Titre: Title:	Interactions entre robots : aspects dynamiques et sensoriels
Auteur: Author:	Christian Zanardi
Date:	1998
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Zanardi, C. (1998). Interactions entre robots : aspects dynamiques et sensoriels [Thèse de doctorat, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/6800/

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI®

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UNIVERSITÉ DE MONTRÉAL

Interactions entre Robots : Aspects Dynamiques et Sensoriels

CHRISTIAN ZANARDI
DÉPARTEMENT DE GÉNIE ÉLECTRIQUE ET DE GÉNIE INFORMATIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE PHILOSOPHIÆ DOCTOR (Ph.D.)
(GÉNIE ÉLECTRIQUE)
Octobre 1998

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée :

Interactions entre Robots : Aspects Dynamiques et Sensoriels

présentée par : ZANARDI Christian
en vue de l'obtention du diplôme de : Philosophiæ Doctor (Ph.D.)
a été dûment acceptée par le jury d'examen constitué de :

M. HURTEAU Richard, Ph.D., président
M. HERVÉ Jean-Yves, Ph.D., membre et directeur de recherche
M. COHEN Paul, Ph.D., membre et codirecteur de recherche
M. MERLO Ettore, Ph.D., membre
M. MILGRAM Paul, Ph.D., membre

Je dédicace ma thèse à ma mère, mon père, mon oncle et mon frère dont les sacrifices m'ont enseigné la vrai valeur de la vie.

Remerciements

Je voudrais remercier mon directeur de thèse Jean-Yves Hervé, pour son soutien durant l'ensemble de ma thèse. Merci également à mon co directeur Paul Cohen. Je remercie également l'ensemble des membres du jury pour avoir pris le temps d'évaluer mon travail et d'accepter d'assister à la soutenance. Je voudrais aussi remercier Alain Ayache de l'ENSEEIHT grâce à qui j'ai pu avoir un financement (bourse MRSE) pendant deux ans.

Je remercie aussi l'ensemble de mes amis et collègues à Polytechnique : V. Béranger pour ses coups de fouet bénéfiques, F. Labrosse et F. Labonte pour m'avoir supporté dans les mauvais et ce qui auraient pu être les bons moments et aussi pour l'ensemble des conseils que vous m'avez donnés. Sans oublier M. Akhloufi, P. Lacroix pour sa fantastique envie de tout savoir sur tout le monde, G. Girard, M. Pilon, P. Debanné, D. Fourt, F. Lathuilière, A. Lima, Nguyen Hong Hai et V. Polotski.

Bien sûr, je n'oublie pas tous mes camarades à l'ENSEEIHT : Particulièrement je remercie mon binôme pour m'avoir donné le goût de faire une thèse (je me vengerai aussi!), *etc.*

Je remercie également le réseau d'excellence IRIS, ainsi que les partenaires de la chaire en automatisation minière CRSNG-Noranda pour l'aide financière durant l'ensemble de ma thèse.

Résumé

Cette thèse présente l'étude de la modélisation des interactions qui surviennent lorsque plusieurs robots mobiles, équipés de senseurs visuels, partagent un même environnement.

Dans un premier temps, la modélisation des capacités motrices et sensorielles d'un robot mobile est envisagée. Le concept de Carte Dynamique est ainsi introduit pour représenter de façon géométrique la mobilité du robot dans son environnement. Ce concept est basé sur la théorie des régions atteignables d'un système dynamique. Les méthodes de construction des Cartes Dynamiques de plusieurs types de robot mobile sont décrites, ainsi qu'un algorithme d'apprentissage de la Carte d'un robot par lui-même. La relation intime existant entre la caméra du robot et ses capacités de déplacement est également explorée au travers du concept de Carte Dynamique.

Dans un deuxième temps, le concept de Carte Dynamique est utilisé pour générer une représentation des interactions entre les robots mobiles. Ainsi, à partir des conditions environnementales et des positions relatives des robots récupérées par les senseurs du robot considéré, la Carte Dynamique du robot auto-apprise par un robot particulier et les Cartes Dynamiques estimées des autres robots sont combinées. Cette combinaison permet de faire apparaître la distribution de l'espace atteignable du robot en fonction non seulement de ses capacités mais également de celles des autres robots dans le cadre d'une tâche à accomplir en coopération ou en compétition. Cette représentation est utilisée par la suite pour permettre la planification de trajectoires pour le robot considéré. L'exemple de la planification d'une trajectoire d'évasion par

un robot poursuivi est présenté.

Plusieurs méthodes d'apprentissage des Cartes Dynamiques sont également envisagées. Par ailleurs, les résultats obtenus par un système d'apprentissage d'une stratégie de poursuite par renforcement permettent de prouver l'utilité de la Carte Dynamique comme un outil d'analyse des interactions entre robots.

Un banc d'essai expérimental basé sur des robots radioguidés a été développé afin de mettre en pratique les concepts décrits dans le cadre de cette thèse. Une implantation du système planification basé sur les Cartes Dynamiques ainsi que les résultats expérimentaux associés sont par ailleurs présentés.

Abstract

This thesis presents a study of the interactions created by several vision-guided mobile robots that must operate within the same workspace, and therefore must coordinate their actions.

We have proposed the concept of Dynamic Map, which allows to represent the intrinsic dynamics capabilities of a vehicle, and therefore to control its displacements when the geometry of this map is known. This concept is based on the theory of reachable regions of a dynamic system. Our work on the concept of Dynamic Map has concentrated on (1) the construction and representation of the map, (2) the learning by a robot of its own map, and (3) describing the sensori-motor relation between the robot's camera and displacement using the Dynamic Map.

The exploitation of the Dynamic Map to model interactions between robots in the environment has been explored. The estimated Maps of the other robots are combined on the learned Map of the robot considered in order to define a new distribution of the reachable space of the robot according to a task to accomplish. This representation is then used to plan optimal trajectories in function of the robot kinetic capabilities as well as the other robots' and the environmental conditions. This representation is shown to be effective in pursuit/evasion purposes.

A complementary approach to the pursuit strategies learning for a mobile robot by reinforcement has proven the feasibility of the Dynamic Map as a tool to analyze interactions between robots.

An experimental setup based on radio-controlled robots has been developed to

valid the concept presented in this thesis. The planning procedure based on the Dynamic Map concept has been implemented on the testbed and various experimental results are presented.

Table des matières

Liste des tableaux

Liste des figures

Liste des annexes

Liste des symboles

$\dot{x}$ représente la dérivée par rapport au temps dx/dt de la variable x.

$\mathbf{x}$ indique qu'il s'agit d'un vecteur avec des élements désignés par $(x_1, x_2 \ldots)$.

$E(x)$ désigne l'espérance mathématique de la variable aléatoire x.

$Var(x)$ représente la variance de la variable aléatoire x.

atan2(y, x) fournit la valeur de l'angle polaire du point (x, y) entre $-\pi$ et π.

Chapitre 1

Introduction

Initialement, la recherche en robotique mobile s'est intéressée au problème de navigation autonome d'un seul robot dans un environnement statique structuré. Par la suite, des environnements dynamiques ont été considérés pour en arriver finalement à des études intégrant à la fois plusieurs robots évoluant dans des environnements dynamiques partiellement connus ou totalement inconnus.

Cependant, malgré les efforts investis dans l'étude des systèmes à multiples robots, le développement d'outils d'analyse et de représentation des interactions existant dans ces systèmes reste encore un problème en quête de solution. Nous voulons introduire dans cette thèse un concept de modélisation des capacités motrices d'un robot afin de représenter et d'analyser les interactions entre robots. Cette modélisation des interactions permet par la suite de générer des trajectoires à court terme, prenant en compte la situation courante des divers éléments de l'environnement et les capacités motrices du robot considéré. Un des aspects intéressants de cette modélisation réside dans son aptitude à relier la physique du robot (et donc ses capacités motrices) à son système sensoriel.

Nous allons donc tout d'abord présenter la problématique considérée, puis donner un aperçu de la solution proposée pour finalement présenter le cheminement de la thèse.

1.1 Problème considéré

Le contexte de ce travail se situe au niveau de l'étude des interactions qui se produisent lorsque plusieurs robots dotés d'un système visuel sont mis en présence dans un même environnement. Les robots agissent en coopération ou en compétition afin de résoudre une tâche commune. Les tâches considérées dans cette thèse seront typiquement compétitives. Un exemple de tâche de planification est la recherche d'une trajectoire d'évasion pour un robot poursuivi dans un environnement contenant des obstacles. Dans le cadre de ce travail, l'existence de communication explicite (autre que visuelle) entre les différents robots ne sera pas considérée. Nous faisons l'hypothèse que le système visuel des robots permet d'appréhender l'environnement. Le lien de communication sera assuré par le système d'acquisition d'images de chaque robot. En effet, les informations pertinentes sur l'environnement et les autres robots seront supposées disponibles lors des phases de planification de trajectoires.

Le concept de Carte Dynamique que nous allons proposer pour représenter et gérer les interactions dynamiques entre robots doit s'intégrer à un système plus général de planification. La Carte Dynamique permettra en effet une planification à court terme et réactive (c'est à dire sensible au changement de l'environnement et des actions des autres robots). La figure 1.1 présente une vue d'ensemble d'un exemple de système intégrant notre approche. Au niveau supérieur se situe généralement un planificateur global géométrique qui prendra en compte un point initial, un but final et éventuellement des contraintes supplémentaires, comme la longueur du chemin, ou la présence d'obstacles statiques dans l'environnement. Au second niveau, ce chemin global est découpé en un ensemble de tronçons sur lesquels le chemin est transformé pour prendre en compte les différentes contraintes non-holonomes[1] du robot. Notre travail concerne le niveau de base de cette pyramide: le robot obtient, à partir de ses senseurs, un ensemble d'informations sur la situation courante. Grâce à une

[1] Le terme sera introduit plus en détail dans le chapitre suivant. Considérons ici qu'il s'agit de contraintes sur le mouvement du robot

