| | |
|---|---|
| **Titre:** Title: | Contrôle décentralisé pour des systèmes multi-robots coopératifs |
| **Auteur:** Author: | Philippe Lacroix |
| **Date:** | 1998 |
| **Type:** | Mémoire ou thèse / Dissertation or Thesis |
| **Référence:** Citation: | Lacroix, P. (1998). Contrôle décentralisé pour des systèmes multi-robots coopératifs [Master's thesis, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/6903/ |

UNIVERSITÉ DE MONTRÉAL

# CONTRÔLE DÉCENTRALISÉ POUR DES SYSTÈMES MULTI-ROBOTS COOPÉRATIFS

PHILIPPE LACROIX
DÉPARTEMENT DE GÉNIE ÉLECTRIQUE ET DE GÉNIE INFORMATIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES (M.Sc.A.)
(GÉNIE ÉLECTRIQUE)
Avril 1998

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

# CONTRÔLE DÉCENTRALISÉ POUR DES SYSTÈMES MULTI-ROBOTS COOPÉRATIFS

présenté par: LACROIX Philippe
en vue de l'obtention du diplôme de: Maîtrise ès science appliquées
a été dûment accepté par le jury d'examen constitué de:

M. DE SANTIS Romano, Ph.D., président
M. COHEN Paul, Ph.D., membre et directeur de recherche
M. POLOTSKI Vladimir, Ph.D., membre et codirecteur de recherche
M. HURTEAU Richard, D.Ing., membre

À Agraféna Alexandrovna

# Remerciements

Je souhaite premièrement remercier mon directeur Paul Cohen qui m'a guidé et motivé tout au cours de ce projet de recherche et mon codirecteur Vladimir Polotski, pour son aide et ses judicieux conseils. Je tiens aussi à remercier Noranda et le CRSNG pour leur soutien financier.

Je remercie également les autres membres du Groupe de Recherche en Perception et Robotique de l'École Polytechnique de Montréal, étudiants, assistants de recherche et professeurs pour leur soutien tout au long de ma maîtrise, tout particulièrement Jean-Yves Hervé pour ses critiques constructives.

# Résumé

Ce projet s'inscrit dans le cadre des recherches sur les interactions entre robots mobiles autonomes. Nous étudions le problème de la coordination des stratégies de contrôle au sein d'une équipe de robots pour l'accomplissement d'une tâche commune. Nous nous intéressons plus particulièrement au développement d'une approche pour le design de systèmes multi-robots décentralisés accomplissant, de manière coopérative, une tâche ayant la particularité de nécessiter un niveau constant d'interaction de la part des robots.

L'approche proposée a comme avantage de ne nécessiter aucune forme de communication explicite ou implicite à l'intérieur du groupe de robots pour que ceux-ci puissent interagir entre eux afin d'accomplir efficacement une tâche. On propose de définir la coordination entre les robots de manière rigide en étendant la spécification de la tâche fournie à chaque robot. Cette spécification de la tâche est faite en décrivant l'évolution de la tâche dans l'espace des configurations au moyen d'un champ de vecteurs qui propose, pour chaque configuration de la tâche, un déplacement instantané pour la tâche. À partir de cette nouvelle manière de spécifier la

tâche, chaque membre d'une équipe peut prédire localement l'évolution de la tâche et utiliser cette information de prédiction pour gouverner ses interventions sur la tâche.

Notre approche repose sur une stratégie de contrôle décentralisé à deux niveaux. Au niveau supérieur, un contrôleur donne l'évolution de la tâche dans l'espace et le temps afin de garantir la qualité des comportements coopératifs du groupe de robots. Au niveau inférieur, un contrôleur détermine les actions individuelles des robots afin que ceux-ci suivent les requêtes du niveau supérieur.

Une application de transport coopératif a été plus profondément étudiée pour illustrer l'approche proposée. La tâche consiste à déplacer une poutre d'une configuration initiale vers une configuration finale en la soulevant par ses extrémités à l'aide de deux robots mobiles. Les robots, de type voiture, sont munis d'un levier à l'avant et d'une caméra pointant vers la poutre.

Le contrôleur décentralisé de chaque robot cherche à accomplir la tâche en ne faisant usage que d'informations visuelles provenant de l'observation de la poutre et de l'information de la position relative à la configuration finale souhaitée pour la poutre. Comme pour l'approche générale, le contrôleur est divisé en deux niveaux de contrôle. Au niveau supérieur, la spécification de la tâche donne au robot les vitesses longitudinales et angulaires désirées pour la poutre à chaque instant. La poutre étant modélisée comme un uni cycle, ce niveau correspond au contrôle d'un système non holonomique dans l'espace cartésien. Au niveau inférieur, un contrôleur simple, non linéaire, cherche à atteindre les caractéristiques dynamiques désirées pour la poutre.

Une méthode pour éviter les obstacles imprévus le long du chemin de la poutre sans faire usage de communication fut inclue. Cette méthode repose sur une particularité du haut niveau de contrôle de notre approche. Elle consiste pour un robot à modifier temporairement le déplacement désiré pour la poutre et ainsi à délibérément perturber le système afin de faire dévier la poutre de sa route et ainsi éviter les éventuels obstacles.

Des éléments d'expérimentation pour cette application spécifique de transport d'une poutre ont été implantés. Deux méthodes pour faire l'extraction visuelle des informations de position et d'orientation de la poutre furent développées. Un banc d'essai expérimental fut créé, entre autres pour les besoins de ce projet d'études sur les interactions entre robots. Des expérimentations ont su démontrer les possibilités de l'utilisation du feedback visuel et du contrôle de bas niveau pour coordonner des robots mobiles lors de l'exécution d'une tâche simple.

# Abstract

This work is related to research on interactions between autonomous mobile robots. We have studied the coordination of control strategies in a team of robots accomplishing a common task. We are concerned with the development of an approach for the design of decentralized multi-robots systems which must accomplish, in a cooperative way, the tasks that demand constant interactions between the robots.

The proposed approach has the advantage of not requiring any use of explicit communication between robots in order to efficiently accomplish a given task. In order to ensure the coordination between robot we have proposed to extend the task specification given to each robot. This extension of the task specification takes the form of a description of the task evolution in the configuration space. With the use of a vector field, a suggested instantaneous motion is specified for all possible task configurations. With this new way for specifying the task, every member of the team can predict locally the task evolution and use this prediction to guide its intervention over the task.

Our approach is based on a two level decentralized control strategy. The high

level of control concerns the ways the task will evolve in time and space in order to guarantee the quality of cooperation. The low level of control determine the robot's individual actions in order for the robots to follow the highest level requests.

A cooperative transport application has been studied deeply to illustrate the proposed approach. For this case the task consists of moving a beam from an initial location to a specified final configuration. Two robots involved in this transportation task are four-wheeled front-steering vehicles equipped with frontal lift and a camera pointing towards the beam. In order to execute the task, the two robots must lift the beam and move it in close coordination.

The decentralized controller of each robot tries to accomplish the task by using visual information given by the beam observation and the position information relative to the final position of the beam. As in our general approach, the control is divided in two levels. For the highest level, the task specification gives the robot the beam desired longitudinal and angular velocities. The beam was modeled as a unicycle, thus this level corresponds to the control of a nonholonomic system in Cartesian space. For the lowest level, a simple non linear controller is proposed to ensure the desired dynamic characteristics for the motion of the beam.

A method for obstacle avoidance along the beam's path that make no use of communication has been analized. This method is based on a particularity of the high level controller we have proposed. In order to avoid an obstacle obstructing the path, a given robot must temporarily modify the desired beam motion. Thus the

robot can deliberately disturb the system, hereby enforcing it to change its path and therefore avoid the obstacle.

We have partially implemented the control strategy, developed for this specific beam transport application. Two methods for visual information extraction were tested. An experimental test bed was built for the group of projects related to the study on robot's interactions, among which our project is a part. Experiments have shown the possibilities to use visual feedback for control and the reliability of the low level controllers for the coordination of mobile robots when they are accomplishing a common task.

# Table des matières

# Liste des figures

# Liste des tableaux

# Liste des annexes

# Liste des notations

- $G$ : l'espace d'état du système ;
- $x_1, \ldots, x_n$ : les coordonnées généralisées du système ;
- $\tau_G$ : une trajectoire dans l'espace d'état du système ;
- $T$ : l'espace de la tâche ;
- $\vec{t} = (t_1, \ldots, t_m)$ : le vecteur d'état de la tâche ;
- $t$ : l'état de la tâche ;
- $X(t)$ : le champ de vecteurs définit sur l'espace d'état de la tâche ;
- $\tau_t$ : une trajectoire dans l'espace d'état de la tâche ;
- $r_i$ : un robot mobile ;
- $R_i$ : l'espace d'état d'un robot ;
- $(x_i, y_i)$ : les coordonnées cartésiennes du point situé au centre de l'essieu arrière d'un robot ;

- $\varphi_i$ : l'orientation d'une voiture par rapport à l'orientation finale de la poutre ;
- $v_i$ : la vitesse de la voiture ;
- $\delta_i$ : l'angle de braquage d'une voiture ;
- $L_i$ : la distance entre l'essieu avant et arrière d'une voiture ;
- $\omega_i$ : la vitesse de rotation instantanée d'une voiture ;
- $(X_0, Y_0)$ : les coordonnées cartésiennes du centre de la poutre par rapport aux coordonnées cartésiennes désirées ;
- $\varphi_0$ : l'orientation de la poutre par rapport à l'orientation finale désirée ;
- $V_t$ : la vitesse transversale de la poutre ;
- $V_l$ : la vitesse longitudinale de la poutre ;
- $\omega_0$ : la vitesse angulaire de la poutre ;
- $D_i$ : la distance le long de la poutre entre le levier d'un robot et du centre de la poutre ;
- $\psi_i$ : l'angle entre l'orientation d'une voiture et l'orientation de la poutre ;
- $e_0$ : la distance d'erreur du centre de la poutre au but ;
- $\alpha$ : la direction du but par rapport à la direction du déplacement transversal du centre de la poutre ;

- $\theta$ : l'orientation du but par rapport au véhicule dans le référentiel du but ;
- $\beta$ : l'angle entre le vecteur vitesse du centre de la poutre et sa direction transversale ;
- $\rho$ : la direction du but par rapport à la direction du déplacement du centre de la poutre.

# Chapitre 1

# Introduction

Les systèmes multi-robots ont reçu beaucoup d'attention dans la communauté scientifique ces dernières années (Cao, Fukunaga, Kahng et Mang, 1995). Lorsque plusieurs robots travaillent en collaboration pour accomplir une même tâche, on dit qu'ils forment un système multi-robots de type coopératif (Barnes et Gray, 1991). Les systèmes multi-robots coopératifs sont souvent caractérisés par l'utilisation d'ensemble de robots simples mécaniquement et dotés de fonctionnalités complémentaires, alors que l'accomplissement de la même tâche par un système mono-robot nécessiterait un robot substantiellement plus complexe. L'utilisation de systèmes multi-robots travaillant en coopération pour accomplir une tâche suggère plusieurs avantages, comme indiqué au tableau 1.1.

Les systèmes multi-robots ont entre autres l'avantage d'avoir des caractéristiques d'accessibilité spatiale ou temporelle supérieures à celles des systèmes mono-robots.

Tableau 1.1 : Avantages et inconvénients des systèmes multi-robots

| | Systèmes multi-robots | Systèmes mono-robot |
|---|---|---|
| Accessibilité spatio/temporelle | supérieure | inférieure |
| Simplicité mécanique | supérieure | inférieure |
| Évolutivité | supérieure | inférieure |
| Simplicité de contrôle | inférieure | supérieure |

Il leur est ainsi possible d'accomplir des tâches ne pouvant pas être exécutées d'une manière séquentielle par un robot unique. Il existe, en effet des tâches dites "fortement coopératives" (Brown et Jennings, 1995), dans lesquelles plusieurs robots doivent agir ensemble et en parallèle. Prenons l'exemple d'une équipe de robots qui aurait la tâche de combattre des feux de forêt. Pour une telle tâche, les limitations spatiales du robot travaillant seul le rendent inadéquat.

De plus, la faible complexité mécanique des robots utilisés dans des systèmes multi-robots, rend ces derniers plus robustes mécaniquement et moins coûteux à produire. Enfin, les ressources d'une équipe de petits robots simples peuvent être gérées d'une manière plus efficace. Il en va de même pour les capacités d'amélioration du système où l'augmentation des capacités d'une équipe pourrait se faire par le simple ajout de robots à l'intérieur de l'équipe.

Il y a, par contre, des difficultés reliées à l'utilisation d'équipes de robots. En effet, pour un robot unique, tous les outils utilisés sont liés physiquement et les informations perçues par les senseurs sont gérées par un système de contrôle centralisé. Dans une équipe de robots, les différents senseurs et actionneurs étant séparés physiquement,

le contrôle du système est plus délicat surtout lorsque les robots doivent *interagir* entre eux, c'est-à-dire lorsque leurs actions imposent des contraintes restrictives sur les actions des autres robots.

On distingue deux principaux types d'organisation pour le contrôle d'équipes de robots (Noreils, 1993) : les architectures centralisées et les architectures décentralisées (voir figure 1.1).

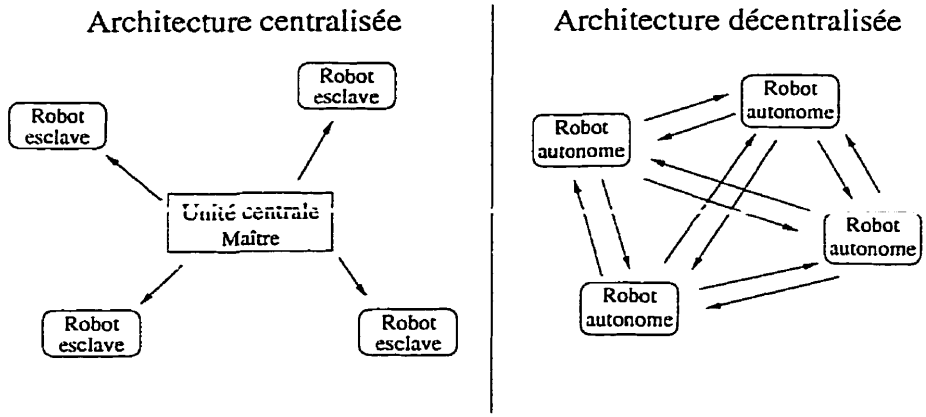



Figure 1.1 : Architectures centralisées et décentralisées

Dans les architectures centralisées, la prise de décision est faite par une entité unique se basant sur des informations globales (Parker, 1993). Dans ce cas, les robots du système sont perçus comme les actionneurs d'un même robot. Dans les architectures décentralisées, la prise de décision est répartie sur plusieurs robots. Dans ce cas, chaque robot utilise uniquement les informations locales perçues par ses senseurs, et n'a pas une connaissance complète de l'état global du système. Ces deux architectures sont des concepts d'architecture extrêmes et il existe une vaste gamme d'architectures hybrides, utilisant un mélange de ces deux types.

Dans les systèmes centralisés, la principale limitation réside dans la communication entre robots. En effet, dans de tels systèmes, le contrôle de chaque robot nécessite d'obtenir les informations provenant de ses senseurs, de traiter ces informations pour choisir les actions à initier, et d'envoyer au robot les commandes nécessaires à son contrôle. Ces informations doivent être transmises au robot via un canal dont la capacité peut affecter les performances du système. En effet, le temps nécessaire à la transmission de l'information affecte le temps de réponse des robots et le succès de la tâche. De plus, l'information peut être altérée ou détruite par la transmission en raison de perturbations. En général, on cherche à limiter le flot de communication entre robots par l'utilisation de protocoles (Mataric, Nilsson et Simsarian, 1995; Wang, 1994), par une structure de communication appropriée (Chen et Luth, 1993; Hashimoto, Oba et Eguchi, 1993), ou même en estimant l'état ou les intentions des autres robots en faisant usage de modèles probabilistiques (Fukuda et Sekiyama, 1994). Enfin, le contrôle de tous ces robots par un système central devient rapidement complexe et exigeant lorsque le nombre de robots contrôlés augmente.

Pour les systèmes décentralisés, la difficulté majeure provient du fait que les robots n'ont pas une connaissance de toutes les informations sur l'état globale du système. Chaque unité ne peut donc pas connaître, en tout temps, l'état des autres robots de même que leurs intentions. Il est par conséquent difficile pour un robot de choisir ses actions en accord avec celles du reste de l'équipe afin de faire réussir la tâche.

L'idée centrale de ce travail est donc de développer une architecture utilisant

principalement des informations locales pour faire face aux changements dynamiques de l'environnement, tout en utilisant des informations globales pour bien coordonner les actions des robots entre eux. Plusieurs de ces architectures mixtes ont été développées utilisant soit des structures hiérarchiques (Wang, Nakano et Matsukawa, 1994; Taipale et Hirai, 1993; Noreils, 1993; Ota, Miyata, Arai, Yoshida, Kurabayashi et Sasaki, 1995), des structures distribuées faisant usage d'une forme de négociation entre robots (Ishida, Asama, Tomita, Ozaki, Matsumoto et Endo, 1994; Ozaki, Asama, Ishida, Matsumoto, Yokota, Kaetsu et Endo, 1993), ou inférant les actions des autres unités par vision (Kuniyoshi, Riekki, Ishii, Rougeaux, Kita, Sakane et Kakikura, 1994; Arkin, 1992) ou par d'autres techniques (Genesereth, Ginsberg et Rosenschein, 1986; Parker, 1995; Mataric, 1994; Fukuda et Sekiyama, 1994). Les inconvénients de ces méthodes résident dans le fait que les robots doivent soit communiquer de façon explicite, avec les problèmes afférents décrits précédemment, ou tenter d'inférer leurs intentions par des méthodes souvent peu robustes.

Notre approche, que nous présenterons au chapitre 2, a l'avantage de ne nécessiter aucune forme de communication implicite ou explicite entre les robots pour que ceux-ci accomplissent efficacement une tâche de manière coopérative. Dans cette approche, la coordination entre les robots n'est pas laissée au robot, mais est définie par des règles précise dans la spécification de la tâche.

Nous définirons en premier un espace d'état de la tâche de même que des espaces d'état pour chaque robot, tous étant des espaces éléments de l'espace d'état du sys-

tème. L'idée générale de l'approche est de spécifier la tâche à chaque membre de l'équipe comme étant, pour un état donné, le changement désiré dans l'espace d'état de la tâche. De cette façon, les robots ont pour chaque configuration du système une idée exacte de la manière dont le système doit évoluer. Cette information globale sur l'évolution de la tâche permet indirectement de centraliser la coordination entre les robots. Le système n'en est pas moins décentralisé au niveau des robots, ceux-ci se choisissant leurs propres actions en se basant sur la spécification de la tâche et sur l'idée qu'ils ont de l'état actuel de la tâche.

Dans le chapitre 3 nous présentons une analyse détaillée d'une application de notre approche consistant à faire déplacer une poutre par deux robots mobiles. Les robots sont des véhicules de type automobile, munis d'un levier à l'avant. Ils soulèveront la poutre pour la déplacer d'une configuration initiale à une configuration finale (voir figure 1.2). Ils doivent mener à bien la tâche en ne faisant usage que d'informations visuelles provenant de l'observation de la poutre et de l'information que chaque robot a de sa propre position relative par rapport à la configuration finale souhaitée pour la poutre.

Nous présenterons ensuite la modélisation des voitures et de la poutre. Nous discuterons aussi des problèmes de modélisation reliés au frottement de la poutre sur les leviers, et de l'algorithme retenu afin de modéliser en simulation ce phénomène de friction.

Le chapitre 4 présentera le contrôleur décentralisé utilisé pour cette application.

Figure 1.2 : Application du déplacement d'une poutre par deux robots mobiles

Notre contrôleur est divisé en deux niveaux de contrôle. Le niveau supérieur correspond à la spécification de la tâche, donnant au robot les vitesses longitudinales et angulaires désirées pour la poutre à chaque instant. En modélisant la poutre comme un uni-cycle, ce niveau correspondra à un contrôle d'un véhicule non holonomique dans l'espace cartésien. Le niveau inférieur consiste en un contrôleur simple, non linéaire, cherchant à atteindre les caractéristiques dynamiques désirées pour la poutre. À la figure 1.3, le diagramme bloc du système complet est présenté.

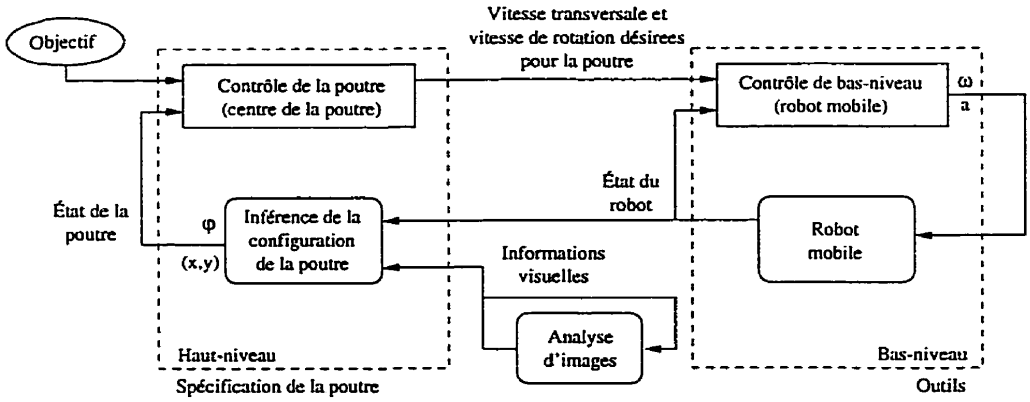



Figure 1.3 : Diagramme bloc de l'application de transport d'une poutre

Au chapitre 5, nous présenterons un ajout qui permettra au système d'éviter les obstacles sans nécessiter de communication explicite entre les robots. En modifiant le déplacement désiré pour la poutre (calculé au niveau supérieur), nous montrerons qu'un robot peut délibérément perturber le système afin de faire dévier la poutre de sa route et ainsi éviter les éventuels obstacles.

Nous présenterons enfin les éléments d'expérimentation de cette application spécifique de transport d'une poutre. Nous commencerons par décrire au chapitre 6 les méthodes employées pour extraire en temps réel les infomations nécessaires au contrôle en observant la poutre au moyen d'une caméra montée sur chaque voiture. Enfin, le chapitre 7 présentera une brève description du banc d'essai expérimental utilisé. Ce dernier comporte deux voitures radioguidées modifiées munies d'un levier à l'avant. Quelques résultats expérimentaux pour notre application de contrôle seront enfin présentés.

À l'annexe A nous présenterons un aperçu des divers éléments du problème de coopération dans des systèmes multi-robots. Nous décrirons les diverses caractéristiques des architectures multi-robots. Nous exposerons enfin les grandes classes d'architecture de même que celles ayant le plus marqué ce domaine de la robotique et de l'intelligence artificielle ces dernières années. Enfin, l'annexe B présente une bibliographie sur les principales méthodes utilisées pour le contrôle de systèmes non holonomiques.

# Chapitre 2

# Interactions et contrôle décentralisé

Pour certaines architectures de robots décentralisés, comme celles étudiées dans le domaine de la vie artificielle (Brooks, 1986; Arbib et Liaw, 1995), le but individuel de chaque robot peut-être décrit en termes de méthodes d'interaction avec l'environnement et d'un désir de survivre dans cet environnement. Atteindre un état désiré précis avec succès dans pareilles architectures est chose peu facile puisque aucune planification n'est utilisée. Pour des applications industrielles, il est essentiel que l'état désiré du système soit atteint selon un scénario planifié (tâche), tout en satisfaisant des contraintes sur la façon de remplir les tâches. Pour un système multi-robots décentralisé orienté vers l'accomplissement de tâches, le comportement de chaque robot doit aussi découler d'un contrôle local et de l'interaction avec d'autres robots. L'ap-

proche présentée ici porte sur le contrôle décentralisé orienté vers l'accomplissement de tâches pour des systèmes multi-robots de type coopératif. Avant de présenter les éléments de l'approche, nous présenterons une taxonomie sur les interactions entre robots.

## 2.1 Taxonomie des interactions

- **Tâche** Une tâche implique un déplacement d'un état du système à un autre. Elle est considérée accomplie lorsqu'un état particulier ou un état à l'intérieur d'un sous-ensemble d'états, parmi l'ensemble des états possibles, est atteint.

  L'exécution de la tâche découle des *interventions* de chaque robot sur le système global constitué des robots et de la tâche. Dans un système décentralisé, ces interventions doivent être choisies par le robot avec l'aide de ce que nous appellerons les *outils* du robot, *i.e.* un ensemble de procédures décrivant les comportements et la manière d'interagir avec les autres unités du groupe tout au long de l'exécution de la tâche (figure 2.1).

- **Interaction** Il y a une *interaction* entre deux robots ou plus lorsque l'un d'entre eux contraint les mouvements des autres dans leurs interventions sur la tâche. Les interactions peuvent être de plusieurs formes selon qu'ils ont des effets positifs ou négatifs sur la tâche ou sur les interventions des autres robots.

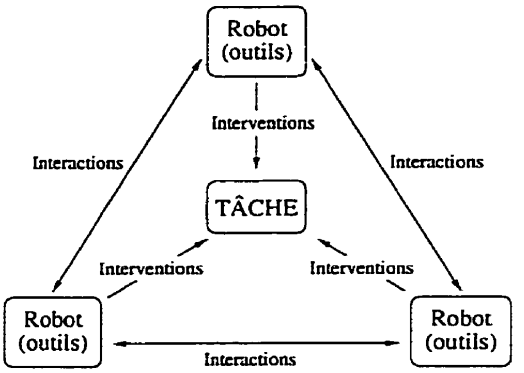



Figure 2.1 : Accomplissement d'une tâche dans un système multi-robots

- *Coopération.* Une interaction est définie comme coopérative si les robots sont impliqués dans l'exécution d'une même tâche et si leurs interventions ont des effets positifs dans l'accomplissement de la tâche et sur les interventions des autres robots. On peut retrouver plusieurs exemples de cette forme d'interaction notamment dans des applications de transport d'objets (Fukuda et Sekiyama, 1994; Mataric et al., 1995).

- *Compétition positive.* Pour une tâche, un système multi-robots a un comportement compétitif positif si, 1) les diverses interactions entres les robots découlent de la volonté de remplir la même tâche prescrite, 2) le robot initiant une interaction a pour but d'optimiser ses interventions personnelles sur la tâche, au détriment, éventuellement, des interventions des autres robots du groupe. Ce type d'interaction est souvent rencontré dans les architectures telles que les essaims de robots (swarm (Dudek, Jenkin, Milios et Wilkes, 1993)), ou les architectures réactives comme dans (Brooks,

Tableau 2.1 : Effets d'une interaction sur les interventions des autres robots et sur leur tâche respective

| Type d'interaction | Effets sur les interventions des autres robots | Effets sur la tâche du robot |
|---|---|---|
| Coopération | positive | positive |
| Compétition positive | négative | positive |
| Compétition négative | négative | négative |

1986), où le manque de connaissance des robots pris individuellement se traduit souvent par des actions allant à l'encontre des interventions des coéquipiers, mais où cet aspect gênant est compensé par le fait que l'équipe comprend un grand nombre de robots.

- *Compétition négative.* La compétition négative concerne les interactions entre robots dont les tâches sont opposées, de sorte que le progrès de chacun dans l'accomplissement de sa tâche fait régresser la tâche des autres robots, et vice versa. Les problèmes de poursuite et d'évasion (Zanardi, Hervé et Cohen, 1996; Isaacs, 1965) ou d'autres types de jeu illustre cette forme d'interaction.

Les différentes formes d'interaction sont illustrées au tableau 2.1.

- **Degré de l'interaction**

Dans un système coopératif, le degré d'interaction entre les différentes unités du groupe est souvent un indicatif de la complexité du contrôle. Lorsque le couplage des interventions augmente, les contraintes sur les robots se resser-

rent, améliorant généralement la qualité globale des comportements coopératifs, mais augmentant en même temps les exigences au niveau des contraintes de mouvement des robot du système.

Les interactions peuvent être classifiées selon leur niveau de coercition (voir la figure 2.2). Le niveau le plus bas n'implique aucune interaction entre les robots, ceux-ci évoluant d'une façon indépendante du reste de la population. Le niveau suivant concerne les situations d'interactions spatiales, dans lesquelles les contraintes sont exprimées dans l'espace des configurations du système et n'affecte pas les mouvements des robots d'une manière continue. Une tâche consistant pour un robot à dégager la route d'un autre robot, comme dans (Kuniyoshi, Riekki, Ishii, Rougeaux, Kita, Sakane et Kakikura, 1994), requiert ce type d'interaction.

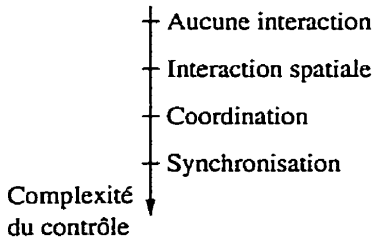



Figure 2.2 : Niveau d'interaction

Le niveau suivant concerne les situations d'interactions impliquant des contraintes autant spatiales que temporelles. Dans ce cas, la *coordination* entre les unités devient une condition importante pour l'accomplissement de la tâche, nécessitant l'attention de chaque robot envers la tâche et/ou envers les interventions des autres unités. Les

tâches de déplacement coopératif d'objets, comme celles présentées dans (Wang et al., 1994; Mataric et al., 1995), utilisent ce type d'interaction de coordination.

Le niveau le plus élevé d'interaction concerne ce que nous appellerons les interactions *synchronisées*, où la tâche demande une coordination constante pour atteindre l'état désiré. Ce type d'interaction est entre autres illustré dans (Hashimoto et al., 1993), où une équipe de robots hiérarchisée déplace un objet en le soulevant du sol.

À mesure que le niveau d'interaction croit, le design du système devient de plus en plus difficile car : (1) les contraintes limitent davantage les mouvements de chaque robot, et (2) l'attention mutuelle entre les robots doit augmenter en intensité et en fiabilité pour permettre aux robots d'ajuster leurs interventions en fonction de ceux des autres unités.

Notre approche dans cette étude concerne le design de stratégies de contrôle décentralisé pour des systèmes multi-robots dotés de comportements *coopératifs* et *coordonnés*.

Notre objectif est de trouver une manière d'accomplir des tâches coopératives exigeant des interactions de coordination ou de synchronisation. Le système de robot est orienté vers l'accomplissement de tâches par le fait que chaque robot impliqué dans l'exécution de la tâche est conscient de cette dernière dans sa totalité et peut délibérément modifier son cours tout au long de sa progression. Ceci diffère du modèle "Pusher/Steerer" de (Brown et Jennings, 1995) où la tâche des deux robots est de déplacer une boîte le long d'une trajectoire, le robot pousseur procurant uniquement

la propulsion du système, tandis que l'autre robot est responsable du suivi de la trajectoire.

Comme dans (Ota et al., 1995; Hashimoto et al., 1993; Samson, Borgne et Espiau, 1991), notre stratégie est divisée en deux niveaux de contrôle. Le niveau supérieur concerne la manière dont la tâche va évoluer dans l'espace et le temps pour garantir la qualité des comportements coopératifs du groupe. Le niveau inférieur concerne le contrôle individuel des robots afin que ceux-ci suivent les requêtes du niveau supérieur. Dans (Wang et al., 1994; Ota et al., 1995), le niveau supérieur est rempli par un membre de l'équipe qui transmet ses décisions aux autres comme à l'intérieur d'une architecture *Maître/Esclaves*. Notre approche, entièrement décentralisée, diffère de cette dernière. Dans notre cas, chaque robot connaît la façon dont la tâche doit évoluer sans avoir besoin d'utiliser une forme de communication et de hiérarchie entre les coéquipiers.

Les informations concernant la façon d'accomplir la tâche sont obtenues par l'intermédiaire de la spécification de la tâche. Notre approche consiste principalement à étendre la définition de la tâche (définie précédemment comme un point ou un ensemble de points désirés dans l'espace des configurations du système), en ajoutant un champ de vecteurs défini sur l'espace de la tâche. Ce champ de vecteurs dote l'espace de la tâche de la structure d'un attracteur, assurant ainsi la convergence de la tâche vers la ou les configurations désirées.

À partir de cette nouvelle manière de spécifier la tâche, chaque membre d'une équipe peut prédire localement l'évolution de la tâche et utiliser cette information de prédiction pour déterminer ses déplacements ou son contrôle de bas niveau. Les robots utilisent enfin leurs outils respectifs pour gouverner leurs interventions conformément aux changements désirés à appliquer à la tâche.

Comme dernière remarque, nous devons mentionner le concept plus général de tâche donné dans (Samson et al., 1991), qui n'est pas orienté vers les systèmes robotiques coopératifs. Nous croyons qu'en spécifiant la tâche avec un champ de vecteurs, non seulement nous procurons des capacités de coordination, mais nous donnons aussi la possibilité à un robot de modifier la manière dont évolue la tâche. Le champ de vecteurs étant défini pour chaque état de la tâche, il peut être pris en compte par tous les membres de l'équipe sans exiger d'échange d'informations de manière explicite. Les événements imprévisibles tels les obstacles peuvent aussi être traités même s'ils sont localement détectés que par un seul robot de l'équipe.

## 2.2 Architecture de contrôle décentralisé

Une tâche est spécifiée ici par un point désiré $t_d$ dans l'espace de la tâche $T$ ou par un sous-ensemble de points désirés $D \subset T$ tel que la tâche est considérée comme complétée lorsqu'un point $t \in D$ est atteint. Les connaissances concernant la description de la tâche, *i.e.* position de $t_d$ ou $D$ dans $T$, sont partagées par tous les robots impliqués dans son exécution.

Nous discuterons en premier de la relation entre l'espace de la tâche $T$ et l'espace d'état du système $G$, décrite en terme de $n$ coordonnées généralisées $x_1, \ldots, x_n$. Une tâche est décrite par le vecteur d'état de la tâche $\vec{t} = (t_1, \ldots, t_m)$, où l'espace de la tâche $T$ est un manifold. Cette hypothèse est naturelle puisque l'espace de la tâche peut être vu comme une surface de dimension $m$ dans $G$. Selon la définition normale d'une tâche, celle-ci est décrite comme une position dans l'espace des configurations, augmentée parfois d'exigences sur la configuration du système en ce point. Ces restrictions peuvent être formulées par :

$$\begin{aligned} x_1 &= f_1(t_1, \ldots, t_m), \\ &\ldots \\ x_n &= f_n(t_1, \ldots, t_m). \end{aligned} \tag{2.1}$$

où la fonction non linéaire $f_i$ définie une correspondance $T \rightarrow G$. Notons que l'espace de la tâche peut parfois être défini simplement comme un sous-espace d'état approprié du système $G$. Pour des situations plus générales, $T$ est une surface de dimension $m$, pour laquelle la structure de manifold peut être facilement introduite (Samson et al., 1991). L'exécution de la tâche est représentée par une trajectoire continue sur $G$, d'une configuration initiale à une configuration finale. La trajectoire comme telle peut être vue comme une fonction de correspondance $\tau_G$ appliquée à l'espace d'état du système $G$.

$$\tau_G : [0, 1] \rightarrow G.$$

Soit $\tau$, la projection de la courbe $\tau_G$ sur l'espace de la tâche $T$. Il existe une infinité de trajectoires à l'intérieur de $T$ pouvant satisfaire les exigences de la tâche. Ceci est une conséquence directe de la façon de spécifier une tâche comme un but ponctuel dans l'espace de la tâche $T$, ne contraignant pas la progression comme telle de la tâche. Comme nous allons le montrer, les déplacements dans l'espace d'état du système $G$ sont contraints, mais cette liberté sur les trajectoires peut être utilisée pour des fins de coordination.

