrid Systems.* 33rd IEEE Conference on Decision Control, pp. 3498–3503, 1993.

[Brian 02] P.G. Brian & M. J. Mataric'. *Sold ! : Auction Methods for Multirobot Coordination.* IEEE Transactions on Robotics and Automation, vol. 18, pp. 758–768, 2002.

[Brogan 97] D.C. Brogan & J.K. Hodgins. *Group Behaviors for Systems with Significant Dynamics.* Autonomous Robots, vol. 4, pp. 137–153, 1997.

[Brogliato 97] B. Brogliato, S. Niculescu & Orhant. *On the control of finite dimensional mechanical systems with unilateral constraints.* IEEE Transactions on Automatic Control, vol. 42(2), pp. 200–215, 1997.

[Brooks 85] R. A. Brooks. *A robust layered control system for a mobile robot.* IEEE Journal of Robotics and Automation, vol. 2, pp. 14–23, 1985.

[Cariou 10] C. Cariou, R. Lenain, B. Thuilot & P. Martinet. *Autonomous maneuver of a farm vehicle with a trailed implement : motion planner and lateral-longitudinal controllers.* IEEE International Conference on Robotics and Automation, 2010.

[Clark 92] R. Clark, R. C. Arkin & A. Ram. *Learning Momentum : Online Performance Enhancement for Reactive Systems.* IEEE International Conference on Robotics and Automation, pp. 111–116, 1992.

[Collins 05] S. Collins, A. Ruina & M. Tedrake. *Efficient bipedal robots based on passive-dynamic walkers.* Science, vol. 307, pp. 1082–1085, 2005.

[Crichton 03] M. Crichton. La proie. Robert Laffont, 2003.

[Cullen 65] J.M. Cullen, E. Shaw & H.A. Baldwin. *Methods for Measuring the Three-Dimensional Structure of Fish Schools.* Animal Behavior, vol. 13, pp. 534–543, 1965.

[Dahl 08] T. S. Dahl, M. Mataric' & Sukhatme G. S. *Multi-robot task allocation through vacancy chain scheduling.* Robotics and Autonomous Systems, vol. 57, pp. 674–687, 2008.

[Das 02] A. K. Das, R. Fierro, V. Kumar, , J.P. Ostrowski, J. Spletzer & J.C. Taylor. *A Vision-Based Formation Control Framework.* IEEE Transactions on Robotics and Automation, vol. 18, pp. 813–825, 2002.

[Do 07] K. D. Do. *Formation Tracking Control of Unicycle-type Mobile Robots.* IEEE International Conference on Robotics and Automation, pp. 527–538, 2007.

[Egerstedt 99] M. Egerstedt, K. H. Johansson, J. Lygeros & S. Sastry. *Behavior Based Robotics Using Regularized Hybrid Automata.* Proceedings of the 38th IEEE Conference on Decision and Control, vol. 4, pp. 3400 – 3405, 1999.

[Egerstedt 00] M. Egerstedt, K. H. Johansson, J. Lygeros & S. Sastry. *Behavior Based Robotics Using Regularized Hybrid Automata.* Computer Science ISSN 0302-9743, vol. 1790, pp. 103–116, 2000.

[Estlin 01] T. Estlin, R. Volpe, I. Nesnas & F. Mutz Fisher. *Decision-Making in a Robotic Architecture for Autonomy.* 6th International Symposium on Artificial Intelligence, Robotics and Automation in Space, pp. 92152–7383, 2001.

[Fazenda 07] P. V. Fazenda & P. U. Lima. *Non-Holonomic Robot Formations with Obstacle Compliant Geometry.* 6th IFAC Symposium on Intelligent Autonomous Vehicles, 2007.

[Feddema 02] J.T Feddema, C. Lewis & D.A. Schoenwald. *Decentralized Control of Cooperative Robotic Vehicles : theory and application.* IEEE Transactions on Robotics and Automation, vol. 18, pp. 852–864, 2002.

[Fend 06] M. Fend, S. Bovet & F. Pfeifer. *On the influence of morphology of tactile sensors for behavior and control.* Robotics and Autonomous Systems, vol. 54, pp. 686–695, 2006.

[Fierro 01] R. Fierro, A. K. Das, V. Kumar & J.P. Ostrowski. *Hybrid Control of Formations of Robots.* International Conference on Robotics and Automation, vol. 1, pp. 157–162, 2001.