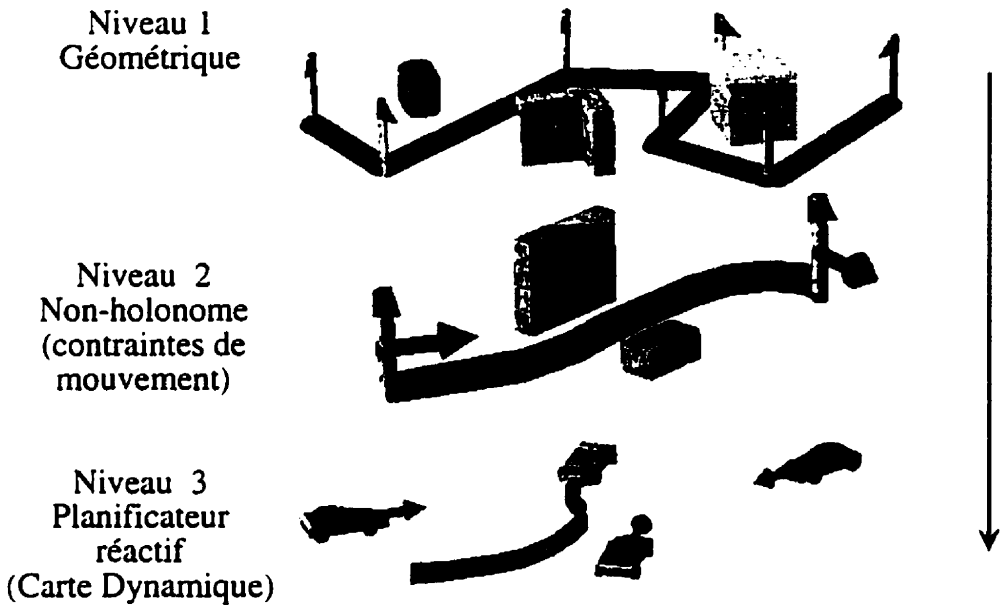


Figure 1.1 : Système pyramidal pour la génération de chemin et trajectoire englobant le concept de Carte Dynamique

interprétation de ces données. exprimée sous la forme de Carte Dynamique. il pourra alors planifier une trajectoire locale en accord avec la tâche qu'il faut accomplir.

Prenons l'exemple d'un robot devant se rendre d'une position initiale à une position finale tout en évitant les obstacles et les éventuels robots qui tentent de l'intercepter (ou simplement gênent son mouvement) en cours de chemin. Les deux premiers niveaux de la pyramide génèrent un chemin que le robot peut alors suivre. Lorsqu'un autre robot apparaît dans le champ de vision et que suffisamment d'information est collectée, il est nécessaire de planifier une trajectoire permettant d'éviter cette interférence et les obstacles détectés dans l'environnement. La Carte Dynamique permet l'intégration de l'ensemble de ces informations sur une même représentation géométrique. Une trajectoire en accord avec cette nouvelle représentation est alors planifiée. Une fois que la source de la perturbation est évitée, le robot peut faire appel aux planificateurs de plus haut niveau pour regénérer un chemin afin d'atteindre l'objectif final. après avoir repéré sa position dans l'environnement.

Nous allons maintenant présenter la solution que nous proposons pour représenter

les interactions entre robots.

1.2 Solution proposée

Nous allons introduire dans cette thèse le concept original de Carte Dynamique, permettant de représenter les capacités motrices d'un robot mobile en se basant sur le concept de régions atteignables[2]. Il s'agit d'une représentation permettant, à partir de l'état courant du robot considéré, de représenter chaque position de son espace de travail en termes d'accessibilité et d'efficacité d'accès, en fonction de ses capacités motrices. Ces informations sont combinées par la suite pour l'ensemble des robots considérés afin de représenter leurs interactions dans l'accomplissement d'une tâche.

Comme nous le verrons dans l'ensemble de la thèse, les principaux avantages de la Carte Dynamique pour la planification sont :

- de fournir un outil générique d'analyse des interactions. En effet, nous verrons qu'il est possible de représenter les capacités motrices d'un grand nombre de modèles dynamiques de robots par l'intermédiaire du seul concept de Carte Dynamique. La Carte Dynamique permet ainsi une indépendance structurelle appréciable, non seulement lorsque l'on veut considérer une population hétérogène de robots, mais également lorsque l'on désire apprendre les capacités motrices d'un autre robot sans pour autant connaître le modèle sous-jacent.

- de faire de la replanification dynamique de trajectoires. Puisque le but de la Carte est de générer des trajectoires efficaces à relativement court terme, cela permet de transférer la puissance de calcul en permettant à des processus lents de pouvoir s'exécuter complètement dés lors que la trajectoire est obtenue. Il est bien connu, par exemple, que les systèmes d'analyse d'images sont généralement lents et qu'il est nécessaire de planifier une trajectoire entre deux analyses.

[2]Elément classique de l'Automatique et de la théorie du contrôle optimal que nous introduirons dans le chapitre 3

Le robot ne devant généralement pas s'arrêter entre deux analyses (nous discuterons des différents modèles de contrôle visuel dans le chapitre 2), la Carte Dynamique permet de donner au robot un chemin à suivre dans le laps de temps nécessaire aux traitements des images.

- d'obtenir un système dont la complexité croît linéairement avec le nombre de robots. En effet, nous verrons dans le chapitre 4 que la planification de trajectoires est identique quelque soit le nombre de robots considéré, malgré le fait que chaque nouveau robot impose une combinaison supplémentaire avec la nouvelle Carte Dynamique. Beaucoup de méthodes ont une complexité algorithmique qui augmente de façon polynomiale, voire exponentielle avec le nombre de robots. En d'autres termes, rajouter un robot supplémentaire dans l'espace de travail augmente le temps de calcul de la solution d'un facteur (au mieux) polynomial (voir, par exemple, une description des complexités d'algorithme dans le livre de Latombe (Latombe 1991)). Par exemple, une méthode se basant sur l'utilisation des états de tous les robots devra considérer le produit cartésien des espaces de travail individuels de chaque robot.

- de générer des trajectoires correspondant à un modèle complexe de robots. Bien que dans le cadre de cette thèse nous n'ayons envisagé que des modèles relativement simples de robots mobiles, il est possible d'utiliser des modélisations plus complexes sans modifier la base théorique des Cartes Dynamiques.

- d'intégrer les capacités sensorielles aux capacités motrices du robot. L'imperfection sensorielle est souvent peu considérée et elle est souvent difficile à insérer dans des méthodes dites complètes. Comme nous le verrons dans le chapitre 3 la Carte Dynamique permet de relier plus intimement l'aspect sensoriel et le contrôle du robot pour l'étude du couplage visuo-moteur.

- d'intégrer les observations de l'environnement en fonction des capacités motrices des robots. En effet, la Carte Dynamique constitue une observation du monde

à un instant donnée en fonction des capacités motrices du robot. L'environnement est donc perçu par le robot suivant ses propres capacités sensorielles et physiques, pour ensuite être intégré aisément dans la Carte Dynamique.

- de permettre d'analyser des phénomènes de compétition (thème central de l'application des Cartes Dynamiques dans cette thèse) mais également de coopération (la problématique et l'application des Cartes Dynamiques à la représentation de ce problème seront présentées dans le chapitre 4)

La Carte Dynamique offre un point de vue unificateur à la planification de trajectoires (grâce, par exemple, à l'indépendance structurelle), dans le cas de plusieurs robots et de différentes tâches à accomplir, ainsi qu'à l'étude du couplage visuo-moteur pour un robot unique.

Il est nécessaire avant de poursuivre, de mentionner les différences qui existent entre notre approche et les approches purement réactives utilisant par exemple un réseau de neurones (plus de détails sur ces approches seront donnés dans l'ensemble du chapitre 2 et l'annexe B). Dans ce genre d'approche réactive, il est généralement facile de définir une action en fonction d'un seul événement dans l'environnement (comme nous le verrons dans le chapitre 5). Toutefois, lorsque le nombre de robots à considérer augmente, les lois reliant perception et action[3] deviennent complexes et il est généralement inévitable de connaître *a priori* le nombre d'événements[4] avant de procéder à l'apprentissage du couplage visuo-moteur. De plus, dans ces systèmes cités comme "schizophréniques" par Connell (Connell 1990), il est parfois difficile de prouver la convergence des algorithmes vers la résolution de la tâche désirée, où même de faire des analyses d'optimalité suivant les paramètres des modèles. La Carte Dynamique permet d'inclure directement des critères d'optimalité qui dépendent à la fois du robot et de la tâche. Des trajectoires non-holonomes satisfaisant ces critères

[3]Ces lois correspondent à la description de l'action qu'il faut appliquer en fonction des données acquises sur l'environnement pour résoudre la tâche

[4]Les événements désignent ici les données à extraire des senseurs

d'optimalité sont ensuite calculées à partir des informations combinées sur les Cartes.

1.3 Survol de la thèse

Dans le chapitre 2, nous allons explorer les différentes facettes du problème que nous considérons dans cette thèse au travers des recherches accomplies dans plusieurs domaines. Nous commencerons par évoquer le contrôle de robots mobiles à l'aide de senseurs visuels, puisque nous avons choisi d'équiper nos robots de caméras et de fermer la boucle de contrôle par une contre-réaction sensorielle. La planification de chemin et les nombreuses ramifications des algorithmes seront décrites par la suite. Les liens entre notre concept de Carte Dynamique et les algorithmes de planification seront donnés dans l'ensemble de ce chapitre. Finalement, un aperçu des systèmes à multiples robots sera introduit. Notre problème sera comparé aux méthodes déjà proposées dans la littérature.

Le chapitre 3 introduira le concept de Carte Dynamique pour la représentation des capacités motrices d'un robot. Nous évoquerons tout d'abord les méthodes de construction de ces Cartes pour différents types de robot. Par la suite, une représentation minimaliste de ces Cartes Dynamiques sera présentée, afin d'en faciliter l'auto-apprentissage par un robot ou l'apprentissage de la Carte Dynamique d'un autre robot par la vision. L'apprentissage des Cartes Dynamiques sera décrites dans le chapitre 5. Finalement, la relation étroite existant entre le senseur du robot et les capacités motrices du robot matérialisées par la Carte Dynamique sera explorée.

Dans le chapitre 4, nous décrirons comment le concept de Carte Dynamique peut être exploité afin de représenter les interactions entre des robots dans le cadre d'une tâche à accomplir. La façon d'utiliser cette représentation pour planifier les trajectoires sera ensuite introduite. Afin d'étendre les capacités des Cartes Dynamiques à un système temps-réel dans l'avenir, une modélisation appropriée sera présentée[5].

[5]Cette modélisation a pour but de pouvoir utiliser simplement des cartes de traitement d'images

L'exemple d'un robot devant planifier une trajectoire d'évasion sera utilisé pour montrer l'efficacité des Cartes Dynamiques en simulation. Les problèmes de la poursuite et de la coopération entre robots, ainsi que notre solution basée sur les Cartes Dynamiques, seront brièvement introduits à la fin de ce chapitre.