Dans des systèmes coordonnés ou synchronisés, les robots doivent disposer de moyens pour inférer les intentions des autres, sinon leurs interventions lors de l'exécution de la tâche peuvent être en conflit. L'utilisation de communication explicite pour informer les autres sur la façon dont la tâche devrait évoluer localement est sans doute la manière la plus évidente d'établir un comportement coopératif. Cependant, cette approche hérite de tous les inconvénients des systèmes de contrôle centralisé, apporte d'importants délais dans la boucle de contrôle et en est inadéquate pour des équipes de plusieurs robots.

Comme alternative à la communication explicite, nous proposons d'ajouter aux connaissances du robot, une information sur la manière d'accomplir la tâche localement. Cette information peut consister en un déplacement désiré, dans l'espace de la tâche, défini pour chaque état de la tâche. Ainsi, de nouvelles contraintes sont imposées sur les mouvements des robots. Par contre, ces contraintes ne doivent pas être trop restrictives, pour éviter que de l'énergie soit inutilement gaspillée pour les

remplir. Les contraintes devraient aussi donner à l'équipe de robots des capacités leur permettant de faire face à des événements imprévus tels qu'un obstacle sur le chemin.

Le terme *spécification de la tâche* va donc être utilisé, en élargissant sa définition en une spécification des changements désirés à apporter à l'état de la tâche $t$ pour tous les états $t \in T$ possibles. Formellement cette spécification va consister en un champ de vecteurs de déplacement désiré $X(t)$, défini pour tous les points $t$ dans $T$ comme montré à la figure 2.3.

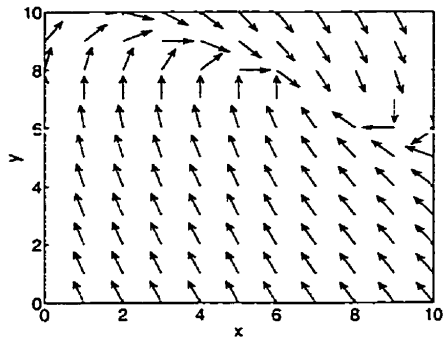



Figure 2.3 : Champ de vecteurs

Nous devons, à ce point, introduire quelques concepts de base en géométrie différentielle. Un champ de vecteurs défini sur un manifold $T$ décrit une correspondance qui associe un vecteur $X(t)$, $t \in T$ à l'espace tangent $TT_t$. L'espace tangent au point $t$ est un espace linéaire de dérivés directionnelles. Une courbe intégrale du champ de vecteur $X$ qui passe par les points $t$ est une correspondance $\tau_t : I \rightarrow T$, $I$ étant un intervalle ouvert. Cette correspondance est définie comme suit :

- il existe $s_0 \in I$ tel que $\tau_t(s_0) = t$,

- pour tout $s \in I$ l'équation différentielle suivante s'applique :

$$\dot{\tau}_t(s) = X(\tau_t(s)). \tag{2.2}$$

Cette équation différentielle définit localement une courbe, qui lorsque étendue à la solution globale, donne une trajectoire $\tau(s)$ dans l'espace de la tâche. La famille de courbes intégrales correspondant au champ de vecteurs de la figure 2.3 est présentée à la figure 2.4. De manière rigoureuse (Nijmeijer et der Schaft, 1990), $\dot{\tau}$ doit être traité comme un opérateur de correspondance linéaire $R^1 \to TT$, qui associe la dérivé d'une fonction scalaire à la dérivé le long de la direction du champ $X$ de cette fonction sur $T$. En utilisant la notation conventionnelle pour les coordonnées, nous pouvons écrire l'équation (2.2) comme :

$$\begin{aligned} \dot{\tau}_1(s) &= X_1(\tau_1(s), \ldots, \tau_m(s)), \\ &\ldots \\ \dot{\tau}_m(s) &= X_m(\tau_1(s), \ldots, \tau_m(s)). \end{aligned} \tag{2.3}$$

où $(X_1, \ldots, X_m)$ sont les composantes de $X$ dans le système de coordonné de la tâche $(t_1, \ldots, t_m)$.

Le champ de vecteur $X$ est ainsi associé de manière surjective au système d'équations différentielles (2.3). En prenant comme hypothèse que les exigences du théorème d'unicité tiennent, il en découle que la courbe intégrale passant par chaque point de

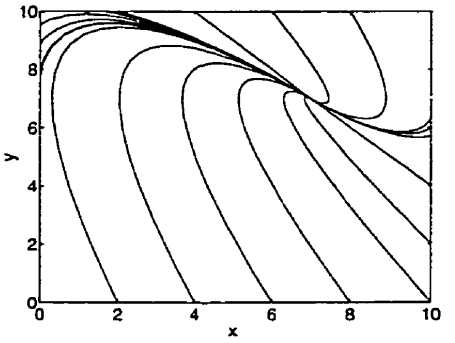



Figure 2.4 : Trajectoires dans l'espace de la tâche

$T$ peut être définie. L'idée principale de cette construction est de définir un champ de vecteurs de telle sorte qu'il garantira la convergence de toutes les courbes intégrales vers le point désiré $t_d$ ou l'ensemble des points désirés $D$.

Différentes méthodes peuvent être utilisées pour désigner le champ de vecteurs $X$. Pour des problèmes simples, le choix de $X$ peut être basé sur des fonctions de potentiel appropriées, assurant ainsi une attraction des trajectoires vers une région désirée dans l'espace de la tâche. En pratique, le choix d'un champ de vecteurs adéquat est limité par le fait que les trajectoires ou les courbes intégrales qui lui correspondent doivent être réalisables par l'équipe de robots. Même si cette exigence de faisabilité ne conduit pas directement au choix de $X$, elle doit être prise en considération pour toutes configurations particulières du système.

Une fois définie, la connaissance du champ $X$ est partagée par tous les robots impliqués dans l'exécution de la tâche, garantissant ainsi que les intentions des robots sur la tâche sont les mêmes (au moins localement), sans qu'il soit nécessaire d'utiliser une forme de communication explicite quelconque.

Au niveau des robots, la tâche doit être exécutée par une équipe de $k$ unités $r_1, \ldots, r_k$, chacune dotée de ses propres caractéristiques cinématiques et dynamiques. Chaque robot tentera d'arranger ses interventions de telle sorte que les changements apportés à la tâche suivent la direction du champ de vecteurs, au moins localement. Ainsi, les outils des robots doivent être conçus de telle sorte qu'ils génèrent un contrôle qui amène des changements sur l'état de la tâche approximant ceux désirés décrits par le champ de vecteurs $X$. Les caractéristiques des robots doivent aussi être prises en compte dans la procédure de conception des outils.

Si le champ de vecteurs suggère une direction pour la tâche qui est irréalisable par les robots, le système ne pourra pas converger vers le but. Pour s'assurer qu'un champ de vecteurs puisse être approximé correctement par les robots, les contraintes imposées aux déplacements des robots doivent être considérées lors de l'étape de spécification de la tâche. Dans l'application présentée au chapitre 3, les contraintes non holonomiques des robots nous ont amenées à modéliser la poutre comme un uni-cycle, et à utiliser un champ de vecteurs $X$ relativement complexe.

Un autre aspect important des outils de chaque robot est que ce dernier ne peut compter que sur les informations fournies par ses propres senseurs. Pour un robot donné $r_i$, ces informations vont être représentées par l'espace d'état observable $RT_i$. Soit $R_i$ un manifold de $G$ contenant les états possibles $r_i$ du $i$-ième robot, avec $R_i \cap R_j = \emptyset$, $(\bigcup_{i=1}^{n} R_i \cup T) \subset G$ (figure 2.5). L'espace d'état observable $RT_i$ est défini par

l'opérateur d'observation $P_i : G \rightarrow RT_i$.

$$RT_i = \{z_i\}, \quad z_i = P_i(x_1, \ldots, x_n).$$

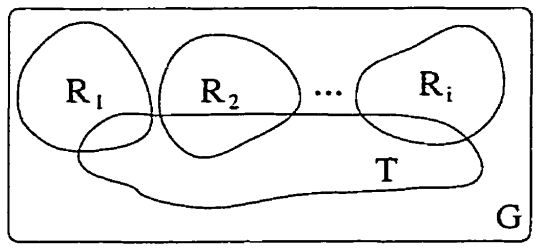



Figure 2.5 : Espace d'état du système

Les outils utilisés par $r_i$ doivent être conçus de manière à n'utiliser que les informations appartenant à son espace observable $RT_i$. Il est évident que $RT_i$ doit procurer suffisamment d'information sur l'état de la tâche afin que le robot puisse décider où il devra se diriger pour accomplir la tâche et quand cette tâche sera finalement accomplie. La connaissance de l'état d'un coéquipier n'est par contre pas requise pour remplir une tâche. Ainsi, le contrôleur de chaque robot doit être conçu sous la forme $U_i(z_i, X(t))$ et non sous une forme telle $U_i(x_1, \ldots, x_n)$ ou $U_i(z_1, \ldots, z_k)$.

En incluant de façon explicite $X(t)$ comme second argument de $U_i(.)$, nous faisons l'hypothèse que l'état de la tâche $t$ peut être obtenu de $z_i$. Cette hypothèse est souvent trop restrictive et peut être relâchée en utilisant une estimation de $t$ basée sur les mesures disponibles $\hat{t} = \hat{t}(z_i)$.

Finalement, les outils des robots seront conçus de telle sorte qu'ils permettront

d'approximer au mieux possible le déplacement désiré pour la tâche $\partial t_{des} = X(t)$, garantissant ainsi qu'une bonne coordination soit maintenue tout au long de l'exécution de la tâche. Les outils vont aussi devoir prendre en compte les contraintes intrinsèques des robots telles les limites de vitesse, les contraintes non holonomiques, etc.

Dans le prochain chapitre, nous présenterons une application de notre approche, montrant ainsi les bénéfices apportés par son utilisation dans la conception de systèmes multi-robots.

# Chapitre 3

# Exemple d'application : transport coopératif d'une poutre

Dans le but d'illustrer les idées présentées dans le chapitre précédent, nous décrivons ici une application nécessitant un haut niveau de coordination entre deux robots mobiles. La tâche consiste à déplacer une poutre d'une configuration initiale vers une configuration finale en la soulevant par ses extrémités. Nous présenterons dans ce chapitre une description de la tâche que nous nous sommes fixée, puis nous décrirons le modèle proposé pour représenter la tâche et les robots conformément au formalisme exposé au chapitre précédent.

## 3.1 Description de la tâche

La tâche consiste à transporter un objet, dans ce cas-ci une poutre, d'une configuration donnée à une autre à l'aide de deux robots travaillant en coopération (voir figure 3.1). Les deux robots sont des véhicules de type automobile équipés à l'avant d'un levier à un degré de liberté. Afin d'exécuter la tâche, les robots doivent soulever la poutre à l'aide de leur levier et la déplacer en coopération.

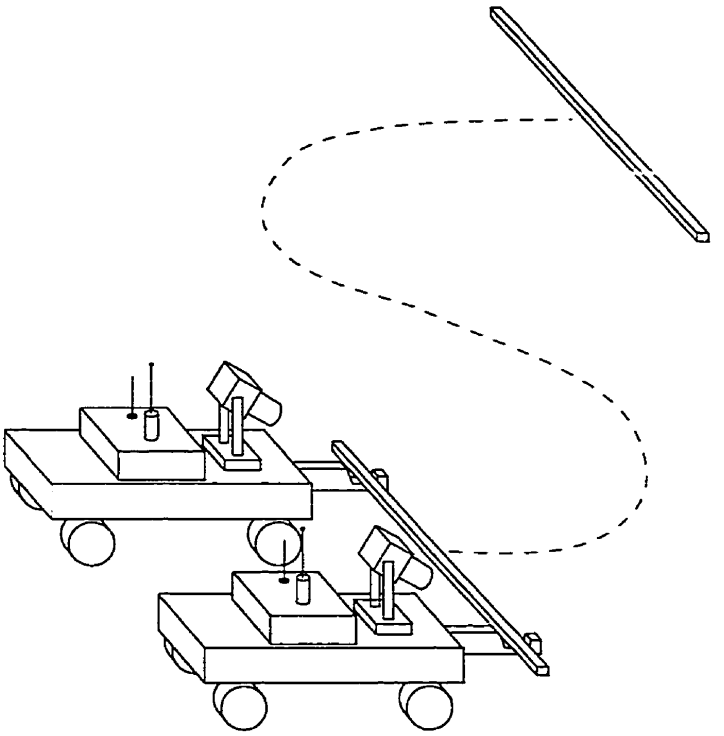


Figure 3.1 : La tâche à accomplir

Les robots ne disposent d'aucun moyen de communication explicite pour coordonner leurs mouvements dans l'accomplissement de la tâche. Seule l'extraction d'informations visuelles simples (voir le chapitre 6) par l'observation de la tâche leur est

disponible pour générer leur contrôle. Le contrôle devra, en premier lieu, permettre de faire déplacer les robots dans un environnement sans obstacles (voir le chapitre 4) de manière à ce que ceux-ci amènent la poutre à sa configuration finale (position et orientation). Nous étudierons ensuite le système en présence d'obstacles (voir le chapitre 5). L'exécution de la tâche se fera tout en satisfaisant des contraintes sur la configuration de la poutre et sur les déplacements des robots.

Nous considérons qu'au début d'un essai expérimental, la poutre est déjà soulevée par les deux robots. Ainsi, nous ne traiterons pas les problèmes reliés à l'approche visuelle de la poutre et à la capture de cette dernière. Pour simplifier davantage le problème, nous ferons l'hypothèse que les robots disposent en tout temps d'une connaissance de leur configuration (position/orientation) relative à la position finale désirée pour la poutre.

Nous présenterons dans ce qui va suivre la modélisation du système étudié. Cette modélisation cherche à décrire les divers éléments du banc d'essai expérimental (décrit au chapitre 7), afin d'étudier la résolution de ce problème de transport dans l'optique d'appliquer notre approche à des robots réels. Nous décrirons en premier les robots utilisés ainsi que leur cinématique. La modélisation de la poutre transportée sera ensuite décrite. Enfin, nous exposerons la modélisation du frottement entre la poutre et les leviers.

## 3.2 Modèle des robots

Dans la discussion qui suit, toutes positions et orientations sont données relativement au système de référence monde $F_W$, attaché à la configuration désirée de la poutre comme illustré à la figure 3.2.

Le système utilisé est composé de deux robots mobiles $r_i (i = 1, 2)$, chacun muni d'un levier frontal permettant de soulever la poutre transportée. Le mouvement des véhicules est contraint à des déplacements sur une surface planaire $F_W = R^2$. La configuration de chaque robot est caractérisée par le quadruple $(x_i, y_i, \varphi_i, v_i)$, où $(x_i, y_i)$ sont les coordonnées du point situé au centre de l'essieu arrière dans l'espace des coordonnées cartésiennes, $\varphi_i$ est l'orientation relative à l'orientation finale de la poutre et $v_i$ est la vitesse (Figure 3.2). L'espace d'état correspondant $R_i$ est de dimension quatre et difféomorphique à $R^3 \otimes C^1$ (Figure 3.2).

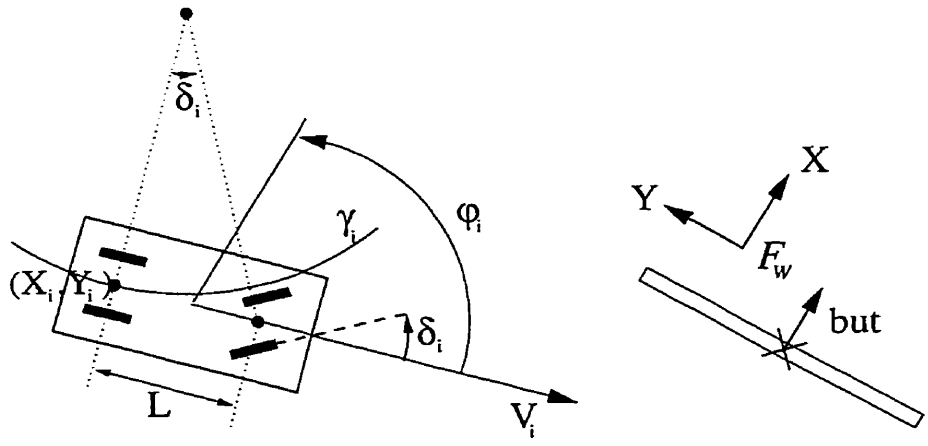



Figure 3.2 : Système de coordonnées

Le déplacement des robots est régi par des contraintes non holonomiques. En

supposant que les voitures ne sont pas sujettes au glissement latéral des roues, les voitures peuvent se déplacer uniquement le long de la courbe instantanée $\gamma_i$. On peut représenter cette contrainte par l'équation suivante :

$$-\sin(\varphi_i)\dot{x} + \cos(\varphi)\dot{y} = 0. \tag{3.1}$$

Cette équation est non intégrable, il en résulte que l'espace des déplacements possibles $(\dot{x}_i, \dot{y}_i, \dot{\varphi}_i)$ est réduit à un espace à deux dimensions. Ainsi, toutes les configurations de l'espace $F_W$ sont atteignables, mais pour une configuration donnée, seulement un nombre restreint de directions de déplacement est permis.

On considère que la distance entre les roues d'un même essieu, la voie, est suffisamment petite relativement à l'empattement $L_i$ (distance entre l'essieu avant et arrière). Cette considération et le fait que l'angle de braquage est borné $\delta_{min} \leq \delta_i \leq \delta_{max}$, nous permet de faire l'approximation du modèle cinématique et dynamique d'une voiture par celui d'une bicyclette en remplaçant les roues d'un même essieu par une roue médiane située au centre de l'essieu.

### 3.2.1 Description du modèle dynamique d'une bicyclette

Les robots utilisent le modèle dynamique d'une bicyclette suivant :

$$\begin{cases} \dot{v}_i = a_i, \\ \dot{x}_i = v_i \cos(\varphi_i), \\ \dot{y}_i = v_i \sin(\varphi_i), \\ \dot{\varphi}_i = \dfrac{v_i \tan \delta_i}{L_i}, \\ |\delta_i| \leq \delta_{max}. \end{cases} \tag{3.2}$$

où $a_i$ est l'accélération du véhicule et $\delta_i$ son angle de braquage.

Notons que nous avons simplifié le modèle dynamique des voitures pour le développement de notre solution. Ce choix est justifié car, étant donné la vitesse relativement faible des robots, il est possible de négliger l'inertie du véhicule, de même que les effets des forces centripètes.

## 3.3 Modèle de la poutre

Un levier est installé à l'avant de chaque véhicule pour leur permettre de soulever une des extrémités la poutre.

### 3.3.1 Description de la poutre

La poutre est caractérisée par sa longueur $D$ et par sa position/orientation $(X_0, Y_0, \varphi_0)$ dans l'espace cartésien relatif à la configuration finale $F_W$. La tâche étant décrite en terme de position désirée pour la poutre, nous pouvons décrire l'espace de la tâche comme $R^2 \otimes C^1$ avec ses trois coordonnées naturelles $(X_0, Y_0, \varphi_0)$.

Étant déposée sur les leviers situés à l'avant des robots, la poutre ne peut pas se déplacer latéralement relativement au levier (c'est-à-dire le long du levier). Seuls les déplacements longitudinaux (le long de la poutre) et angulaires sont permis (voir figure 3.3). Pour empêcher la poutre de tomber hors des leviers, nous avons fixé des embouts à ses extrémités.

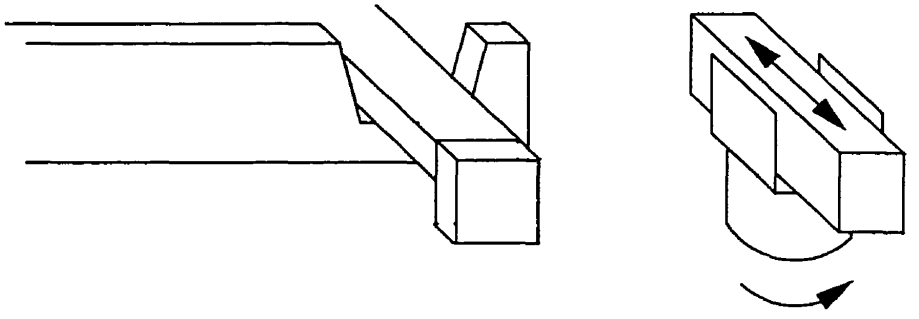


Figure 3.3 : Degrés de liberté de la poutre

Le comportement cinématique de la poutre est décrit par ses vitesses de déplacement transversal $V_t$ et longitudinal $V_l$, et par sa vitesse angulaire $\omega_0$ selon les équations suivantes :

$$\begin{aligned} \dot{X}_0 &= V_t \cos\varphi_0 + V_l \sin\varphi_0, \\ \dot{Y}_0 &= V_t \sin\varphi_0 - V_l \cos\varphi_0, \\ \dot{\varphi}_0 &= \omega_0. \end{aligned} \tag{3.3}$$

Il peut sembler que le système défini par les deux robots et la poutre doit être décrit par $11 = 4 + 4 + 3$ paramètres et qu'ainsi l'espace d'état du système $G$ soit de dimension 11. En fait, le système complet peut être vu comme étant une voiture équipée de deux roues mobiles pouvant s'orienter indépendamment l'une de l'autre (comme dans (Yun et N.Sarkar, 1996)), et se déplacer le long de l'axe comme illustré dans la figure 3.4. Ainsi, seule la configuration des roues relativement à la poutre doit être explicitement incluse dans l'espace d'état du système. La position et l'orientation des deux robots relativement à la poutre sont ainsi déterminées par la distance le long de la poutre à partir de son centre $D_i$, et par l'angle entre l'orientation de la voiture et celle de la poutre $\psi_i$.

### 3.3.2 Déplacement longitudinal de la poutre

La poutre peut tourner librement autour de son point de contact sur chaque levier de même que faire une translation le long de sa direction longitudinale. Bien que le premier type de mouvement soit entièrement déterminé par la configuration et les

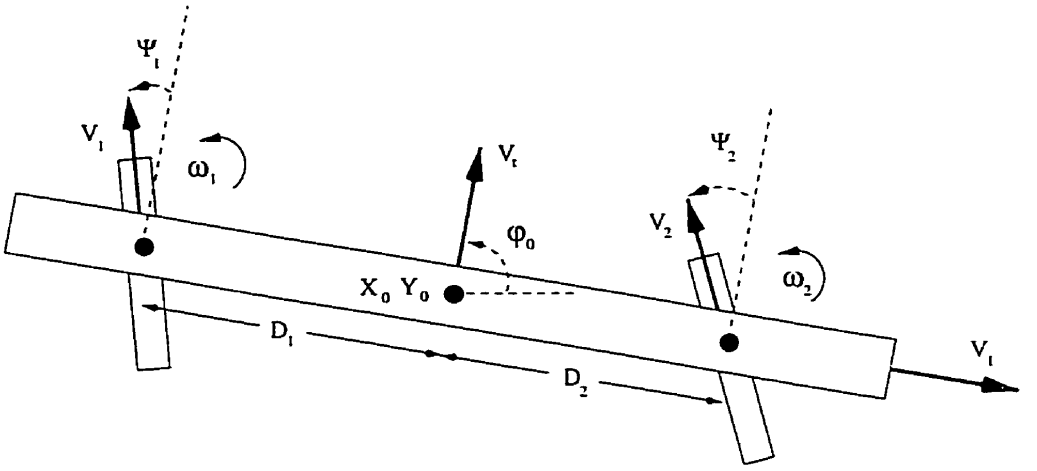



Figure 3.4 : Les paramètres du système

vitesses des robots (Figure 3.4), le déplacement longitudinal de la poutre sur le levier ne peut pas être déterminé sans utiliser un modèle de friction aux points de contact.

Le déplacement latéral de la poutre sur les leviers est régi par les lois des forces de frottement sec. Lorsque la distance entre les leviers des robots varie, chaque robot applique une force pour amener la poutre dans sa direction de mouvement. La force $f_p$ appliquée sur la poutre par l'autre robot tente de faire glisser la poutre sur le levier du robot $r_i$. Il en résulte une force de friction, $f_t = \mu f_n$, fonction de la masse de la poutre sur le levier, dans la direction opposée à la force $f_p$ (figure 3.5(b)). Le coefficient de friction $\mu$ varie selon que la poutre est immobile sur le levier (friction statique), ou qu'elle est déjà en mouvement (friction dynamique) (McKerrow, 1991).

La force de friction peut aussi être représentée comme dépendante, de façon explicite, de la vitesse de glissement de la poutre relative au levier. Cette dépendance peut être exprimée par une courbe ayant la même forme que celle de la

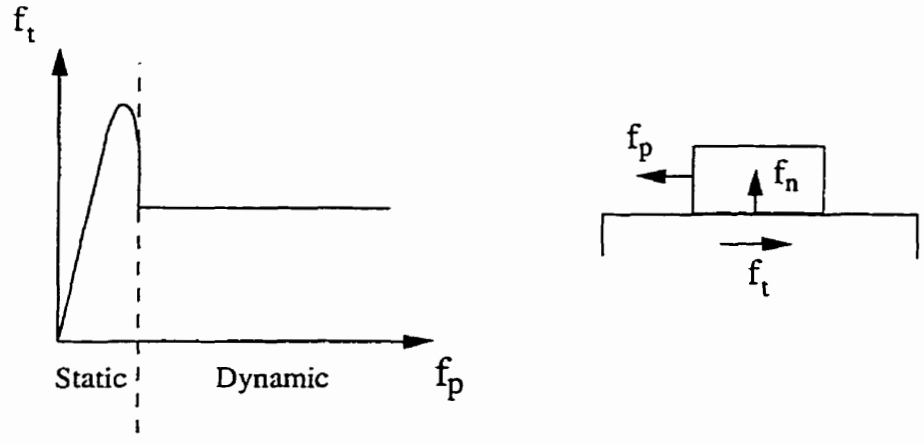



Figure 3.5 : Modèle de friction

figure 3.5(a) (Armstrong, 1988). On fait une approximation de cette courbe par une fonction de valeur constante excepté pour $V_l = 0$ où elle prend une valeur plus élevée.

Dans notre modèle on considère que le coefficient de friction statique est beaucoup plus grand que celui de friction dynamique. Ainsi, lorsque la poutre commence à glisser sur un levier, elle a tendance à continuer sa course sur le même levier jusqu'à ce qu'elle s'immobilise. Le changement de vitesse de glissement de la poutre sur les leviers est donc fonction du poids de la poutre et du changement dans la distance entre les robots.

$$\dot{V}_l = \frac{1}{m}\left(f(\dot{D}_1) - f(\dot{D}_2)\right). \tag{3.4}$$

### 3.3.3 Algorithme pour modéliser la friction

La modélisation de la friction est très étudiée dans la littérature (Armstrong-Hélouvry, Dupont et Wit, 1994). Cette modélisation est généralement employée à des fins de contrôle, dans le but de compenser les forces de friction provoquées à l'intérieur des machines. Pour ce type d'application, plusieurs algorithmes pour modéliser la friction donnant de bons résultats ont été développés.

Le modèle de friction doit, dans notre cas, nous permettre de déterminer l'évolution des déplacements longitudinaux de la poutre sur les leviers. La difficulté vient du fait que le déplacement longitudinal de la poutre est uniquement généré par deux forces de friction provenant des deux contacts avec les leviers. Ces deux contacts étant à priori du même type, il en résulte qu'il est très difficile d'évaluer les forces de frictions appliquées. Nous avons donc créé un algorithme nous permettant de faire une approximation du comportement réel de la poutre.

Il est possible d'évaluer la composante de la vitesse des voitures le long de la poutre avec l'équation ci-dessous (voir figure 3.6) :

$$V_{\perp_i} = -V_i \sin \psi_i. \tag{3.5}$$

Soit $V_\tau$, la composante transversale de la vitesse de la poutre. La position du centre de la poutre relative à un levier est déterminée à l'instant $n+1$ par :

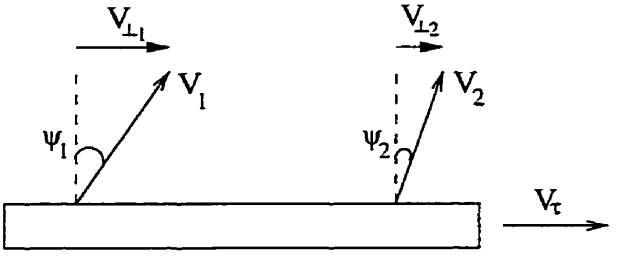



Figure 3.6 : Système de coordonnées de la poutre

$$D_i^{n+1} = D_i^n + (-1)^{i+1} \frac{(V_{\perp_i}^n - V_\tau^n) + (V_{\perp_i}^{n+1} - V_\tau^{n+i})}{2} \Delta t. \tag{3.6}$$

Lorsque la vitesse de la poutre n'est pas égale à l'une des deux projections des vitesses des voitures sur la poutre, on peut affirmer que la force de friction est uniquement du type *frottement dynamique*. Dans pareils cas, la vitesse de la poutre est calculée par :

$$V_\tau^{n+1} = V_\tau^n + a\Delta t, \tag{3.7}$$

$$a = \frac{1}{m}(f_1 + f_2), \tag{3.8}$$

$$f_i = \begin{cases} +f & \text{Si } V_{\perp_1}^{n+1} - V_\tau^n > 0 \\ -f & \text{Si } V_{\perp_1}^{n+1} - V_\tau^n < 0 \end{cases}. \tag{3.9}$$

Si, par contre, la poutre se déplace à la même vitesse que l'une des deux voitures et donc reste immobile sur un levier, la force de friction sera beaucoup plus grande et la poutre aura tendance à s'immobiliser sur ce levier, glissant sur l'autre. Dans un pareil cas, on considère que la poutre garde la même vitesse que la composante

longitudinale de la vitesse de la voiture sur la poutre .

$$V_\tau^{n+1} = V_{\perp_i}^{n+1}. \tag{3.10}$$

Dans le cas où, à l'intérieur d'une itération, il y a un changement dans la direction de la force $f_i$, on peut affirmer que le système passera par un état où la vitesse de déplacement de la poutre par rapport au levier est nulle. À cet instant, la force de friction de la poutre sur le levier de $r_i$ augmente brusquement, et la poutre conserve une vitesse nulle par rapport à ce levier. On peut déterminer l'instant, après le début de l'itération, où $V_{\perp i} - V_\tau = 0$ par l'équation suivante :

$$\Delta\tilde{t} = \frac{\Delta t(V_\tau^n - V_{\perp_i}^n)}{V_\tau^n - V_{\perp_i}^n - V_\tau^{n+1} + V_{\perp_i}^{n+1}}. \tag{3.11}$$

Dans le cas où la valeur de $\Delta\tilde{t}$ est plus petite que $\Delta t$ pour un des deux robots, la vitesse de la poutre devient égale à la projection de la vitesse du robot $r_i$ sur la poutre. Dans le cas où $\Delta\tilde{t}$ est plus petit que $\Delta t$ pour les deux robots, on doit choisir le robot ayant le plus petit $\Delta\tilde{t}$.

Si la projection de la vitesse des deux robots s'égale à un instant à l'intérieur d'un intervalle donné, on peut affirmer qu'à cet instant $V_\tau = V_{\perp_1} = V_{\perp_2}$. Il devient alors très difficile, voir impossible de déterminer les changements qui seront apportés à la vitesse de la poutre. Nous avons grandement simplifié la modélisation de la friction en réglant cette situation de façon aléatoire et en choisissant la vitesse future de la

poutre comme étant l'une des trois possibilités suivantes :

$$\begin{aligned} V_\tau^{n+1} &= V_{\perp_1}^{n+1}, \\ V_\tau^{n+1} &= V_{\perp_2}^{n+1}, \end{aligned} \tag{3.12}$$

$$V_\tau^{n+1} = \frac{(V_{\perp_1}^{n+1} + V_{\perp_2}^{n+1})}{2}. \tag{3.13}$$

Ce modèle de friction est très simplifié. Plusieurs autres modèles (Armstrong-Hélouvry et al., 1994) tous aussi valables pourraient aussi être utilisés pour cette application.

# Chapitre 4

# Description du contrôleur décentralisé

Dans cette application, chaque robot contrôle ses propres déplacements sans faire usage de communication avec l'autre robot, afin d'accomplir une tâche consistant à déplacer une poutre d'une configuration initiale à une configuration finale. Pour simplifier la coordination, des informations additionnelles sur la manière d'accomplir la tâche doivent être ajoutées à la spécification de la tâche, tel que mentionné au chapitre 2.

Le contrôleur responsable de spécifier la tâche, que l'on nommera le contrôleur de la poutre, choisit un déplacement désiré $(V_t, \omega_0)$ pour la poutre (figure 3.4) en se basant sur ses connaissances de l'état actuel de la tâche. Cette spécification est décrite à l'aide d'un champ de vecteurs $X$ (voir chapitre 2), proposant un déplacement à la poutre qui

lui permettra de converger vers la configuration finale spécifiée. Les outils des robots doivent garantir un comportement correct pour le centre de la poutre en contrôlant adéquatement la propulsion et la direction des roues des véhicules. Dans ce chapitre, nous présenterons en premier une analyse détaillée du système voitures/poutre, nous décrirons ensuite le contrôleur de la poutre, puis nous décrirons l'implantation des fonctionnalités des outils dans le contrôleur de bas niveau. Des simulations seront enfin présentées à la fin de ce chapitre.