[Filippov 88] A.F. Filippov. Differential equations with discontinuous righthand side. Kluwer academic publisher, 1988.

[Firby 87] R. Firby. *An Investigation into Reactive Planning in Complex Domains.* 6th National Conference on Artificial intelligence, pp. 202–206, 1987.

[Fox 97] D. Fox, W. Burgard & S. Thrun. *The Dynamic Window Approach to Collision Avoidance*, 1997.

[Fraichard 99] T. Fraichard. *Trajectory planning in a dynamic workspace : a state time approach.* Advanced Robotics, vol. 13, pp. 75–94, 1999.

[Fukuda 89] T. Fukuda, S. Nakagawa, Y. Kawauchi & M. Buss. *Structure decision method for self organising robots based on cellstructures-CEBOT.* IEEE International Conference on Robotics and Automation, pp. 695–700, 1989.

[Gat 97] E. Gat, R.P. Bonasso, R. Murphy & A. Press. *On Three-Layer Architectures.* Artificial Intelligence and Mobile Robots, pp. 195–210, 1997.

[Ghommam 08] J. Ghommam, M. Saad & F. Mnif. *Formation Path Following Control of Unicycle-type Mobile Robots.* IEEE International Conference on Robotics and Automation, pp. 1966 – 1972, 2008.

[Gil Pinto 05] A. Gil Pinto, P. Fraisse & R. Zapata. *A Decentralized Adaptive Trajectory Planning Approach for a Group of Mobile Robots.* Towards Autonomous Robotic Systems, pp. 65–72, 2005.

[Goldberg 03] D. Goldberg, V. Cicirello, M. Dias, R. Simmons, S. Smith & A. Stentz. *Market-Based Multi-Robot Planning in a Distributed Layered Architecture.* Proceedings of the 2003 International Workshop on Multi-Robot Systems, pp. 27–38, 2003.

[Grassi Junior 06] V. Grassi Junior, S.P. Parikh & J. Okamoto Junior. *Hybrid Deliberative/Reactive Architecture for Human-Robot Interaction.* ABCM Symposium Series in Mechatronics, vol. 2, pp. 563–570, 2006.

[Gulec 05] N. Gulec & M. Unel. *A Novel Coordination Scheme Applied to Nonholonomic Mobile Robots.* IEEE Conference on Decision and Control, and European Control Conference., pp. 5089 – 5094, 2005.

[Guo 95] D. Guo & W. Rugh. *A stability result for linear parameter-varying systems.* Systems Control Lett., vol. 24, pp. 1–5, 1995.

[Gustavi 08] T. Gustavi & X. Hu. *Observer-Based Leader-Following Formation Control using Onboard Sensor Information.* IEEE Transactions on Robotics, vol. 24, pp. 1457–1462, 2008.

[Ha 05] H.M. Ha, A.D. Nguyen & Q.P. Ha. *Controlling formations of multiple mobile robots with inter-robot collision avoidance.* Australian Conference on Robotics and Automation, 2005.

[Harper 06] C.J. Harper & A.F.T. Winfield. *A methodology for provably stable behavior-based intelligent control.* Robotics and Autonomous Systems, vol. 54, pp. 52–73, 2006.

[Hespanha 99] J.P. Hespanha & A.S. Morse. *Stability of switched systems with average dwell-time.* Proceedings of the 38th IEEE

Conference on Decision and Control, vol. 3, pp. 2655–2660, 1999.

[Huntsberger 03] T. Huntsberger, P. Pirjanian, A. Trebi-ollennu, H. A. Nayar, A. J. Ganino, M. Garrett, S.S. Joshi & S. P. Schenker. *CAMPOUT : A Control Architecture for Tightly Coupled Coordination of Multi-Robot Systems for Planetary Surface Exploration.* IEEE Trans. Systems, Man & Cybernetics, Part A : Systems and Humans, vol. 33, pp. 550–559, 2003.

[Hussein 02] A.M. Hussein & A. Elnagar. *Motion planning using Maxwell's equations.* IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 3, pp. 2347 – 2352, 2002.

[Iida 05] F Iida, G. Gomez & G. Pfeifer. *Exploiting body dynamics for controlling a running quadruped robot.* 12th Int. Conf. on Advanced Robotics, ICAR05, 2005.

[Iocchi 01] L. Iocchi, D. Nardi & M. Salerno. Balancing reactivity and social deliberation in multi-agent systems. Springer Berlin / Heidelberg, 2001.