Le chapitre 5 présentera deux approches pour l'apprentissage de la Carte Dynamique d'un robot. La première approche est une alternative au point de vue réactif pour l'apprentissage d'une stratégie de poursuite par renforcement. Une revue bibliographique sur l'apprentissage par renforcement est donnée dans l'annexe B. Bien que cette méthode soit quasiment à l'opposé de l'approche par les Cartes Dynamiques, nous avons constaté d'importantes ressemblances dans les résultats. Nous verrons ainsi que la Carte Dynamique est un bon moyen d'expliquer et éventuellement de prédire les résultats obtenus, comme par exemple la formation de régions atteignables dans le plan image de la caméra d'un robot. La deuxième approche se base sur le volume de représentation décrit dans le chapitre 3. L'objectif est ainsi de transformer des trajectoires acquises par l'intermédiaire des systèmes sensoriels (le système d'observation central dans le cas de l'auto-apprentissage des Cartes et le système de vision embarqué pour l'apprentissage des Cartes d'un autre robot) pour trouver la forme adéquate du volume de représentation.

Finalement, notre effort pour construire un banc d'essai expérimental sera décrit dans le chapitre 6. Le but est de prouver qu'il est possible de créer un système flexible à faible coût, pour être utilisé dans le cadre d'études sur des coopérations de robots à guidage sensoriel. Nous montrerons comment nous avons pu baser ce banc d'essai sur des modèles réduits radioguidés pour construire des robots ayant une caméra active avec un système rudimentaire d'analyse d'images, une odométrie visuelle et une capacité de contrôle assurée par un réseau de stations de travail. Des exemples d'applications de notre banc d'essai expérimental seront présentés à la fin de ce chapitre. Finalement, nous montrerons que le système de planification basé

---

pour générer les déplacements du robot

sur les Cartes Dynamiques et décrit dans le chapitre 4 peut-être testé en temps-réel sur notre banc d'essai expérimental. La description de l'implantation du système ainsi que les conditions expérimentales seront introduites et plusieurs exemples de trajectoires exécutés par les robots dans des situations de poursuite-évasion seront par ailleurs présentés.

Dans la conclusion, nous résumerons les divers éléments originaux introduits dans cette thèse. Les extensions futures seront également discutées.

Chapitre 2

Revue bibliographique

Dans ce chapitre, nous allons présenter une étude bibliographique des travaux liés à cette thèse. Comme le sujet de ce travail englobe un grand nombre de domaines (allant de la vision pour les robots mobiles, jusqu'à la planification non-holonome, en passant par l'apprentissage par renforcement à base de réseaux de neurones et la collaboration de robots) la revue bibliographique est particulièrement extensive.

Nous présenterons tout d'abord les travaux liés à la vision comme un outil nécessaire à la navigation de robots mobiles. Par la suite, nous présenterons une revue détaillée sur les méthodes de planification pour des robots mobiles. Nous évoquerons ensuite les études portant sur les régions atteignables, thème central aux Cartes Dynamiques, avant de poursuivre par une bibliographie (légère par rapport à la masse d'articles existants) sur la collaboration de robots mobiles. Une revue bibliographique sur l'apprentissage par renforcement sera donnée dans l'annexe B.

2.1 La vision pour les robots mobiles

Pour qu'un robot soit autonome dans un environnement, il ne peut se baser complètement sur les données odométriques de ses capteurs. En effet les erreurs pour ce type de senseurs sont cumulatives et, après un temps relativement court, le

robot n'est plus capable de se repérer dans son environnement de façon fiable. Des senseurs externes sont donc très importants pour fournir une information supplémentaire de recalage. Deux types principaux de senseurs existent : actifs (comme un laser qui explore l'environnement) et passifs (comme une caméra qui observe simplement). Puisque nous avons fait le choix d'équiper nos robots de caméra, nous allons ici nous intéresser seulement aux travaux faits en vision par ordinateur avec des senseurs passifs.

2.1.1 Les premiers temps de la vision par ordinateur

Les senseurs devaient fournir initialement une carte de profondeur de l'environnement, afin de planifier ensuite un chemin dans l'environnement reconstruit. A cause de la complexité de la reconstruction, l'approche s'est ensuite étendue à des méthodes plus qualitatives et à des systèmes où vision et action sont plus intimement liés.

Dans (Marr 1982), Marr présente une des premières tentatives de formalisation de l'approche par reconstruction. Son étude est à trois niveaux :

- la théorie calculatoire (*quoi*).
- les représentations et les algorithmes (*comment*).
- l'implantation (*où*).

Son approche a généré une profusion d'articles. Horn (Horn 1986) résume assez bien les progrès qui ont été accomplis. Toutefois malgré l'énorme quantité de travail mise en œuvre, les systèmes réels se sont heurtés rapidement à une barrière calculatoire difficile à franchir.

Dans le cas d'analyse à la structure à partir du mouvement, on relève trois étapes fondamentales :

- l'étude de l'existence des solutions (Longuet-Higgins 1981) ;
- la recherche de l'unicité de ces solutions (Poggio et Koch 1985) :

- enfin la robustesse des méthodes de calculs (Tomasi et Kanade 1991).

Un exemple assez significatif des progrès dans le domaine est donnée par Tomasi et Kanade (Tomasi et Kanade 1991), où la reconstruction presque parfaite d'un objet est présentée. Les calculs sont faits directement dans un référentiel centré sur l'objet, ce qui permet d'utiliser une profondeur relative moins bruitée que la profondeur absolue par rapport au référentiel de la caméra. Toutefois, les résultats impressionnants cachent des défauts majeurs pour une utilisation dans le cadre d'un robot mobile. Principalement, le nombre d'images utilisées pour obtenir cette qualité est trop important pour s'appliquer à un quelconque système temps réel.

Dans (Baker et Bolles 1988, Baker et Garvey 1991), le volume temporel des images (x, y, t) est analysé directement pour trouver des informations intéressantes pour le contrôle d'un robot. Ce genre d'approche nécessite une grande puissance de calcul du fait de la grande masse d'informations à traiter.

2.1.2 Les indices visuels

Plutôt que de reconstruire complètement la scène ou les objets d'intérêt, on peut s'intéresser à des indices visuels simples à extraire des images, portant des informations pertinentes sur l'environnement où le robot évolue. Il s'agit d'une forme du principe d'effort minimum.

Par exemple, Nelson et Aloimonos (Nelson et Aloimonos 1989) introduisent un opérateur donnant la divergence (taux d'expansion) suivant une direction du flux optique (variation temporelle des points) permettant de savoir si un objet en mouvement indépendant se rapproche du robot. Ce type d'opérateur est un moyen simple de décider comment éviter un obstacle. Cet opérateur permet d'obtenir ensuite un autre indice précieux : le temps avant collision, qui représente le temps qu'un objet se déplaçant en direction du robot va mettre avant de l'atteindre. (Duric, Rosenfeld et Duncan 1994) présente une méthode de calcul de cet indice en reliant l'intégrale de la divergence sur une surface à une intégrale sur un contour fermé du flux op-

tique. Le calcul intégral permet généralement une plus grande robustesse par rapport au bruit. Dans (Cipolla et Blake 1992), Cipolla et Blake proposent une approche similaire pour calculer le temps avant collision et l'orientation des surfaces des objets à partir des valeurs des descripteurs du premier ordre du flux optique (divergence, rotationnel et déformation. Voir les travaux de Kœnderink *et al.* (Koenderink et van Doorn 1975, Koenderink 1986) pour une description mathématique formelle de ces descripteurs). Ces descripteurs ont également la propriété intéressante de fournir des bornes aux valeurs du temps avant collision (Subbarao 1990).

Dans (François 1991), après une segmentation en régions de mouvement homogènes, les différentes régions sont étiquetées suivant la prédominance des valeurs des descripteurs du flux optique. Ceci permet d'identifier par exemple des régions de danger potentiel afin d'y concentrer l'analyse.

L'article (Kundur et Raviv 1994) expose un indice visuel similaire au temps avant collision: le VTC ("Visual Threat Cue" ou indice visuel de danger) qui indique le changement relatif de distance. Cet indice est inversement proportionnel au temps mais ne dépend pas de la profondeur. C'est un avantage par rapport au temps avant collision. Les lignes de même valeurs de VTC dans l'espace permettent de faire une partition en zones de danger et de sécurité.

Cette partition donnée par le VTC se rapproche des résultats présentés dans le chapitre 3 lorsque la Carte Dynamique est projetée dans le plan image des caméras des robots. En effet, la Carte Dynamique va permettre de gérer des zones d'attention dans l'image pour le robot en fonction de ces capacités motrices.

2.1.3 Analyse d'images par rupture de contraintes

Il est très important de pouvoir distinguer le mouvement d'un objet du mouvement propre du robot. Thompson et Pong (Thompson et Pong 1990) présentent des méthodes basées sur le flux optique et la rupture de contraintes de rigidité. Le même type de contrainte est utilisée par Nelson dans (Nelson 1991). L'idée principale est

que lorsque le mouvement de l'utilisateur est connu. la vitesse 3D projetée en chaque point de l'image est contrainte de s'étendre sur une droite unidimensionnelle (dépendant seulement des paramètres de mouvement de l'observateur) dans l'espace des vitesses. Le flux normal (projection du flux optique suivant le gradient de l'image) est utilisé car il peut être extrait de façon plus fiable. L'ambiguïté que cela introduit remplace la contrainte de la droite par l'appartenance des valeurs du flux à une région dont les frontières sont connues. Le mouvement indépendant est simplement repéré par la rupture de cette contrainte.

L'article (Zhang, Weiss et Hanson 1994) utilise un test sur la consistance d'un système d'équations. Ce système relie les coordonnées des points dans l'image suivant une contrainte coplanaire. L'inconsistance indique la présence de points au dessus du plan et donc d'obstacles potentiels.

Il est possible d'utiliser le foyer d'expansion (point particulier d'où émanent les vecteurs de mouvement dans le cas d'une translation pure) pour extraire des informations de l'image. (Burger et Bhanu 1992) donne l'exemple d'un système de navigation entièrement basé sur le foyer d'expansion. Plusieurs algorithmes découplant l'effet de la translation et de la rotation sur le flux optique sont donnés. Ils peuvent ensuite interpréter et qualifier les différents vecteurs de mouvements par rapport à une région où doit se trouver le foyer d'expansion.

(Aloimonos et Duric 1994, Sinclair, Blake et Murray 1994, Fermüller 1993*b*) sont d'autres méthodes de calcul du foyer à partir des vecteurs de flux normaux, qui examinent également l'aspect de robustesse du calcul. Elles se basent sur la contrainte suivante : le foyer d'expansion se trouve toujours dans le demi espace opposé à la direction du vecteur de mouvement.