## 4.1 Analyse préliminaire

Le module de spécification de la tâche informe les robots du déroulement désiré de la tâche dans le temps. Pour que cette information soit utile aux robots, ces derniers infèrent l'état actuel de la tâche au moyen de leurs senseurs. Dans notre application spécifique, l'état de la tâche est obtenu en observant la poutre à l'aide d'une caméra dirigée vers la poutre (voir chapitre 6). Les robots doivent aussi connaître leur état respectif de manière à pouvoir intervenir sur la tâche selon le plan dicté. Ces informations, correspondant à l'espace observable du robot $RT_i$, proviennent des senseurs montés sur chaque robot. Ces senseurs n'étant pas toujours d'une très grande précision, il est possible que les robots aient une connaissance légèrement erronée de leur état actuel ou de celui de la tâche. Pour que la spécification de la tâche puisse permettre aux robots de coordonner correctement leurs interventions, il faut que le déplacement désiré pour la poutre soit similaire pour les deux robots même si ces

derniers n'ont pas exactement la même idée de l'état actuel de la tâche. Il serait donc délicat de spécifier la tâche comme une trajectoire définie pour le centre de la poutre. Comme il a été mentionné au chapitre 2, nous décrirons la tâche comme un champ de déplacement $X$ désiré pour le centre de la poutre, défini pour tous les points $t$ dans $T$. Pour éviter que les robots aient des intentions opposées, nous décrirons le champ par un contrôle qui aura la particularité de varier doucement dans l'espace de la tâche $T$.

Le choix du contrôle pour déterminer un champ de vecteurs $X$ adéquat est limité par la nécessité que les trajectoires ou les courbes intégrales lui correspondant, soient réalisables par les deux robots. Deux aspects importants doivent être considérés dans le design des équations de contrôle pour décrire le champ de vecteurs $X$. Le champ doit être conçu de manière à :

1. satisfaire les contraintes physiques des robots mobiles,

2. maintenir la poutre dans une configuration contrôlable par les robots.

Ces aspects doivent être considérés tout en n'empêchant pas le champ de vecteurs de garantir à la poutre la convergence vers une configuration finale désirée.

Comme décrit dans le chapitre 3, une des contraintes les plus restrictives des robots mobiles concerne la manière dont ils peuvent se déplacer dans le plan. Les robots étant des véhicules de type automobile, leur déplacement est restreint par des contraintes non holonomiques. Il est donc impossible de leur demander de changer brusquement leur orientation à un instant donné. Ainsi, pour obtenir des changements d'orientation

lisse pour la poutre, et par le fait même doux pour les robots, nous modéliserons la poutre comme un uni-cycle placé au centre de la poutre. À cela s'ajoute des contraintes de vitesse et d'accélération exigeant que les changements de vitesse ou les rotations désirées pour la poutre soient limités. Notons qu'il n'est pas possible d'inclure efficacement les contraintes non holonomiques des voitures directement dans la spécification de la tâche. En effet, il faudrait pour cela que chaque robot soit informé de la configuration actuelle de l'autre robot afin de pouvoir considérer ces contraintes. Ces contraintes sont donc considérées uniquement par le contrôleur de bas niveau.

Pour que la poutre soit contrôlable par les robots, il faut que ces derniers puissent intervenir en tout temps sur la poutre. Étant donné que les déplacements longitudinaux de la poutre ne sont pas contrôlables par les robots, le déplacement désiré de la poutre (et par conséquent, le champ de vecteurs $X$) doit être maintenu orthogonal à la poutre $V_l^{des} \equiv 0$. Le choix de modéliser la poutre par une uni-cycle situé en son centre et dirigé perpendiculairement à celle-ci répond donc à cette exigence.

La modélisation de la poutre par un uni-cycle apporte donc plusieurs avantages au système. Cette modélisation a, entre autres, pour effet de diminuer les déplacements longitudinaux. De plus, une trajectoire générée avec ce modèle à l'avantage d'être plus facilement réalisable par des robots de type automobile. Enfin, il existe dans la littérature (Samson, 1993; M'Closkey et Murray, 1993; Bloch et Drakunov, 1994; Aicardi, Casalino, Bicchi et Balestrino, 1995) plusieurs méthodes de contrôle pour déplacer un uni-cycle dans un plan d'une configuration initiale à une configuration

finale.

Un dernier élément important à considérer pour accomplir cette tâche concerne la nécessité de maintenir les robots à une certaine distance l'un de l'autre afin d'éviter qu'ils ne laissent échapper la poutre ou qu'il y ait une collision entre eux. Mais cet élément n'étant pas directement relié à la tâche, il sera traité au niveau des outils des robots en exigeants que chaque robot reste dans une plage de distance par rapport à l'extrémité correspondante de la poutre.

## 4.2 Relations entre la poutre et les robots

En prenant pour hypothèse que les déplacements longitudinaux de la poutre sont de faibles amplitudes, il est possible de modéliser la poutre par un uni-cycle situé en son centre, et ayant comme orientation la direction perpendiculaire à celle-ci. Grâce à ce modèle, on peut représenter la cinématique de la poutre par les équations suivantes :

$$\begin{aligned} \dot{X}_0 &= V_t \cos\varphi_0 + V_l \sin\varphi_0, \\ \dot{Y}_0 &= V_t \sin\varphi_0 - V_l \cos\varphi_0, \\ \dot{\varphi}_0 &= \omega_0. \end{aligned} \tag{4.1}$$

où $(X_0, Y_0)$ est la position du centre de la poutre, et $\varphi_0$ son orientation dans l'espace de la tâche $T$. La composante normale de la vitesse de la poutre $V_t$ et la vitesse angulaire $\omega_0$ sont reliées aux vitesses des robots par les équations de cohésion globale

suivantes :

$$V_t = \frac{D_2 V_1 \cos\Psi_1 + D_1 V_2 \cos\Psi_2}{D_1 + D_2},$$
$$\omega_0 = \frac{V_2 \cos\Psi_2 - V_1 \cos\Psi_1}{D_1 + D_2}. \tag{4.2}$$

Notons à ce point qu'il serait possible, pour un système centralisé, d'évaluer, à tout instant, la vitesse nécessaire à appliquer aux voitures pour qu'il en résulte une vitesse transversale et angulaire désirée pour la poutre. Les configurations des vitesses transversales et angulaires des robots seraient alors déterminées par les équations suivantes :

$$V_i = \frac{V_l + (-1)^i \dot{D}_i}{\sin\Psi_i},$$
$$\omega_i = \dot{\varphi}_0 + \dot{\varphi}_i. \tag{4.3}$$

Selon les équations ci-dessus, en prenant pour hypothèse que la vitesse latérale de la poutre $V_l$ est connue ou peut être négligée, la vitesse linéaire de chaque robot peut être déterminée lorsque la composante normale de la vitesse du centre de la poutre $V_t$ et sa vitesse angulaire $\omega_0$ sont données.

En combinant les équations du modèle cinématique de l'uni-cycle (4.1) avec celles des relations cinématiques de la poutre avec les robots (4.2), on obtient les équations cinématiques suivantes :

$$\begin{aligned} \dot{X}_0 &= V_l \sin\varphi_0 + \frac{D_2 V_1 \cos\Psi_1 + D_1 V_2 \cos\Psi_2}{D_1 + D_2} \cos\varphi_0, \\ \dot{Y}_0 &= V_l \cos\varphi_0 + \frac{D_2 V_1 \cos\Psi_1 + D_1 V_2 \cos\Psi_2}{D_1 + D_2} \sin\varphi_0, \end{aligned} \tag{4.4}$$

$$\dot{\varphi}_0 = \frac{V_2 \cos \Psi_2 - V_1 \cos \Psi_1}{D_1 + D_2}.$$

On complète le modèle cinématique des robots en évaluant le changement d'orientation de la poutre relatif à un robot et la vitesse du changement dans la distance entre le centre de la poutre et le levier du robot.

$$\dot{D}_1 = V_l - V_1 \sin \Psi_1,$$

$$\dot{D}_2 = -V_l + V_2 \sin \Psi_2,$$

$$\dot{\Psi}_1 = \omega_1 - \omega_0,$$

$$\dot{\Psi}_2 = \omega_2 - \omega_0.$$

En ajoutant les équations de glissement de la poutre sur les leviers, on obtient finalement le modèle cinématique complet du système représenté dans le tableau 4.1.

## 4.3 Contrôleur de la poutre

L'objectif du contrôleur de la poutre est de déterminer un déplacement pour la poutre qui contribuera à approcher celle-ci vers sa configuration (position/orientation) finale désirée.

En prenant en compte les contraintes non holonomiques des robots, nous modéli-

Tableau 4.1 : Modèle cinématique du système

$$
\begin{aligned}
\dot{X}_0 &= V_l \sin\varphi_0 + \frac{D_2 V_1 \cos\Psi_1 + D_1 V_2 \cos\Psi_2}{D_1 + D_2} \cos\varphi_0, \\
\dot{Y}_0 &= V_l \cos\varphi_0 + \frac{D_2 V_1 \cos\Psi_1 + D_1 V_2 \cos\Psi_2}{D_1 + D_2} \sin\varphi_0, \\
\dot{\varphi}_0 &= \frac{V_2 \cos\Psi_2 - V_1 \cos\Psi_1}{D_1 + D_2}, \\
\dot{D}_i &= (-1)^i V_i \sin\Psi_i - V_l, \\
\dot{\Psi}_i &= \omega_i - \omega_0, \\
\dot{V}_l &= \frac{1}{m}\left(f(\dot{D}_1) - f(\dot{D}_2)\right).
\end{aligned}
$$

serons la poutre comme un véhicule uni-cycle placé en son centre. Dans sa forme la plus générale, ce problème tombe dans les limites du théorème de Brockett (Brockett, 1983) qui stipule qu'il n'existe pas de contre-réaction stabilisante et lisse pour la manœuvre d'un véhicule dans un système de coordonnées cartésiennes. On a par contre démontré qu'il est possible de contourner ce problème, notamment (Astolfi, 1994; Aicardi et al., 1995; de Wit et Sørdalen, 1992; Sørdalen et de Wit, 1992) en utilisant une autre représentation du véhicule dans l'espace, de manière à y insérer des discontinuités. D'autres approches basées sur l'utilisation de sinusoïdes dans les lois de contrôle ont été proposées (Samson, 1993; M'Closkey et Murray, 1993; Murray et Sastry, 1993). Il est aussi possible d'utiliser des approches faisant usage du mode de glissement (Bloch et Drakunov, 1994; Gulder et Utkin, 1994; Bloch et Drakunov, 1995) pour le contrôle d'un véhicule dans l'espace cartésien. On trouvera en annexe B

des détails concernant les différentes approches présentées dans la littérature sur le contrôle de systèmes non holonomiques.

Les approches utilisant des sinusoïdes dans les lois de contrôle ou celles utilisant un mode de glissement sont de peu d'intérêt pour notre application. En effet, pour ces approches, les lois de contrôle ont tendance à demander au véhicule d'osciller, généralement en faisant passer sa vitesse de positive à négative continuellement. Cet effet sur la poutre n'est pas intéressant car, pour de telles lois de contrôle, la dynamique demandée pour la poutre varierait beaucoup trop d'une configuration à l'autre. Ainsi, un tel contrôle ne générerait pas un champ de potentiel $X$ doux comme notre approche le demande.

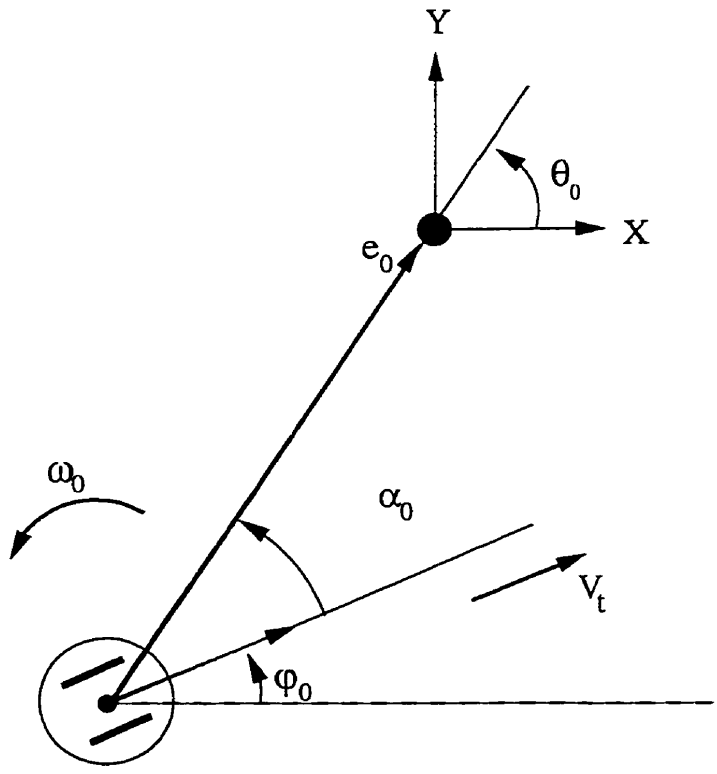



Figure 4.1 : Système en coordonnées polaires

Nous avons donc adopté une approche inspirée de celle présentée par Aicardi *et al.* (Aicardi et al., 1995) qui consiste à faire un changement de représentation dans le plan polaire générant une discontinuité sur la position du but. Le système poutre/robots est modélisé comme un uni-cycle dont la configuration relative au référentiel monde du système est représentée par ses coordonnées polaires (voir figure 4.1), soit la distance $e$ d'erreur au but, l'orientation $\theta$ du but relative au véhicule dans le référentiel du but, et la direction $\alpha$ du but relative à la direction du déplacement transversal du centre de la poutre ($V_t$). Sous cette paramétrisation de l'espace de travail et en utilisant le modèle simplifié des véhicules, il est alors possible, en utilisant des techniques de Lyapunov, de générer un contrôle en boucle fermée qui garantit la convergence globale du système.

Les équations cinématiques du système dans le nouveau système de coordonnées sont les suivantes :

$$\begin{cases} \dot{e_0} = -V_t \cos(\theta_0 - \varphi_0), \\ \dot{\theta}_0 = V_t \dfrac{\sin \varphi_0}{e_0}, \\ \dot{\varphi}_0 = \omega_0. \end{cases} \tag{4.5}$$

En introduisant la variable $\alpha_0 = \theta_0 - \varphi_0$ comme étant l'angle entre la direction du

robot et la direction de l'objectif, on obtient les équations simplifiées suivantes :

$$\begin{cases} \dot{e_0} = -V_t \cos\alpha_0, \\ \dot{\theta_0} = V_t \dfrac{\sin\alpha_0}{e_0}, \\ \dot{\alpha_0} = -\omega_0 + V_t \dfrac{\sin\alpha_0}{e_0}. \end{cases} \tag{4.6}$$

Pour vérifier la convergence du système vers l'objectif, le critère de stabilité de Lyapunov est employé. Une fonction d'énergie est premièrement donnée. Cette fonction donne une indication de la distance entre la configuration courante et la configuration du but. Elle est définie comme étant la distance quadratique des angles $\alpha_0$ et $\theta_0$ et de la distance à vol d'oiseau au but $e_0$.

$$W = \frac{1}{2}\lambda {e_0}^2 + \frac{1}{2}({\alpha_0}^2 + h{\theta_0}^2). \tag{4.7}$$

Le critère de stabilité exige que $\dot{W} < 0$ pour toutes configurations dans $F_W$. Les lois de contrôle choisies satisfaisant cette contrainte sont :

$$\begin{aligned} V_t &= (\gamma \cos\alpha_0) e_0, \\ \omega_0 &= k\,\alpha_0 + \gamma \tfrac{\cos\alpha_0 sin\alpha_0}{\alpha_0}(\alpha_0 + h\theta_0), \end{aligned} \tag{4.8}$$

où $h > 0$, $\lambda > 0$ et $\gamma > 0$. Le contrôle ainsi généré garantit la convergence du déplacement de la poutre vers la configuration finale.

### 4.3.1 Modèle utilisant le glissement de la poutre

Du fait de l'hypothèse de l'uni-cycle pour modéliser le comportement de la poutre, les lois de contrôle précédentes ne sont valables que dans le cas où le glissement de la poutre est négligeable. Il est possible de modifier la représentation du système de façon à tenir compte du déplacement longitudinal de la poutre. Cette nouvelle représentation prendra le vecteur réel du déplacement de la poutre pour orienter l'uni-cycle (figure 4.2).

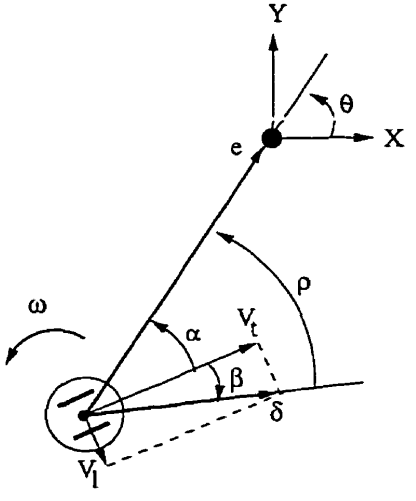



Figure 4.2 : Représentation avec glissement de la poutre

Pour ce, on représente la vitesse réelle $\delta$ du centre de la poutre en fonction de la vitesse latérale et longitudinale de la poutre :

$$\delta = \sqrt{V_l^2 + V_t^2}. \tag{4.9}$$

L'angle entre le vecteur vitesse du centre de la poutre et celui de la vitesse transver-

sale est déterminé par

$$\beta = \arctan \frac{V_l}{V_t}. \tag{4.10}$$

On introduit $\rho$ comme étant la direction du but relative à la direction du déplacement du centre de la poutre, soit $\rho = \alpha - \beta$. En utilisant cette nouvelle représentation, on peut décrire les équations cinématiques du système par le système suivant :

$$\begin{cases} \dot{e} = -\delta \cos \rho, \\ \dot{\theta} = \delta \dfrac{\sin \rho}{e}, \\ \dot{\rho} = -\omega + \delta \dfrac{\sin \rho}{e} + P_{ert}. \end{cases} \tag{4.11}$$

Les perturbations du système se manifestent principalement sur le déplacement latéral de la poutre. Nous pouvons représenter ces perturbations comme étant des variations infligées à l'angle $\beta$. Nous caractérisons ces variations comme étant des perturbations puisque la vitesse longitudinale de la poutre est difficilement contrôlable par les robots. Ces perturbations peuvent être représentées par l'équation :

$$P_{ert} = -\dot{\beta} = \frac{V_l \dot{V}_t - \dot{V}_l V_t}{V_l^2 + V_t^2}. \tag{4.12}$$

En utilisant la fonction de Lyapunov définie comme étant la distance quadratique des angles $\rho_0$ et $\theta_0$ et la distance $e_0$ :

$$W = \frac{1}{2}\lambda e^2 + \frac{1}{2}(\rho^2 + h\theta^2), \tag{4.13}$$

nous cherchons des lois de contrôle sur la vitesse et la vitesse angulaire de la poutre qui auront la caractéristique de satisfaire le critère de stabilité de Lyapunov, définit par (4.14) pour toutes les configurations, excepté celles où $e = 0$.

$$\dot{W} = -\lambda e\delta \cos\rho + \rho(-w + \delta\frac{\sin\rho}{e} + P_{ert}) + h\theta\delta\frac{\sin\rho}{e} < 0. \tag{4.14}$$

Les lois de contrôle choisies, satisfaisant la contrainte de Lyapunov sont les suivantes :

$$\begin{cases} \delta = \gamma e \cos\rho, \\ \omega = \gamma \cos\rho \sin\rho - h\gamma\theta\frac{\cos\rho\sin\rho}{\rho} + \kappa\rho - \frac{V_l\dot{V}_t - \dot{V}_l V_t}{V_t^2 + V_l^2}. \end{cases} \tag{4.15}$$

Il est donc possible d'affirmer que le système converge vers la configuration désirée pour les cas où l'on est capable de produire les commandes définies précédemment. Connaissant, à partir des mesures locales, la vitesse $V_l$ longitudinale de la poutre, il est possible de déterminer la vitesse $V_t$ désirée pour atteindre la vitesse réelle $\delta$ souhaitée pour le centre de la poutre.

$$V_t = sign(\delta)\sqrt{\delta^2 - V_l^2}. \tag{4.16}$$

Vu les contraintes non holonomiques des voitures, il n'est pas toujours possible d'obtenir la commande $\omega$. Soit $r_{min}$ le rayon de courbure minimum des voitures, pour des conditions idéales où il n'y a pas de glissement de la poutre et où les voitures sont perpendiculaires à la poutre, la vitesse angulaire maximum que l'on peut appliquer à

la poutre est :

$$\omega_{max} = \frac{V_t}{r_{min} + \frac{D}{2}}. \tag{4.17}$$

## 4.4 Contrôle de bas niveau

Le contrôleur de la poutre de chaque robot génère un chemin désiré pour le centre de la poutre en se basant sur les informations accessibles par ce robot au sujet de la configuration du but et de sa propre position et orientation par rapport à ce but. Les outils des robots doivent maintenant produire un contrôle qui permettra au centre de la poutre de suivre du mieux possible les déplacements désirés, soit la vitesse $V_t$ et la vitesse angulaire $\omega_0$.

Chaque robot $R_i$ doit maintenant générer un contrôle de bas niveau pour sa direction— angle de braquage $\delta_i$—et sa propulsion—accélération $a_i$. Il doit faire cela en utilisant uniquement les informations du niveau supérieur de contrôle et les informations locales d'orientation $\Psi_i$ relative à la poutre et de distance $D_i$ du centre de la poutre. Le contrôleur de bas niveau doit disposer de capacités d'adaptation pour réagir aux variations dans la dynamique désirée pour la poutre tout en évitant des situations où la configuration de la poutre par rapport au levier rend le contrôle difficile. Cette capacité d'adaptation est aussi étudiée au chapitre 7 lors de la présentation des expérimentations.

En faisait l'hypothèse que $\Psi_i$ est maintenu petit par une procédure de contrôle que l'on présentera plus tard, il est possible d'éliminer des termes de haut niveau

utilisant cette variable, pour finalement obtenir l'équation simplifiée suivante :

$$V_i = V_t + (-1)^i \omega_0 D_i.$$

La rotation instantanée du véhicule doit contribuer à garder l'orientation de la voiture perpendiculaire à la poutre et le plus près possible d'une position nominale $D_{\text{nom}}$ le long de la poutre. Ceci implique que la rotation instantanée $\omega_i$ doit être calculée de manière à garantir une décroissance de $|\Psi_i|$ et de l'erreur sur la position longitudinale $|D_i - D_{\text{nom}}|$.

Les dérivés dans le temps des variables $\Psi_i$ et $D_i$ sont les suivantes :

$$\begin{aligned} \frac{d\Psi_i}{dt} &= \omega_i - \omega_0, \\ \frac{dD_i}{dt} &= (-1)^i(-V_t\Psi_i + \omega_i D_i \Psi_1 - V_l). \end{aligned} \tag{4.18}$$

En utilisant une technique standard de placement des pôles (Kailath, 1980) pour le signal de contrôle $\omega_i$, on peut garantir la décroissance de $|\Psi_i|$ et $|D_{\text{nom}} - D_i|$ vers zéro. La vitesse longitudinale de la poutre $V_l$ peut être traitée comme étant une perturbation externe, mais peut aussi être traitée comme partiellement contrôlable connaissant la forme générale de son comportement qui est :

$$\dot{V}_l = f(\dot{D}_1) - f(\dot{D}_2). \tag{4.19}$$

Dans le but de maintenir $\Psi_i$ petit, un contrôleur linéaire qui tente de minimiser

cette variable est introduit. On obtient finalement les équations de contrôle suivantes :

$$\begin{aligned} a_i &= k_1(V_t - V_i + (-1)^i D_i \omega_0), \\ \omega_i &= \omega_0 + g(D_{\text{nom}} - D_i) + k_2 \Psi_i, \end{aligned} \tag{4.20}$$

où $g(D_{\text{nom}} - D_i)$ est une fonction de gain non-linéaire. Connaissant la vitesse de rotation instantanée du robot et sa vitesse courante, on peut déterminer l'angle de braquage à appliquer à la voiture par l'équation suivante :

$$\delta_i = \arctan\left(\frac{\omega_i L}{V_u}\right). \tag{4.21}$$

## 4.5 Simulations

Les stratégies décrites précédemment pour le contrôle du déplacement coopératif d'une poutre ont été validées par simulation.

### 4.5.1 Contrôle utilisant le modèle de l'uni-cycle

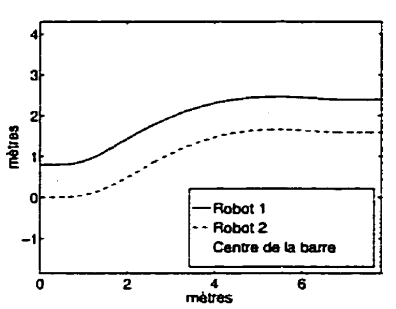
Des simulations ont été faites pour le contrôle développé en utilisant le modèle de l'uni-cycle. Pour ce faire, les lois de contrôle (4.8) sont injectées dans le modèle cinématique de chaque robot donnée par (3.2). Dans l'objectif de rendre la simulation plus réaliste, des bornes sur la vitesse, l'accélération et la vitesse angulaire des roues des robots mobiles ont été fixées conformément aux caractéristiques dynamiques du banc d'essai expérimental (chapitre 6).

Les estimations de la position de l'objectif ont aussi été bruitées pour chaque robot de sorte que ceux-ci ne génèrent pas exactement le même contrôle pour le centre de la poutre. Enfin, nous avons perturbé les estimations de la position et de l'orientation de la poutre sur le levier pour chaque voiture, en ajoutant une composante de bruit blanc aux valeurs réelles de ces variables.

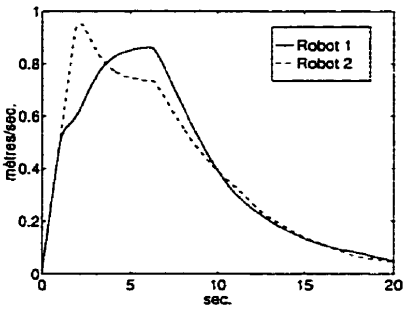
Le contrôleur de bas niveau à tendance à maintenir la valeur de $(D_{\text{nom}} - D_i)$ le plus près de zéro. En pratique une telle approche fait osciller un peu les voitures le long de cet attracteur. Il est préférable de maintenir la valeur de $D_i$ à l'intérieur d'un intervalle de valeurs centré en $D_{\text{nom}}$, définissant ainsi une zone morte pour le contrôleur de bas niveau qui, par le fait même, est non linéaire.

Les figures (4.3,4.4) montrent des simulations de notre système où l'objectif est de placer le centre de la poutre aux coordonnées (8,2) et (2,-12) avec une orientation selon l'axe des $x$.

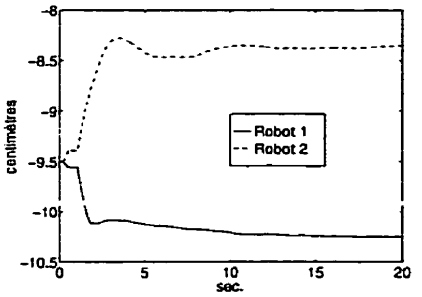
En $(a)$, la position de chaque robot et la position du centre de la poutre sont présentées en coordonnées cartésiennes. La vitesse de chaque voiture est représentée en $(b)$. Cette figure illustre bien comment les voitures adaptent leur vitesse pour se coordonner afin d'effectuer un virage. La position longitudinale de la poutre est présentée en $(c)$. On peut constater qu'au-delà d'un certain écart de la position nominale, les voitures se réorientent pour ramener la poutre dans une configuration plus sûre. En $(d)$, on montre l'orientation de la poutre relative au levier $(\Psi_i = 0)$. En $(e)$, la distance quadratique des coordonnées cartésiennes courantes de la poutre



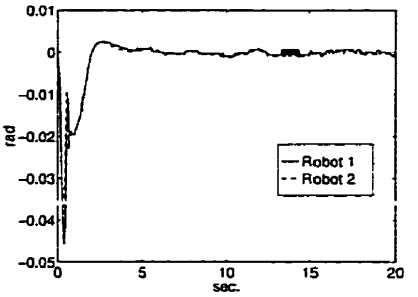

(a) Positions



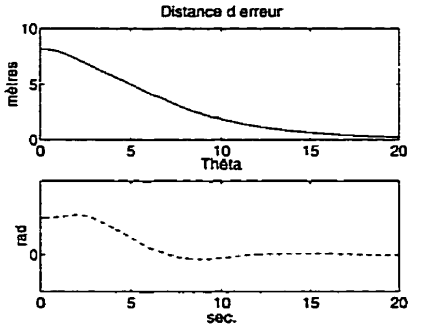

(b) Vitesse




(c) Position longitudinale de la poutre

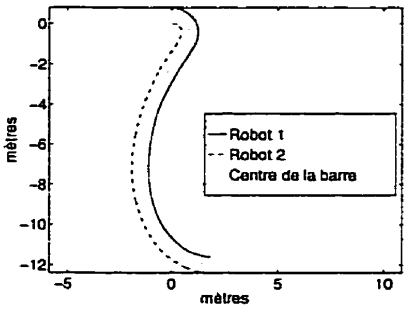



(d) Angle formé par la poutre et la perpendiculaire au levier

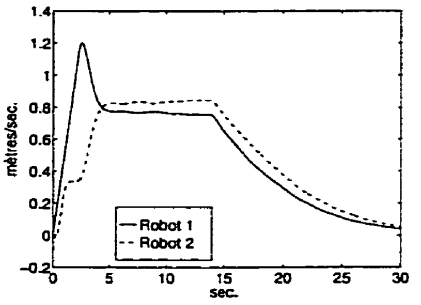



(e) Distance de la poutre à la cible et orientation relative à l'orientation du but

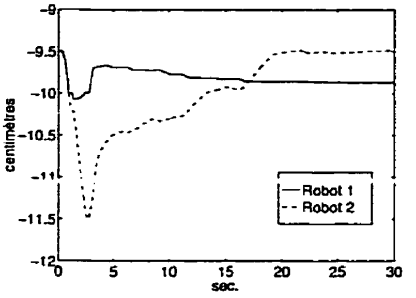
Figure 4.3 : Simulation pour un objectif en (8,2)



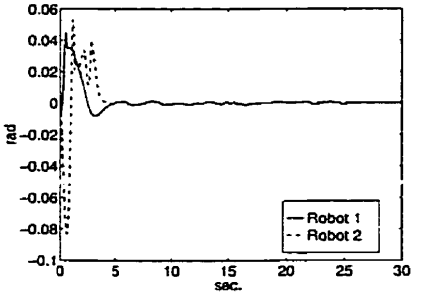

(a) Positions



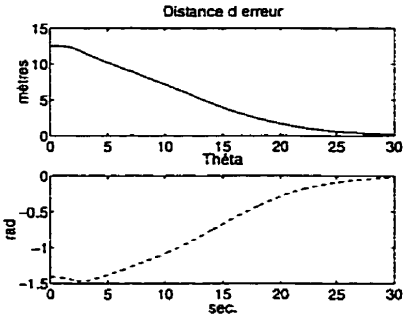

(b) Vitesse




(c) Position longitudinale de la poutre




(d) Angle formé par la poutre et la perpendiculaire au levier




(e) Distance de la poutre à la cible et orientation relative à l'orientation du but

Figure 4.4 : Simulation pour un objectif en (2,-12)

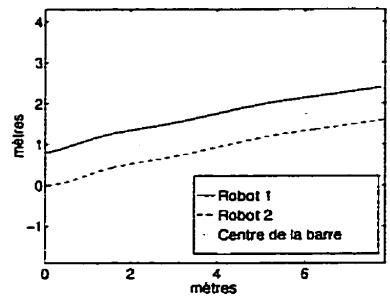
relatives à celles de la configuration finale $e_0$ est présentée, de même que l'angle $\alpha_0$ qui correspond à l'angle entre l'orientation actuelle et l'orientation finale de la poutre.

On peut constater que le système converge bien vers l'objectif tout en gardant la poutre dans une configuration acceptable.

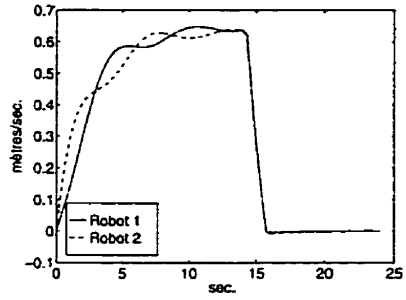
### 4.5.2 Modèle utilisant le glissement de la poutre

Des simulations ont aussi été menées sur le contrôleur utilisant les informations de glissement de la poutre dans le modèle de déplacement désiré de celle-ci. Pour ce, les lois de contrôle (4.15) sont injectées dans le modèle cinématique des robots. La figure 4.5 montre une simulation de notre système où l'objectif est de placer le centre de la poutre aux coordonnées (8,2) et selon l'orientation de l'axe des $x$.

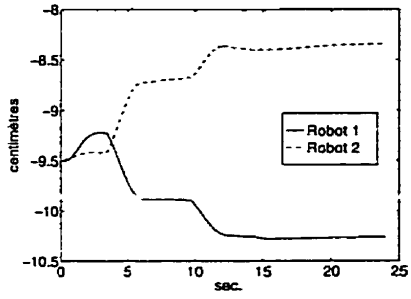
Le comportement du système utilisant ce type de contrôleur pour la poutre donne des résultats moins intéressants que celui ne faisant pas usage de cette information. Ceci peut sembler surprenant, le modèle étant plus complet lorsque l'on utilise cette information de glissement. Le problème rencontré dans cette version du contrôleur est que la trajectoire désirée a beaucoup plus tendance à osciller. De plus, le déplacement latéral de la poutre n'étant pas contrôlable, ces oscillations sont difficiles à stabiliser par les robots.



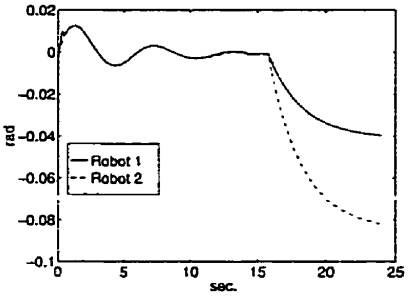

(a) Positions



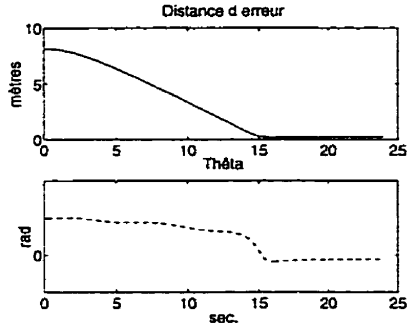

(b) Vitesse




(c) Position longitudinale de la poutre




(d) Angle formé par la poutre et la perpendiculaire au levier




(e) Distance de la poutre à la cible et orientation relative à l'orientation du but

Figure 4.5 : Simulation pour un objectif en (8,2) avec le modèle de frottement

# Chapitre 5

# Évitement d'obstacles

Dans une application réelle, des obstacles imprévus pourraient surgir le long de la trajectoire de la poutre. Si les deux robots détectent l'obstacle à temps, ils peuvent réagir en modifiant temporairement la position de la configuration finale de la poutre. Dans certaines situations, par contre, seulement un des robots observera l'obstacle, rendant ainsi la tâche d'évitement d'obstacles difficile surtout si aucune forme de communication n'est permise entre les deux robots.