[Jeong 99] I. Jeong & J. Lee. *Evolving Fuzzy Logic Controllers for Multiple Mobile Robots Solving a Continuous Pursuit Problem.* IEEE International Conference on Fuzzy Systems, pp. 685–690, 1999.

[Jin 94] K. Jin, P. Liang & G. Beni. *Stability of synchronized distributed control of discrete swarm structures.* International Conference on Robotics and Automation, pp. 1033â€“1038, 1994.

[Johansson 99] K. H. Johansson, Egerstedt M., Lygeros J. & Sastry S. *On the regularization of Zeno hybrid automata.* Systems & Control Letters, vol. 38, pp. 141–150, 1999.

[Jones 01] H. L. Jones & M. Snyder. *Supervisory Control of Multiple Robots Based on Real-Time Strategy Game Interaction Paradigm.* International Conference on Systems, Man and Cybernetics, pp. 383–388, 2001.

[Kalman 60] R. E. Kalman. *A New Approach to Linear Filtering and Prediction Problems.* Trans. ASME, Journal of Basic Engineering, vol. 82, pp. 34–45, 1960.

[Karam 09] N. Karam. *Agrégation d'informations délocalisées pour la mise à jour dynamique d'une cartographie de l'environnement.* Thèse de doctorat, Université Blaise Pascal, 2009.

[Khatib 86] O. Khatib. *Real time obstacle avoidance for manipulators and mobile robots.* International Journal of Robotics Research, vol. 5, pp. 90–99, 1986.

[Khoshnevis 98] B. Khoshnevis & G. Bekey. *Centralized sensing and control of multiple mobile robots.* Computers & Industrial Engineering, vol. 35, pp. 503–506, 1998.

[Kim 03a] D. Kim & J. Kim. *A real-time limit-cycle navigation method for fast mobile robots and its application to robot soccer.* Robotics and Autonomous Systems, vol. 42, pp. 17–30, 2003.

[Kim 03b] J.O. Kim, C.J. Im, H. Shin, K. Y. Yi & H. G. Lee. *A new Task-Based Control Architecture for Personal Robots.* International Conference on Intelligent Robots and Systems, vol. 2, pp. 1481–1486, 2003.

[Ko 98] N. Y. Ko, R. Simmons, R. Ko & G. Simmons. *The Lane-Curvature Method for Local Obstacle Avoidance.* IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 3, pp. 1615–1621, 1998.

[Koren 91] Y. Koren & J. Borenstein. *Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation.* IEEE Conference on Robotics and Automation, pp. 1398–1404, 1991.

[Krogh 86] B. H. Krogh & C. E. Thorpe. *Integrated Path Planning and Dynamics Steering Control for Autonomous Vehicles.* IEEE International Conference on Robotics and Automation, pp. 1664–1669, 1986.

[Kube 97] C. Kube. *Collective Robotics : From Local Perception to Global Action.* Thèse de doctorat, University of Alberta Canada, 1997.

[Lalish 06] E. Lalish, K.A. Morgansen & T. Tsukamaki. *Formation Tracking Control using Virtual Structures and Deconfliction.* IEEE Conference on Decision and Control, 2006.

[Latombe 91] J-C. Latombe. Robot motion planning. Boston, MA : Kluwer academic publishers, 1991.

[Lébraly 10] P. Lébraly, C. Deymier, O. Ait-Aider, E. Royer & M. Dhome. *Flexible Extrinsic Calibration of Non-Overlapping Cameras Using a Planar Mirror : Application to Vision-Based Robotics.* IEEE International Conference on Intelligent Robots and Systems, pp. to appear, 2010.

[Léchevin 06] N. Léchevin, C. Rabbath & P. Sicard. *Trajectory tracking of leader-follower formations characterized by constant line-of-sight angles.* Automatica, vol. 42, pp. 2131–2141, 2006.

[Lee 00] S-O. Lee, Y-J. Cho, M. Hwang-Bo, B-J. You & S-R Oh. *A Stable Target-Tracking Control for Unicycle Mobile Robot.* IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1822–1827, 2000.

[Leonard 01] N. E. Leonard & E. Fiorelli. *Virtual Leaders, Artificial Potentials and Coordinated Control of Groups.* Proceedings of the 40th IEEE Conference on Decision and Control, vol. 3, 2001.