Dans le chapitre 6, nous allons présenter un algorithme de détection de mouvement basé sur cette approche de rupture de contraintes.

2.1.4 Vers des architectures visuelles intégrées

Cependant, la recherche d'indices dans l'image ne permet pas de réaliser des robots capables d'appréhender totalement le monde extérieur. Il est alors nécessaire d'envisager la construction d'architectures de systèmes robotiques permettant d'intégrer ces informations.

L'approche d'Aloimonos (Aloimonos 1990) est complètement à l'opposé du principe d'effort minimum. La vision étant trop complexe pour être considérée de façon générale, il faut identifier les tâches à exécuter par le robot et résoudre les problèmes de vision dans le cadre particulier de cette application. L'architecture d'un robot mobile simple, capable de réaliser des tâches complexes est présentée. Dans (Rivlin et Rosenfeld 1994), les auteurs ont décomposé l'attitude d'un robot en évitement d'obstacles, interception des objets et référence par rapport aux couloirs. Les différents comportements sont connectés avec un ordre de priorité.

Selon cette même méthodologie, (Fermüller 1993*a*) propose l'approche "synthétique" de conception de système robotique. Les points essentiels de la méthode sont la conception d'une hiérarchie de capacités visuelles et l'utilisation de méthodes qualitatives pour avoir plus de robustesse. Les capacités visuelles de base sont ici la recherche du mouvement propre (paramètres du mouvement de l'observateur) grâce aux calculs du foyer d'expansion et l'estimation du mouvement des objets.

Brooks (Brooks 1991*b*, Brooks 1991*a*) introduit l'architecture "subsumption". Elle est composée de comportements indépendants parallèles qui communiquent directement par perception et action avec le monde plutôt qu'entre eux. Le monde est ici considéré comme le dépositaire de l'information et les représentations intermédiaires n'ont plus alors de raisons d'exister (Horswill et Brooks 1988).

Connell (Connell 1990) donne un exemple de ce type de décomposition verticale du système de contrôle où la tâche de perception est distribuée à l'ensemble des comportements. L'esprit du robot est donné comme une collection d'impulsions schizophréniques qui se battent pour le contrôle du corps.

Deux autres types de théorie sont également apparu : La vision active et la vision animée. Ces théories tentent de franchir le pas existant entre perception et action. Bajcsy *et al.* (Bajcsy 1988, Bajcsy et Campos 1992) définissent les concepts de la perception active pour résoudre les problèmes "mal-posés" de la vision. Ballard et Brown dans (Ballard 1991, Bandopadhay et Ballard 1991, Grosso et Ballard 1993) présentent les 3 aspects fondamentaux de la vision animée : l'importance du contrôle de la fixation de l'attention ; le traitement séquentiel pour réduire la complexité calculatoire : l'apprentissage. L'utilisation d'un repère centré sur l'objet pour la reconnaissance plutôt que sur celui de la caméra est également une des idées clefs de ce concept.

Proposer une nouvelle architecture visuelle est une tâche bien au-delà du cadre cette thèse. Toutefois, le chapitre 3 expose la relation intime qui existe entre la Carte Dynamique et les senseurs du robot.

2.1.5 Le contrôle visuel

On peut approcher le problème du contrôle d'un robot différemment. Plutôt que d'extraire des images des informations et ensuite prendre des décisions sur le chemin à suivre et finalement envoyer une commande aux moteurs, on peut relier la perception et l'action de façon plus intime. En robotique classique, le contrôle est un concept bien maitrisé. Sanderson et Weiss (Sanderson et Weiss 1983) présentent les différents aspects du contrôle à partir de la vision.

Il y a deux types de contrôle possibles :

- basé sur l'image (extraction de caractéristiques saillantes de l'image). Ce type de contrôle est à contre-réaction non linéaire (à cause de la complexité du couplage entre les caractéristiques et les positions).

- basé sur la position qui est à contre-réaction linéaire (c.a.d. que la différence entre les valeurs de positions désirées et obtenues est linéaire), mais instable à

cause de la cinématique du robot à recalculer.

Il existe également deux types de structure de contrôleur :

- "regarder-puis-bouger" où il y a une boucle fermée sur le contrôle des joints, avec un point de vue statique (perception et action non-coïncidents) et dynamique (perception et action en parallèle à des temps d'échantillonnage différents).
- "suivi visuel": boucle fermée directement sur les caractéristiques sensorielles.

Lorsqu'on s'intéresse à des systèmes avec de multiples robots, il n'est pas possible de travailler dans le cadre du paradigme "regarder-puis-bouger". Un robot ne peut en effet rester statique et analyser son environnement alors que les autres robots sont toujours actifs. Nous avons donc choisi de considérer que nos robots sont toujours en mouvement quel que soit le temps nécessaire au traitement des images. Comme nous allons le montrer dans les deux chapitres suivants, la Carte Dynamique est bien adaptée à ce genre de paradigme.

Un exemple de système qui ne peut pas se baser sur une reconstruction "hors-ligne" de l'environnement est donnée dans (Anderson 1989). En effet, le système de contrôle d'un joueur de ping-pong ne peut séparer les phases de planification, suivi de trajectoire et perception. Malgré le fait que l'environnement soit très contrôlé, des problèmes de perception subsistent. Ce système est basé sur un grand nombre de connaissances *a priori*.

Pour faire le contrôle d'une voiture sur une route, une caméra regardant dans le sens de la marche est utilisée dans (Raviv et Hermann 1991). Les points singuliers (point de valeur de flux optique nulle) sont reliés alors directement aux paramètres de mouvement du robot par le biais d'équations différentielles. Un contrôleur sur les paramètres utilisant la position de ces points peut alors être construit pour suivre la route à une certaine distance. Les méthodes de type "suivi de route" (Crisman et Thorpe 1993) ont connu un grand succès grâce au travaux de Dickmanns (Dickmanns et Graefe 1988*b*, Dickmanns et Graefe 1988*a*) — basés sur une approche dynamique.

où le but est de traiter le plus rapidement possible le minimum d'information utile — et ceux de Pomerleau (Jochem et Pomerleau 1996) par l'intermédiaire de réseau de neurones.

Les deux articles (Coombs et Roberts 1993, Santos-Victor, Sandini, Curotto et Garibaldi 1993) parlent d'un même type de contrôle pour la navigation d'un robot mobile dans un environnement contraint. Les techniques sont inspirées par les système de guidage des abeilles qui ont tendance à équilibrer les flux optiques sur les zones périphériques opposées des yeux. Le contrôle est alors simplement formulé comme la différence des valeurs moyennes ou maximales du flux optique dans les deux zones périphériques opposées.

Une autre exemple d'approche, dans le cadre un peu différent de la coordination main-œil est donnée dans (Hervé 1993), avec une méthode de contrôle direct à partir d'images non calibrées et un apprentissage géométrique des configurations singulières.

2.2 Planification de chemin pour les robots mobiles

Nous allons donner dans cette section une vue d'ensemble des méthodes de planification usuelles.

2.2.1 Planification de base

La planification de chemin a pour but de trouver un ensemble de points par lesquels le robot doit passer pour aller d'une configuration initiale q_i à une configuration finale q_f sans que le robot ne rentre en contact avec un obstacle. Lorsqu'une relation entre le robot et sa position dans le temps est envisagé on parle alors de planification de trajectoire plutôt que de chemin. Latombe (Latombe 1991) et Huang et Ahuja (Hwang et Ahuja 1992) présentent des revues des travaux dans ce domaine.

Il est nécessaire de définir tout d'abord la différence entre des contraintes holonomes et non-holonomes.

Les contraintes holonomes peuvent être caractérisées par un ensemble d'équations prenant en compte seulement l'état du système. Cela signifie que les contraintes holonomes réduisent la dimension de l'espace de configuration permettant alors l'utilisation de méthodes usuelles (géométriques) de planification de chemin. Les contraintes de connexion entre les articulations d'un robot sont un exemple de contraintes holonomes (Craig 1989).

Une contrainte non-holonome est caractérisée par une équation non intégrable, fonction non seulement de l'état du système mais aussi des dérivées par rapport au temps des variables d'états. Parce que ces équations ne peuvent être intégrées, les contraintes non-holonomes ne réduisent pas la dimension de l'espace de configuration. Cependant, elles permettent d'imposer des restrictions sur la direction du mouvement. Donc pour un robot non-holonome, tous les chemins sans collision ne sont pas des chemins admissibles.

Par exemple, un robot de type voiture est non-holonome. Physiquement, la non-holonomie est due au fait qu'on suppose que les roues ne doivent pas glisser sur le sol. Cela impose une relation entre l'orientation et la vitesse admissible. Cette relation s'exprime par l'équation non intégrable suivante[1] :

$$\dot{x}\sin(\theta) - \dot{y}\cos(\theta) = 0 \tag{2.1}$$

où x, y, θ représentent la position et l'orientation du robot (voir figure 3.3) dans l'espace de configuration.

L'ensemble des configurations où le robot n'a aucune intersection avec les obstacles forme l'espace C_{libre}. Il existe plusieurs méthodes pour trouver cet espace à partir des différentes représentations du robot et de l'environnement. Toutefois l'idée est

[1] $\dot{x}$ désigne la dérivée par rapport au temps de la variable x (dx/dt)

globalement la même : on "grossit" les obstacles de façon à avoir une représentation équivalente de l'espace dans laquelle le robot n'est plus référencé que par un point (voir figure 2.1). Il y a principalement 3 approches possibles aux problèmes de planification

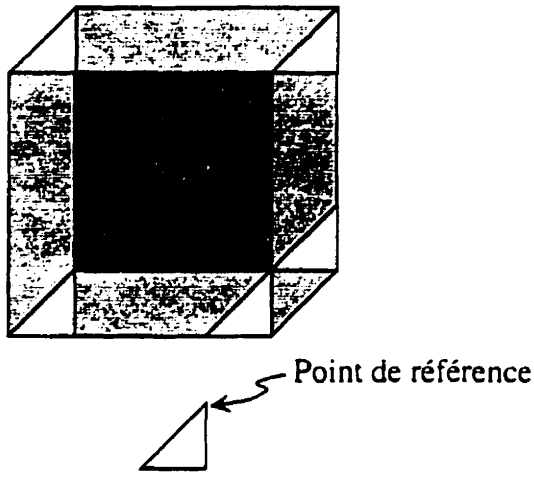


Figure 2.1 : Transformation des obstacles pour trouver l'espace libre C_{libre} pour un robot se déplaçant en translation seulement

de chemin :

1. approche par réduction de l'espace libre (squelettisation ou "carte routière") :
2. décomposition et approximation de l'espace libre (décomposition en cellules) :
3. planification par l'intermédiaire de flux de potentiel.