Dans ces situations, notre approche pour le contrôle décentralisé donne la possibilité à un robot de modifier favorablement le progrès de la tâche de manière à ce que le système poutre/robots puisse éviter l'obstacle, et cela, sans faire usage de communication entre robots. Il sera démontré qu'en utilisant des règles simples et une connaissance des intentions des autres robots, ce genre de problèmes imprévus peut être résolu sans pour autant compromettre l'accomplissement de la tâche.

Dans ce chapitre, nous discuterons des idées développées pour permettre au système d'éviter les obstacles imprévus. On discutera premièrement de la détection d'obstacles, puis l'on décrira l'algorithme d'évitement proposé. Enfin, des simulations pour diverses situations seront présentées à la fin du chapitre.

## 5.1 Détection d'obstacles

Plusieurs méthodes de détections d'obstacles ont été proposées dans la littérature ces dernières années. Certaines utilisent une caméra et appliquent des méthodes de segmentation du flux optique dans l'image (Nelson et Aloimonos, 1989), d'autres utilisent des télémètres laser (Evans, 1991), ou d'autres type de systèmes à lumière structurée (J.L.Moigne et Waxman, 1988), ou même des sonars (Kuc, 1990) pour n'en nommer que quelques-unes.

Nous ne nous intéresserons pas aux aspects sensoriels de la détection d'obstacles dans ce chapitre. Nous ferons donc l'hypothèse qu'un senseur quelconque de détection d'obstacles est monté à l'avant de chaque robot et dirigé vers l'avant de celui-ci. Le senseur permettra d'évaluer grossièrement la direction d'un obstacle et une approximation de sa distance relative au robot.

Ces senseurs ont généralement la caractéristique commune de pouvoir détecter les obstacles situés à l'intérieur d'un champ de détection donné, et d'une plage de distance au robot donnée. Nous ferons l'hypothèse que le système de détection d'obstacles fonctionne à l'intérieur d'un champ de vision de $2\nu$ degrés, de même qu'à partir d'une

certaine distance $d$ du robot.

En utilisant ces hypothèses sur les capacités de détections des senseurs, on peut diviser l'espace des configurations à l'avant de la poutre, en plusieurs sous régions (figure 5.1). Une première sous région correspond à l'espace visible par les deux robots. Deux autres sous régions correspondent à l'espace visible par seulement un des deux robots. Enfin, une dernière sous région correspond à l'espace non visible par les robots, soit parce que ces positions sont trop près du centre de la poutre et sortent du champ de détection des robots, ou parce qu'elles sont à une distance trop éloignée pour que les robots les observent.

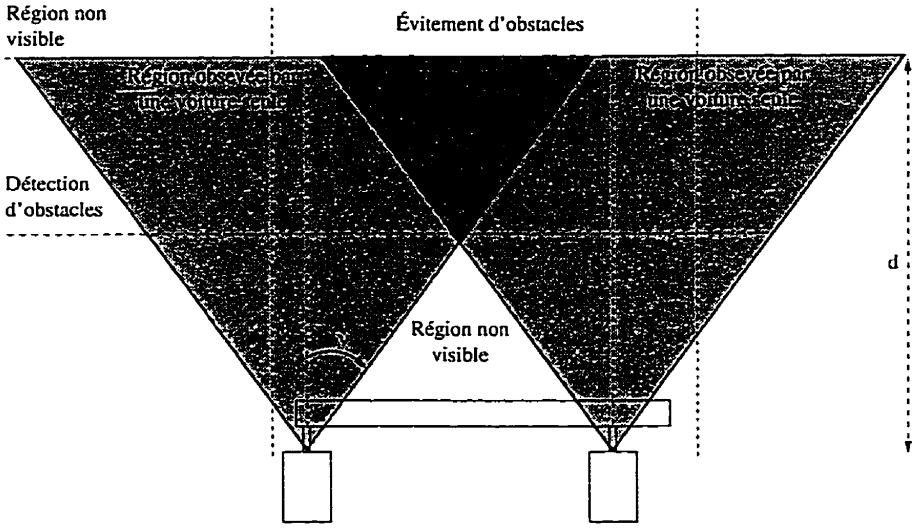



Figure 5.1 : Détection d'un obstacle

Représentons en premier un obstacle comme étant un emplacement ponctuel dans l'espace des configurations, et modélisons la contrainte de non-collision en stipulant que la poutre ne peut franchir ce point dans toute sa longueur. Pour un robot donné,

une procédure d'évitement d'obstacles doit être activée si un obstacle est détecté à l'intérieur d'une région bornée par un couloir, délimité par les extrémités de la poutre (en y ajoutant une certaine marge de sécurité).

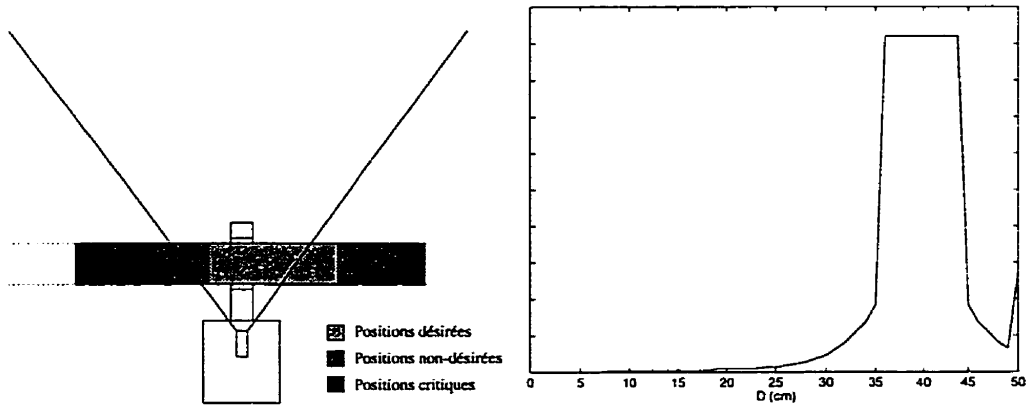
Étant donné que soit un seul soit les deux robots peuvent détecter un éventuel obstacle, l'algorithme d'évitement d'obstacles doit être capable de bien fonctionner pour ces deux cas. Il pourrait être intéressant, pour un robot, d'utiliser une stratégie plutôt qu'une autre selon qu'un ou les deux robots détectent l'obstacle. Pour ce, un robot détectant un obstacle n'aurait qu'à vérifier que l'obstacle se situe bien dans la sous région de l'espace visible par les deux robots. Il est malheureusement difficile de déterminer précisément les limites des différentes sous régions de l'espace des configurations étant donné qu'un robot ne dispose pas de la configuration de son coéquipier. Il n'est donc pas possible, dans l'absolu, de savoir, pour un robot donné, si son coéquipier est lui aussi conscient de l'obstacle sur le chemin.

Sachant à priori que les robots essaient de conserver une certaine configuration relative à la poutre, il est possible d'avoir une idée des chances qu'un obstacle soit détecté par les deux robots. Dans la figure 5.1, on voit que les obstacles situés en ligne avec la région centrale de la poutre ont de bonnes chances d'être détectés par les deux robots.

Deux informations manquent pour qu'un robot puisse connaître la configuration de l'autre robot, soit sa position longitudinale $D_i$ le long de la poutre et son orientation $\Psi_i$ relative à la poutre. Les lignes limites de visibilité de l'autre robot vont varier en

fonction de ces paramètres.

Il est possible de générer des courbes donnant la probabilité qu'un de ces paramètres prenne telle ou telle valeur. Sachant que le contrôle de bas niveau cherche à ramener la poutre à l'intérieur d'une plage de distance $D_i$ (figure 5.2), on peut évaluer, par simulation ou même par expérimentation, la probabilité que la poutre se situe à une position particulière sur le levier de l'autre robot (figure 5.2). Notons qu'à la figure 5.2, on considère que la demi-longueur, $\frac{D}{2}$, de la poutre est de 0.500 mètre. De même il est possible de générer des courbes similaires pour l'orientation $\Psi_i$ de la voiture relative à la poutre (figure 5.3). Ces courbes pourraient éventuellement découler de modèles plus complexes en utilisant d'autres informations, telles que la commande provenant du contrôleur de la poutre, la vitesse de rotation de la poutre, etc.




(a) Plage de distance $D_i$

(b) Probabilité pour les différentes position

Figure 5.2 : Position longitudinale $D_i$ du coéquipier

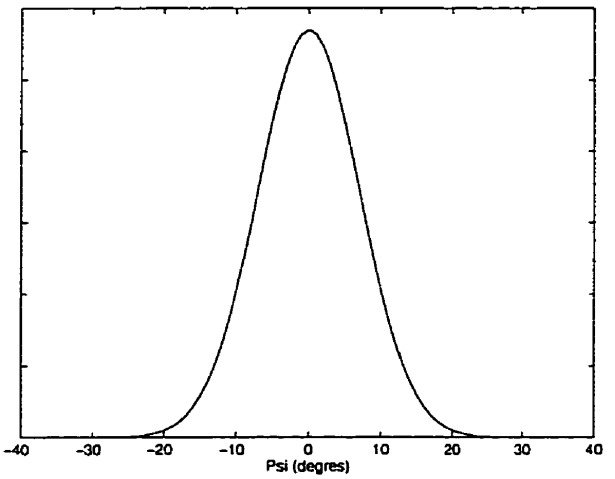


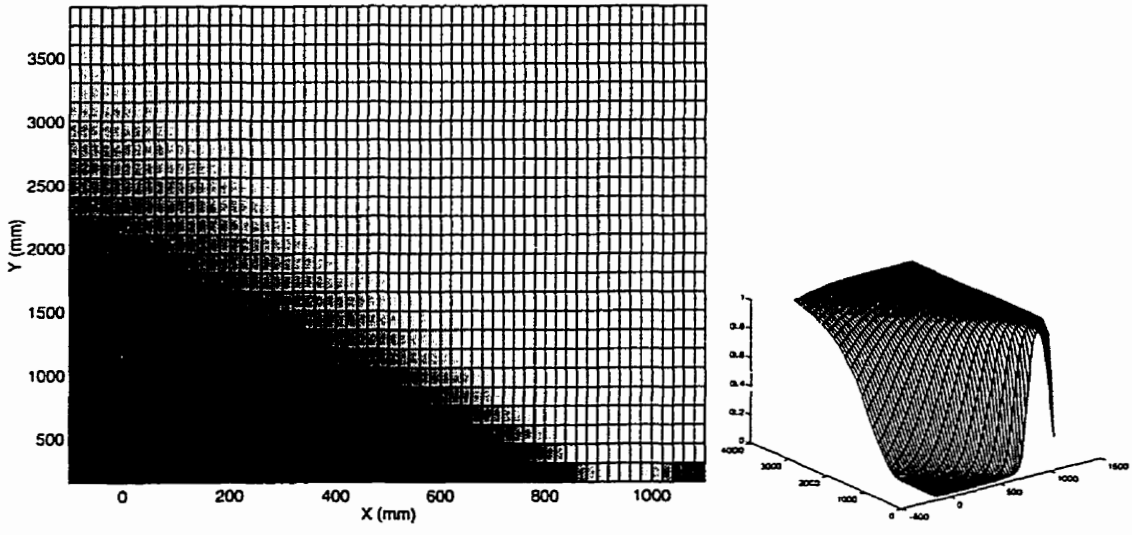

Figure 5.3 : Angle $\Psi_i$ du coéquipier

À l'aide de ces courbes, on évalue, pour un ensemble de positions à l'avant de la poutre, la probabilité qu'un obstacle soit détecté par le coéquipier. Soit un système de coordonnées cartésiennes dont l'axe des $x$ correspond à la poutre dans sa plus grande dimension, l'axe $y$ est dirigé vers l'avant de la poutre et l'origine est située à son extrémité gauche. Pour un point $(x, y)$, on cherche la probabilité que les deux limites de visibilité du robot passent de chaque côté de ce point, et ainsi que la relation (5.1) soit satisfaite :

$$\frac{D}{2} + D_i + \frac{y}{\tan(90 + \nu + \Psi)} < x < \frac{D}{2} + D_i + \frac{y}{\tan(90 - \nu + \Psi)}. \qquad (5.1)$$

Dans la figure 5.4, on présente, au moyen d'une grille de probabilité et de sa représentation tridimentionnelle, la probabilité qu'un obstacle soit détecté par le robot situé à l'extrémité droite de la poutre.

De cette façon, un robot détectant un obstacle a une information supplémentaire




(a) (b)

Figure 5.4 : Probabilité qu'un obstacle soit détecté par la voiture de droite

concernant la probabilité que son coéquipier détecte lui aussi l'obstacle, et peut ainsi modifier sa stratégie d'évitement en conséquence.

## 5.2 Évitement d'obstacles

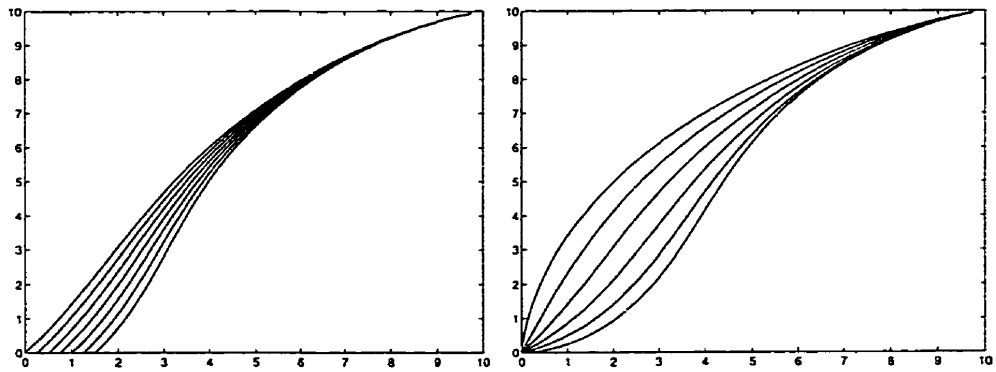
Nous désirons un algorithme d'évitement d'obstacles qui soit applicable dans les cas où l'obstacle est observé par les deux robots simultanément comme dans ceux où seul un des robots l'observe. Ainsi, la connaissance de l'obstacle par les deux robots ne doit pas être nécessaire pour éviter correctement l'obstacle, mais devrait permettre de simplifier la manœuvre.

Pour éviter un obstacle, il est nécessaire d'appliquer des modifications ou de sup-

primer les contrôles générés pour la poutre. Sachant que le coéquipier peut ne pas avoir détecté l'obstacle, l'idée sera d'utiliser les propriétés géométriques de la trajectoire générée par le module de spécification de la tâche, et de modifier la trajectoire de la poutre en perturbant le système d'une manière appropriée.

### 5.2.1 Propriétés des chemins générés pour la poutre

En raison des contraintes non holonomiques que nous avons appliquées artificiellement à la poutre, le système est relativement sensible aux perturbations dans le sens où une variation dans la position ou l'orientation de la poutre à un instant donné résultera en une grande déviation de la trajectoire en comparaison à la trajectoire non perturbée. Ce phénomène est illustré à la figure 5.5.



(a) Chemin pour différentes positions de départ

(b) Chemin pour différentes orientations de départ

Figure 5.5 : Effets d'une perturbation

### 5.2.2 Algorithme d'évitement d'obstacles

Pour éviter un obstacle, un robot doit donc perturber délibérément le système, le forçant ainsi à modifier son chemin. Les effets du frottement de la poutre sur les leviers des robots étant imprévisibles, les déplacements latéraux de la poutre ne peuvent être contrôlés par les robots. Il leur est donc impossible de perturber le système de cette façon. Une solution alternative consiste à perturber le système dans le but de modifier l'orientation de la poutre. Lorsqu'un obstacle le long du chemin de la poutre est perçu par l'une des voitures, celle-ci modifiera temporairement le but de la tâche dans l'objectif de réorienter la poutre afin de forcer le système à dévier de sa route et ainsi se positionner dans une configuration plus sécuritaire.

Le robot déplacera temporairement la configuration finale désirée de la poutre dans une direction perpendiculaire à l'orientation désirée pour la poutre. Les outils des robots étant responsables de s'assurer que la configuration de la poutre relative au levier soit maintenue dans un certain domaine, le système modifiera sa course en s'efforçant de satisfaire les requêtes du robot et en gardant la configuration du système dans un état acceptable. En faisant un choix approprié de la position temporaire de la configuration finale de la poutre, il est possible d'éviter un obstacle même dans le cas où seulement un des robots modifie temporaire sa spécification de tâche.

Lorsqu'un obstacle est détecté, le robot évalue sa position relative. En connaissant la position exacte du centre de la poutre, il est possible de déterminer la direction de l'obstacle relative au centre de la poutre. L'angle $\chi$ formé entre ce vecteur de

direction et l'orientation de la poutre nous permettra de choisir le côté vers lequel le système évitera l'obstacle (figure 5.6).

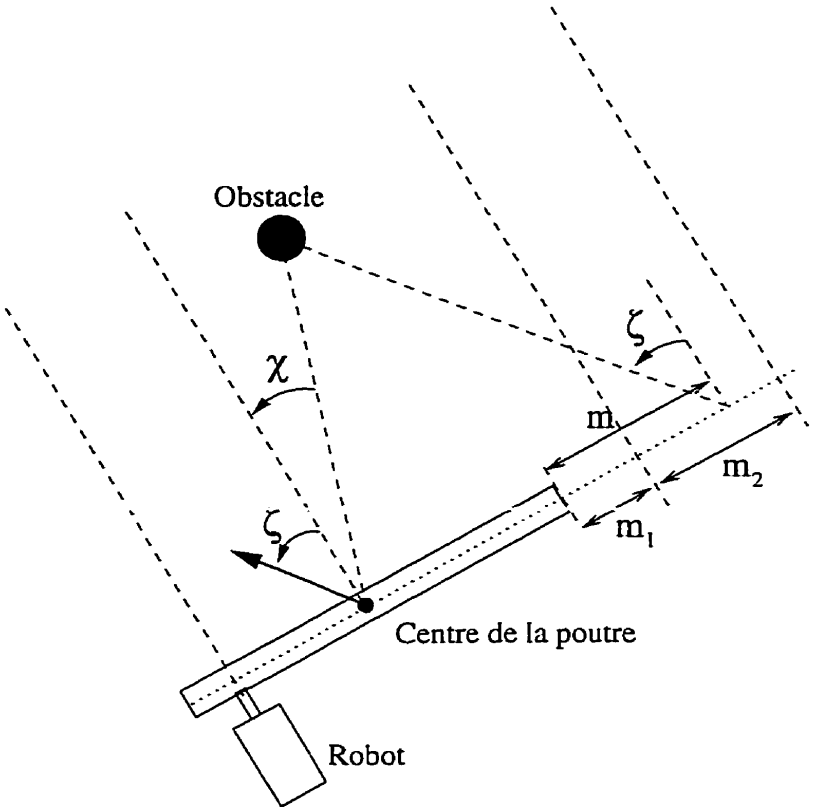



Figure 5.6 : Évitement d'un obstacle

Si l'orientation $\chi$ est positive (négative), le robot amorcera l'évitement d'obstacles par la gauche (droite) de l'obstacle.

Pour des raisons de sécurité, la poutre sera étirée d'une manière virtuelle par une longueur $m$ à l'une de ses extrémités, soit à l'extrémité gauche (droite respectivement) lorsque la manœuvre d'évitement nécessitera un virage vers la droite (gauche respectivement). Cette longueur ajoutée à la poutre sert à pallier aux erreurs provenant entre autres de la vitesse de réaction du système, du glissement de la poutre sur les

leviers et de la possibilité que l'autre robot ne soit pas conscient de l'obstacle.

On calcule ensuite, l'angle $\xi$ entre l'orientation de la poutre et une droite passant par l'extrémité virtuelle de la poutre et par l'obstacle. Cet angle correspondra à la direction désirée pour le centre de la poutre. La spécification de la tâche est enfin temporairement redéfinie en choisissant comme configuration finale désirée du centre de la poutre une position le long d'une droite passant par le centre de la poutre et d'angle $\xi$ et une orientation désirée pour la poutre selon cet angle $\xi$.

La possibilité que les deux robots voient ou non l'obstacle a des répercussions importantes sur la façon dont la poutre évitera l'obstacle. En effet, pour les cas où les deux robots perçoivent l'obstacle, ceux-ci auront tous deux un désir de dévier la poutre de son chemin. Si, par contre, seulement l'un d'eux perçoit l'obstacle, le système aura tendance à converger vers une direction médiane entre les deux objectifs des robots. Pour que le système puisse s'adapter aux différentes situations, évitant correctement l'obstacle lorsque le coéquipier ne le voit pas, tout en empêchant que le système évite inutilement un obstacle non menaçant dans le cas où les deux robots ont détecté ce dernier, nous modifions la valeur de $m$ selon la probabilité que l'autre voiture a de voir l'obstacle. La dimension $m$ est ainsi calculée par l'équation suivante :

$$m = m1 + m2 * P(x, y), \tag{5.2}$$

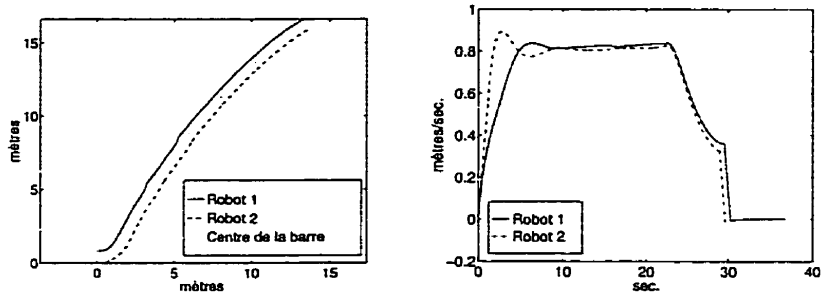
avec $P(x, y)$ étant la probabilité que l'obstacle situé en $(x, y)$ soit détecté par l'autre voiture.

## 5.3 Simulations

Des simulations ont été faites pour évaluer la faisabilité de cette procédure d'évitement d'obstacles décentralisée. On considère que les obstacles ne peuvent être détectés que s'ils se trouvent face au robot. On considère aussi que la détection de l'obstacle est faite à partir d'une certaine distance de celle-ci, et que les robots n'ont pas connaissance de cet obstacle lorsqu'ils en sont trop éloignés.

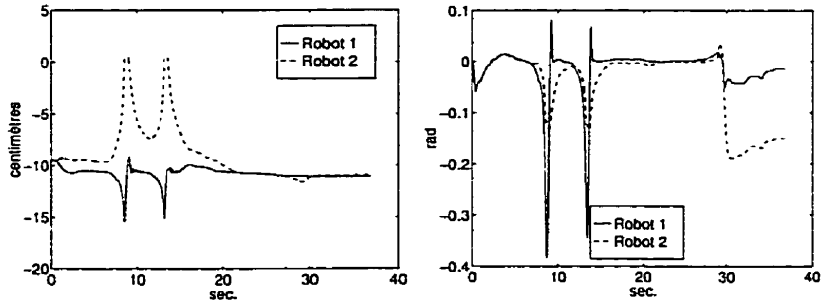
La configuration finale de la poutre pour ces simulations se situe aux coordonnées cartésiennes (14,16) avec une orientation selon l'axe des $x$. Dans la figure 5.7, on présente une simulation où il n'y a aucun obstacle dans l'espace de travail. Lorsque les deux robots perçoivent l'obstacle, le système arrive facilement à modifier sa trajectoire [1] comme illustré à la figure 5.8. Dans la figure 5.9, on présente une situation où seulement l'un des deux robots a détecté l'obstacle. Dans le cas illustré, seul le robot de gauche détecte l'obstacle et réussit à dévier le robot de droite vers la gauche pour éviter correctement l'obstacle.

[1] Il est bon de noter que l'obstacle peut être placé artificiellement de telle sorte (*e.g.* aligner avec la configuration finale de la poutre et avec une orientation donnée pour la poutre) que même si la poutre est détectée par les deux robots, l'obstacle ne pourra être évité par cette procédure. Pour surmonter cette difficulté, il serait envisageable d'insérer une asymétrie dans la stratégie de contrôle des deux robots.




(a) Positions

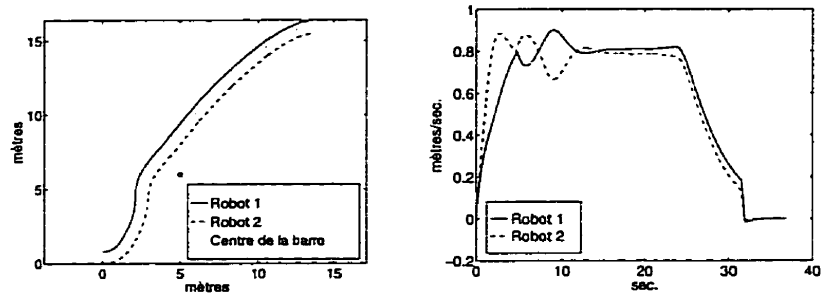
(b) Vitesse




(c) Position longitudinale de la poutre

(d) Angle formé par la poutre et la perpendiculaire au levier

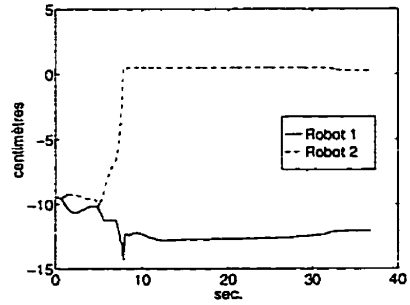
Figure 5.7 : Simulation pour objectif en (14,16) sans obstacle



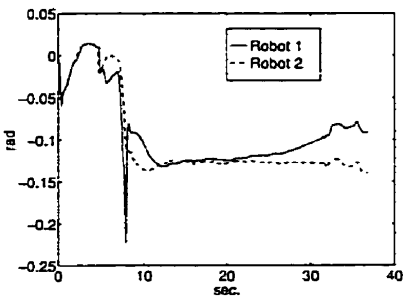

(a) Positions




(b) Vitesse

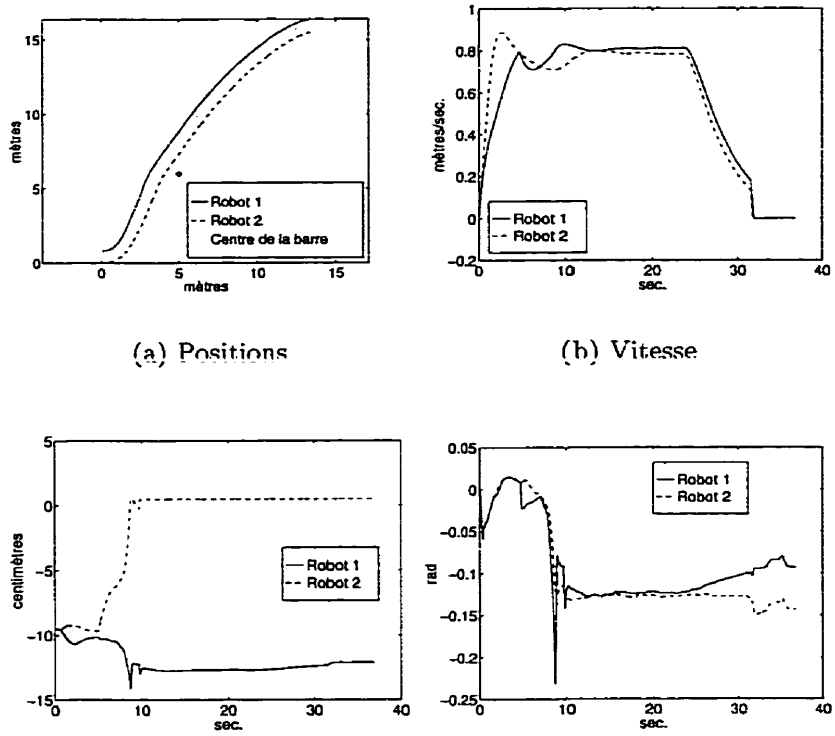



(c) Position longitudinale de la poutre




(d) Angle formé par la poutre et la perpendiculaire au levier

Figure 5.8 : Simulation pour objectif en (14,16) avec détection et évitement de l'obstacle par les deux robots




(a) Positions

(b) Vitesse

(c) Position longitudinale de la poutre

(d) Angle formé par la poutre et la perpendiculaire au levier

Figure 5.9 : Simulation pour objectif en (14,16) avec détection et évitement de l'obstacle par le robot de gauche

# Chapitre 6

# Analyse sensorielle

Des efforts ont été investis afin d'implanter l'application de transport d'une poutre décrite précédemment sur un banc d'essai expérimental. Ce chapitre et le chapitre suivant présenteront la description de l'implantation de cette application sur un banc d'essai, ainsi que des résultats obtenus suites aux expérimentations.

Dans ce chapitre, nous décrirons les méthodes utilisées pour analyser les images prises par la caméra montée sur les robots, afin d'extraire les informations nécessaires au contrôle des robots mobiles dans l'accomplissement de leur tâche de transport. Notons que les algorithmes développés ici ne constituent pas des méthodes générales d'extraction de l'information dans une image mais plutôt des solutions efficaces et peu complexes appliquées à un problème très spécifique.

## 6.1 Informations nécessaires au contrôle

Pour intervenir judicieusement sur la tâche, les robots doivent posséder des informations sur l'état actuel de la tâche, de même que sur leur configuration par rapport à celle de la poutre. Ces informations sont fournies au robot par l'entremise de leurs senseurs et correspondent à l'état observable $RT_i$ du robot. Nous considérerons dans ce chapitre que chaque robot connaît son état $R_i$ en tout temps, grâce à l'utilisation d'un moyen adéquat, tel qu'une combinaison GPS/Encodeurs optiques.

Pour déduire l'état actuel de la tâche, chaque robot cherche à obtenir une bonne approximation de la valeur de l'orientation $\Psi$ de la poutre et de la position longitudinale $L$ de celle-ci par rapport au levier. Pour ce faire, une caméra CCD est montée à l'avant du robot sur une plate-forme "pan" et "tilt". Pour observer la poutre, la caméra est dirigée vers l'avant avec un angle $\beta$ relativement à la ligne d'horizon. Les informations $\psi$ et $l$ sont extraites de l'image obtenue par la caméra tel qu'illustré à la figure 6.1.

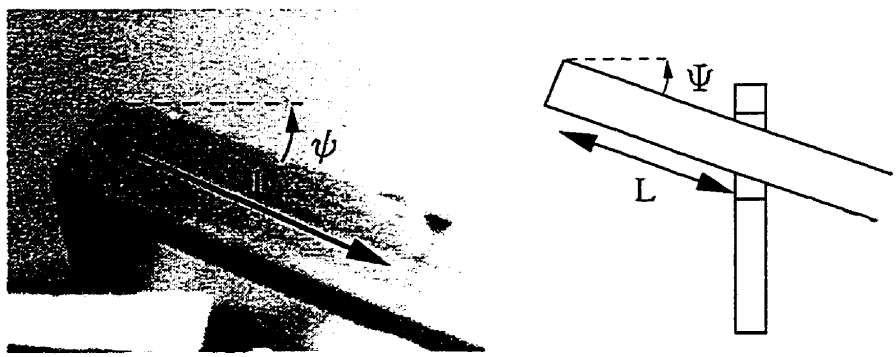



Figure 6.1 : (a) Image obtenue par la caméra, (b) Informations visuelles extraites

L'obtention d'un contrôle stable, réagissant rapidement aux changements dynamiques du système, demande que l'analyse d'images soit exécutée d'une manière efficace. Le choix de l'algorithme est principalement restreint par une contrainte de vitesse de calcul afin d'obtenir un taux de rafraîchissement élevé des informations extraites de l'image, soit de l'ordre des 30 images par seconde. Il n'est donc pas possible d'envisager une analyse exhaustive de l'image. Une recherche locale et minime doit être exécutée. Ainsi, nous croyons que l'usage de masques de convolution complexes pour la détection de la poutre requiert trop de calcul pour nos contraintes de temps.

Une solution envisageable pour déterminer l'orientation de la poutre consisterait simplement à extraire des points le long de l'une des bordures longitudinales de la poutre et d'évaluer l'orientation à partir de ces points. De même nous pourrions envisager une solution semblable pour inférer la position latérale de la poutre, en faisant l'extraction de la position de l'extrémité de la poutre dans l'image. L'inconvénient avec cette dernière solution est que le champ de vision de la caméra est restreint et qu'il est possible qu'à certains moments, l'extrémité de la poutre sorte de ce champ. Sachant que la caméra est montée sur une plate-forme 'pan' et 'tilt', nous pourrions suivre l'extrémité de la poutre en modifiant de façon dynamique l'orientation de la caméra. Par contre, une telle solution exigerait une calibration précise de la plate-forme afin de pouvoir faire une correspondance entre les points dans l'image et ceux dans le monde. Une telle précision dans la calibration est impossible à obtenir pour le type de plate-forme dont nous disposons, ce qui implique que la caméra doit donc

être maintenue dans une position fixe.

Pour contourner cette difficulté nous pourrions imaginer des solutions qui consisteraient à évaluer les déplacements de la poutre sur le levier à l'aide d'une méthode faisant usage du champ de mouvement par exemple, et à intégrer la position de la poutre. Mais l'accumulation d'erreurs pour un traitement de ce genre donnerait vite une très mauvaise approximation de la position de la poutre. Finalement, pour cette application, deux solutions ont été retenues, chacune avec leurs avantages et leurs inconvénients propres.

Une des approches consiste à peindre des lignes noires sur la poutre dans la direction transversale à celle-ci. Ces lignes sont espacées de façon irrégulière de sorte que la distance entre deux lignes adjacentes correspond à une position de ces lignes sur la poutre (figure 6.2). Ainsi, en connaissant la position d'une ligne dans l'image et la position correspondante de la ligne sur la poutre il est aisé d'inférer la position de la poutre relative au levier. Cette solution a, par contre, l'inconvénient d'exiger que l'on dessine sur la poutre des marques à des positions spécifiques, ce qui diminue la généralité de la solution.