[Lerman 01] K. Lerman, A. Galstyan, A. Martinoli & A. Ijspeert. *A Macroscopic Analytical Model of Collaboration in Distributed Robotic Systems.* Artificial life, vol. 7, pp. 375–393, 2001.

[Lewis 97] A. Lewis & K-H. Tan. *High Precision Formation Control of Mobile Robots Using Virtual Structures.* Autonomous Robots, vol. 4, pp. 387–403, 1997.

[Li 05] X. Li, J. Xiao & Z. Cai. *Backstepping Based Multiple Mobile Robots Formation Control.* IEEE International Conference on Intelligent Robots and Systems, pp. 887 – 892, 2005.

[Liberzon 03] Daniel Liberzon. Switching in systems and control. Birkhauser, 2003.

[MacArthur 07] E.Z. MacArthur & C.D. Crane. *Compliant Formation Control of a Multi-Vehicle System.* International Symposium on Computational Intelligence in Robotics and Automation, pp. 479 – 484, 2007.

[Maes 89] P. Maes. *The Dynamics of Action Selection.* Proceedings of the 11th International Joint Conference on Artificial Intelligence, pp. 991–997, 1989.

[Martin 99] D. L. Martin, A.J. Cheyer & D.B. Moran. *The Open Agent Architecture : A Framework for Building Distributed Software Systems.* Applied Artificial Intelligence, vol. 13, pp. 91 – 128, 1999.

[Mastellone 07] S. Mastellone, D.M. Stipanovic & M.W. Spong. *Remote Formation Control and Collision Avoidance for Multi-Agent Nonholonomic Systems.* IEEE International Conference on Robotics and Automation, pp. 1062–1067, 2007.

[Mataric' 92a] M. J. Mataric'. *Designing Emergent Behaviors : From Local Interactions to Collective Intelligence.* International Conference on Simulation of Adaptive Behavior : From Animals to Animats 2, pp. 432–441, 1992.

[Mataric' 92b] M. J. Mataric'. *Minimizing Complexity in Controlling a Mobile Robot Population.* IEEE International Conference on Robotics and Automation, pp. 830–835, 1992.

[Mataric' 95a] M. J. Mataric'. *Issues and Approaches in Design of Collective Autonomous Agents.* Robotics and Autonomous Systems, vol. 16, pp. 321–331, 1995.

[Mataric' 95b] M. J. Mataric', M. Nilsson & K. Simsarian. *Cooperative multi-robot box pushing.* IEEE International Conference on Intelligent Robots and Systems, vol. 3, pp. 556–561, 1995.

[Mataric' 97] M. J. Mataric'. *Behavior-Based Control : Examples from Navigation, Learning, and Group Behavior.* Journal of Experimental and Theoretical Artificial Intelligence, vol. 9, pp. 323–336, 1997.

[Mechraoui 10] A. Mechraoui, J-M. Thiriet & Gentil S. *On-line Distributed Bayesian Decision and Diagnosis of Wireless Networked Mobile Robots.* 18th Mediterranean Conference on Control and Automation, 2010.

[Morse 96] A. S. Morse. *Supervisory control of families of linear set-point controllers, part 1 : Exact matching.* IEEE Transactions on Automatic Control, vol. 41, pp. 1413 – 1431, 1996.

[Nguyen 06] K. D. Nguyen D.B. Do. *Formation Control of Mobile Robots.* Inlernutional Journal of Computers. Communications & Control, vol. I, pp. 41–59, 2006.

[Noreils 93] F.R. Noreils. *Toward a robot architecture integrating cooperation between mobile robots : application to indoor environment.* International Journal of Robotics Research, vol. 12, pp. 79–98, 1993.

[Ogren 02] P. Ogren, E. Fiorelli & Leonard N. E. *Formations with a Mission : Stable Coordination of Vehicle Group Maneuvers.* 15th International Symposium on Mathematical Theory of Networks and Systems, 2002.

[Ogren 05] P. Ogren & N. E. Leonard. *A convergent Dynamic Window Approach to obstacle avoidance.* IEEE Transactions on Robotics, vol. 21, pp. 188–195, 2005.

[Paquier 03] W. Paquier & R. Chatila. *Learning New Representations and Goals for Autonomous Robots.* IEEE International Conference on Robotics and Automation, vol. /, pp. 803–808, 2003.