2.2.1.1 Approche par réduction de l'espace libre

Le concept de la squelettisation est basé sur la réduction de l'espace libre C_{libre} à un réseau de courbes unidimensionnelles. Le chemin se décompose alors en trois sous-chemins :

1. un chemin allant de q_i au réseau ;

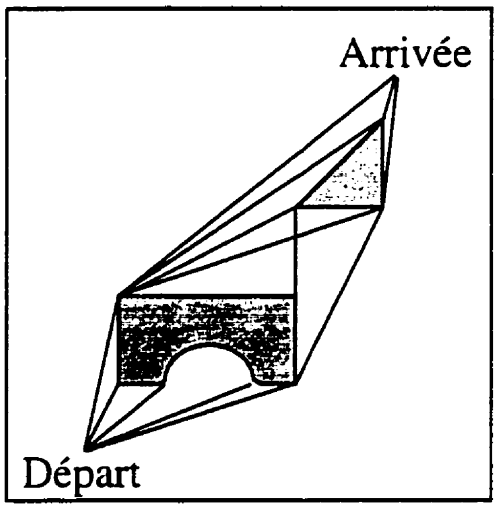


Figure 2.2 : Exemple de graphe de visibilité pour un environnement simple

2. un chemin intra-réseau ;

3. un chemin entre le réseau et q_f.

Dans la plupart des méthodes, ces réseaux de courbes peuvent être représentés par un graphe. Une fois cette réduction faite, la planification n'est plus alors que le problème d'intelligence artificielle bien connu de recherche dans un graphe. On relève 4 méthodes pour ce concept de squelettisation :

1. Le graphe de visibilité où les sommets du graphe sont les sommets des obstacles, q_i et q_f (voir figure 2.2). Deux sommets sont liés si, dans l'espace des configurations, les sommets correspondants peuvent être joints par un segment n'ayant aucune intersection avec les obstacles. La planification est faite par l'algorithme A^*, par exemple, pour trouver le chemin le plus court.

2. La rétraction dont un exemple est le diagramme de Voronoï. Un point appartient au diagramme si la distance minimale à un obstacle se vérifie en au moins deux points des obstacles. Dans le cas d'obstacles polygonaux le diagramme de Voronoï est constitué de segments de droite et d'arcs de paraboles. Il suffit alors de chercher un chemin dans cet espace unidimensionnel.

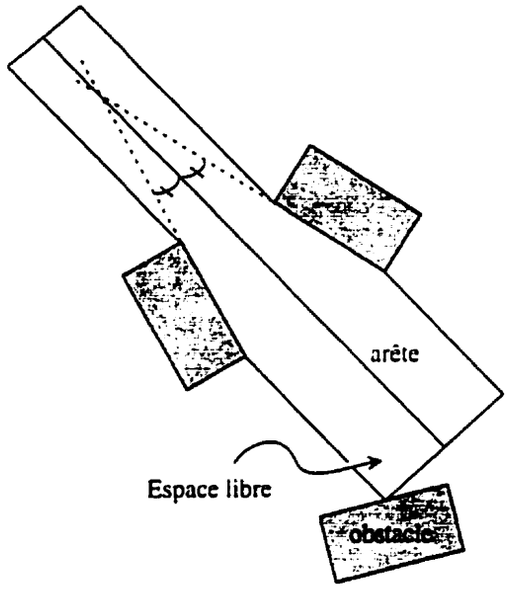


Figure 2.3 : Construction d'un réseau d'autoroute

3. Le réseau d'autoroutes consiste à extraire des figures géométriques appelées autoroutes de l'espace de travail et à les connecter par un graphe (figure 2.3). Une "freeway" est un cylindre généralisé linéaire strict dont l'axe est annoté avec une description conservatrice de l'orientation libre du robot. Le graphe est construit en liant les axes dont les orientations libres se touchent. Une simple recherche dans un graphe permet alors de trouver un chemin.

4. La silhouette est une méthode complexe permettant de réduire progressivement la dimension de l'espace C_{libre}. Pour cela, l'espace courant (de dimension m par exemple) est balayé par un hyperplan de dimension inférieure $(m-1)$ et les points extrémaux sont retenus. Ces points forment alors le nouvel espace dans lequel de nouveau sera fait un balayage. Finalement un graphe contenant des courbes de dimension croissante est construit, sur lequel une recherche de type A^* est possible.

(Svestka et Overmars 1995) propose une approche statistique à la construction de graphes pour la planification. Les nœuds sont choisis de façon aléatoire dans l'espace C_{libre}, et une planification locale (prenant en compte des contraintes non-holonomes)

est ensuite utilisée pour déterminer si un chemin admissible peut-être trouvé entre deux nœuds. Chaque chemin admissible fournit alors un arc supplémentaire dans le graphe. L'idée principale de l'algorithme est la rapidité, qui dépendra fortement de l'algorithme de planification locale.

2.2.1.2 Décomposition et approximation de l'espace libre

La décomposition en cellules est une approche en deux étapes :

1. l'espace C_{libre} est décomposé en cellules. Deux types de cellules existent : exactes (l'union est C_{libre}) ou approximative (l'union des cellules est incluse dans C_{libre}) :
2. la proximité des cellules est utilisée pour construire un graphe dans lequel est faite une recherche de chemin.

Il existe beaucoup de méthodes de décomposition : par arbre quaternaire, par balayage, par division et étiquetage, *etc.*

2.2.1.3 Planification par l'intermédiaire de champs de potentiel

L'approche par flux de potentiel provient d'une analogie avec la physique des particules. En effet, on peut considérer que la configuration q_f induit une force artificielle attractive sur le robot. Les obstacles exercent des forces répulsives pour empêcher le robot d'entrer en contact avec eux. La méthode de résolution est alors simple, il faut tout d'abord générer le champ de potentiel en chaque point, résultant des forces artificielles de q_f et des obstacles, et laisser un programme d'optimisation trouver un chemin vers le minimum global q_f. L'approche est intéressante, toutefois les fonctions de potentiels sont souvent choisies arbitrairement et bien souvent le flux de potentiel créé présente plusieurs minimum locaux. Cela nécessite alors souvent des méthodes de planification dites aléatoires (par exemple la marche aléatoire pour sortir d'un minimum local). Khatib (Khatib 1985), Gutsche *et al.* (Gutsche, Laloni

et Wahl 1994), Arkin (Arkin 1990, Arkin 1992) utilisent ce principe pour planifier les trajectoires de robots dans un espace semi inconnu. Afin de pallier à des problèmes de minimums locaux, (Nam, Lee et Ko 1995) utilise une notion supplémentaire de région atteignable pour un obstacle en mouvement. Dans l'approche unificatrice présentée dans (Schöner et Dose 1992), l'ensemble des éléments du système sont représentés par des systèmes dynamiques dont les propriétés peuvent être caractérisées. A partir de cette modélisation, la création d'un champ de potentiel cohérent est possible.

Comme nous le verrons par la suite, notre solution se rapproche de ce type de méthodes. Toutefois, la plupart des systèmes par flux de potentiel se basent sur des potentiels arbitraires, alors que nous proposons ici une base cohérente par l'intermédiaire des Cartes Dynamiques représentant les capacités motrices des robots. Les potentiels sont créés par rapport à la physique des modèles dynamiques des robots.

2.2.2 Méthodes par optimisation

(Hwang et Ahuja 1992) décrit également une quatrième approche classique au problème de planification, où l'évitement d'obstacles est formulé comme un problème d'optimisation mathématique pour trouver une courbe minimisant une certaine quantité scalaire.

Dans l'article (Shiller et Gwo 1991), une fonction simple basée sur la distance est utilisée afin de planifier la trajectoire d'un robot sous la forme d'une courbe B-spline sur un terrain représenté de façon similaire. Bien que la fonction soit simple, l'optimisation prend en compte un grand nombre de contraintes, allant d'une contrainte de retournement du véhicule à des contraintes de vitesse ou de dynamique du robot.

A partir d'un ensemble d'inéquations algébriques et d'une modélisation en système d'équations différentielles, l'article (Spiteri, Ascher et Pai 1995) propose une solution numérique au problème de planification. La méthode se base sur un glissement (mathématique) à proximité des frontières définies par les contraintes.

(Connolly et Grupen 1994) utilise des fonctions harmoniques pour représenter

les contraintes non-holonomiques. Ces fonctions permettent ainsi de représenter le problème de planification par un réseau de type "resistant" (représentation cubique de l'espace (x, y, θ) avec des résistances entre chaque point de l'espace discrétisé). Le potentiel produit par le réseau fournit un chemin qui est une approximation du chemin "non-holonomement" idéal.

On doit noter ici que la plupart des méthodes par optimisation numérique tiennent compte des contraintes non-holonomes d'un système contrairement aux méthodes de planification "géométrique".

A partir des Cartes Dynamiques combinées pour représenter les interactions, pour planifier des trajectoires optimales basées sur les flux de potentiel mentionnés précédemment, nous utiliserons de simples algorithmes d'optimisation.

2.2.3 Extension temporelle des approches classiques

La plupart des méthodes citées fonctionnent bien dans le cas d'obstacles statiques. Malheureusement, la complexité du problème, quand des obstacles sont en mouvement ou quand il y a d'autres robots, interdit souvent de les utiliser telles quelles.

Dans ce cas, une extension temporelle des méthodes est parfois envisagée, le temps induisant une contrainte non négligeable d'irréversibilité des actions. On peut ainsi représenter l'espace C_{libre} par une ensemble de couches à des temps discrets et planifier alors dans cet espace sur-dimensionné de couche en couche de façon ascendante.

Fujimura et Samet (Fujimura 1989, Fujimura et Samet 1989) présentent un tel type d'extension, où une étude centralisée sur la planification de trajectoire de robot dans un environnement dynamique est envisagée. La résolution du problème dans le cas où le robot est plus rapide que les obstacles est faite grâce à un graphe de visibilité prenant en compte l'aspect temporel par une notion de front de collision. (Rude 1994) propose une extension supplémentaire à ce genre d'approche en introduisant une notion de priorité afin de planifier la trajectoire de plusieurs robots.

Kant et Zucker (Kant et Zucker 1988) présentent pour ce même type de problé-

matique, une méthode en deux étapes :

- une planification est faite avec seulement les obstacles statiques :
- les horaires où le robot ne doit pas se trouver sur le chemin qui est fourni par le premier algorithme indiquent les différents changements de vitesse que doit faire le robot pour éviter les obstacles en mouvement.