La deuxième approche consiste simplement à chercher dans l'image l'extrémité de la poutre, en avertissant éventuellement le contrôleur que la position de la poutre est supérieure à une valeur limite lorsque l'extrémité est hors du champ de vision de la caméra. Cette méthode à l'avantage de ne pas faire usage de repères artificiels pour bien fonctionner et d'être ainsi plus applicable à des problèmes industriels. Elle a par

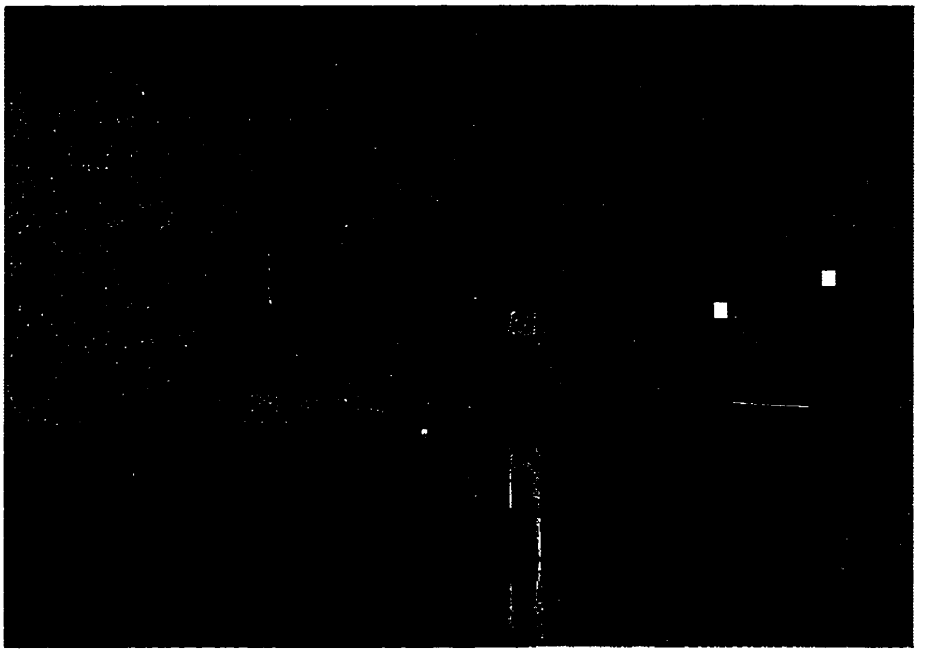


Figure 6.2 : Analyse d'une image

contre l'inconvénient de permettre l'extraction d'une plage limitée de valeur pour la position latérale de la poutre.

Ces deux solutions ont été implantées et seront décrites en détail dans les deux sections qui suivent.

## 6.2 Utilisation de repères artificiels sur la poutre

Pour la première solution, l'algorithme d'analyse d'images doit être en mesure de chercher deux lignes noires adjacentes sur la poutre. La position des lignes détectées doit être déterminée de façon identique dans l'image pour chaque ligne afin qu'il y ait concordance dans les valeurs de distance calculées. On caractérisera ainsi chaque ligne

par la position de son coin supérieur droit. On utilisera une prédiction de premier ordre (l'ancienne configuration de la poutre et des lignes détectées) pour accélérer la recherche dans l'image. La recherche s'effectuera au moyen d'une séquence de masques appliqués sur des petites régions de l'image.

La détermination de la position/orientation de la poutre dans l'image est faite en trois étapes distinctes soit la recherche des lignes noires, la détermination de la position du coin supérieur droit des lignes et la caractérisation de la configuration de la poutre à partir des positions de ces lignes.

### 6.2.1 Recherche des lignes dans l'image

À chaque analyse, deux lignes doivent être trouvées dans l'image pour inférer la configuration de la poutre. En prenant pour hypothèse que la poutre ne s'est pas déplacée de façon marquée depuis la dernière itération, on débute la recherche de la première ligne à partir de la position d'une des lignes trouvées lors de l'itération précédente. De même on considère que l'orientation de la poutre n'a pas changé.

Ainsi, la recherche de la première ligne consiste à faire un balayage vers la droite le long d'une ligne ayant la même orientation que l'ancienne orientation de la poutre. On débute cette recherche sur un point situé à la gauche de la ligne la plus à gauche des deux lignes trouvées lors de la dernière itération. Le balayage se fait en vérifiant chaque pixel le long de la ligne d'une manière séquentielle en s'arrêtant sur les pixels ayant une intensité inférieure à un certain seuil.

Lorsque l'on rencontre un pixel satisfaisant la condition de seuil, on applique un masque pour vérifier si ce pixel fait parti d'une région contenant plusieurs pixels où l'intensité satisfait la contrainte de seuil. Dans un pareil cas, on peut affirmer qu'une ligne a été trouvée. Si par contre le masque ne satisfait pas la valeur de seuil, on continue la recherche. La recherche se termine si le balayage dépasse les limites de recherche dans l'image ou si une ligne est trouvée.

Dans le cas où aucune ligne n'est trouvée, on applique la recherche vers la gauche selon la même orientation et en partant du même point de départ. Si, pour cette image, l'orientation de la poutre a changé considérablement, il est possible de ne pas avoir rencontré de ligne durant cette opération. On exécute alors des balayages successifs aux alentours de l'orientation de la poutre de la dernière itération en prenant comme point initial la position du centre du levier du robot. Lorsqu'une ligne est trouvée, on détermine la position de son coin supérieur droit.

On recherche ensuite la deuxième ligne en exécutant des balayages aux alentours de la première ligne trouvée et en utilisant une approche semblable à celle décrite précédemment. La différence étant que la recherche débute toujours dans un voisinage immédiat de la ligne précédemment trouvée afin de s'assurer que les deux lignes sont adjacentes.

### 6.2.2 Détermination de la position de la ligne dans l'image

Une fois qu'une ligne est trouvée, on cherche à déterminer la position de son coin supérieur droit afin de bien la positionner relativement au levier. Pour ce, on longe cette ligne vers le haut de la bordure droite de la ligne trouvée en appliquant successivement deux masques (figure 6.3).

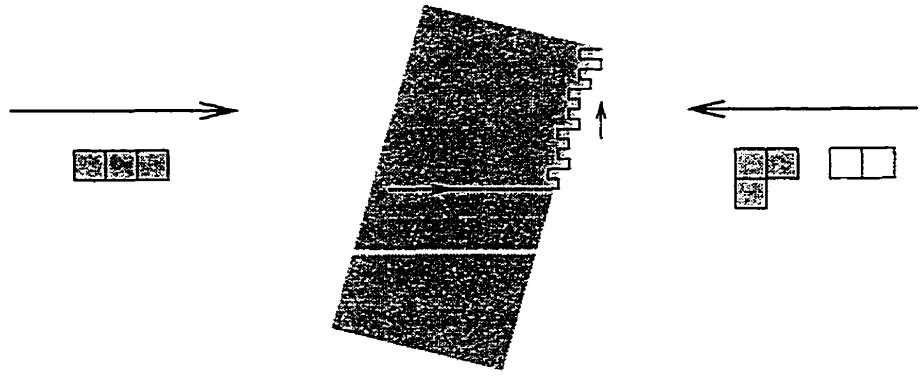


Figure 6.3 : Suivie de la bordure

À chaque itération, on se déplace d'un pixel vers le haut de l'image. Dans le cas où l'intensité du pixel est inférieure à une valeur de seuil, on peut affirmer que l'on est à l'intérieur de la ligne noire. Dans pareil cas, on se déplace vers la droite dans l'image jusqu'à ce que l'on atteigne la bordure de la ligne. Si, par contre, la valeur de l'intensité est supérieure au seuil, on considère que l'on est à la droite de la ligne noire. On se déplace alors vers la gauche de l'image en appliquant un masque de détection de bordure jusqu'à ce qu'une bordure soit détectée. La recherche du point se termine lorsque la recherche d'une bordure vers la gauche demande trop d'itération. On considère alors que la position du sommet est trouvée.

### 6.2.3 Caractérisation de la configuration de la poutre

Une fois que la position des lignes est obtenue, il faut déterminer les informations de positions et d'orientation de la poutre relativement au levier. En réorganisant les deux points dans l'image, correspondant à la position des lignes, de façon à ce que le premier point se situe à la gauche du deuxième, il est possible d'évaluer l'orientation de la poutre dans l'image grâce à l'équation (6.1) :

$$\psi = \arctan\left(\frac{-(Y_{point2} - Y_{point1})}{X_{point2} - X_{point1}}\right). \tag{6.1}$$

Les coordonnées tridimensionnelles des points dans le monde sont obtenues en appliquant une déprojection des points dans l'image. Les paramètres intrinsèques et extrinsèques de la caméra, nécessaires pour cette déprojection, peuvent être obtenus par l'algorithme d'étalonnage de Tsai-Wilson (Wilson, 1994).

L'étalonnage de la caméra est effectué en disposant au préalable des cibles sur le sol suivant une disposition connue, et en extrayant la position de ces cibles dans l'image. Un logiciel d'étalonnage fournit alors les paramètres permettant de faire la correspondance entre les points dans l'image et les positions tridimensionnelles relatives à la caméra. La connaissance de la hauteur du point dans le monde est requise pour la déprojection.

Lorsque l'extraction de la position des points tridimensionnels est terminée, on calcule l'orientation de la poutre dans le monde à l'aide de l'équation (6.1). Pour

déterminer la position de ces deux lignes sur la poutre, on évalue la distance entre les deux lignes dans le monde par l'équation (6.2) :

$$k = \sqrt{((Y_{point2} - Y_{point1})^2 + (X_{point2} - X_{point1})^2)}. \tag{6.2}$$

Les lignes sont espacées sur la poutre de façon à ce que l'information de distance entre les deux lignes nous permette de connaître la position de ces lignes sur la poutre. Pour limiter les risques d'erreur reliés à la détermination de la distance entre les lignes, l'espacement entre les lignes a été choisi de façon à ce que l'incrément d'espace soit très grand. Ce choix a comme inconvénient de ne permettre qu'un nombre restreint de distance entre deux lignes données. Pour remédier à ce problème, on recopie plusieurs fois le motif d'espacement entre les lignes sur un même coté de la poutre (figure 6.4). De cette façon, un espacement entre deux lignes correspond à plusieurs positions sur la poutre. En prenant pour hypothèse que la poutre ne se déplace pas d'une grande distance entre deux itérations, on peut, en utilisant une table "look-up", retrouver la position de cette paire de lignes sur la poutre.

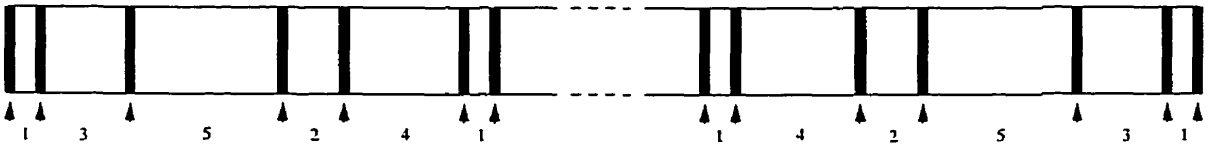



Figure 6.4 : Repères sur la poutre

La méthode consiste donc à calculer, premièrement, une approximation de la position de la ligne la plus rapprochée de l'extrémité de la poutre, en supposant que

la poutre ne s'est pas déplacée longitudinalement depuis la dernière itération grâce à l'équation suivante (6.3) :

$$\textit{Position ligne sur poutre} = \textit{Ancienne position de poutre} - \textit{Distance ligne au levier}. \tag{6.3}$$

On associe, ensuite, au tableau de correspondance, la distance entre les deux lignes trouvées de manière à ce que la position de la ligne extrême correspondant dans le tableau s'approche du mieux possible de celle approximée précédemment. La distance du levier à l'extrémité de la poutre peut être facilement évaluée en faisant la somme de la distance de la ligne extrême au levier et de cette même ligne à l'extrémité de la poutre.

L'utilisation de ces motifs d'espacement répétés sur la poutre n'est possible que si les déplacements de la poutre ne dépassent pas une valeur limite entre deux itérations. Pour un motif d'espacement de $p$ mètres, la poutre ne doit jamais faire un déplacement longitudinal de plus de $\frac{p}{2}$ mètres entre deux itérations. Dans notre implantation, le motif d'espacement est de 0.15 mètre. En considérant une analyse d'image faite à un taux supérieur à 10 images/seconde, la poutre peut se déplacer à une vitesse de près de 0.75 mètre/seconde sans que le système d'analyse perde trace de la position de la poutre. Ceci est amplement suffisant puisque, pour une telle vitesse de la poutre, aucun contrôle ne permettrait de stabiliser le système.

La méthode de détermination des positions des points dans l'image étant basée sur

l'intensité des pixels, la précision de cette localisation est limitée. En effet, la position peut s'écarter quelque peu de la position optimale. De plus, cette localisation étant donnée en terme de pixels dans l'image, elle est limitée par la taille des pixels. Pour augmenter la précision, une procédure permettant d'aller chercher les coordonnées d'une troisième ligne noire sur la poutre a été ajoutée (figure 6.5).

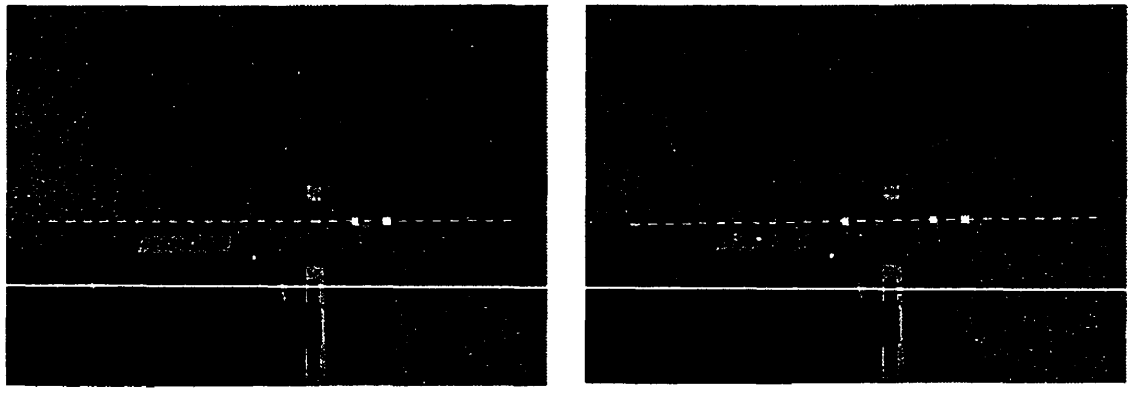


Figure 6.5 : Amélioration de la précision des données

En utilisant l'orientation déjà évaluée à l'aide des deux points, on recherche un troisième point de chaque côté des lignes déjà trouvées. Si une troisième ligne est trouvée, on réévalue l'orientation de la poutre en trouvant la droite de la forme $y = ax + b$ qui minimise l'écart quadratique à ces points (6.4) :

$$\begin{aligned} a &= \frac{\sum_i x_i y_i - \frac{1}{N}\sum_i \sum_j x_i y_j}{\sum_i x_i^2 - \frac{1}{N}\sum_i \sum_j x_i x_j}, \qquad N = 3, \\ b &= -\sum_i a x_i - y_i. \end{aligned} \tag{6.4}$$

L'angle formé par la poutre et le levier peut être ainsi calculé par $\Psi = \arctan(a)$.

De la même manière, il est possible d'améliorer l'évaluation de la position longitudinale de la poutre relativement au levier. Pour ce, on projette les points sur la droite $y = ax + b$ grâce aux équations (6.5) :

$$\begin{aligned} x^* &= \frac{\tilde{x} + a\tilde{y} - ab}{1 + a^2}, \\ y^* &= \frac{b + a\tilde{x} + a^2\tilde{y}}{1 + a^2}. \end{aligned} \tag{6.5}$$

Connaissant les distances exactes entre les lignes, on minimise l'écart entre les positions actuelles des points le long de la droite et une configuration de points où le bon espacement entre les points est respecté. Pour un ensemble de trois points dont les coordonnées en x sont $\tilde{x_a}$, $\tilde{x_b}$ et $\tilde{x_c}$, et pour un espacement entre les points de $D_{ab}$ et $D_{bc}$, on peut calculer les coordonnées corrigées par (6.7) :

$$\begin{aligned} x_a &= \frac{x_a^* + x_b^* + x_c^* - (D_{ab} + D_{bc})\cos(\Psi)}{3}, \\ x_b &= x_a + D_{ab}\cos(\Psi), \\ x_c &= x_a + (D_{ab} + D_{bc})\cos(\Psi), \\ y_i &= ax_i + b. \end{aligned} \tag{6.6}$$

Cette recherche d'une troisième ligne pouvant être exigeante en temps de calcul, elle ne sera exécutée que lorsque le temps le permet.

## 6.3 Extraction de l'extrémité

La deuxième solution consiste à extraire de l'image la bordure supérieure de la poutre de même que la position de l'extrémité de la poutre. Ces informations devraient nous permettre de générer les paramètres d'orientation $\Psi$ et de position $L$, nécessaires au contrôle.

Afin de bien distinguer la poutre des autres détails de l'image, nous avons peint celle-ci en noir (figure 6.6). La détermination de la position/orientation de la poutre dans l'image se fait en quatre étapes distinctes soit l'acquisition des points sur la bordure supérieure de la poutre, l'évaluation de son orientation, l'acquisition des points sur la bordure de l'extrémité de la poutre et l'évaluation de la position latérale de la poutre.

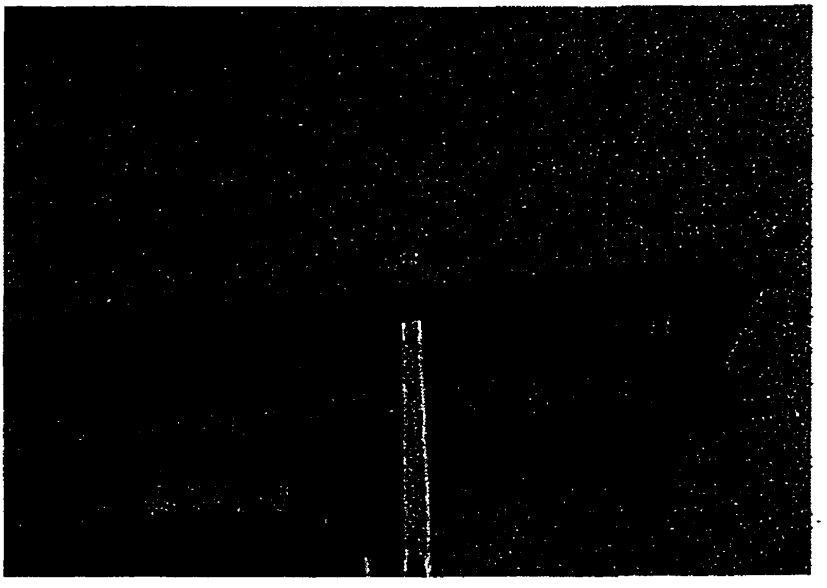


Figure 6.6 : Image de la poutre

### 6.3.1 Acquisition des points sur la bordure supérieure de la poutre

On recherche en premier un point faisant partie de la bordure supérieure de la poutre dans l'image. Connaissant à priori la position de l'extrémité du levier dans l'image, on exécute un balayage verticalement de cette position vers le bas de l'image en appliquant un masque de convolution (de détection de bordure) tel que montré à la figure 6.7. Pour vérifier que la bordure détectée ne soit pas une ombre ou une autre caractéristique de l'image, on vérifie à l'aide d'un autre masque de convolution qu'il s'agit bien de la bordure de la poutre. Ce dernier masque vérifie si une grande surface noire se situe sous la bordure détectée. La position du point situé dans le réceptacle du levier et donnant la plus grande valeur pour le masque de détection de bordure (tout en satisfaisant la contrainte du deuxième masque) est considérée comme étant un point valide appartenant à la bordure supérieure de la poutre.

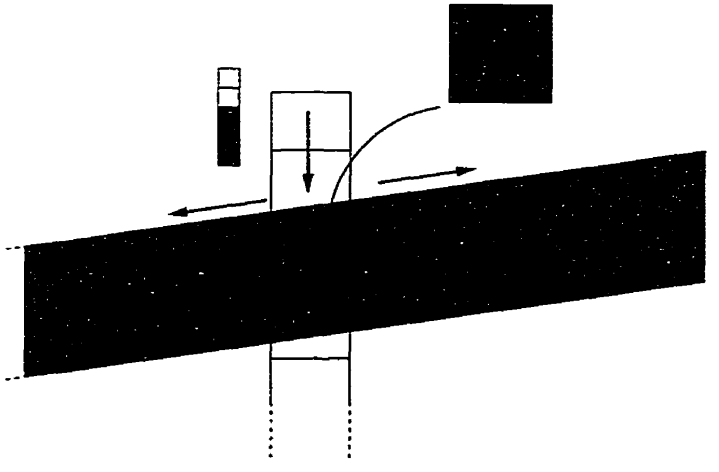


Figure 6.7 : Acquisition des points de la bordure supérieure de la poutre

On fait ensuite le suivi de la bordure de la poutre en conservant chaque point trouvé dans une liste de points. La recherche des points s'effectue en balayant l'image de ce premier point trouvé sur la bordure vers le centre de la poutre puis de ce même point vers l'extrémité de la poutre. Pour chaque coordonnée en $x$, on applique un masque de convolution tel qu'illustré dans la figure 6.7, le long d'un intervalle de coordonnées en $y$. L'emplacement de cet intervalle est choisi en faisant au préalable une prédiction sur l'emplacement du prochain point sur la bordure. Encore une fois, on considère que la bordure est située à la coordonnée $y$ pour laquelle la valeur du masque est la plus élevée.

Pour guider la recherche des points le long de la poutre, on prédit la position du prochain point de la bordure au fur et à mesure que l'on fait l'acquisition des points. Cette prédiction a deux utilités : elle permet premièrement de réduire la zone de recherche du prochain point, ce qui accélère grandement le traitement de l'image; elle permet aussi de déterminer le moment où l'on a atteint l'extrémité de la poutre. La prédiction peut se faire de multiples façons, en utilisant, par exemple, des filtres prédictifs tels que le filtre de Kalman. En utilisant de telles méthodes, la prédiction pourrait servir à elle seule à évaluer l'orientation de la poutre d'une façon suffisamment précise. Pour éviter des imprécisions aux extrémités de la poutre, nous utilisons plutôt une approximation simple de l'orientation par intégration comme représenté dans l'équation (6.7) :

$$\tilde{a}_i = \left(1 - \frac{\zeta}{\zeta + i}\right) \tilde{a}_{i-1} + \frac{\zeta}{\zeta + i} \frac{(y_0 - y_i)}{(x_0 - x_i)}. \tag{6.7}$$

où $\psi = \arctan(a)$. En utilisant cette méthode grossière mais rapide pour évaluer l'orientation de la poutre dans l'image, il est possible de prédire l'emplacement du prochain point de la bordure dans l'image.

Pour certaines configurations de la poutre (figure 6.8), la méthode utilisée pour l'acquisition des points le long de la bordure considérera les points du côté de l'extrémité de la poutre comme faisant parti de sa bordure supérieure de la poutre. Pour contrecarrer ce problème, on vérifiera que chaque point trouvé ne s'éloigne pas de l'emplacement prédit. Lorsqu'une suite de plusieurs points dévie de l'emplacement prédit, on considère que l'on a atteint l'extrémité de la poutre.

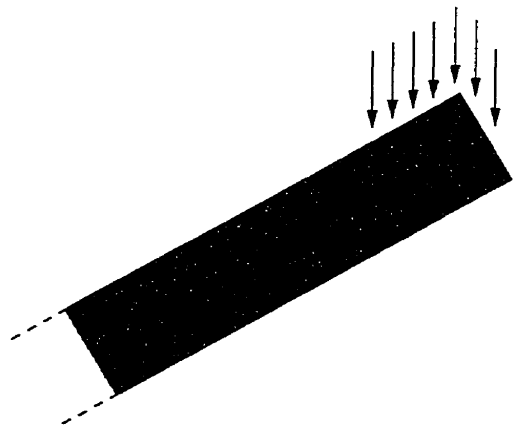


Figure 6.8 : Configurations problématiques

### 6.3.2 Calcule de l'orientation de la poutre

Une fois que la liste de points est obtenue, il nous faut déterminer l'orientation de la poutre par rapport au levier dans l'espace réel. L'évaluation de l'orientation de la poutre peut se faire par une minimisation par les moindres carrés. L'équation de la droite $y = ax + b$ minimisant l'écart quadratique à ces $N$ points est obtenue par les équations (6.8) :

$$\begin{aligned} a &= \frac{\sum_i x_i y_i - \frac{1}{N}\sum_i \sum_j x_i y_j}{\sum_i x_i^2 - \frac{1}{N}\sum_i \sum_j x_i x_j}, \\ b &= -\sum_i a x_i - y_i. \end{aligned} \tag{6.8}$$

L'angle formé par la poutre et le levier peut être ainsi calculé par $\Psi = \arctan(a)$. L'ordre de léquation de minimisation étant d'ordre $O(n^2)$, nous limiterons les calculs en simplifiant l'évaluation de la droite. Le calcul se fera en utilisant la moyenne des $k$ droites calculées à partir de triplets de points pris dans la liste. Ainsi, la complexité de calcul diminuera de l'ordre de $O(n)$. L'équation de la droite pour les $N$ points d'une liste $L$ devient donc :

$$\begin{aligned} a &= \frac{1}{k}\sum_k a_j, \\ b &= \frac{1}{k}\sum_k b_j. \end{aligned}$$

L'orientation de la poutre peut être ainsi calculée par une liste de points. Pour que cette orientation corresponde à l'orientation réelle de la poutre dans le monde, les coordonnées tridimensionnelles des points dans le monde doivent être obtenues en appliquant une déprojection des points dans l'image. Cette opération est remplie par la calibration préalable de la caméra au moyen de l'algorithme de Tsai-Wilson (Wilson, 1994). Dans la figure 6.9, la droite calculée est affichée sur l'image.

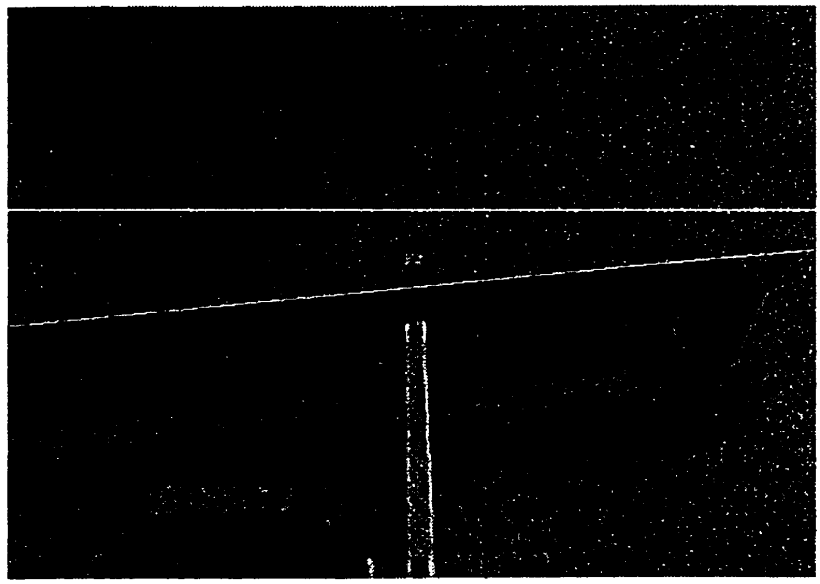


Figure 6.9 : Résultat de l'extraction de l'orientation

### 6.3.3 Détermination de la position latérale de la poutre

L'étape suivante consiste à évaluer la distance $L$ entre l'extrémité de la poutre et le centre du levier. La méthode choisie consiste, premièrement, à extraire la droite passant le long de la bordure de l'extrémité de la poutre. On évaluera ensuite le point dans l'espace monde qui correspond à l'intersection de cette droite et d'une droite de

même orientation que la poutre et passant par le centre de cette dernière (figure 6.10). La dimension $L$ recherchée correspondra à la longueur séparant ce point du centre du levier.

Pour extraire les points de la bordure de l'extrémité de la poutre, on utilise sensiblement la même approche que celle illustrée précédemment pour extraire la bordure supérieure. On débute l'extraction à la position du point extrême trouvé auparavant. On extrait les points correspondant à la bordure par l'application d'un masque de détection d'arêtes orienté horizontalement (figure 6.10). On balaye ainsi, pour chaque coordonnée en $y$, une plage de coordonnées en $x$ à l'aide de ce masque. Pour une coordonnée $y$, le point ayant la valeur la plus élevée est un point de la bordure. Les points sont ainsi insérés dans une liste. L'acquisition des points s'arrête lorsque l'on note un changement d'orientation ou que l'on ne trouve plus de points faisant partie de la bordure.

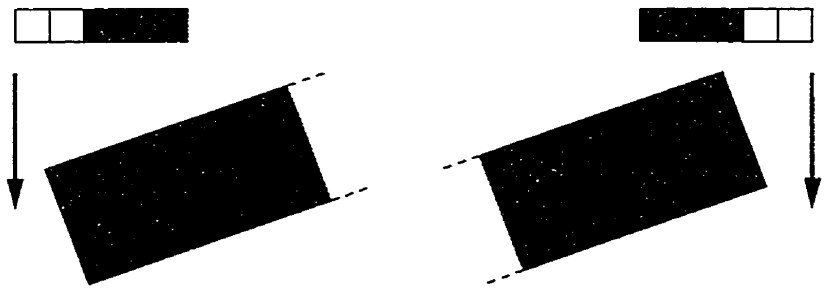


Figure 6.10 : Acquisition des points de la bordure de l'extrémité de la poutre

Le calcul de la droite, dans l'espace monde, passant par les points extraits est effectué par la méthode des moindres carrés à l'aide de l'équation (6.8). On calcule ensuite le point $p_1$ correspondant à l'intersection entre cette droite et une droite

passant par le centre de la poutre et d'orientation égale à l'orientation de la poutre par l'équation (6.9).

$$\begin{aligned} x_{p_1} &= \frac{b_{Sup} + LARGEUR\ POUTRE - b_{Ext}}{a_{Ext} - a_{Sup}}, \\ y_{p_1} &= \frac{a_{Ext}(b_{Sup} + LARGEUR\ POUTRE) - a_{Sup}b_{Ext}}{a_{Ext} - a_{Sup}}. \end{aligned} \tag{6.9}$$

La distance de l'extrémité de la poutre est finalement calculée en évaluant la distance entre le point trouvé et un point correspondant à l'intersection entre une droite passant par le centre de la poutre et une autre traversant le centre du levier (voir figure 6.11). L'équation de distance $L$ ainsi obtenue après simplification est la suivante :

$$L = \sqrt{(1 + {a_{Sup}}^2)(x_{p_1} - x_0)}. \tag{6.10}$$

Cette étape ne peut être exécutée que si l'extrémité de la poutre est visible dans l'image. Dans le cas contraire, on considère que l'extrémité de la poutre correspond au point situé à la limite de l'image. La distance $L$ est considérée, dans pareils cas, comme étant supérieure à :

$$L > \sqrt{(1 + {a_{Sup}}^2)((x_{i_{max}} + \frac{LARGEUR\ POUTRE}{a}) - x_0)}. \tag{6.11}$$

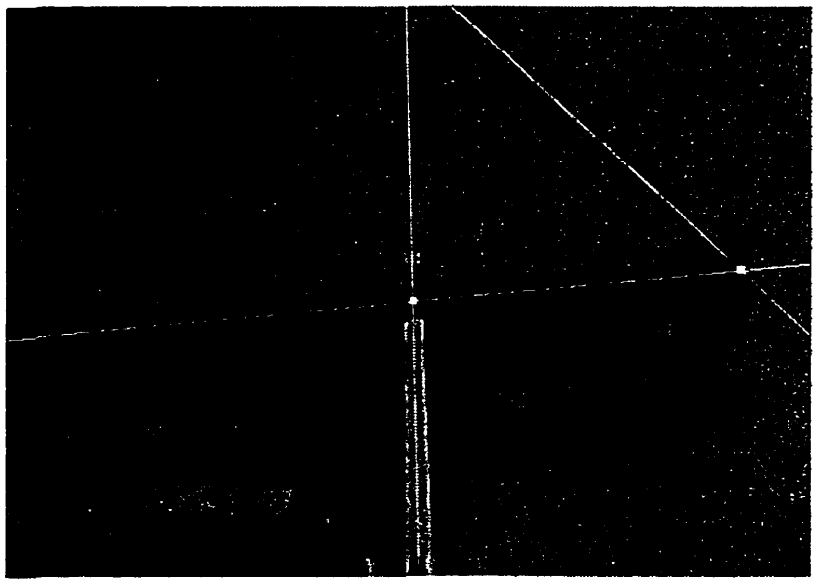


Figure 6.11 : Distance de l'extrémité

## 6.4 Résultats expérimentaux

L'implantation du système d'analyse d'images fut réalisée sur une plate-forme Macintosh. La première solution utilisant des repères artificiels nécessite peu de calcul et peut être exécutée au même taux que l'acquisition d'images, soit à un taux de 30 images par seconde. La deuxième approche qui consiste à rechercher l'extrémité de la poutre dans l'image est un peu plus exigeante en calcul et est normalement exécutée à un taux variant entre 15 et 30 images par seconde.

La deuxième méthode a été retenue pour être utilisée lors de l'implantation sur notre banc d'essai expérimental décrit au chapitre 7. Pour vérifier la robustesse et la justesse de notre analyse d'images, nous avons effectué des tests pour différentes configurations de la poutre. Pour chaque configuration de la poutre dans le monde,

l'analyse d'image a été exécutée 200 fois pour montrer à quel point les résultats sont stables dans le temps. La médiane et la variance des valeurs de positions latérales et d'orientation de la poutre ont été calculées et sont présentées au tableau 6.1

Tableau 6.1 : Résultats de l'analyse d'images

| Valeurs réelles | | Distance calculée (mm) | | Angle calculé (degré) | | Images |
|---|---|---|---|---|---|---|
| Distance | Angle | Médiane | Variance | Médiane | Variance | refusées |
| 12 mm | 0 degré | 11.6374 | 9.542e-08 | 1.7172 | 2.514e-03 | 9 |
| 50 mm | 0 degré | 50.6520 | 3.256e-05 | 1.6048 | 3.571e-02 | 0 |
| 100 mm | 0 degré | 100.684 | 8.703e-02 | 1.8337 | 2.801e-00 | 0 |
| infini | 0 degré | 139.180 | 1.223e-05 | 1.5050 | 4.300e-04 | 0 |
| 11 mm | 10 degrés | 9.8120 | 1.108e-05 | 10.986 | 9.387e-03 | 0 |
| 50 mm | 10 degrés | 49.6138 | 2.054e-01 | 11.421 | 1.546e-03 | 0 |
| 100 mm | 10 degrés | 99.6539 | 5.296e-01 | 11.566 | 2.485e-00 | 0 |
| infini | 10 degrés | 167.872 | 2.270e-02 | 14.721 | 3.861e-04 | 0 |
| 11 mm | 25 degrés | 9.5321 | 8.355e-03 | 25.101 | 5.298e-01 | 0 |
| 50 mm | 25 degrés | 48.5676 | 2.634e-02 | 26.016 | 1.725e-01 | 6 |
| 100 mm | 25 degrés | 97.9078 | 4.158e-01 | 25.117 | 2.300e-01 | 9 |
| infini | 25 degrés | 191.172 | 2.850e-00 | 23.705 | 1.098e-00 | 14 |
| 11 mm | -10 degrés | 11.7452 | 9.452e-07 | -7.9573 | 1.087e-03 | 0 |
| 50 mm | -10 degrés | 51.0906 | 1.931e-05 | -7.6815 | 1.334e-03 | 0 |
| 100 mm | -10 degrés | 101.597 | 3.652e-01 | -7.9016 | 2.283e-00 | 0 |
| infini | -10 degrés | 125.602 | 5.757e-03 | -9.8360 | 3.327e-04 | 0 |
| 12 mm | -25 degrés | 16.0070 | 3.632e-01 | -23.938 | 1.210e-03 | 0 |
| 50 mm | -25 degrés | 51.9763 | 5.654e-02 | -24.307 | 3.119e-04 | 0 |
| 100 mm | -25 degrés | 101.974 | 4.324e-01 | -23.763 | 2.132e-04 | 0 |
| infini | -25 degrés | 119.424 | 2.249e-02 | -22.691 | 1.435e-04 | 0 |

Dans le tableau 6.1, une distance "infinie" indique que l'extrémité de la poutre sort du champ de vision de la caméra. Notons que la mesure réelle de la configuration dans le monde a été exécutée manuellement avec un outillage rudimentaire. Ainsi, la manipulation manuelle de la poutre est en grande partie responsable des erreurs sur les valeurs médianes des configurations. Notons aussi que le rejet d'images lors de l'analyse est principalement causé soit par une mauvaise acquisition de l'image ou

par des interférence lors de la transmission sans fils de l'images (voir le chapitre 7 pour plus de détails à ce sujet).