[Parker 94] L.E. Parker. *ALLIANCE : An Architecture for Fault-Tolerant, Cooperation Active Control of Heterogeneous Mobile Robots.* IEEE/RSJ International Conference on Intelligent Robots and Systems, 1994.

[Parker 96] L.E. Parker. *On The Design Of Behavior-Based Multi-Robot Teams.* Journal of Advanced Robotics, vol. 10, pp. 547–578, 1996.

[Parker 98] L.E. Parker. *ALLIANCE : An Architecture for Fault-Tolerant Multirobot Cooperation.* IEEE Transactions on Robotics and Automation, vol. 14, pp. 220–240, 1998.

[Parker 00] L.E. Parker. *Lifelong Adaptation in Heterogeneous Multi-robot Teams : Response to Continual Variation in Individual Robot Performance.* Autonomous Robots, vol. 8, pp. 239–267, 2000.

[Parra-Gonzalez 08] E.F. Parra-Gonzalez & J.G. Ramirez-Torres. *Cooperative Multi-robot Box-Pushing in a Cluttered Environment.* Electronics, Robotics and Automotive Mechanics Conference, pp. 514 – 519, 2008.

[Ranganathan 03] A. Ranganathan & S. Koenig. *A Reactive Robot Architecture with Planning On Demand.* IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1462–1468, 2003.

[Reif 99] J. H. Reif & H. Wang. *Social Potential Fields : A Distributed Behavioral Control for Autonomous Robots.* Robotics and Autonomous Systems, vol. 27, pp. 171–194, 1999.

[Rekleitis 01] I. Rekleitis, G. Dudek & E. Milios. *Multi-robot collaboration for robust exploration.* Annals of Mathematics and Artificial Intelligence, vol. 31, pp. 7–40, 2001.

[Ren 04] W. Ren & R.W. Beard. *Formation feedback control for multiple spacecraft via virtual structures.* IEEE Proceedings on Control Theory and Applications, vol. 151, pp. 357 – 368, 2004.

[Reynolds 87] C. Reynolds. *Flocks, herds and schools : A distributed behavioral model.* Computers Graphics, vol. 21, pp. 25–34, 1987.

[Schulz 01] D. Schulz, W. Burgard, D. Fox & A. B. Cremers. *Tracking Multiple Moving Objects with a Mobile Robot.* IEEE Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 371–377, 2001.

[Shorten 05] R. Shorten, F. Wirth, O. Mason, K. Wulff & C. King. *Stability Criteria for Switched and Hybrid Systems.* SIAM Review, vol. 49, pp. 545–592, 2005.

[Siegwart 04] R. Siegwart & Nourbakhsh R. I. Introduction to autonomous mobile robots. The MIT Press, Massachusetts Institute of Technology Cambridge, 2004.

[Simmons 96] R. Simmons. *The Curvature-Velocity Method for Local Obstacle Avoidance.* IEEE International Conference on Robotics and Automation, pp. 3375–3382, 1996.

[Skjetne 02] R. Skjetne, S. Moi & T. Fossen. *Nonlinear formation control of marine craft.* in Proceedings of the 41st Conference on Decision and Control, 2002.

[Slack 93] M.G. Slack. *Navigation Template : Mediating Qualitative Guidance and Quantitative Control in Mobile Robots.* IEEE Transactions on Systems, Man and Cybernetics, vol. 23, no. 2, pp. 452–466, 1993.

[Spletzer 01] J. Spletzer, A. K. Das, R. Fierro, V. Taylor J.C. Kumar & J.P. Ostrowski. *Cooperative Localization and Control for Multi-Robot Manipulation.* IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 24, pp. 1457 – 1462, 2001.

[Stilwell 05] D.J. Stilwell, B.E. Bishop & C.A. Sylvester. *Redundant Maniplulator Techniques for Partially Decentralized Path Planning and Control of a Platoon of Autonomous Vehicles.* IEEE Transactions on Systems, Man and Cybernetics, vol. 35, pp. 842–848, 2005.

[Stonier 98] R. Stonier & M. Mohammadian. *Knowledge Acquisition for Target Capture.* IEEE International Conference on Evolutionary Computation, pp. 721–726, 1998.

[Stuart 96] A. Stuart & A. Humphries. *Dynamical Systems and Numerical Analysis.* Cambridge University Press, 1996.