En considérant le chemin déjà planifié, (Aronov, Fortune et Wilfong 1991) introduit une méthode basée sur des graphes de visibilité pour les obstacles statiques afin de contrôler la vitesse d'un robot qui doit éviter des obstacles en mouvement. Dans le même type d'approche (Griswold et Eem 1990) utilise une fonction à optimiser en tant qu'information statistique pour contrôler la vitesse du robot. (Ferrari, Pagello et Arai 1995) étend ce genre de résultat pour la coordination de plusieurs robots sur des graphes de visibilité en créant des graphes sans collision.

Il faut noter que l'ensemble des méthodes citées ici ne sont pas complètes, c'est-à-dire qu'il n'y a aucune garantie de trouver une solution même si elle existe.

2.2.4 Planificateur à plusieurs niveaux

Dans la plupart des méthodes de planification usuelles, ni la dynamique ni la cinématique, du robot ne sont prises en compte, à cause principalement du coût de calcul et de l'augmentation de la complexité du problème.

Afin de pallier ces problèmes, la planification peut être faite en plusieurs étapes. Lors de la première étape, un premier chemin est défini grâce à une planification holonome classique. Par la suite, le chemin initialement obtenu est modifié pour refléter les contraintes non-holonomes imposées au robot.

Laumond *et al.* (Laumond, Jacobs, Taïx et Murray 1994) proposent un planificateur exact à deux niveaux pour un robot de type automobile. Dans cette approche, le chemin initial est divisé en un ensemble de sous-chemins holonomes qui sont alors modifiés par l'intermédiaire des courbes de type "Reeds et Shepp" (Reeds et Shepp

1990) (suite d'arcs de cercle et de segments de droite) pour faire l'approximation du chemin géométrique. (Bicchi, Casalino et Santilli 1995) présente une approche très similaire au même problème.

Barraquand et Latombe (Barraquand et Latombe 1993) proposent une heuristique "complète" pour la planification de chemin d'un véhicule avec remorque. L'algorithme consiste à construire heuristiquement un graphe à partir de cellules qui sont connectées s'il existe un chemin permettant d'aller d'un configuration à un autre dans les deux cellules. La complexité de l'algorithme diminue cependant l'intérêt de l'approche.

Cherif et Laugier (Cherif et Laugier 1995) proposent le même type d'approche en ajoutant également des contraintes de contact et d'environnement pour un robot d'exploration spatiale à trois axes indépendants.

Dans le rapport technique (Sekhavat, Svestka, Laumond et Overmars 1996), les auteurs proposent un algorithme de planification de chemin à plus de deux niveaux. Les contraintes non-holonomes du robot sont d'abord décomposées en un ensemble de contraintes "sous-non-holonomes". La planification se fait ensuite en rajoutant une contrainte "sous-non-holonomes"[2] qu'il faut satisfaire à chaque niveau. Le premier niveau est créé par l'intermédiaire d'un algorithme probabiliste de planification (Overmars et Svestka 1996). Il est ainsi possible de planifier la trajectoire d'un véhicule avec plusieurs remorques.

Il existe également des représentations comme les diagrammes de sauts (diagramme dont les liens sont des trajectoires génériques entre des configurations de l'ensemble C_{libre}) pour un véhicule qui peuvent permettre de contraindre la recherche à des courbes cohérentes par rapport à la dynamique. Les résultats de ce genre d'approche permettent une étude théorique des problèmes mais pas pratique, principalement à cause de la complexité et de la restriction sur les trajectoires possibles.

(Chatila, Alami, Degallaix et Laruelle 1992) propose une architecture à trois niveaux pour l'intégration de système de planification et d'exécution des contrôles

[2]correspondant à une décomposition d'une contrainte non-holonome en un ensemble d'équations

pour un robot autonome. Comme nous l'avons suggéré dans l'introduction, notre modèle de Carte Dynamique s'intègre parfaitement à ce genre de système à plusieurs niveaux de planification.

2.2.5 Planification avec incertitude

Il faut aussi envisager les incertitudes qui peuvent exister dans les contrôleurs, afin d'obtenir une convergence quasi-certaine. Ainsi, quand un modèle de l'incertitude est disponible, la planification peut en tenir compte et des méthodes comme le "chaînage arrière" peuvent permettre d'identifier des zones où, même avec un contrôle incertain, on est certain d'obtenir le résultat désiré.

Dans (Takeda, Facchinetti et Latombe 1994), l'incertitude sensorielle et la planification dans un environnement connu sont étudiées conjointement. Le but est d'avoir un contrôle qui se base à la fois sur des données odométriques et sur des données sensorielles. Pour cela, il faut connaître les positions pour lesquelles l'incertitude des senseurs est la plus faible. L'espace C_{libre} est donc modifié en accord avec les différents modèles, et une planification à mi-chemin entre une approche par odométrie pure et une approche par senseurs seulement peut alors être faite.

(Bouilly, Siméon et Alami 1995) utilise le même paradigme, en faisant l'étude de l'influence de "landmarks" sur l'incertitude globale du chemin et en utilisant un méthode d'optimisation numérique.

Page et Sanderson (Page et Sanderson 1995) représentent les incertitudes par un ensemble de configurations où des mesures peuvent être prises. Une méthode de chaînage arrière ou d'optimisation numérique est ensuite utilisée pour calculer la trajectoire optimale. L'article (Barraquand et Ferbach 1995) utilise la même approche pour représenter l'incertitude de façon cumulative et ensuite appliquer un algorithme de programmation dynamique.

Une autre approche aux problèmes d'incertitudes est d'utiliser une modélisation simple des probabilités liées au système. Dans (Zhu 1991) des chaînes de Markov

sont utilisées pour prédire la transition d'un obstacle vers une autre position. (Saito et Fukuda 1995) se base sur un modèle de prédiction multiple : à base d'une décomposition en série temporelle, à base de connaissance (modèle d'inférence de type intelligence artificielle) et adaptatif (apprentissage continu). Des alarmes, ainsi qu'une modélisation de la replanification sont utilisées dans (Sharma 1992) pour planifier en milieu incertain. Le fait que la modélisation soit souvent faite sans tenir compte du sens physique réel constitue le principal inconvénient de ces approches.

Nous verrons dans notre approche de modélisation de l'incertitude du mouvement des autres robots que nous avons rajouté un sens physique cohérent par rapport aux capacités motrices du robot.

2.2.6 Planification non-holonome d'un véhicule automobile

L'automobile a été souvent le modèle de prédilection pour la planification de chemin non-holonome, principalement à cause de l'attrait naturel de ce type de véhicule.

Reeds et Shepp (Reeds et Shepp 1990), en se basant sur les travaux de (Dubins 1957) concernant les courbes de longueur minimale ayant une contrainte sur la courbure, introduisent une classification exhaustive des trajectoires optimales pour une voiture pouvant aller en avant ou en arrière. Les trajectoires sont formées d'un ensemble d'arcs de cercle ou de segments de droite. Nous nous sommes également inspirés de ce genre de travaux pour fonder notre concept de Carte Dynamique.

Par la suite, (Bui, Boissonnat, Souères et Laumond 1993) ont utilisé cette classification pour caractériser les trajectoires de plus court chemin suivant une partition de l'espace. Cette volonté de trouver le plus court chemin est assez commune au problème de planification et a toujours reçu beaucoup d'attention (Lyusternik 1964). Bui et Boissonnat (Boissonnat et Bui 1994) proposent une suite des travaux précédents pour obtenir les régions accessibles d'un robot mobile allant en marche avant. Notre approche au problème des régions atteignables d'un robot mobile est similaire

à ces travaux. Nous avons cependant étendu ces idées dans le cadre moins restreint de tous les types de robots mobiles.

Lumelsky *et al.* (Lumelsky, Mukhopadhyay et Sun 1990), (Lumelsky et Shkel 1995) ont proposé une approche identique pour éviter des obstacles en suivant leur contour. L'idée est que le robot essaye toujours de tourner au maximum de ses capacités et qu'en fonction du résultat le robot ajustera le but à atteindre pour éviter l'obstacle.

2.3 Les régions atteignables d'un système dynamique

Le concept de régions atteignables est un élément très important du domaine du contrôle optimal. Le point de départ essentiel de ces approches consiste à modéliser le robot comme un système d'équations différentielles ou système dynamique (voir (Campion, Bastin et D'Andréa-Novel 1996) pour avoir un aperçu des différents modèles dynamiques de robots).

Snow (Snow 1967), Gupta (Gupta 1981*a*, Gupta 1981*b*) présentent bien le problème lié au contrôle pour trouver des trajectoires optimales pour un robot. Le principe de maximum de Pontryagin évoqué dans (Snow 1967) est un outil important pour étudier et produire une condition nécessaire sur les trajectoires optimales. Nous verrons dans le chapitre 3 la théorie des régions atteignables plus en détail.

Le principe de Pontryagin est utilisé dans (Zheng et Moore 1995) afin d'obtenir le contrôle optimal d'un robot à deux roues motrices qui tentent d'appréhender une cible en mouvement.

(Nam, Lee et Ko 1995) exploite les régions atteignables d'un obstacle en mouvement pour ainsi créer un champ de potentiel cohérent.

Bui et Boissonnat (Boissonnat et Bui 1994) utilisent les courbes de "Reeds et Shepp" afin d'obtenir les régions atteignables d'un robot mobile ayant une courbure

maximale, mais dont le changement de courbure est infiniment rapide. Dans (Fleury, Souères, Laumond et Chatila 1995), un modèle plus complexe de courbes en forme de clothoïde est utilisé pour modifier une trajectoire initialement optimale. Les régions accessibles pour un véhicule à deux roues motrices sont également proposées. Pour le même type de véhicule, Reister et Pin (Reister et Pin 1994) proposent une étude complète des régions accessibles suivant un certain nombre de conditions finales désirées. Notre approche au problème est inverse puisque nous étudions l'accessibilité d'un robot par rapport à un ensemble de condition initiale alors que la destination et la configuration finale sont inconnues.

Simmons (Simmons 1996) transforme la notion d'atteignabilité dans l'espace des courbures/vitesses afin de planifier des trajectoires à court terme pour l'évitement d'obstacles. En utilisant le même genre de représentation (Singh et Kelly 1996) propose également de réduire l'espace des configurations en éliminant les trajectoires impossibles. La méthode est utilisée à la fois pour un véhicule et pour un robot excavateur.