# Chapitre 7

# Présentation du banc d'essai et des résultats expérimentaux

Un banc d'essai expérimental a été développé dans notre laboratoire pour l'étude des interactions dans les systèmes multi-robots (Zanardi, Hervé et Cohen, 1997). Ce banc d'essai a été conçu grâce à la collaboration de Christian Zanardi, étudiant de doctorat, avec l'aide de plusieurs personnes dont le professeur Jean-Yves Hervé, Christian Meunier, Fabienne Lathuilière et Laurence Raffin, stagiaires.

Dans ce chapitre, nous présenterons en premier les différents modules de ce banc d'essai expérimental. Nous présenterons ensuite les expériences effectuées pour vérifier certaines idées avancées concernant les capacités de coordination de ces robots mobiles.

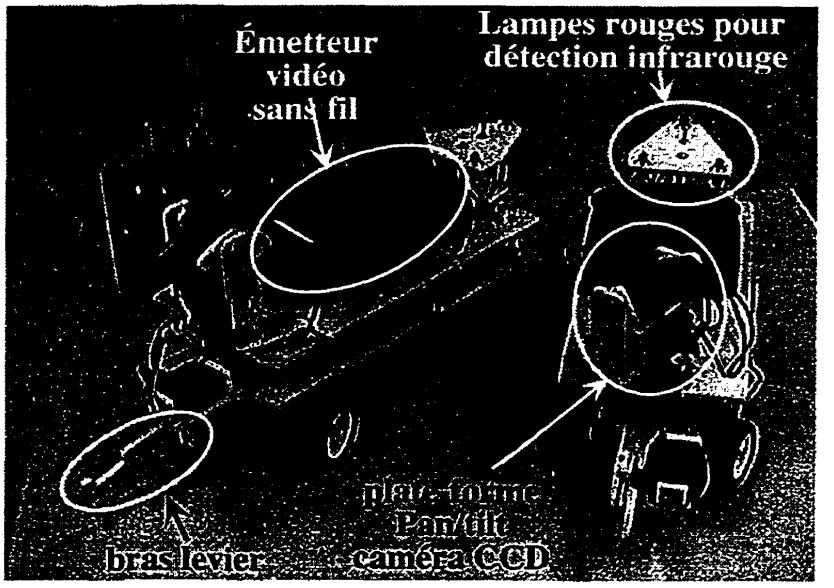



Figure 7.1 : Les robots du banc d'essai expérimental

## 7.1 Banc d'essai expérimental

Le banc d'essai expérimental comporte deux robots mobiles construits sur la base de véhicules radio-commandés de type grand public (figure 7.1). Ces véhicules ont l'inconvénient de ne pas posséder un système de codeurs de position intégré. Pour pallier ce manque majeur, un odomètre a été simulé en utilisant un filtre de Kalman. Une caméra au plafond permet de détecter des cibles placées sur les robots, et grâce à l'étalonnage de la caméra, d'extraire les coordonnées dans le repère monde. Le filtre de Kalman sert alors d'outil de prédiction et fournit aussi des valeurs de retour utilisées dans les boucles de contrôle visuel.

L'opération du système d'estimation d'état par observation (un exemple d'un tel système est donnée dans (Ishikawa et Sampei, 1995)) et l'analyse des images provenant des caméras sur les robots sont distribués sur un ensemble de stations de

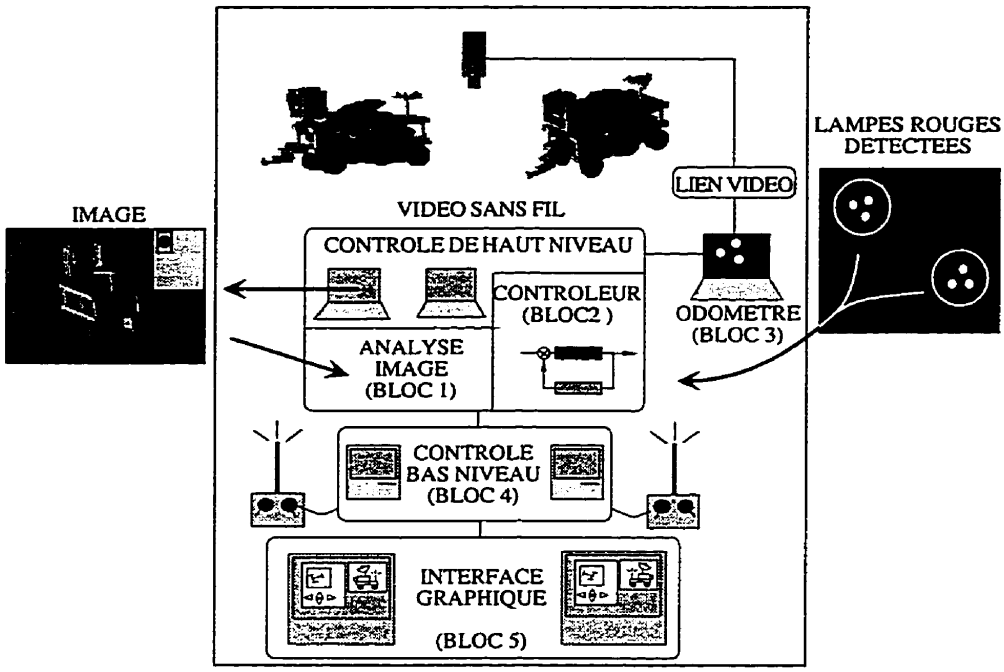



Figure 7.2 : Vue d'ensemble du système

travail en réseau afin de répartir la charge de calcul. Une vue d'ensemble du banc d'expérimentation est présentée à la figure 7.2.

Les différents blocs de la figure 7.2 décrivent les principaux éléments fonctionnels du banc. Une interface graphique permet de faire le contrôle de chaque véhicule et d'afficher leur signal vidéo (bloc 1). L'ensemble des commandes fournies par l'opérateur est ensuite transmis à une boucle de contrôle (bloc 2).

Un système d'odométrie simulée (bloc 4) procure aux robots des informations sur leur état dynamique et leur configuration spatiale. Ces informations sont utilisées dans la boucle de contrôle des robots. Les valeurs de contrôle ainsi générées sont ensuite envoyées au port sériel d'une station de travail afin d'être transmises au véhicule par l'intermédiaire de la télécommande (bloc 3). Ce type d'architecture s'est montré

très efficace pour des problèmes typiques de coordination de robots en intelligence artificielle (Sahota, 1994).

Ce banc d'essai est très versatile, et permet, entre autres, de développer des stratégies de coordination décentralisées pour des systèmes multi-robots autonomes équipés d'un système visuel. Les informations odométriques étant accessibles par le système, il est possible d'expérimenter des stratégies autant centralisées que décentralisées. Grâce au système de vision autonome monté sur le robot, des informations sur les autres robots et l'environnement peuvent aussi être utilisées dans les boucles de contrôles

Les stations de travail utilisées pour le banc d'essai comprennent : un Power Macintosh 8500/120 pour calculer les valeurs d'odométrie (bloc 4), deux Power Macintosh 7500/100 (blocs 2 et 5) pour analyser les images et transmettre les valeurs de contrôles, deux Macintosh IIci qui sont utilisés pour fournir les valeurs de tension directement aux télécommandes des véhicules (Bloc 3), et finalement deux Silicon Graphics (type Indy et Indigo 2) pour l'affichage de l'interface graphique et l'envoi des commandes (bloc 1). Notons que, pour notre application spécifique de transport coopératif, l'interface graphique (Bloc 2) n'est pas utilisée.

### 7.1.1 Description des robots mobiles

Comme il a été mentionné, les robots utilisés sont des véhicules radio-commandés de loisir, facilitant ainsi les modifications physiques qui pourraient y être apportées.

Ces voitures sont physiquement petites, leur longueur d'empattement étant de l'ordre de 25 cm.

Les télécommandes ont la capacité de contrôler jusqu'à quatre servomoteurs montés sur les robots. Deux de ces servomoteurs sont utilisés pour le contrôle de l'orientation des roues et de la vitesse du véhicule, ce qui laisse deux voies libres pour la commande d'une plate-forme tilt/pan sur laquelle se trouve une caméra. Dans notre application spécifique du transport d'une poutre, un petit levier a été ajouté à l'avant du véhicule. La possibilité d'orienter la plate-forme pan/tilt n'étant pas utilisée, le contrôle du levier est substitué à un des deux servomoteurs contrôlant cette plate-forme.

Chaque voie de la télécommande correspond à un potentiomètre dont la valeur oscille entre 0 et 5 volts. Les tensions sont envoyées par l'intermédiaire d'un générateur de tension contrôlé à l'aide d'un langage de commandes et d'un port d'accès sériel RS232. Notons que chaque valeur de tension appliquée à la télécommande est conservée dans un registre. Ainsi, une tension reste active sur la télécommande tant qu'une nouvelle valeur n'est pas venue la remplacer.

Pour réduire la zone morte des moteurs et ainsi obtenir un meilleur contrôle sur la vitesse, un contrôleur de vitesse a été rajouté, limitant la vitesse maximale des voitures à 0.8 m/s et leur accélération à environ 0.5 $m/s^2$.

#### 7.1.1.1 Étalonnage de la dynamique des véhicules

Une phase d'étalonnage a été appliquée pour connaître à l'avance les relations existant entre les différents actuateurs du robot (dans notre cas les potentiomètres des télécommandes) et les caractéristiques dynamiques réelles du robot (sa vitesse, la courbure de son déplacement ou son accélération en fonction des tensions envoyées à la télécommande). Lorsque ces relations sont connues, il est possible de faire abstraction des détails techniques (ici la modélisation des tensions applicables) et ainsi de commander le robot par l'intermédiaire de techniques usuelles de contrôle en accélération ou en vitesse. L'étalonnage obtenu permet ainsi de procéder à une simple interpolation entre les valeurs désirées de contrôle et les tensions à appliquer aux actuateurs.

Une automatisation de la technique d'étalonnage a été réalisée dans le cadre du projet de stage étudiant de Fabienne Lathuilière (Lathuilière, 1997) sous la supervision conjointe de Christian Zanardi et Philippe Lacroix.

#### 7.1.1.2 Transmission des images en provenance des caméras des robots.

L'espace disponible sur le robot étant limité, il a été choisi de ne pas faire le traitement des images provenant de la caméra directement sur le robot. Le traitement étant fait sur des stations de travail non relié au véhicule au moyen d'un câble vidéo, il est nécessaire de transmettre le signal NTSC en sortie des caméras vers ces stations à l'aide d'un système de communication sans fil. Pour diminuer les interférences entre

les signaux vidéo provenant des deux voitures, nous avons choisi deux systèmes sans fil opérant dans des gammes de fréquence séparées.

### 7.1.2 Système d'odométrie simulée

Le principal inconvénient du genre de véhicule utilisé provient de l'absence de système odométrique monté sur la voiture. En effet, pour pouvoir positionner les voitures dans l'espace et connaître leur dynamique, il est essentiel de transmettre ces informations aux contrôleurs afin d'alimenter leur boucle de contrôle.

Pour pallier à cette lacune, nous utilisons un observateur central (caméra CCD muni d'un filtre infrarouge) afin de détecter la présence des lampes rouges montées sur un support triangulaire à l'arrière de chaque robot. Cet observateur nous permettra de recueillir et de redistribuer aux contrôleurs de chaque robot, les données d'odométrie nécessaires au contrôle.

On repère chaque voiture par les coordonnées $(x,y)$ du centre de gravité du triangle (qui correspond aussi au centre de l'essieu arrière) ainsi que par l'orientation $\theta$ relative à l'axe des $x$. L'angle de braquage $\phi$ des roues par rapport à la direction de la voiture permet d'obtenir le rayon de courbure $R$ défini par l'équation suivante :

$$\frac{1}{R} = \kappa = \frac{\tan\phi}{L},$$

où $L$ désigne la longueur de la voiture. Le modèle des voitures est représenté à la figure 7.3

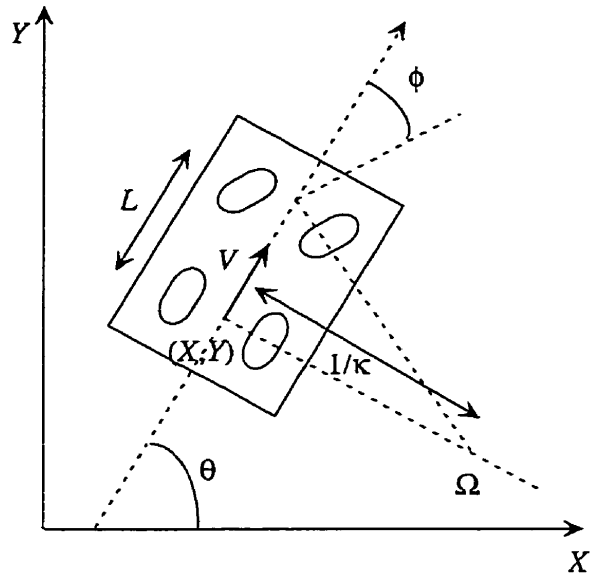



Figure 7.3 : Modèle cinématique des véhicules utilisés

Pour relier les coordonnées images aux coordonnées tridimensionnelles, nous avons utilisé l'algorithme de calibrage de Tsai-Wilson (Wilson, 1994) afin de connaître les paramètres intrinsèques et extrinsèques de la caméra.

La localisation de l'usage des lampes est facilitée à la fois par le filtre infrarouge sur l'objectif de la caméra et par la sortie d'un filtre de Kalman utilisé pour prédire les positions des lumières dans l'image. La sortie du filtre de Kalman correspond au vecteur d'état suivant :

$$\underline{x}(k) = (X_k, Y_k, \theta_k, V_k, \kappa_k)^T.$$

Dans (Zanardi, 1998) on présente une description détaillée du système d'odométrie simulée et de l'utilisation du filtre de Kalman à cette fin.

### 7.1.3 Protocole de communication entre les stations de travail

Comme il a été mentionné, plusieurs ordinateurs de différents types ont été utilisés pour analyser efficacement les informations provenant des caméras montées sur les robots mobiles et du système d'odométrie simulée. L'ensemble des stations de travail étant connectées sur un réseau *Ethernet*, un protocole pour la communication entre les machines fut développé afin de transmettre les résultats des différents traitements aux stations de travail impliquées.

Ce protocole comporte deux niveaux : au niveau supérieur, un langage permettant de coder les diverses informations de manière à standardiser les données transmises entre les diverses machines, et un protocole de communication bas niveau basé sur l'utilisation de *sockets Ethernet*. Nous avons choisi d'employer des sockets de type *stream* qui ont l'avantage d'être plus sécuritaires, renvoyant les messages perdus et garantissant un ordonnancement correct des messages (le dernier message envoyé ne peut arriver avant le message précédent).

### 7.1.4 Limites du banc d'essai

Le banc d'essai expérimental, bien qu'opérationnel, comporte certaines limitations au niveau de ses performances. En effet, dans son état actuel, il comporte des limitations au niveau de :

- la plage de vitesse,

- la plage d'accélération,
- les délais reliés à l'acquisition des images et leur analyse,
- le délai relié à l'envoie des commandes,
- le délai relié à la transmission par sockets,
- l'espace de travail disponible pour les expériences,
- les interférences lors de la transmission vidéo.

L'inefficacité du contrôleur de vitesse actuel, limite grandement la plage de vitesse d'opération des voitures. En effet, les résultats de calibrage présentés pour ces voitures dans (Lathuilière, 1997), donne les valeurs suivantes :

- Voiture 1 (la petite ou Jerry)

| | | |
|---|---|---|
| $2V < U_{vit} < 2.41V$ | $0.34m/s <$ vitesse $< 0.48m/s$ | en marche avant |
| $2.61V < U_{vit} < 2.63V$ | $-0.29m/s <$ vitesse $< -0.21m/s$ | en marche arrière |

- Voiture 2 (la grosse ou Tom)

| | | |
|---|---|---|
| $1.5V < U_{vit} < 2.33V$ | $0.28m/s <$ vitesse $< 0.53m/s$ | en marche avant |
| $2.8V < U_{vit} < 3.5V$ | $-0.39m/s <$ vitesse $< -0.30m/s$ | en marche arrière |

L'accélération est non seulement limitée en amplitude, mais elle dépend aussi de la vitesse à laquelle se déplace la voiture. Ceci rend difficile le contrôle en accélération

puisque la plage d'accélération n'est pas constante et très limitée. De plus, l'accélération et la vitesse des voitures dépendent, de façon marquée, du niveau de charges des batteries alimentant les voitures. D'autres facteurs tel le poids des voitures, les émetteurs, le frottement, font que la correspondance entre les tensions et la dynamique des voitures comportent beaucoup d'erreurs. Enfin, le type de contrôleur de vitesse monté sur les voitures a été conçu pour les modélistes, et comporte une protection qui ajoute un délais de près d'une seconde lorsque la tension appliquée sur le moteur servant à la propulsion passe d'une valeur positive à une valeur négative.

Plusieurs délais rendent l'utilisation du système complet difficile pour des applications exigeants des informations de feedback précis et à un taux de rafraîchissement élevé. Une partie de ces délais est mesurable et correspond entre autre à l'acquisition des images ainsi qu'à leur traitement. Certains délais varient dans le temps. En effet, le réseau *Ethernet* du laboratoire sert de support pour la communication entre la différentes machines impliquées dans le contrôle des robots, générant des délais variant en fonction de l'utilisation du réseau au moment des essais expérimentaux. Enfin, d'autres délais sont induits dans la phase d'application des commandes, soit au niveau de l'écriture sur le port sériel, au niveau du convertisseur numérique/analogique, ou au niveau de l'exécution des commandes sur les moteurs. Entre l'acquisition de l'image par le système d'odométrie simulée, et l'exécution de la commande désirée, il peut s'écouler souvent plus de 0.5 seconde.

La caméra utilisée pour l'odométrie simulée ne couvre qu'une superficie très limitée de l'espace de travail. En effet, la caméra ne recouvre qu'un superficie d'environ 2 mètres par 3 mètres. Ceci limite grandement les expérimentations réalisables. Ce phénomène n'a malheureusement pas comme unique cause la position de la caméra, puisque pour une situation où la caméra observe une plus grande surface, la détection des lampes montées sur les voitures devient très difficile, voir impossible.

Enfin, le système utilisé pour la transmission des images vidéo est très sensible aux interférences en haute-fréquence. En effet, les émetteurs montés sur les voitures sont peu puissants et lorsqu'un appareil extérieur émettant des ondes à des fréquences similaires est activé, l'image n'est pas transmise correctement. Un exemple d'une image transmise pendant qu'un four micro-onde fonctionnait est présenté à la figure 7.4. Dans pareils situations, toute tentative pour analyser l'image ne mène à aucun résultat valable.

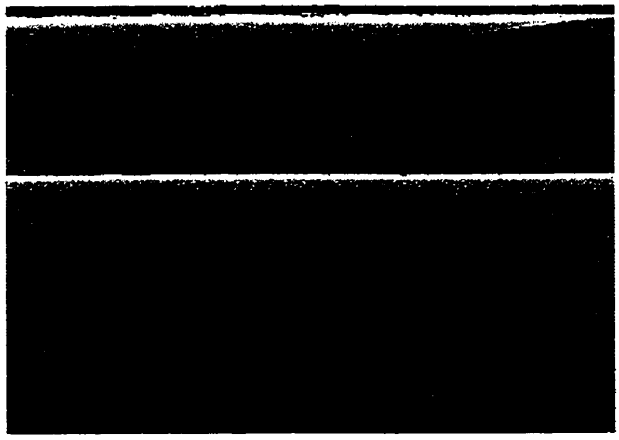


Figure 7.4 : Interférence causée par l'utilisation d'un four micro-onde

Tous ces limitations ne sont pas sans solutions, mais elles restreignent le genre d'expérimentation réalisable dans la version actuelle du banc d'essai.

## 7.2 Expériences

Les expérimentations ont été réduites vu les limitations diverses mentionnées précédemment. Les problèmes du banc d'essai ayant le plus d'impact sur ses performances sont les délais induits dans la communication par *sockets*, et les délais survenant lors de l'envoi des commandes sur les ports sériels des Macintosh IIci. Pour réduire ces délais, nous avons premièrement éliminé les deux Macintosh IIci du banc d'essai. L'envoi des commandes s'est fait en utilisant les Power Macintosh faisant le traitement des images en provenance des caméras montées sur les voitures. Pour obtenir certains résultats, nous avons aussi dû renoncer à utiliser le système d'odométrie simulée éliminant ainsi toutes communications entre les machines. L'odomètre dans son état actuel est de toute manière de peu d'utilité pour cette tâche de transport d'une poutre, vu son champ d'action très limité.

Le sous-ensemble du banc d'essai utilisé fait usage de deux Power Macintosh, tel qu'illustré à la figure 7.5. Chaque ordinateur est responsable d'une des voitures, analysant l'image en provenance de sa caméra, calculant la commande à partir des lois de contrôle données et acheminant cette dernière aux effecteurs du robot. Selon l'expérimentation, un signal de synchronisation de départ peut être envoyé d'un robot à un autre à l'aide de communication par *sockets*.

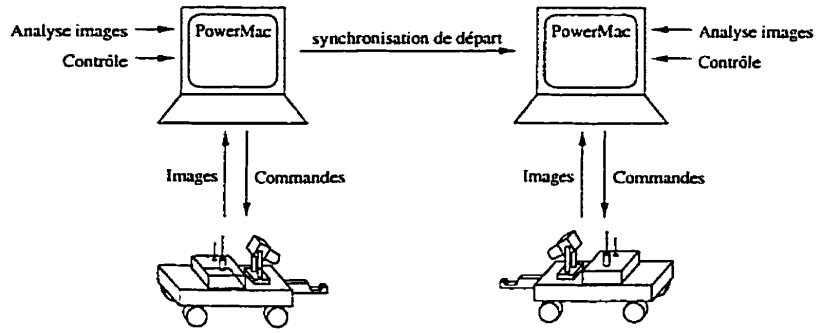
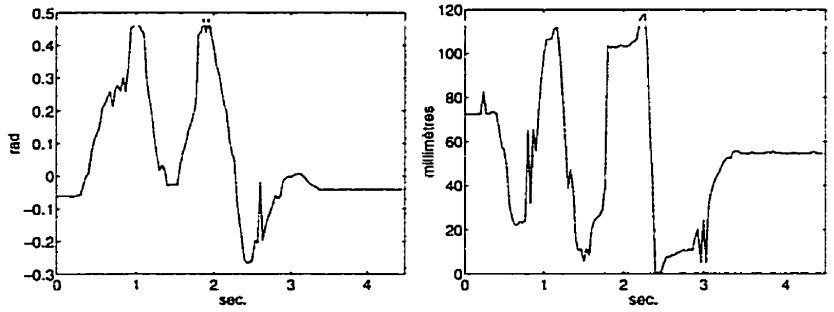



Figure 7.5 : Le banc d'essai expérimental tel qu'utilisé

Le fait de ne pas utiliser l'odomètre nous empêche de vérifier expérimentalement le contrôleur, développé au chapitre 4, dans sa totalité. En effet, le contrôleur de la poutre a besoin de l'information de position relative de la poutre à sa position finale pour bien fonctionner. Cette information ne pouvant être obtenue par le contrôleur des voitures, il en résulte une impossibilité de déterminer une direction désirée pour la poutre. Il fut donc nécessaire lors des expérimentations d'ajouter une procédure pour, soit imposer une direction désirée commune pour la poutre ou éliminer cette nécessité de connaître la direction désirée de la poutre.

**Feedback visuel** Une première expérimentation consista à déplacer une des voitures en boucle ouverte selon une trajectoire prédéfinie, l'autre voiture s'adaptant à sa coéquipière en utilisant des informations de *feedback* visuel en provenance de sa caméra. Ici, le système prend la forme d'une architecture hiérarchique où un robot maître dirige la poutre, choisissant ses déplacements désirés, et le robot esclave adapte ses déplacements pour maintenir la poutre dans une certaine configuration par rap-

port à lui, sans pour autant chercher à la diriger vers un point précis. Dans cette expérience, le robot transportant l'extrémité gauche de la poutre suit une trajectoire rectiligne en boucle ouverte. Le robot supportant l'extrémité droite essaie de maintenir sa configuration relative à la poutre dans une certaine plage de valeurs en utilisant les informations de *feedback* visuel d'orientation relative de la poutre et de sa position longitudinale (chapitre 6). Nous présentons à la figure 7.6, les valeurs de position et d'orientation de la poutre telles que perçues par la caméra de la voiture de droite lors d'une expérimentation.

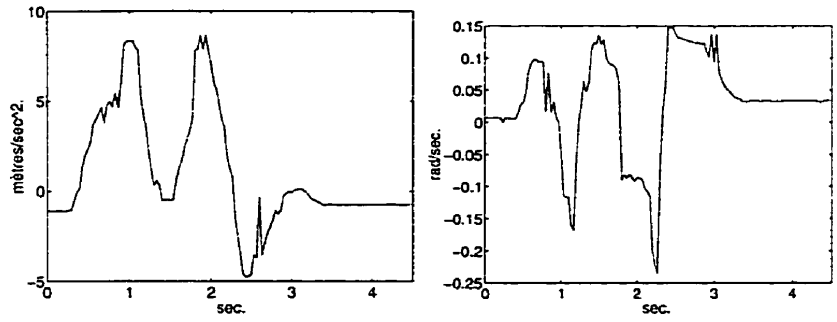



(a) Orientation (b) Position longitudinale

Figure 7.6 : Position longitudinal et orientation de la poutre tel que perçu par la caméra

Malheureusement, le contrôleur de bas niveau, tel que présenté au chapitre 4, ne peut être utilisé comme tel. En effet, celui-ci cherche à compenser les erreurs de position longitudinale et d'orientation relative de la poutre en appliquant uniquement des changements d'orientation sur les roues. La commande d'accélération appliquée au moteur des robots sert à orienter la poutre et à lui donner sa vitesse latérale.

En utilisant un tel contrôleur pour la voiture de droite, cette dernière, n'ayant pas la connaissance exacte de la vitesse de sa coéquipière, aura tendance à se déplacer plus vite ou moins vite que celle-ci ce qui se répercutera en un bris de la poutre. Un contrôle linéaire simple, n'ayant pas cet inconvénient, est ainsi utilisé pour maintenir la voiture dans une configuration relative à la poutre acceptable. Pour ce cas, les écarts sur l'orientation de la poutre sont corrigés en appliquant une accélération appropriée à la voiture. La position longitudinale de la poutre est maintenue dans une certaine plage en choisissant des vitesses de rotation du véhicule appropriées. À la figure 7.7, les commandes envoyées au robot sont illustrées.




(a) Accélération

(b) Changement d'orientation

Figure 7.7 : Accélération et changement d'orientation désirés pour la la voiture

**Distributions expérimentales** Cette expérimentation, en plus de démontrer les capacités d'adaptation des robots, a permis de générer des statistiques sur les configurations des robots. Comme il a été mentionné au chapitre 5, ces statistiques

constituent un outil intéressant pour qu'un robot puisse déterminer les chances que la poutre se situe à une position particulière sur le levier de l'autre robot ou selon une orientation donnée. Les figures 7.8 et 7.9 illustrent respectivement la position longitudinale de la poutre et l'orientation de la voiture sur celle-ci, à partir d'un échantillon de 10 expérimentations.

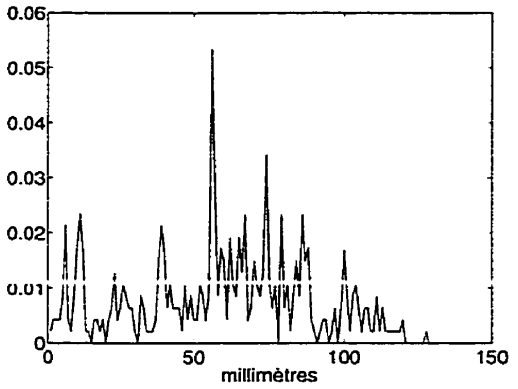



Figure 7.8 : Distribution des positions longitudinales du coéquipier

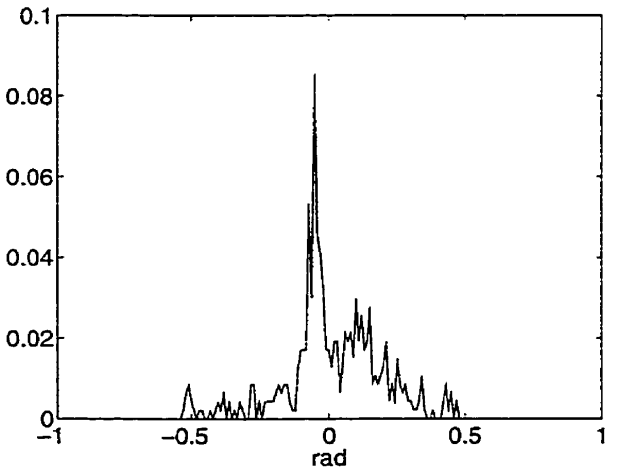



Figure 7.9 : Angle $\Psi_i$ du coéquipier

Comme il a été mentionné au chapitre 5, il est possible d'évaluer à l'aide de ces

courbes, la probabilité qu'un obstacle soit détecté par le coéquipier pour un ensemble de positions à l'avant de la poutre. Dans la figure 7.10, on présente, au moyen d'une grille de probabilité et de sa représentation tridimentionnelle, la probabilité qu'un obstacle soit détecté par le robot situé à l'extrémité droite de la poutre.

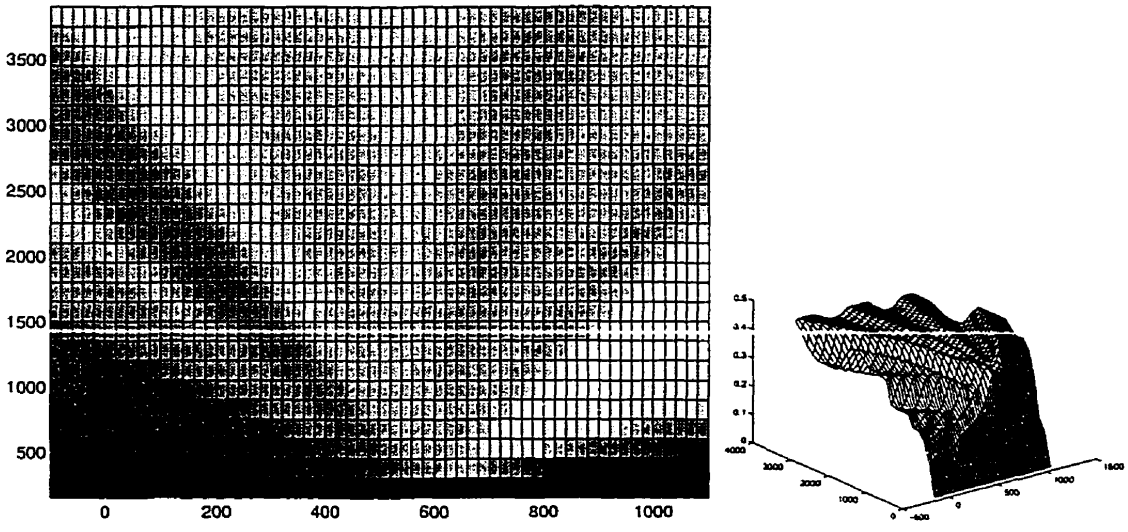
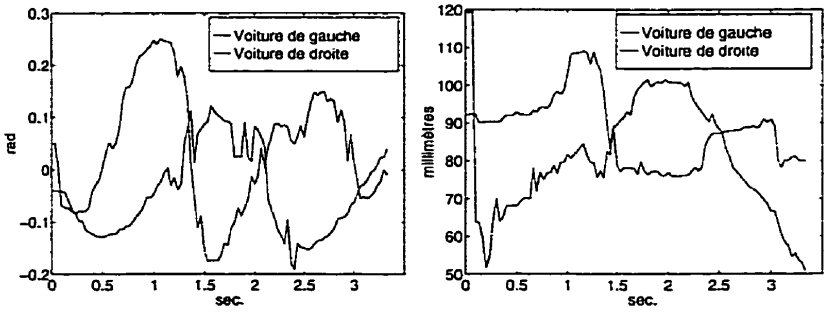



Figure 7.10 : Probabilité qu'un obstacle soit détecté par la voiture de droite

**Synchronisation** Une deuxième expérimentation consista à appliquer un même contrôle en boucle fermée aux deux voitures en se basant sur les informations du *feedback* visuel. Comme mentionné précédemment, les robots, n'ayant pas d'informations sur leur position dans l'espace, ne peuvent pas choisir un déplacement désiré à appliquer à la poutre. Pour contourner cette limitation, une simplification est faite telle que chaque robot cherche à déplacer la poutre perpendiculairement à son orientation

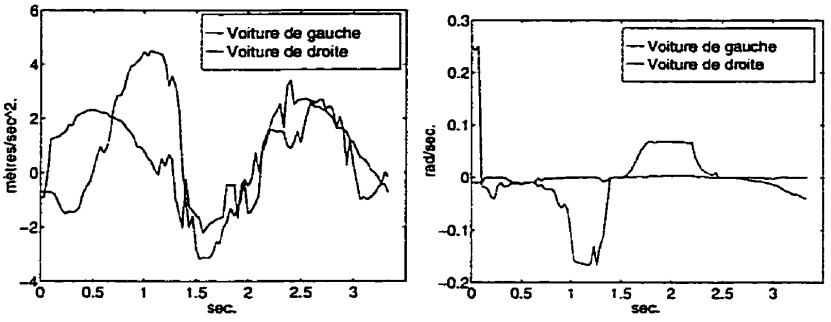
courante et selon une vitesse donnée. Ce type d'expérimentation résulte en général en un déplacement chaotique dans des directions aléatoires. Encore là, le contrôleur de bas niveau, tel que décrit au chapitre 4, ne peut être utilisé puisque la tension appliquée à la propulsion serait toujours constante ce qui n'est pas souhaitable. Un problème majeur pour cette expérimentation réside dans le comportement global de la poutre qui a tendance à suivre une trajectoire saccadée. En effet, pour ce type de trajectoire, trop souvent les robots sont dans des configurations difficiles, amenant à l'occasion le débordement des conditions permises. Nous présentons à la figure 7.11, les valeurs de position et d'orientation de la poutre telles que perçues par les caméras des voitures lors d'une expérimentation. À la figure 7.12, les commandes envoyées aux robots sont illustrées.