[Sty 01] K. Sty. *Using situated communication in distributed autonomous mobile robots.* 7th Scandinavian Conference on Artificial Intelligence, 2001.

[Sycara 96] K. Sycara, K. Decker, A. Pannu & M. Williamson. *Distributed Intelligent Agents.* IEEE Intelligent Systems, vol. 11, pp. 36–46, 1996.

[Tanner 01] H.G. Tanner, S. Loizou & K.J. Kyriakopoulos. *Nonholonomic stabilization with collision avoidance for mobile robots.* 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 3, pp. 1220 – 1225, 2001.

[Tanner 04] H.G. Tanner, G.J Pappas & V. Kumar. *Leader-to-formation stability.* IEEE Transactions on Robotics and Automation, vol. 20(3), pp. 433–455, 2004.

[Tong 98] W. Tong & T. Li. *Realization of Two-Dimensional Target Tracking Problem via Autonomous Mobile Robot Using Fuzzy Sliding Mode Control.* IEEE International Conference on Industrial Electronics, pp. 1158–1163, 1998.

[Tu 94] X. Tu & D. Terzopoulos. *Artificial Fishes : Physics, Locomotion , Perception, Behavior.* SIGGRAPH Conference Proceedings, pp. 43–50, 1994.

[Ulrich 98] I. Ulrich & J. Borenstein. *VFH+ : Reliable Obstacle Avoidance for Fast Mobile Robots.* IEEE International Conference on Robotics and Automation, pp. 1572–1577, 1998.

[Urcola 08] P. Urcola, L. Riazuelo, M.T. Lazaro & L. Montano. *Cooperative navigation using environment compliant robot formations.* IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2789 – 2794, 2008.

[Van den Berg 05] J. Van den Berg & M. Overmars. *Roadmap-based motion planning in dynamic Environments.* IEEE Transactions on Robotics, vol. 21, pp. 885–897, 2005.

[Vehrencamp 79] S.L. Vehrencamp. *The roles of individuals, kin and group selection in the evolution of sociality.* P. Marler and J. G. Vandenbergh, Editors, Handbook of Behavioral Neurobiology, vol. 3, pp. 351–394, 1979.

[Yamada 01] S. Yamada & J. Saito. *Adaptive Action Selection without Explicit Communication for Multi-robot Box-pushing.* IEEE Transactions on Systems, Mand and Cybernetics, vol. 31, no. 3, pp. 398–404, 2001.

[Yamaguchi 01] H. Yamaguchi, T. Arai & G. Beni. *A Distributed Control Scheme for Multiple Robotic Vehicles to Make Group Formations.* Robotics and Autonomous Systems, vol. 36, pp. 125–147, 2001.

[Yamauchi 04] B. Yamauchi. *PackBot : a versatile platform for military robotics*. Unmanned Ground Vehicle Technology VI. Edited by Gerhart, Grant R.; Shoemaker, Chuck M.; Gage, Douglas W. Proceedings of the SPIE, vol. 5422, pp. 228–237, 2004.

[Yang 03] Y. Yang, O. Brock & R.A. Grupen. *Exploiting Redundancy to Implement Multi-objective Behavior*. IEEE International Conference on Robotics and Automation, pp. 3385–3390, 2003.

[Zadeh 75] L. A. Zadeh. *The concept of a linguistic variable and its application to approximate reasoning-1*. Information sciences, vol. 8, pp. 199–249, 1975.

[Zapata 94] R. Zapata & P. Lepinay. *Reactive behaviors of fast mobile robots*. Journal of Robotics Systems, vol. 11, pp. 13–20, 1994.

[Zapata 99] R. Zapata & P. Lepinay. *Flying among obstacles*. Third European Workshop on Advanced Mobile Robots, pp. 81 – 88, 1999.

[Zheng 93] J. Zheng. *General lemmas for stability analysis of linear continuous-time systems with slowly time-varying parameters*. International Journal of Control, vol. 58, pp. 1437–1444, 1993.

[Zhiqiang 06] C. Zhiqiang, M. Tan, L. Li, N. Gu & S. Wang. *Cooperative hunting by distributed mobile robots based on local interaction*. IEEE transactions on robotics, vol. 22, pp. 403–407, 2006.

[Zhu 09] S. Zhu & Q. Wang D. Chen. *Standoff Tracking Control of Moving Target in Unknown Wind*. Proceedings of the 48th IEEE Conference on Decision and Control, pp. 776–781, 2009.