Il est également intéressant de constater que les régions atteignables sont utilisées dans le cadre de la coopération de robot manipulateur (Bien et Lee 1992). Dans (Sundar et Shiller 1995), le système dynamique d'un robot manipulateur est résolu par l'intermédiaire des outils de contrôle optimal comme l'équation de Hamilton-Jacobi-Belmann. L'approche est étendue par la suite dans (Fiorini et Shiller 1996), où des cônes de collision sont introduits pour déduire les vitesses afin de générer des manœuvres d'évitement.

2.4 La coopération et l'interaction de robots mobiles

Nous allons utiliser dans cette section la taxonomie décrite dans l'article (Cao, Fukunaga, Kahng et Mang 1995).

L'évolution vers l'étude des systèmes à plusieurs robots s'est fait de façon naturelle depuis plusieurs années. L'augmentation des tâches qu'un robot n'est capable d'accomplir qu'avec l'aide d'un autre robot, ou l'étude des sociétés de robots pousse de plus en plus la robotique vers l'étude des coopérations et des interactions entre robots.

Les axes principaux de recherche sont :

- l'architecture du groupe :
- les conflits de ressources :
- l'étude des comportements émergeants ;
- l'apprentissage de coopération ;
- les problèmes géométriques ("rester en formation" et la planification multiple) :
- la coopération de robot pour le déplacement d'objet.

2.4.1 Architecture du groupe

L'architecture d'un système multi-agent est primordiale. Elle fournit l'infrastructure sur laquelle les interactions vont être implantées et détermine également les capacités du système à résoudre une classe de problème.

Une première distinction importante se fait si on caractérise le système de façon centralisé ou décentralisé. Le premier type de système revient à dire qu'il y a en fait un seul robot qui contrôle à toute instant l'ensemble de ces éléments constituants comme par exemple la population des robots. Une architecture purement décentralisée suppose que chaque robot prend des décisions pour lui même sans qu'il se concerte avec les autres. Cette dichotomie est parfois discutable comme le décrit Parker (Parker 1993) puisqu'il arrive que des systèmes soient décentralisés de façon hiérarchique. Par exemple dans (Wang, Nakano et Matsukawa 1994) les robots sont

coordonnés par un station centrale de façon grossière puis de façon fine par une architecture de type "subsumption". Sahota (Sahota 1994) présente une architecture où le contrôle des robots est décentralisé mais où les capacités sensorielles sont centralisées par l'intermédiaire d'une caméra au dessus de l'aire de jeu.

Le système que nous allons proposer dans cette thèse est basé sur un paradigme complètement décentralisé.

L'homogénéité/hétérogénéité de la population de robots est également un critère important à considérer. Dans la plupart des travaux, l'homogénéité des robots est considérée, toutefois l'hétérogénéité est une réalité qu'il ne faut pas oublier suivant le type de tâche à accomplir. Notre approche au problème permet de prendre en compte l'hétérogénéité des robots tout en gardant une représentation unique. Cette indépendance structurelle est appréciable.

Les méthodes de communication (explicite ou non) conditionnent aussi l'architecture d'un système multi-robots. Les communications sont faites généralement soit par l'environnement lui-même (comme une mémoire partagée), soit par les senseurs ("rester en formation" en regardant le leader), soit finalement par communication explicite (peu d'attention a été porté sur ce genre de problème: voir (Yamaguchi et Arai 1994),(Yoshida, Arai, Ota et Miki 1994)).

Il existe plusieurs architectures typiques comme : SWARM, qui correspond à avoir un nombre très important de robots: CEBOT Fukuda *al.* (Fukuda et Iritani 1995), une architecture inspirée de l'organisation cellulaire des entités biologiques: ALLIANCE pour des équipes assez réduites de robots: une architecture basée sur des comportements comme par exemple dans la thèse de Mataric (Matarić 1994). Dans (Kosecka, Christensen et Bajcsy 1993) et (Kosecka 1996) l'architecture du système est à mi-chemin entre des comportements simples et une description temporelle discrète (comme les réseaux de Petri) pour décrire les modèles coopératifs des robots.

Dans (Mitsumoto, Fukuda, Shimojima et Ogawa 1995) et (Ishiguro, Watanabe et Uchikawa 1995) une analogie avec les systèmes biologiques humains est faite et les

auteurs proposent une architecture ayant des ressemblances avec le système immunitaires.

2.4.2 Conflits de ressources

Le problème de coopération entre robot peut également être vu comme un problème d'accès à une ressource. Cependant, cela nécessite généralement une mémoire partagée explicite comme dans (Alami, Robert, Ingrand et Suzuki 1995), ou implicite (Rude 1994).

Beaucoup des méthodes de planification temporelles présentées précédemment se caractérisent comme un conflit de ressources (Kant et Zucker 1988).

L'article (Svestka et Overmars 1995) propose une approche où un premier graphe est utilisé pour construire l'espace libre de l'ensemble des robots. Un super-graphe est ensuite construit itérativement, où chaque élément du graphe contient un ensemble de positions dans lesquelles aucun des robots n'est en collision avec un autre.

(Vidal, Ghallab et Alami 1996) utilise une allocation incrémentale des différentes missions des robots par l'intermédiaire d'un graphe afin de planifier les déplacements d'un grand nombre de robots devant transporter des containers.

2.4.3 Etude des comportements émergeants

Plusieurs approches tentent de prendre à contre-pied la traditionnelle décomposition hiérarchique de haut en bas. Ainsi les comportements dans les systèmes multi-robots sont appréhendés comme le résultat de la coopération de comportements décentralisés de robots. Par exemple Gaussier et Zrehen (Gaussier et Zrehen 1994) présentent comment un ensemble de robots ayant des comportements simples peuvent produire un comportement complexe de formation de tas d'objets. Cependant, comme pour l'approche de Brooks, on remarque qu'il y a ici un barrière difficile à pénétrer. Ainsi, pour obtenir des comportements simples d'animaux primitifs il est assez simple d'y arriver. Mais des comportements plus complexes qui requièrent plus

qu'un simple interaction de comportements élémentaires sont difficiles à concevoir dans ce genre d'approche. Notre approche au problème de coordination est totalement inverse, nous avons une tâche à accomplir, le but est maintenant de savoir comment représenter au mieux les informations disponibles aux robots pour arriver à résoudre le problème de façon efficace.

2.4.4 Apprentissage de coopération

Il existe peu de systèmes qui possèdent une capacité inhérente d'apprentissage. Les systèmes à comportements émergeants sont censés être à apprentissage, mais il est plus raisonnable de penser qu'il s'agit d'une manipulation de paramètres sans que le vrai processus d'apprentissage ne soit connu. Il en est un peu de même pour les approches par réseau de neurones où finalement le mécanisme fondamental de l'apprentissage est complètement escamoté derrière une formulation mathématique complexe. Beaucoup de système ont basé l'apprentissage sur le paradigme d'enseignement par renforcement (voir l'annexe B).

Nous proposons dans notre approche de permettre un apprentissage géométrique des capacités d'un robot afin de permettre la représentation des interactions entre robots mobiles.

2.4.5 problème géométrique

Les méthodes de planification usuelles, que nous avons décrites dans ce chapitre, sont parfois utilisées pour résoudre des problèmes de coopération de robots (Latombe 1991). Comme par exemple, en ajoutant le temps à l'espace d'états des robots. Cela se rapproche des méthodes par conflits de ressources, où la ressource est l'espace libre dans l'environnement. Ces méthodes bénéficient des solides bases établies dans les approches de planification pour un seul robot. Par exemple, (Valle et Hutchinson 1996) proposent d'utiliser des réseaux d'autoroutes augmentés d'index de performances individuels afin de planifier les trajectoires de plusieurs robots ayant des buts

indépendants.

Les problèmes de formation de groupe géométrique sont également très intéressants. Le but ici est de proposer un seul algorithme pour chacun des robots d'un groupe pour qu'ils convergent vers une figure géométrique stable (Parker 1993). Nous avons tous à l'esprit les bancs de poissons, ou les formations en triangle de certains oiseaux. Le problème est complexe puisqu'à ce jour et à ma connaissance, il n'existe pas d'algorithme permettant à des robots de décrire un cercle (Cao, Fukunaga, Kahng et Mang 1995).

2.4.6 Coopération de robots pour le déplacement d'objet

Une attention considérable a été dévolue au problème du déplacement coopératif d'objets (Chen et Luh 1994).

(Brown et Jennings 1995) propose une approche avec un robot qui pousse pendant que l'autre contrôle la direction de l'objet. Dans (Ota, Miyata, Arai, Yoshida, Kurabayashi et Sasaki 1995) la planification se fait en deux étapes : la trajectoire de l'objet est tout d'abord planifié, sans tenir compte des robots ; dans la deuxième étape, les trajectoires individuelles de chaque robot sont planifiées afin de porter conjointement l'objet suivant la trajectoire planifiée à la première étape. Dans (Noreils 1993), une architecture à plusieurs niveaux est proposée pour intégrer la coopération entre robot. Les robots s'organisent en équipes qui coopèrent pour déplacer l'objet. Chaque équipe se coordonne de façon interne. L'approche de (Matarić, Nilsson et Simsarian 1995) est basée sur des comportements simples, mais la présence d'un protocole de communication détruit l'esprit du concept utilisé.

2.5 Conclusions sur la bibliographie

La modélisation du mouvement d'un robot est un tâche particulièrement complexe, considérer plusieurs robots complique d'autant plus les choses. Il est clair

d'après la bibliographie que l'étude des interactions entre robots mobiles requiert d'appréhender un grand nombre de techniques couvrant un large spectre de domaines. Bien que cela puisse sembler être un problème, il s'agit en fait d'une bonne chose. En effet, le cadre trop restreint de certain domaine est brisé par cette interpénétration d'expériences, ce qui fournit une opportunité à une recherche plus intéressante.

Nous avons essayé de coaliser plusieurs techniques et domaines dans le concept de la Carte Dynamique que nous allons présenter dans le prochain chapitre. En effet, la Carte Dynamique est basée sur le concept des régions atteignables, et s'inspire ainsi des travaux décrits dans la section 2.3. Afin d'ajouter à l'information contenue dans les régions atteignables, nous nous sommes inspirés des travaux faits dans le domaine de la modélisation de l'incertitude. Nous proposerons également par la suite la façon de relier les senseurs d'un robot et ses capacités motrices à l'aide de la Carte Dynamique dans la même ligne des études sur le couplage visuo-moteur. La combinaison des Cartes Dynamiques afin de planifier les trajectoires des robots se rapproche des modélisations par champs de potentiel et des techniques de planification non-holonome pour un véhicule automobile. Le banc d'essai expérimental, que nous allons présenté, est également le résultat de l'intégration de plusieurs techniques et modélisations concernant les systèmes multi-agents.