(a) Orientation (b) Position longitudinal

Figure 7.11 : Position longitudinale et orientation de la poutre telles que perçues par les caméras

Ces expérimentations, bien que non complètes, permettent de renforcer les divers choix pris lors de l'élaboration du contrôle décrit au chapitre 4. Elles ont montré




(a) Accélération

(b) Changement d'orientation

Figure 7.12 : Accélération et changement d'orientation désirés pour la la voiture

les capacités d'adaptation nécessaires aux robots telles qu'il a été mentionné à la section 4.4, en démontrant qu'avec un contrôle de bas niveau approprié les robots peuvent éviter les situations où la configuration de la poutre par rapport au levier rend le contrôle difficile. Les expérimentations ont en plus servi à recueillir diverses données statistiques sur la distribution des configurations d'un robot par rapport à la poutre, tel qu'utilisé dans l'algorithme d'évitement d'obstacles décrit au chapitre 5. Enfin, elles ont démontré les capacités déjà prometteuses du banc d'essai développé au laboratoire du Groupe de Recherche en Perception et Robotique.

# Chapitre 8

# Conclusion

L'étude des systèmes à plusieurs robots accomplissant des tâches en coopération est un domaine récent mais déjà très vaste, incorporant autant des aspects de contrôle robotique que des concepts provenant du domaine de l'intelligence artificielle (voir annexe A). La difficulté principale du contrôle de systèmes multi-robots autonomes concerne la coordination des actions des robots dans la tâche. Dans ce travail, nous avons présenté une approche innovatrice pour coordonner les stratégies de contrôle des différents robots dans l'accomplissement d'une tâche.

Nous nous sommes intéressés plus particulièrement au contrôle de systèmes multi-robots accomplissant des tâches coopératives exigeant des interactions de coordination ou de synchronisation. L'approche proposée ici (chapitre 2) a comme avantage de ne nécessiter aucune forme de communication explicite ou implicite à l'intérieur d'un groupe de robots pour que ceux-ci puissent accomplir efficacement une tâche

de manière coopérative. Pour ce faire, la coordination entre les robots est définie de manière rigide en étendant la spécification de la tâche fournie à chaque robot. Cette spécification de la tâche est faite en décrivant l'évolution de la tâche dans l'espace des configurations au moyen d'un champ de vecteurs qui propose, pour chaque configuration de la tâche, un déplacement instantané pour la tâche. À partir de cette nouvelle manière de spécifier la tâche, chaque membre d'une équipe peut prédire localement l'évolution de la tâche et utiliser cette information de prédiction pour gouverner ses interventions sur la tâche.

L'architecture ainsi proposée comporte deux niveaux de contrôle, soit au niveau supérieur un contrôle donnant l'évolution de la tâche dans l'espace et le temps, garantissant la qualité des comportements coopératifs du groupe, et un niveau inférieur concernant le contrôle individuel des robots afin que ceux-ci suivent les requêtes du niveau supérieur. Cette architecture générale peut être considérée comme la première contribution de ce travail.

Pour illustrer l'approche proposée, une application a ensuite été décrite en détail au chapitre 3. La tâche consiste à déplacer une poutre d'une configuration initiale vers une configuration finale en la soulevant par ses extrémités à l'aide de deux robots mobiles. Les robots, de type voiture, sont munis d'un levier à l'avant et d'une caméra pointant vers la poutre. La modélisation des robots mobiles et de la poutre, nécessaire à l'analyse du système, de même que des éléments sur le phénomène de friction présent entre la poutre et les leviers, furent ensuite présentés.

Le contrôleur décentralisé de chaque voiture, utilisé pour cette application (chapitre 4), cherche à accomplir la tâche en ne faisant usage que d'informations visuelles provenant de l'observation de la poutre et de l'information de la position relative à la configuration finale souhaitée pour la poutre. Tel que décrit dans l'approche générale, le contrôleur est divisé en deux niveaux de contrôle. Au niveau supérieur, la spécification de la tâche donne au robot les vitesses longitudinales et angulaires désirées pour la poutre à chaque instant. La poutre étant modélisée comme un uni-cycle, ce niveau correspond au contrôle d'un véhicule non holonomique dans l'espace cartésien. Un survol des méthodes de contrôle pour pareils systèmes est présenté à l'annexe B. Deux contrôleurs ont été étudiés ici, l'un faisant usage de l'information de glissement de la poutre sur les leviers l'autre ne faisant pas usage de cette information. Au niveau inférieur, un contrôleur simple, non linéaire cherche à atteindre les caractéristiques dynamiques désirées pour la poutre. Ce contrôleur distribué pour le transport d'un poutre par deux voitures est la deuxième contribution de ce travail.

Une méthode pour éviter les obstacles imprévus le long du chemin de la poutre sans faire usage de communication fut ensuite présentée au chapitre 5. Nous avons montré qu'en modifiant le déplacement désiré pour la poutre, un robot peut délibérément perturber le système afin de faire dévier la poutre de sa route et ainsi éviter les éventuels obstacles.

Enfin, nous avons présenté les éléments d'expérimentation de cette application spécifique de transport d'une poutre. Deux méthodes pour faire l'extraction visuelle

des informations de position et d'orientation de la poutre ont premièrement été présentées au chapitre 6. Au chapitre 7 nous avons présenté le banc d'essai expérimental ainsi que ses limitations dans l'état actuel de son implantation. Quelques résultats expérimentaux pour notre application de contrôle ont enfin été présentés. Ces expérimentations ont su démontrer les possibilités de l'utilisation du feedback visuel et du contrôle de bas niveau pour coordonner des robots mobiles lors d'une tâche simple. Ces derniers chapitres ont contribué à démontrer les possibilités d'implantation de la stratégie de contrôle choisie.

En conclusion, ce travail présente trois principales contributions innovatrices :

1. Sur le plan théorique, nous avons proposé une approche concernant le design de stratégies de contrôle décentralisé pour des systèmes multi-robots dotés de comportements *coopératifs* et *coordonnés.*

2. Sur le plan pratique, nous avons décrit une solution efficace pour le contrôle décentralisé pour remplir une tâche consistant à transporter une poutre à l'aide de deux robots en utilisant l'approche de contrôle que nous avons proposé.

3. Sur le plan expérimental, nous avons développé un banc d'essai pour étudier les interactions dans des systèmes multi-robots. Nous avons de même validé certains éléments du contrôleur de bas niveau pour notre application de transport d'une poutre.

Enfin, il serait intéressant d'appliquer notre approche pour le contrôle décentra-

lisé dans d'autres applications notamment pour des applications d'exploration d'environnement, de mouvement en formation ou pour d'autres applications de transport faisant l'usage de plus de deux robots. De plus, il serait intéressant d'apporter diverses amélioration au banc d'essai, notamment en augmentant la superficie couverte par le système odométrique et en améliorant le système de communication entre les stations de travail. Ces améliorations permettrait de compléter l'expérimentation de notre contrôleur pour le transport d'une poutre.

# Bibliographie

AICARDI, M., CASALINO, G., BICCHI, A. et BALESTRINO, A. (1995), Closed loop steering of unicycle-like vehicles via lyapunov techniques, dans *IEEE Robotics and Automation Magazine*, pp. 27–35.

ALAMIR, M. et KHENNOUF, H. (1995), Discontinuous receding horizon based stabilizing feedback for nonholonomic systems in ower form, dans *Proceedings of the 34rd Conference on Decision and Control*, pp. 4300–04.

ARAI, T. et OTA, J. (1995), Collision avoidance among multiple robots using virtual impedance, dans *IEEE International Workshop on Intelligent Robots and Systems*, pp. 479–485.

ARBIB, H. et LIAW, J.-S. (1995), Sensorimotor transformations in the worlds of frogs and robots, dans *Artificial Intelligence*, Vol. 72, pp. 53–79.

ARKIN, R. (1990), Integrating behavioral, perceptual, and world knowledge in reactive navigation, *Robotics and Autonomous Systems* **6**, 105–122.

ARKIN, R. (1992), Cooperation without communication: Multiagent schema-based robot navigation, dans *Journal of Robotic Systems*, pp. 351–364.

ARMSTRONG, B. (1988), Friction: Experimental determination modeling and computation, dans *Proceedings of the IEEE International Conference on Robotics and Automation*, Vol. 3, Philadelphia, PA, pp. 1422–27.

ARMSTRONG-HÉLOUVRY, B., DUPONT, P. et WIT, C. D. (1994), A survey of models, analysis tools and compensation methods for the control of machines with friction, *Automatica* **30**, 1083–1138.

ASTOLFI, A. (1994), On the stabilization of nonholonomic systems, dans *Proceedings of the 33rd Conference on Decision and Control*, pp. 3481–86.

BALCH, T., BOONE, G., COLLINS, T., FORBES, H., MACKENZIE, D. et SANTAMAIA, J. C. (1995), Io, ganymede, and callisto: A multiagent robot trash-collecting team, dans *AI Magazine*, Vol. 16, pp. 39–51.

BARNES, D. et GRAY, J. (1991), Behaviour synthesis fo co-operant mobile robot control, dans *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, pp. 1135–1140.

BLOCH, A. et DRAKUNOV, S. (1994), Stabilization of a nonholonomic system via sliding modes, dans *Proceedings of the 33rd Conference on Decision and Control*, pp. 2961–63.

BLOCH, A. et DRAKUNOV, S. (1995), Tracking in nonholonomic dynamic systems via sliding modes, dans *Proceedings of the 34rd Conference on Decision and Control*, pp. 2103–06.

BROCKETT, R. (1983), Asymptotic stability and feedback stabilization, dans R. Brockett, R. Millman et H. Sussman, éditeurs, *Differential Geometric Control*, Birkhauser, Boston, pp. 181–208.

BROOKS, R. (1986), A robust layered control system for a mobile robot, dans *IEEE Journal of Robotics and Automation*, Vol. 1, pp. 14–23.

BROOKS, R. (1990), Elephants don't play chess, dans *Robotics and Autonomous Systems*, Vol. 6, pp. 3–15.

BROOKS, R. (1991), Intelligence without reason, dans *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 569–595.

BROWN, R. et JENNINGS, J. (1995), A pusher/steerer model for strongly cooperative mobile robot manipulation, dans *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, Pittsburgh, PA.

CAO, Y., FUKUNAGA, A., KAHNG, A. et MANG, F. (1995), Cooperative mobile robotics: Antecedents and directions, dans *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, Vol. 1, Pittsburgh, PA, pp. 226–34.

CHEN, Q. et LUTH, J. (1993), Distributed motion coordination of multiple robots, dans *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, Yokohama, Japan, pp. 1493–1500.

CHUI, C. et CHEN, G. (1991), *Kalman Filtering with Real-time Applications*, Springer Verlag.

DECARLO, R., ZAK, S. et MATTHEWS, G. (1988), Variable structure control of nonlinear multivariable systems: A tutorial, dans *Proceedings of the IEEE*, Vol. 76, pp. 212–232.

DONALD, B., JENNINGS, J. et RUS, D. (1994), Analysing teams of cooperating mobile robots, dans *Proceedings of the IEEE International Conference on Robotics and Automation*, San Diego, CA, pp. 1896–1903.

DORIGO, M. (1995), Genetics-based machine learning and behavior-based robotics: A new synthesis, dans *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 23, pp. 141–153.

DUDEK, G., JENKIN, M., MILIOS, E. et WILKES, D. (1993), A taxonomy for swarm robotics, dans *IEEE International Workshop on Intelligent Robots and Systems*, Vol. 1, pp. 441–447.

DUDEK, G., JENKIN, M., MILIOS, E. et WILKES, D. (1995), Experiments in sensing and communication for robot convoy navigation, dans *Proceedings of*

*the IEEE International Conference on Intelligent Robots and Systems*, Vol. 2, Pittsburgh, PA, pp. 268–.

EVANS, J. (1991), Visual navigation and obstacle avoidance structured light system, dans *US Patent 5 040 116*.

FUKUDA, T. et IRITANI, G. (1995), Construction mechanism of group behavior with cooperation, dans *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, Vol. 3, Pittsburgh, PA.

FUKUDA, T., IRITANI, G., UEYAMA, T. et ARAI, F. (1994), Optimization of group behavior on cellular robotic system in dynamic environment, dans *Proceedings of the IEEE International Conference on Robotics and Automation*, Vol. 2, San Diego, CA, pp. 1027–.

FUKUDA, T. et SEKIYAMA, K. (1994), Communication reduction with risk estimate for multiple robotic system, dans *Proceedings of the IEEE International Conference on Robotics and Automation*, Vol. 4, San Diego, CA, pp. 2864–69.

GAUSSIER, P. et ZREHEN, S. (1994), A constructivist approach for autonomous agents, dans N. Tholman, éditeur, *Artificial Life in Virtual Reality*, Wiley and son.

GENESERETH, M., GINSBERG, M. et ROSENSCHEIN, J. (1986), Cooperation without communication, dans *AAAI*, Vol. 1, pp. 51–57.

GULDER, J. et UTKIN, V. (1994), Stabilization of non-holonomic mobile robots using lyapunov functions for navigation and sliding mode control, dans *Proceedings of the 33rd Conference on Decision and Control*, pp. 2967–72.

HASHIMOTO, M., OBA, F. et EGUCHI, T. (1993), Dynamic control approach for motion coordination of multiple wheeled mobile robots transporting a single object, dans *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, Vol. 3, Yokohama, Japan, pp. 1944–51.

HINTON, G. (1989), Connectionist learning procedures, *Artificial Intelligence* **40**(1), 167–216.

HUGLI, H., TIECHE, F. et FACCHINETTI, C. (1994), Coordinating motion of cooperative mobile robots through visual observation, dans *IEEE International Conference on Systems, Man, and Cybernetics*, Vol. 1, pp. 1047–52.

ISAACS, R. (1965), *Differential Games*, John Wiley and Sons, New York.

ISHIDA, Y., ASAMA, H., TOMITA, S., OZAKI, K., MATSUMOTO, A. et ENDO, I. (1994), Functional complement by cooperation of multiple autonomous robots, dans *Proceedings of the IEEE International Conference on Robotics and Automation*, San Diego, CA.

ISHIKAWA, M. et SAMPEI, M. (1995), State estimation of non-holonomic mobile robots using nonlinear observers, dans *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1379–1384.

J.L.MOIGNE et WAXMAN, A. (1988), Structured light patterns for robot mobility, dans *IEEE Journal of Robotics and Automation*, pp. 541–548.

KAILATH, T. (1980), *Linear Systems*, Prentice-Hall, Englewood Cliffs, N.J.

KALMAN, R. et BERTRAM, J. (1960), Control system analysis and design via the "second method" of lyapunov - continuous-time systems, dans *Transactions of the ASME, Journal of Basic Engineering*, pp. 371–393.

KHATIB, O. (1985), Real-time obstacle avoidance for manipulators and mobile robots, *Proceedings of the IEEE International Conference on Robotics and Automation* pp. 500–505.

KOLMANOVSKY, I. et REYHANOGLU, M. (1994), Discontinuous feedback stabilization of nonholonomic systems in extended power form, dans *Proceedings of the 33rd Conference on Decision and Control*, pp. 3469–74.

KOSECKA, J., CHRISTENSEN, H. et BAJCSY, R. (1995), Discrete event modeling of visually guided behaviors, dans *International Journal of Computer Vision*, Vol. 14, pp. 179–91.

KOZA, J. (1990), Evolution and co-evolution of computer programs to control independently-acting agents, dans *Proceedings, Simulation of Adaptive Behavior SAB-90*, The MIT Press, Paris, France, pp. 366–375.

KUC, R. (1990), A spatial sampling criterion for sonar obstacle detection, dans *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 5, pp. 686–690.

KUNIYOSHI, Y., KITA, N., ROUGEAUX, S., SAKANE, S., ISHII, M. et KARIKUA, M. (1994), Cooperation by observation: The framework and basic task patterns, dans *Proceedings of the IEEE International Conference on Robotics and Automation*, Vol. 1, San Diego, CA, pp. 768–773.

KUNIYOSHI, Y., RIEKKI, J., ISHII, M., ROUGEAUX, S., KITA, N., SAKANE, S. et KAKIKURA, M. (1994), Vision-based behaviors for multi-robot cooperation, dans *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, Vol. 2, Munich, Germany, pp. 925–932.

LATHUILIÈRE, F. (1997), Coordination visuelle entre robots mobiles autonomes, dans *Rapport technique GRPR-RT-9710*.

LISCANO, R. et MANZ, A. (1992), Advantages of sector elimination methods over generalized potential fields methods for real-time autonomous navigation, dans *IASTED*, pp. 22–25.

LIZARRALDE, F. et WEN, J. (1996), Feedback stabilization of nonholonomic systems in presence of obstacles, dans *Proceedings of the IEEE International Conference on Robotics and Automation*, Minneapolis, MA, pp. 2682–87.

MARAPANE, S., HOLDER, M. et TRIVEDI, M. (1994), Coordinating motion of cooperative mobile robots through visual observation, dans *IEEE International Conference on Systems, Man, and Cybernetics*, Vol. 3, pp. 2260–2265.

MATARIC, M. (1994), Interaction and Intelligent Behavior, Thèse de doctorat, Massachusetts Institute of Technology.

MATARIC, M., NILSSON, M. et SIMSARIAN, K. (1995), Cooperative multi-robot box-pushing, dans *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, Pittsburgh, PA.

MCKERROW, P. (1991), *Introduction to Robotics*, Addison-Wesley.

M'CLOSKEY, R. et MURRAY, R. (1993), Convergence rates for nonholonomic systems in power form, dans *Proceedings of the IEEE American Control Conference*, pp. 2967–72.

MURRAY, R. et SASTRY, S. (1993), Nonholonomic motion planning: Steering using sinusoids, dans *IEEE Transactions on Automatic Control*, Vol. 38, pp. 700–716.

NELSON, R. et ALOIMONOS, J. (1989), Obstacle avoidance using flow field divergence, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **11**, 1102–1106.

NIJMEIJER, H. et SCHAFT, A. V. D. (1990), *Nonlinear Dynamical Control Systems*, Springer-Verlag, New York.

NOREILS, F. (1992), An architecture for cooperative and autonomous mobiles robots, dans *Proceedings of the IEEE International Conference on Robotics and Automation*, Vol. 3, Nice, France.

NOREILS, F. (1993), Toward a robot architecture integrating cooperation between mobile robots: Application to indoor environment, dans *The International Journal of Robotics Research*, Vol. 12, pp. 79–98.

OTA, J., MIYATA, N., ARAI, T., YOSHIDA, E., KURABAYASHI, D. et SASAKI, J. (1995), Transferring and regrasping a large object by cooperation of multiple mobile robots, dans *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, Vol. 3, Pittsburgh, PA, pp. 543–48.

OZAKI, K., ASAMA, H., ISHIDA, Y., MATSUMOTO, A., YOKOTA, K., KAETSU, H. et ENDO, I. (1993), Synchronized motion by multiple mobile robots using communcation, dans *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, Vol. 2, Yokohama, Japan, pp. 1164–69.

PARKER, L. (1993), Designing control laws for cooperative agent teams, dans *Proceedings of the IEEE International Conference on Robotics and Automation*, Vol. 1, Atlanta, GA, pp. 582–587.

PARKER, L. (1994*a*), Alliance: An architecture for fault tolerant, cooperative control of heterogenous mobile robots, dans *Proceedings of the IEEE International Con-*

*ference on Intelligent Robots and Systems*, Vol. 1, Munich, Germany, pp. 776–783.

PARKER, L. (1994*b*), Heterogeneous Multi-Robot Cooperation, Thèse de doctorat, Massachusetts Institute of Technology.

PARKER, L. (1995), Effect of action recognition and robot awareness in cooperative robotic teams, dans *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, Vol. 1, Pittsburgh, PA, pp. 212–219.

REYNOLDS, C. (1987), Flocks, herds, and schools: A distributed behavioral model, dans *Proceedings of the SIGGRAPH'87*, Vol. 21, pp. 25–34.

RUS, D., DONALD, B. et JENNINGS, J. (1995), Moving furniture with teams of autonomous robots, dans *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, Vol. 1, Pittsburgh, PA, pp. 235–42.

SAHOTA, M. (1994), Reactive deliberation: An architecture for real-time intelligent control in dynamic environments, dans *AAAI*, pp. 1303–1308.

SAMSON, C. (1993), Time-varying feedback stabilization of car-like wheeled mobile robots, dans *The International Journal of Robotics Research*, Vol. 12, pp. 55–64.

SAMSON, C., BORGNE, M. L. et ESPIAU, B. (1991), *Robot Control: The Task Function Approach*, Oxford University Press, Oxford.

SINGH, K. et FUJIMURA, K. (1993), A navigation strategy for cooperative multiple mobile robots, dans *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, Vol. 1, Yokohama, Japan, pp. 283–288.

SLOTINE, J.-J. E. et LI, W. (1987), On the adaptive control of robot manipulators, *The International Journal of Robotics Research* **6**(3).

SØRDALEN, O. et WIT, C. C. D. (1992), Exponential control law for a mobile robot: Extension to path following, dans *Proceedings of the IEEE International Conference on Robotics and Automation*, Nice, France, pp. 2158–2163.

STEELS, L. (1994), Emergent functionality of robot behavior through on-line evolution, dans R. Brooks et P. Macs, éditeurs, *Artificial Life IV, Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, The MIT Press.

STEWART, J. (1995), The implications for understanding high-level cognition of a grounding in elementary adaptive systems, *Robotics and Autonomous Systems* **16**, 107–116.

TAIPALE, T. et HIRAI, S. (1993), A behavior-based control system applied over multiple robot system, dans *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, Vol. 3, Yokohama, Japan, pp. 1941–43.

TEEL, A., MURRAY, R. et WALSH, G. (1992), Nonholonomic control systems: From steering to stabilization with sinusoids, dans *Proceedings of the 31rd Conference on Decision and Control*, pp. 1603–09.

TILOVE, R. (1990), Local obstacle avoidance for mobile robots based on the method of artificial potentials, dans *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 566–71.

UEYAMA, T. et FUKUDA, T. (1993), Self-organization of cellular robots using random walk with simple rules, dans *Proceedings of the IEEE International Conference on Robotics and Automation*. Vol. 3. Atlanta. GA. pp. 595–600.

WANG, J. (1994), On sign-board based inter-robot communication in distributed robotic systems, dans *Proceedings of the IEEE International Conference on Robotics and Automation*, Vol. 2, San Diego, CA, pp. 1045–50.

WANG, Z.-D., NAKANO, E. et MATSUKAWA, T. (1994), Cooperating multiple behavior-based robots for object manipulation, dans *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, Vol. 3, Munich, Germany, pp. 1524–31.

WATKINS, C. (1989), Learning from Delayed Rewards, Thèse de doctorat, Psychology Department, Cambridge University.

WILSON, R. (1994), Modeling and Calibration of Automated Zoom Lenses, Thèse de doctorat, Carnegie Mellon University.

WIT, C. C. D. et SØRDALEN, O. (1992), Exponential control of mobile robot with nonholonomic constraints, *IEEE Transactions on Automatic Control* **37**(11), 1791–1797.

YOSHIDA, E., YAMAMOTO, M. et OTHERS (1995), Design method of local communication range in mutliple mobile robot system, dans *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, Vol. 2, Pittsburgh, PA, pp. 274–279.

YUN, X. et N.SARKAR (1996), Dynamic feedback control of vehicles with two steerable wheels, dans *Proceedings of the IEEE International Conference on Robotics and Automation*, Minneapolis, MA, pp. 3105–10.

ZANARDI, C. (1998), Interactions entre Robots: Aspects Dynamiques et Sensoriels, Thèse de doctorat, Ecole Polytechnique de Montreal.

ZANARDI, C., HERVÉ, J. et COHEN, P. (1996), Mutual learning of unsupervised interactions between mobile robots, dans *Proc. IEEE Int'l Conf. Pattern Recognition.*

ZANARDI, C., HERVÉ, J.-Y. et COHEN, P. (1997), Robocoop: A low cost testbed for the study of interactions between mobile robots, dans *submitted to the International Conference on Robotics and Automation 1997.*

# Annexe A

# Revue bibliographique sur les systèmes multi-robots coopératifs

Cette section dresse un aperçu des avancements dans le domaine de la coopération de systèmes multi-robots.

Beaucoup de travaux ont vue le jour ces dernières années sur la coopération entre robots mobiles (Cao et al., 1995). L'utilisation d'une équipe de robots pour remplir une tâche apporte beaucoup d'avantages face à l'utilisation de système comportant un robot unique très spécialisé. Les équipes de robots ne sont, entre autres, pas limitées dans l'espace et dans le temps. Elles peuvent ainsi accomplir des tâches ne pouvant pas être exécutées séquentiellement. Il existe, en effet des tâches dites "fortement coopératives" (Brown et Jennings, 1995), où les robots doivent agir simultanément à divers endroit dans l'espace de travail pour exécuter la tâche.

De plus, les unités utilisées dans l'équipe peuvent être moins complexes mécaniquement et donc moins coûteux et plus robustes face aux bris. Enfin la capacité de pouvoir ajouter des robots simples disposant de fonctionnalités variées, apporte une plus grande flexibilité et facilite l'évolutivité des systèmes multi-robots face aux systèmes mono-robot.

Dans ce chapitre, nous regarderons premièrement les différentes caractéristiques des architectures à robot multiples. Nous présenterons ensuite les architectures les plus représentatives dans la littérature. Enfin, certaines idées relatives aux comportements émergents et aux méthodes d'apprentissage seront présentées.

# A.1 Architecture

L'architecture constitue la structure rigide d'un système robotique, déterminant ses capacités et ses limitations. C'est elle qui donne une infrastructure aux robots sur lesquels on implante les divers comportements du système.

## A.1.1 Caractéristiques

### A.1.1.1 Organisation du groupe sur le plan décisionnel

Une des caractéristiques importantes des architectures multi-robots concerne sa structure d'organisation décisionnelle (figure A.1). Les principaux types de structure décisionnelle existant sont les suivants (Noreils, 1993; Cao et al., 1995) :

**Centralisé :** Architecture où la prise de décision est faite par une entité unique. Les robots du système sont perçus comme des membres d'un robot unique.

**Décentralisé :** Architecture où la prise de décision est répartie sur plusieurs robots. On peut distinguer deux types d'architecture décentralisée :

**Distribué :** Les prises de décisions sont faites par chaque robot. Ceux-ci sont considérés comme égals sur le plan du contrôle.

**Hiérarchique :** L'organisation des comportements est centralisée localement pour un groupe de robots donné. Certains robots plus élevés en rang contrôlent ou influencent directement les comportements des robots de rang inférieur.

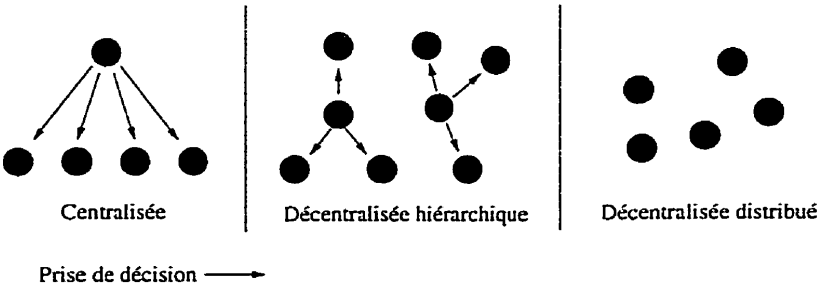



Figure A.1 : Organisation sur le plan décisionnel

Les systèmes centralisés ont l'avantage de disposer de toutes les informations sur l'état du systéme, ce qui facilite grandement les décisions à prendre pour obtenir les comportements désirés pour le système. Ils sont, par contre, sujets aux erreurs ou aux pannes du contrôleur central. De plus, la complexité du système croissant avec

les capacités et le nombre de robots, ce type de système est limité dans le nombre de robots pouvant être contrôlés pour remplir une tâche donnée. Nétant pas limités sur ces points, les systèmes multi-robots décentralisés ont reçus beaucoup d'attention depuis quelques années (Cao et al., 1995).

#### A.1.1.2 Contrôle local/global

Pour décrire les comportements de chaque robot, on utilise des lois de contrôle qui gouvernent les actions du robot en fonction d'une situation donnée (Parker, 1993). Ces lois peuvent être basées sur des informations locales ou globales du système (Parker, 1994*b*). Les informations globales permettent aux robots d'élaborer une stratégie d'intervention. Ce sont ces informations qui guides la coordination entre les robots pour l'obtention de comportements coopératifs. Elles requièrent beaucoup de communication inter robots, nécessitant du temps et des ressources matérielles considérables. Les informations locales sont basées sur l'environnement immédiat du robot. Elles servent dans la boucle de contrôle afin de permettre aux robots de réagir face aux stimuli perçus par leurs senseurs. Un équilibre entre l'utilisation de ces connaissances globales et locales doit être obtenu de manière à ce que les actions à long terme soient guidées par les informations globales et les actions à court terme, en réactions à l'environnement, soient guidées par les informations locales.

### A.1.1.3 Structure de communication

Pour obtenir l'information nécessaire à l'utilisation des lois de contrôle, il est essentiel de doter les robots d'outils leur permettant d'évaluer l'état actuel du système. Ces outils doivent de plus servir la coordination entre les robots, informant ceux-ci des changements désirés à apporter à la tâche. Deux formes de communication peuvent être utilisées à ces fins, soit la communication explicite et la communication implicite.

**Communication explicite**

Pour cette forme de communication, des informations exactes sont transmises entre les agents du système par communication radio, câblé ou tout autre médium. Ces informations peuvent être transmises simultanément à tous les unités (*broadcast*) comme dans (Chen et Luth, 1993), où un agent planifie et dicte aux autres robots, la trajectoire d'un objet à transporter coopérativement. La communication peut aussi se faire un à un comme dans (Hashimoto et al., 1993), où un dirigeant donne ses instructions aux robots à tour de rôle . Plusieurs protocoles de communication sont décrits dans la littérature (Mataric et al., 1995; Wang, 1994).

Des recherches ont été effectuées dans le but de minimiser la quantité d'information à transmettre entre les robots pour obtenir une bonne coordination. Dans (Fukuda et Sekiyama, 1994), on propose une méthodologie pour réduire l'utilisation de communication explicite en faisant usage d'estimations probabilistes du risque. Dans (Yoshida, Yamamoto et al., 1995), on présente une méthode pour calculer le rayon de communication optimal en se basant sur la probabilité qu'une information a d'être transmise

correctement.

### Communication implicite

Pour les formes de communication implicite, les informations sur l'état actuel du système sont extraites par chaque robot, à partir de leurs senseurs. Ces informations peuvent être acquises par l'unique observation de l'environnement comme dans (Arkin, 1992). Dans pareil cas, on peut considérer l'environnement comme étant une mémoire partagée. Les informations sur l'état du système peuvent aussi être acquises par l'observation des actions des autres agents comme c'est souvent le cas dans les applications de convoyage (Dudek, Jenkin, Milios et Wilkes, 1995; Marapane, Holder et Trivedi, 1994). Dans (Parker, 1995; Mataric, 1994), on regarde les effets de la reconnaissance des actions des autres robots sur la performance d'un système de robots coopératifs. Dans (Kuniyoshi, Kita, Rougeaux, Sakane, Ishii et Karikua, 1994), on présente une architecture où les robots observent et classifient qualitativement les actions des autres robots afin d'anticiper les événements futurs. Les systèmes de communication explicite étant fragile aux erreurs de transmissions et générant des délais important dans les boucles de contrôle, l'utilisation d'une forme de communication implicite a l'avantage d'augmenter la robustesse et la réactivité des systèmes robotiques (Parker, 1995).

#### A.1.1.4 Modélisation des autres agents

La modélisation des intentions ou de l'état des autres agents peut améliorer le niveau de coopération entre les robots. Dans (Genesereth et al., 1986), on regarde les systèmes multi-agents ne communiquant pas entre eux. Sachant que chaque robot tente de maximiser l'utilité de ses actions, celles-ci sont choisies en exploitant l'information de rationalité qu'un robot a des autres robots et la similarité entre les robots. Un robot agit de façon rationnelle s'il choisit toujours les actions qui sont dominantes, donc qui à coup sûr, donnent une plus grande utilité au robot.

#### A.1.1.5 Différenciation

Une autre caractéristique importante concerne l'homogénéité ou l'hétérogénéité du groupe de robots. Un groupe de robots est dit homogène si chaque robot est identique (hétérogène dans le cas contraire). L'homogénéité du groupe rend les comportements d'un robot plus prévisible face à ses coéquipiers, ceux-ci pouvant modélisater le robot selon leur propre modèle. De plus, l'augmentation de l'hétérogénéité d'une équipe permet de diminuer le flot de communication inter robot, chaque robot étant plus restreint à un certaines tâches spécifiques propres à ses compétences (Parker, 1995).

### A.1.2 Classes d'architecture

Il est difficile de classifier les architectures de système multi-robots. Il est possible de séparer les architectures actuelles selon l'approche employée pour représenter les

comportements du robot. On peut dénombrer deux principales formes de représentation des comportements, soit la décomposition par fonctions (représentation explicite ou approche constructive) et la décomposition par activités (représentation implicite ou approche "computationnist") (Brooks, 1991; Arbib et Liaw, 1995; Stewart, 1995).

La décomposition par fonctions est une approche plus traditionnelle qui consiste à diviser la résolution d'un problème par l'emploie d'une suite de modules fonctionnels comme la perception, la création d'un modèle, la planification, l'exécution de la tâche et le contrôle des moteurs (figure A.2). Chaque module est exécuté de façon séquentielle en s'échangeant des descriptions symboliques du monde extérieur. Cette approche fait appel aux notions de la pensée et du raisonnement pour obtenir des comportements intelligents entre robots. Les inconvénients majeurs de cette approche concernent les difficultés reliées à la vérification du bon fonctionnement d'un tel système. De plus, il est laborieux d'ajouter un comportement au système une fois sa structure définie, étant donné qu'un tel ajout peut affecter tous les autres modules du système.

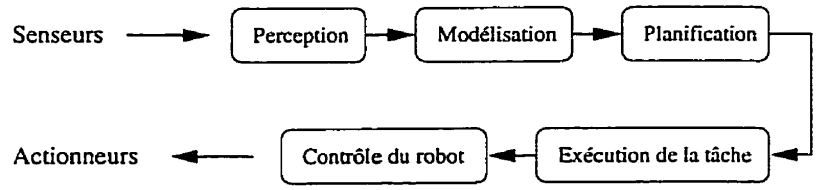



Figure A.2 : Décomposition par fonctions

La décomposition par activités ou par comportements est une approche dite réactive qui subdivise le problème en se basant sur les manifestations externes désirées

(figure A.3). C'est une approche plus biologique de la robotique. Chaque comportement est un modèle d'interaction avec le monde extérieur et connecte les senseurs aux actions du robot sans l'utilisation de représentation interne. Cette méthode se base sur l'hypothèse de l'encrage dans le monde physique (Brooks, 1990), qui stipule que pour qu'un système soit intelligent, il doit avoir comme représentation le monde physique tel quel sans utiliser de représentation symbolique.

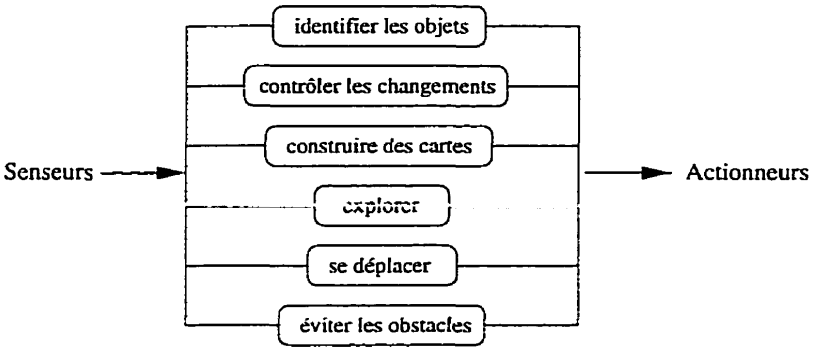



Figure A.3 : Décomposition par activités

L'approche de décomposition par activités n'étant pas conçue autour d'une unité centrale, cette approche a l'avantage d'être parallèlisable et d'être plus robuste aux défaillances. De plus, elle permet le développement de façon incrémentale (architecture dite buttom-up). Par contre, pour de telle approche, les buts des robots étant implicites, il est difficile de faire le design de l'ensemble des comportements élémentaires pour obtenir le comportement global désiré du robot (Arbib et Liaw, 1995).

Les architectures que l'on retrouve dans la littérature se situent entre ces deux architectures extrêmes. Les architectures les plus représentatives vont être décrites dans la section suivante.

### A.1.3 Architectures représentatives

#### A.1.3.1 Systèmes réactifs

Les systèmes "purement" réactifs relient des situations de l'état de leurs senseurs à des actions. On parle ici d'approche connexionniste puisqu'on suggère que les effecteurs soient reliés directement aux senseurs du robot, appliquant seulement un traitement minimal sur les données des senseurs. Dans (Mataric et al., 1995), on présente une expérimentation de coopérations robotiques avec des robots contrôlés uniquement en réagissant aux stimuli de leur environnement. Comme le montrent les résultats expérimentaux (Mataric et al., 1995), les systèmes purement réactifs ne permettent pas de bien coordonner les comportements des robots.

#### A.1.3.2 Essaim

Les essaims de robots (*swarm*) sont des systèmes comprenant un grand nombre de petits robots peu complexes. Contrairement aux équipes de robots, où chaque robot est attentif aux autres membres de l'équipe, dans un essaim les robots ne communiquent qu'avec les robots autours de lui (figure A.4), souvent uniquement dans un but de viabilité. Les robots agissent ici comme une entité unique, cherchant constamment à conserver la cohésion du groupe. Le niveau de cognition d'un tel système étant très faible, ces systèmes ne peuvent être utilisés efficacement que pour accomplir des tâches simples comportant une répétition d'une même activité.

Dans (Dudek et al., 1993), on présente une taxonomie pour ces systèmes robo-

tiques. Cette architecture à été utilisée dans des applications autres que la robottique, notamment en graphisme informatique comme dans (Reynolds, 1987), où l'on présente une méthode pour animer une volée d'oiseaux en décrivant des comportements réactifs individuels pour chaque agent.

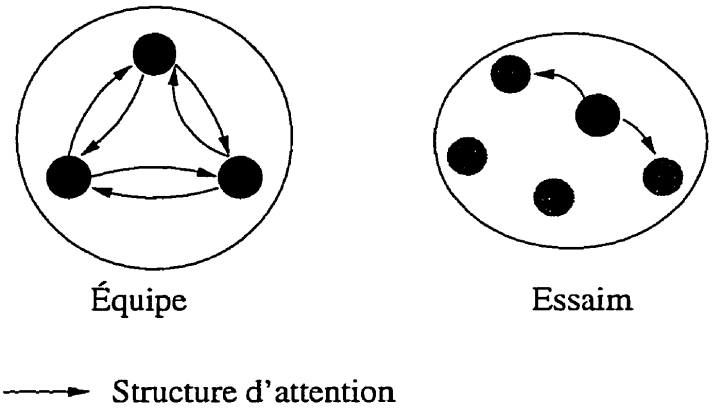



Figure A.4 : Différence entre équipe et essaim de robots

Dans (Fukuda, Iritani, Ueyama et Arai, 1994; Fukuda et Iritani, 1995; Ueyama et Fukuda, 1993), on présente le CEBOT (*Cellular Robotic System*) comme un système robotique décentralisé. Ce système d'essaim de robots s'inspire de l'organisation et des interactions entre les êtres cellulaires en biologie. Le système, composé de plusieurs cellules, s'organise dynamiquement en améliorant sa configuration. Après une expérimentation, chaque cellule est reconfigurée de manière centralisée pour améliorer la coordination et la coopération du groupe.

### A.1.3.3 Subsumption

L'architecture "*subsumption*" (Brooks, 1986) est une architecture dite descendante où la subdivision est faite sur la base des manifestations externes désirées. Son approche est dite basée sur le comportement (behavior-based), et consiste à créer des modules indépendants permettant d'implanter les comportements de base du robot.

Cette architecture ne fait pas ou peu usage de représentation interne, et nécessite un échantillonnage élevé des données de feedback, ce qui la rend mal adaptée à la vision. Plusieurs types d'architecture correspondant à des extensions de l'architecture de Brooks ont été développés.

Dans (Kuniyoshi, Riekki, Ishii, Rougeaux, Kita, Sakane et Kakikura, 1994), on donne une architecture de contrôle pour des robots coopérant par observation. Son architecture est dérivée de l'architecture de Brooks. Pour utiliser la vision, il introduit des représentations intermédiaires de l'environnement perçu. Les comportements utilisent ces représentations comme moyen de communication entre les différents comportements du système.

Dans (Wang et al., 1994), on présente une extension de cette architecture, que l'on nomme "Behavior-based multiple robot system with host". Ici, les divers modules de comportement ne se basent plus uniquement sur les senseurs du système pour déterminer leurs actions, mais utilisent aussi des informations supplémentaires provenant d'un système de communication.

Dans (Taipale et Hirai, 1993), l'approche de contrôle utilise une organisation

hiérarchique de type "master/slave" entre les robots. Les comportements des robots sont organisés de telle sorte que les modules de niveau supérieur ont comme entrées des données de contrôle (*subsumption signal*) provenant du robot maître.

### A.1.3.4 Champ de potentiel

L'idée d'utiliser des champs de potentiel pour contrôler un système robotique est venue de Kahtib (Khatib, 1985). Dans (Arkin, 1990; Arkin, 1992), on propose une approche de contrôle réactif basée sur les méthodes de planification de déplacements par calcul du champ de potentiel. Chaque schéma, ou comportement, produit un vecteur de vitesse pour le robot. La somme des différents vecteurs vitesses des schémas à une position donnée correspond à la direction que le robot doit suivre. Le champ de potentiel est calculé de façon dynamique. Cette architecture a été utilisée avec succès pour une tâche de collecte d'ordure dans (Balch, Boone, Collins, Forbes, Mackenzie et Santamaia, 1995).

Dans (Liscano et Manz, 1992), on présente une approche dérivée du champ de potentiel qui consiste à remplir une carte à partir de données prises par des senseurs ultra-soniques. Cette carte donne la direction où l'on peut rencontrer un obstacle aux alentours du robot. Le robot choisit ensuite sa direction de déplacement en utilisant cette carte dynamique. Une revue des différentes approches d'évitement d'obstacles par champ de potentiel peut être vue dans (Tilove, 1990)

L'approche d'impédance virtuelle (Arai et Ota, 1995) est une approche dérivée des

approches par champ de potentiel. Dans cette approche, des ressorts et des amortisseurs virtuels sont utilisés comme attracteurs. Dans (Ota et al., 1995), l'utilisation de cette approche se fait par la planification des déplacements d'un objet en produisant des forces et des moments virtuels.

#### A.1.3.5 ALLIANCE/L-ALLIANCE

L'architecture ALLIANCE, décrite dans (Parker, 1994*a*), est une architecture basée sur les comportements pour le contrôle de robots hétérogènes fonctionnant en coopération dans des missions comportant des tâches simples. Dans cette architecture, le choix des comportements est déterminé par un processus de motivation. L'emphase a été mise sur la tolérance aux fautes, et la récupération du système après erreurs. Pour ce, deux attitudes ont été implantées sur les robots, soit l'impatience (de voir le travail s'accomplir) et le consentement (face à son incapacité à remplir une tâche).

#### A.1.3.6 Intelligence artificielle distribuée

L'intelligence artificielle distribuée est un domaine de l'intelligence artificielle qui adresse le problème de l'interaction entre agents ou entre robots pour augmenter l'efficacité du système. Dans (Genesereth et al., 1986), on regarde les systèmes multi-agents en traitant de la coopération entre robots ne pouvant communiquer entre eux. On exploite l'information de rationalité qu'un robot a des autres robots et la similarité entre les robots pour choisir les actions des robots. Un robot agit de façon rationnelle

s'il choisit toujours les actions qui sont dominantes, donc qui, à coup sûre, donnent une plus grande utilité au robot.

#### A.1.3.7 Autres

Plusieurs autres approches furent présentées ces derniéres années. Dans (Noreils, 1992; Noreils, 1993), on présente une architecture décentralisée dotée d'une organisation hiérarchisée, où un *leader* décompose les tâches à exécuter et les réparti dans l'équipe. Dans (Kosecka, Christensen et Bajcsy, 1995), on modélise les comportements des robots selon la théorie des systèmes à événements discrets. Une architecture hybride entre l'architecture comportemental et cognitive, incorporant la vision, est présentée dans (Hugli, Tieche et Facchinetti, 1994). Dans (Ishida et al., 1994; Ozaki et al., 1993), des fonctionnalités réduites ou manquantes de certains robots sont complétées par d'autres robots ou activées par le biais de négociations inter robot.

### A.1.4 Analyse des performances

Pour évaluer l'optimalité de l'utilisation des ressources d'un système robotique, on propose une méthode d'analyse "information invariant framework" (Donald, Jennings et Rus, 1994; Rus, Donald et Jennings, 1995). Cette méthode permet d'analyser l'efficacité de l'architecture d'un système robotique en terme de ressources matérielles utilisées pour exécuter une tâche. Dans (Rus et al., 1995), on cherche un protocole pour

réorienter un objet avec des robots utilisant des ressources minimales. Dans (Donald et al., 1994), on décrit l'information nécessaire pour accomplir la tâche de pousser une boîte avec deux robots. Dans (Brown et Jennings, 1995), on montre qu'il est possible de redistribuer les ressources sur plusieurs robots tout en gardant la même fonctionnalité.

### A.1.5 Comportements émergents

L'émergence est un processus prédominant en intelligence artificielle, où un comportement surgit, à un certain niveau, des interactions produites à un niveau inférieur. On pourrait considérer la température comme un phénomène émergent. En effet, la température n'existe que pour un ensemble de molécules interagissant entre elles. Le phénomène de température est inexistant pour des molécules prises individuellement.

Les systèmes à comportements émergent sont donc des systèmes où un comportement global pour le groupe émerge des différentes interactions élémentaires entre les robots du groupe. Ce concept est généralement employé pour les architectures basées sur une décomposition par activités comme les systèmes réactifs, les essaims de robots et l'architecture *subsumption*, pour ne nommer que les plus importants.

À titre d'exemple, dans (Reynolds, 1987), on étudie les comportements d'une volée d'oiseaux émergent des comportements individuels des oiseaux. Dans (Gaussier et Zrehen, 1994), un ensemble de robots dotés de comportements simples, produit un comportement complexe consistant à regrouper en tas des objets.

### A.1.6 Apprentissage

L'apprentissage peut apparaître de plusieurs manières dans un système robotique. Il peut prendre la forme d'une méthode utilisée pour connaître l'environnement du robot, souvent représenté sous la forme de cartes comme dans (Singh et Fujimura, 1993). Il peut servir à l'évaluation du modèle inverse ou direct du système robotique. Le modèle direct permettra au système de prédire la sortie pour un état et une entrée donnée comme c'est le cas pour le filtre de Kalman (Chui et Chen, 1991). Le modèle inverse permettra au système robotique de choisir une action pour obtenir une sortie désirée connaissant l'état actuel du robot. Le contrôle adaptatif (Slotine et Li, 1987) est un exemple de ce type d'apprentissage. Enfin, certaines architectures (Steels, 1994) permettent d'apprendre de nouveaux comportements ou de choisir parmi un ensemble de comportements pour bien interagir dans le monde.

Il serait possible de classifier les systèmes à comportements émergents comme des systèmes à apprentissage, mais l'apprentissage à l'aide de ces systèmes correspond plutôt à une manipulation de paramètres. L'apprentissage par renforcement est sans doute la méthode la plus répandue. D'autres méthodes d'apprentissage se basent sur l'évolution biologique, comme c'est le cas pour les modèles génétiques.

#### A.1.6.1 Apprentissage par renforcement

L'apprentissage par renforcement est une classe de méthodes d'apprentissage où l'agent s'adapte en se basant sur la contre-réaction extérieure provenant de l'envi-

ronnement. La contre-réaction reçue est interprétée comme renforcement positif ou négatif au modèle d'interaction. C'est donc un processus de récompense/punition où une action ayant un résultat positif est récompensée, et où cette même action est punie pour un effet négatif (Mataric, 1994). Des méthodes tels le *Q-learning* (Watkins, 1989), le contrôle adaptatif (Slotine et Li, 1987), ou les réseaux de neurones (Hinton, 1989) font partie de la classe des méthodes d'apprentissage par renforcement.

#### A.1.6.2 Évolution biologique

Plusieurs techniques inspirées de l'évolution biologique sont utilisées pour l'apprentissage des robots. Les algorithmes génétiques font, entre autre, partie de ce groupe de techniques.

**Génétique** La programmation génétique est une classe d'algorithmes génétiques pour l'apprentissage (Koza, 1990). Le principe de la programmation génétique consiste, étant donné une population de programmes, à conserver ceux satisfaisant le mieux une fonction d'évaluation. Des opérateurs de mutation et de croisement sont appliqués sur les meilleurs individus pour générer la nouvelle population de programmes (Koza, 1990). Le principal problème rencontrer dans l'application des algorithmes génétiques vient du fait, qu'étant donné qu'ils opèrent sur des données abstraites du problème, il est nécessaire d'avoir un bon modèle des robots et de l'environnement pour que ces algorithmes soient efficaces.

Dans (Dorigo, 1995), on présente un GBML (*Genetics-based machine learning*), un

algorithme d'apprentissage de la classe des algorithmes d'apprentissage par renforcement, pour générer des lois de contrôle. Dans (Steels, 1994), on donne un mécanisme de sélection, le selectron, qui permet l'évolution de comportements robotiques en ligne sans nécessiter une fonction d'évaluation précise.

# Annexe B

# Revue bibliographique sur le contrôle de systèmes non holonomiques

L'état actuel de la recherche sur les méthodes de contrôle de systèmes non holonomiques est présenté ici.

## B.1 Analyse de stabilité

Pour valider une méthode de contrôle, il est nécessaire de faire une étude de stabilité de la dynamique du système sous les lois de contrôle développées. Traditionnellement, on utilise la seconde méthode de Lyapunov pour faire cette analyse de stabilité (Gulder et Utkin, 1994). Cette méthode est applicable aux systèmes

linéaires et non linéaires. Elle a l'avantage de permettre l'évaluation de la stabilité d'un système différentiable en utilisant les bonnes formes des équations, sans pour autant connaître explicitement la solution. L'idée générale de la méthode consiste créer une fonction d'énergie représentant l'état d'avancement de la tâche. Pour que le système soit stable, il faut s'assurer que le contrôle appliqué au système fasse toujours décroître cette fonction d'énergie.

Soit un système dynamique décrit par $\dot{\chi} = f(\chi, \mu)$, ayant $\chi \in \Re^n$ comme vecteur d'état et $\mu$ comme lois de contrôle. L'approche consiste à se créer une fonction d'énergie définie positive $V(\chi)$, dont la dérivé partielle première est continue et avec $V(0) = 0$. Cette fonction d'énergie peut simplement consister en une distance quadratique du vecteur d'état actuel relative au vecteur d'état à létat d'équilibre. Pour que le système soit stable sous $\mu$, il faut que le taux de changement de la fonction d'énergie soit strictement négatif $\dot{V} < 0$ (Kalman et Bertram, 1960). De cette façon, même perturbé, le système va continuer à perdre de l'énergie jusqu'à l'état de repos. Il est bon de noter que cette méthode, quoique très générale, n'apporte aucune information quant aux performances du contrôleur étudié.

## B.2 Contraintes mécaniques

La majorité des systèmes mécaniques sont soumis à un ensemble de contraintes de position et d'orientation. Ces contraintes, dites holonomiques, réduisent l'espace des configurations du système. Ce sont des contraintes intégrables pouvant être

modélisées par des équations de la forme suivante :

$$h_i(q) = 0. \tag{B.1}$$

où $q$ est le vecteur des coordonnées généralisées (vecteur de configuration).

Les contraintes non holonomiques sont des contraintes non intégrables impliquant les paramètres de configuration et leurs dérivés. Ces contraintes réduisent la dimension de l'espace des déplacements réalisables (l'espace des directions de la vitesse) pour une configuration donnée. Elles permettent donc de décrire les directions vers lesquelles le système ne peut se déplacer. On peut modéliser ce type de contraintes par des équations de la forme :

$$w(q)\dot{q} = 0 \tag{B.2}$$

## B.3 Contrôle stable pour les systèmes non holonomiques

Un problème de contrôle non holonomique largement utilisé dans la littérature est celui de trouver un contrôle stable permettant de déplacer un véhicule jusqu'à une configuration spécifiée (position/orientation). Les robots mobiles classiques (unicycle, bicyclette) ayant des contraintes non holonomiques sont des systèmes possédant trois degrés de liberté, mais seulement deux entrées de contrôle. Le fait que la dimension du vecteur de contrôle soit plus petite que celle du vecteur d'état fait en sorte

que le système n'est pas contrôlable directement, une fois linéarisé par rapport à son point d'équilibre.

Selon le célèbre théorème de Brockett (Brockett, 1983), pour des systèmes de la forme de l'intégrateur non holonomique décrit par (B.3),

$$\begin{cases} \dot{x} = u, \\ \dot{y} = v, \\ \dot{z} = xv - yu. \end{cases} \tag{B.3}$$

il n'existe pas de loi de contrôle continue $(u, v) = (u(x, y, z), v(x, y, z))$ qui rend l'origine du système asymptotiquement stable. Le contrôle d'un véhicule non holonomique dans l'espace Cartésien fait parti de cette classe de problèmes. On peut donc affirmer qu'il n'existe pas de loi de contrôle continue et invariante dans le temps qui permet de stabiliser un robot mobile de cette forme.

Pour contourner le théorème de Brockett, plusieurs méthodes ont été développées. Une des idées employées fréquemment dans la littérature consiste à générer des contrôles variant dans le temps (Samson, 1993), notamment en utilisant des sinusoïdes comme contrôle (M'Closkey et Murray, 1993; Murray et Sastry, 1993). D'autres approches consistent à utiliser des contrôles de forme discontinue, soit par mode de glissement (Bloch et Drakunov, 1994; Gulder et Utkin, 1994; Bloch et Drakunov, 1995), soit en modifiant la représentation du système d'état (Astolfi, 1994; Aicardi et al., 1995; de Wit et Sørdalen, 1992; Sørdalen et de Wit, 1992), ou en utilisant

d'autres approches (Kolmanovsky et Reyhanoglu, 1994; Alamir et Khennouf, 1995). Enfin, certains utilisent des méthodes numériques (Lizarralde et Wen, 1996) pour contrôler leur système.

## B.4 Contre-réaction variant dans le temps

Dans (Samson, 1993), on montre qu'il est possible de stabiliser un système de la forme $\dot{X} = B(X)U$ avec $dim(X) > dim(U)$, en utilisant une contre-réaction qui dépend aussi de la variable du temps. Pour un système décrit par (B.4)

$$\begin{cases} \dot{x_1} = u_1(1 + \alpha(x_1, x_3)x_2), \\ \dot{x_2} = u_2, \\ \dot{x_3} = c(x_1, x_3)u_1x_2. \end{cases} \tag{B.4}$$

ils utilisent une loi de contrôle de la forme (B.5)

$$\begin{cases} u_1 = -k_t - g_1(X,t)(x_1 + k(x_3,t)), \\ u_2 = -u_1[(x_1 + k(x_3,t)(a + c^T k_{x_3}) + c^T x_3] - g_2(X,t)x_2. \end{cases} \tag{B.5}$$

avec $k_i$ étant la dérivé partielle de $k$ relative à $i$. Ils montrent que pour certain type de fonction $k(x_3,t)$, la fonction de Lyapunov $V(X,t) = \frac{1}{2}((x_1 + k)^2 + {x_2}^2 + \|x_3\|^2))$ est non-croissante et que les variables d'état convergent asymptotiquement vers zéro.

## B.5 Utilisation de sinusoïdes

Dans (Murray et Sastry, 1993; Teel, Murray et Walsh, 1992), on propose une procédure pour générer les actions d'un contrôle en boucle ouverte pour des systèmes non holonomiques. Les robots utilisent comme contrôle des fonctions sinusoïdales de différentes fréquences. La contrôlabilité d'un système de la forme $\dot{x} = \sum g_i(x)u_i$ peut être décrite avec l'algèbre de Lie appliquée sur le champ de vecteur $g_i$. Pour un déplacement infinitésimal sur deux champs de mouvement le long de $g_1, g_2, -g_1, -g_2$ pour un temps $\epsilon$ , on peut montrer que le déplacement net dans le vecteur d'état peut être représenté par (B.6)

$$\epsilon^2[g_1, g_2](x_0) + o(\epsilon^3), \tag{B.6}$$

$[g_1, g_2]$ étant le crochet de Lie entre le vecteur $g_1$ et $g_2$ défini par (B.7)

$$[g_1, g_2] = \frac{\partial g_2}{\partial x} g_1 - \frac{\partial g_1}{\partial x} g_2 \tag{B.7}$$

Ainsi, le crochet de Lie peut être vu comme un déplacement infiniment petit le long d'un carré défini par deux vecteurs tangents (figure B.1). Selon le théorème de Chow (Murray et Sastry, 1993), un système est contrôlable s'il est possible de se déplacer dans n'importe quelle direction en utilisant le déplacement dans les crochets de Lie. Il montre que les fonctions sinusoïdales donnent un contrôle qui permet des déplacements dans n'importe quelle direction.

On montre aussi que certains systèmes canoniques ne peuvent pas être contrôlés

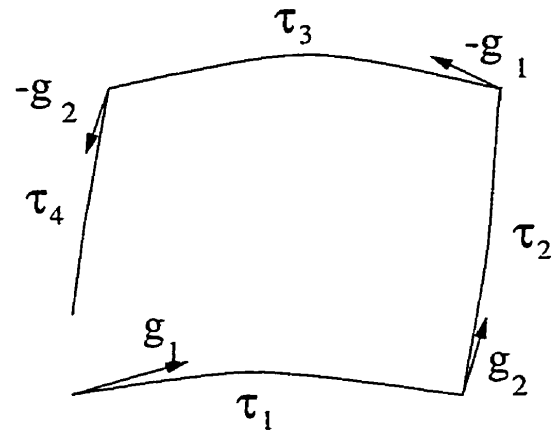



Figure B.1 : Déplacement infinitésimal

par des sinusoïdes, mais pourraient l'être en utilisant des combinaisons de sinusoïdes de différentes fréquences pour chaque entrée, ou d'autres fonctions périodiques plus complexes. Il prouve que, pour la classe des systèmes chaînés, il est toujours possible de faire un contrôle stable en utilisant des fonctions sinusoïdales.

Dans (M'Closkey et Murray, 1993), on utilise un contrôle composé de sinusoïdes de fréquences variées pour contrôler un système de puissance d'ordre trois. Un système non holonomique d'ordre trois dans la forme de puissance peut être représenté par les équations (B.8)

$$\begin{cases} \dot{x_1} = v_1, \\ \dot{x_2} = v_2, \\ \dot{x_3} = x_1 v_2. \end{cases} \tag{B.8}$$

Il étudie les propriétés de convergence de ce type de contrôleur de faible dimension en utilisant une technique d'analyse par échelle de temps multiple. La procédure de perturbation "two-timing" qu'ils utilisent, leur permet d'observer et d'évaluer la

convergence de façon qualitative d'un système lorsque celui-ci est perturbé.

## B.6 Contrôle discontinu

### B.6.1 Mode de glissement

Le contrôle dans le mode de glissement peut être utilisé pour contourner le théorème de Brockett (Bloch et Drakunov, 1994; Gulder et Utkin, 1994; Bloch et Drakunov, 1995). L'idée générale du contrôle dans le mode de glissement est de forcer un système dynamique à restreindre ses déplacements dans un manifold que l'on nomme surface de glissement $s(\chi)$. Ceci est obtenu en dirigeant la trajectoire du système vers le manifold en appliquant un contrôle de chaque côté de la surface, et dirigé vers celle-ci (Gulder et Utkin, 1994). Le principal avantage de ce mode de contrôle est qu'il permet de découpler un problème de dimension élevé en sous problèmes de dimension réduite. De plus ce type de contrôle est généralement très robuste.

Le contrôle par mode de glissement est largement employé dans la littérature (DeCarlo, Zak et Matthews, 1988) sous le nom de contrôle à structure variable. C'est un contrôle par contre-réaction où les gains commutent entre deux valeurs selon certaines lois. Ils utilisent des lois de contrôle de commutation rapide pour diriger la trajectoire du système d'état dans un espace d'état choisi. Le contrôle cherche à maintenir le système sur cet espace d'état que l'on nomme surface de glissement ou de commutation. On l'appelle ainsi puisque, si la trajectoire est située au dessus de la surface

de glissement, le contrôle modifiera ses gains pour permettre à la trajectoire de descendre vers la surface, de même, pour une trajectoire située en dessous de la surface, les gains seront modifiés pour permettre à la trajectoire de monter jusqu'à la surface de glissement. Ainsi, on dit qu'un mode de glissement existe si la vitesse du vecteur d'état est toujours dirigée vers la surface. Pour qu'un mode de glissement existe, il faudrait que le contrôleur soit capable de changer ses gains à une vitesse infinie lorsque le système est près de la surface de glissement, ce qui n'est pas envisageable. Un zone morte est ainsi ajoutée, impliquant ainsi des effets oscillations autour de la surface de glissement. Ce phénomène d'oscillation est rencontré dans d'autres types d'approche nécessitant des changements rapides sur les valeurs de contrôle, notamment lorsque l'on utilise des sinusoïdes comme commandes. Si l'on à un système linéaire simple représenté par (B.9)

$$\dot{x_1} = x_2, \dot{x_2} = \mu, \tag{B.9}$$

et que l'on veut maintenir la trajectoire sur une surface de glissement définie par $\omega(x_1, x_2) = s_1 x_1 + x_2$, on peut, en utilisant un contrôle de la forme $\mu = sgn[\omega(x_1, x_2)]$ avec (B.10),

$$sgn(\omega) = \begin{cases} 1 & \omega > 0, \\ -1 & \omega < 0, \end{cases} \tag{B.10}$$

satisfaire aux exigences dans la mesure où les commutations de gains peuvent se faire dans un laps de temps très court.

(Gulder et Utkin, 1994) présente une approche pour le contrôle d'un véhicule, modélisé par un système constitué de deux roues reliées par un essieu. La modélisation d'un tel système peut être simplifié par le modèle d'un uni-cycle. Il se définit une fonction de Lyapunov $V(x, y)$ dont le gradient est dirigé vers la surface de glissement. Ils décrivent un contrôle qui permettra au vecteur vitesse d'être colinéaire avec le champ de gradient, et permettra ainsi au système de converger vers la position et l'orientation désirées. Le contrôleur est tel que, lorsque le véhicule n'est pas orienté dans la bonne direction relativement au but, le déplacement du véhicule doit être fait en marche arrière. Ils montrent que, pour des fonctions de Lyapunov correspondant à des trajectoires paraboliques, leur système converge de façon exponentielle.

(Bloch et Drakunov, 1994) présente des lois de contrôle invariantes dans le temps pour résoudre le problème de contrôle de l'intégrateur non holonomique qui correspond au système d'équations (B.3). Ce sont des lois discontinues basées sur l'approche de contrôle par mode de glissement. Des lois de contrôle satisfaisant le critère de convergence de Lyapunov sous certaines conditions sont construites (B.11). Ces lois fonctionnent lorsque le système est à l'extérieur de la région parabolique $P$ définie par l'inégalité $\frac{\alpha}{2}(x^2+y^2) > |z|$. Lorsque le système n'est pas dans cette région $P$, une loi de contrôle constante est appliquée, ce qui permet au système de quitter la région en temps fini.

$$\begin{cases} u = -x + \alpha y sign(z), \\ v = -y - \alpha x sign(z), \end{cases} \tag{B.11}$$

Dans (Bloch et Drakunov, 1995), on étend la méthode de contrôle par mode de glissement pour les systèmes non holonomiques présentés dans (Bloch et Drakunov, 1994) pour faire un suivie de trajectoire. On représente le système d'état comme étant une distance à la position sur la trajectoire pour un instant donné. Il présente un contrôle discontinu permettant au système d'approcher une trajectoire avec une erreur inférieure à une certaine constante $\epsilon$, dans un temps fini $t_1$.

### B.6.2 Changement de représentation

Dans (Astolfi, 1994), on donne les conditions suffisantes pour stabiliser un système non holonomique ayant $n$ états et $n-p$ entrées. Il montre que pour qu'un système ait pareilles conditions, il faut qu'il possède un certain type de discontinuité. Il présente différents types de changements de coordonnées pour transformer un système continu en un système discontinu. La transformation de coordonnées doit avoir la propriété de générer de grands changements dans les variables d'état lorsque l'on a un petit déplacement près du point de convergence. Il montre que le système de coordonnées polaires, de même que le processus $\sigma$, permettent d'augmenter la résolution autour d'un point de discontinuité. Il présente un contrôleur discontinu qui stabilise un robot mobile non holonomique en faisant une transformation polaire du système d'état.

Dans (Aicardi et al., 1995), on contourne le problème de Brockett en changeant la représentation des équations d'état du système. On transforme ici la description du système de coordonnées Cartésiennes à une représentation en coordonnées polaires.

En faisant un choix judicieux des variables d'état, on montre qu'il est possible de garantir la propriété de stabilité globale pour un véhicule uni-cycle, en utilisant une loi de contrôle continue. La loi de contrôle est développée en utilisant la seconde méthode de Lyapunov pour garantir la convergence du système. Une extension de leur méthode est aussi décrite pour permettre de naviguer en suivant une trajectoire définie par une courbe paramétrique ou en suivant des points de référence (via points).

Dans (de Wit et Sørdalen, 1992), un changement de variable permet de rendre l'origine d'une voiture dans un espace Cartésien stable de manière exponentielle, à partir de n'importe quelle condition initiale dans l'espace d'état. La transformation de variable amène deux types de discontinuité au système. Il prouve que, pour certaines lois de contrôle, le système ne peut rester dans ces discontinuités ou même les atteindre et que le système converge de manière exponentielle. Une extension de l'approche donnant un contrôle stabilisant relativement à un point arbitraire est présentée dans (Sørdalen et de Wit, 1992).

### B.6.3 Autres approches discontinues

Dans (Kolmanovsky et Reyhanoglu, 1994), on étudie le contrôle des systèmes représentés dans une extension de la forme de puissance (B.8), obtenue par l'ajout d'un second contrôle (B.12)

$$\begin{cases} \dot{v}_1 = u_1, \\ \dot{v}_2 = u_2. \end{cases} \tag{B.12}$$

Il développe une stratégie de contrôle discontinu qui permet le contrôle stable en temps fini des systèmes de la forme (B.8, B.12). Leur méthode se base sur le fait que les changements apportés aux variables $(x_3, x_4, ..., x_n)$, lorsque les variables de base $(x_1, x_2)$ parcourent une boucle fermée, dépendent uniquement de la géométrie de la trajectoire sur les variables de base. Ainsi, en choisissant un ensemble adéquat de $n-2$ trajectoires rectangulaires, il est possible de contrôler les variables $(x_3, x_4, ..., x_n)$. Leur contrôle comporte deux niveaux, un superviseur d'événements discrets et un ensemble de $n-2$ contrôleurs par contre-réaction de bas niveau.

Dans (Alamir et Khennouf, 1995), on propose un contrôleur discontinu invariant dans le temps basé sur la stratégie de contrôle prédictif "receding horizon". Cette méthode permet de formuler le contrôle comme étant un problème de résolution d'un programme quadratique. Le contrôle ainsi généré est un contrôle semi-constant par morceau et permet de contrôler des systèmes formulés dans la forme de puissance.

## B.7 Méthodes numériques

Il existe plusieurs méthodes de planification de trajectoire itératives pour des systèmes non holonomiques. Dans (Lizarralde et Wen, 1996), on montre qu'il est possible de transformer une de ces méthodes de planification en un contrôleur en boucle fermée. Il dérive un contrôle qui permet de minimiser l'erreur de la configuration finale par rapport à celle désirée. Il s'agit donc d'un problème d'optimisation par moindre carré ("best-step Newton-Raphson method"), où l'objectif est de modifier itérativement la

commande pour que l'erreur sur la configuration finale converge vers zéro. Cet algorithme a été étendu en incluant des contraintes d'inégalité pour permettre l'évitement d'obstacles. Pour transformer la planification en contrôle, il exécute simultanément les itérations de l'algorithme et contrôle le véhicule dans l'itération courante.