Chapitre 3

Concept de Carte Dynamique

Dans ce chapitre, nous allons décrire le concept de Carte Dynamique pour un système dynamique, avant de présenter dans le chapitre suivant l'utilisation d'une telle représentation dans le cadre d'un système composé de plusieurs robots mobiles.

Ce chapitre est organisé de la façon suivante : le concept général de Carte Dynamique sera d'abord présenté (Zanardi, Hervé et Cohen 1995, Zanardi, Hervé et Cohen 1996*b*, Zanardi, Hervé et Cohen 1996*a*), puis nous appliquerons ce concept dans le cas de plusieurs véhicules ayant des modèles cinématiques différents. Une représentation bien adaptée à l'apprentissage des Cartes Dynamiques par un robot sera ensuite présentée[1]. Finalement, nous exposerons plus en détails la relation étroite entre la Carte Dynamique et les senseurs du robot.

3.1 Concept général

Nous allons présenter dans cette section le concept général de Carte Dynamique pour un système dynamique quelconque. Nous appliquerons ce concept dans le cas de trois véhicules différents dans la section suivante.

La fonction principale de la Carte Dynamique est de permettre l'intégration des

[1] L'apprentissage des Cartes Dynamiques sera décrit dans le chapitre 5

capacités intrinsèques de déplacement d'un robot ainsi que les interactions de ce robot avec son environnement à l'intérieur d'une même représentation géométrique. En effet, cette représentation indiquera non seulement les positions que le robot est capable d'atteindre (nous utiliserons le terme de région atteignable pour désigner ces positions), mais également des informations supplémentaires sur la façon dont ces positions sont atteintes, ainsi que sur l'environnement.

La Carte Dynamique est à la base de notre modélisation d'interactions entre robots. Nous verrons ainsi dans le chapitre 4 comment les informations contenues dans les Cartes Dynamiques peuvent être combinées de façon à décrire les interactions entre les différents robots en présence. Cette modélisation est alors utilisée pour planifier la trajectoire du robot considéré en fonction d'une tâche à accomplir. Nous avons envisagé dans notre travail seulement la tâche reliée à la planification de trajectoires d'évasion pour un robot mobile poursuivi. Cependant, la généralité de la Carte Dynamique nous permet de pouvoir considérer dans de futurs travaux d'autres tâches coopératives ou compétitives entre robots. Nous discuterons également dans le chapitre 5 d'une méthode alternative aux Cartes Dynamiques pour la planification de trajectoires pour une tâche d'interception d'un robot par un autre, qui permettra également l'apprentissage des zones accessibles d'un robot.

A ce stade, il est nécessaire de préciser que la Carte Dynamique n'est pas conçue pour construire en chaque point de l'espace la trajectoire optimale permettant de l'atteindre (comme dans des méthodes exactes telles que présentées par Bui et Boissonnat (Boissonnat et Bui 1994), Reeds et Shepp (Reeds et Shepp 1990), Dubins (Dubins 1957)), mais plutôt pour obtenir une indication sur la distribution non uniforme de l'espace atteignable par un robot, afin de permettre une planification se basant sur des réalités physiques (la capacité à se déplacer du robot) et sur des informations reliées à la tâche à accomplir (dans notre cas la recherche de trajectoires d'évasion).

Comme nous le verrons, la forme de la Carte Dynamique est fonction des paramètres internes du modèle dynamique ou cinématique du robot considéré. Prenons

par exemple le cas d'une automobile : les contraintes non-holonomes de ce genre de véhicule font que l'espace qu'il est possible d'atteindre en un temps fini dépend de l'orientation initiale des roues du robot. Nous devons faire remarquer également que la Carte Dynamique est sans mémoire, c'est-à-dire qu'elle représente un cliché de la situation du robot dans son environnement à un instant précis, sans tenir compte de la trajectoire éventuellement planifiée auparavant. De ce fait, lorsque nous parlerons de valeurs initiales, la notion sera relative à l'instant considéré.

La Carte Dynamique permet également de se soustraire à une dépendance structurelle des robots et de leurs modèles dynamiques. Dans la section 3.3, nous décrirons ainsi la représentation générique des Cartes Dynamiques d'un robot mobile quelconque. Lorsque l'on considère une population de robots hétérogènes, les Cartes Dynamiques permettent d'unifier l'approche de planification et d'obtenir ainsi une indépendance structurelle intéressante. Cette indépendance est également fondamentale lorsque l'on considère le problème d'apprentissage des Cartes Dynamiques d'un robot. Nous n'envisagerons cependant dans ce travail que l'auto-apprentissage des Cartes (section 5.2).

La Carte Dynamique comporte donc deux aspects, un premier correspondant à la capacité intrinsèque de déplacement (ou région atteignable) que nous définirons formellement dans la section 3.1.1, et un deuxième aspect constitué de fonctions qui fournissent en chaque point de l'espace atteignable des informations concernant l'environnement et les informations pertinentes sur les trajectoires permettant d'atteindre le voisinage de ce point. Ces fonctions seront introduites dans la section 3.1.2.

3.1.1 Concept de régions atteignables

Le concept de régions atteignables est un élément classique de la théorie de la commande optimale (Snow 1967). Le problème fondamental de contrôle se pose sous

la forme d'un système d'équations différentielles

$$\dot{\mathbf{x}}=\mathbf{f}(t,\mathbf{x},\mathbf{u}), \tag{3.1}$$

où

$$\mathbf{f}=(f_1,\ldots,f_n), \tag{3.2}$$

$$\mathbf{x}=(x_1,\ldots,x_n), \tag{3.3}$$

$$\mathbf{u}=(u_1,\ldots,u_m), \tag{3.4}$$

où t est une variable indépendante (par exemple représentant le temps), $\mathbf{x}$ est l'état courant du système et $\mathbf{u}$ est le vecteur de commande. Pour des états initial $\mathbf{x}_0$ et final $\mathbf{x}_T$, il s'agit de produire un contrôle admissible c'est-à-dire une fonction de contrôle $u_\alpha(t)$, $\alpha \in \{1,\ldots,m\}$, telle que $\mathbf{x}(t_0) = \mathbf{x}_0$ et $\mathbf{x}(T) = \mathbf{x}_T$. Le contrôle sera optimal s'il minimise une fonction

$$J[\mathbf{u}]=\int_{t_0}^{T} L_0(t,\mathbf{x},\mathbf{u})dt, \tag{3.5}$$

où L_0 est une fonction scalaire.

Dénotons maintenant $U \subset \mathbb{R}^m$ l'ensemble de tous les vecteurs de contrôle. Une région T-atteignable est définie comme l'ensemble de tous les états qui peuvent être atteints dans un temps T à l'aide d'un contrôle appartenant à l'ensemble U, avec la même condition initiale $\mathbf{x}(t_0) = \mathbf{x}_0$ (voir figure 3.1). Une région T-contrôlable correspond à l'ensemble des conditions initiales qui permettent d'atteindre $\mathbf{x}(T) = \mathbf{x}_T$ (voir figure 3.2). Un système est dit complètement contrôlable si pour deux états quelconques, il existe un contrôle u permettant d'aller de l'un à l'autre. Une voiture est un exemple de système complètement contrôlable (Laumond, Jacobs, Taïx et Murray 1994). Les régions T-contrôlables ont été utilisées pour la planification en milieu incertain, la méthode étant plus connue suivant l'appellation de chaînage

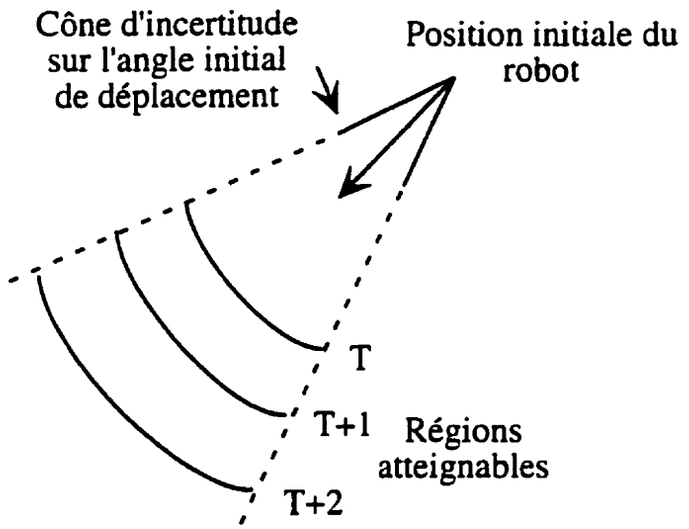


Figure 3.1 : Exemple de régions atteignables : position que peut atteindre le robot aux temps T, $T+1$ et $T+2$ en partant de la même position initiale mais avec des orientations différentes

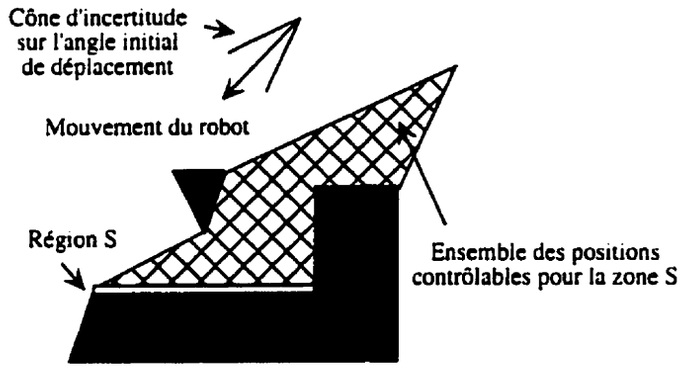


Figure 3.2 : Exemple de région contrôlable : ensemble des positions du robot permettant d'atteindre la zone S avec une contrainte sur l'orientation initiale du robot

arrière (notion de "Preimage Backchaining" présentée dans (Latombe 1991)).

Le problème précédent est équivalent à la résolution d'un système d'équations différentielles d'Hamilton-Jacobi qui peut être résolu à l'aide du principe de maximum de Pontryagin. La démonstration est disponible dans (Snow 1967) et sort ici du cadre de la thèse.

Les méthodes basées sur les ensembles atteignables ont été également utilisées sous la formulation d'un jeu différentiel concernant un problème similaire de guidage de missile (Gupta 1981*a*). En effet, pour intercepter un avion, connaissant ses capacités à

manœuvrer, un missile doit garder à chaque instant l'ensemble de l'espace atteignable de l'avion à l'intérieur de son propre ensemble. En utilisant les modèles dynamiques de l'avion et du missile, les valeurs admissibles pour les paramètres de contrôle de l'avion peuvent être décrites dans l'espace de contrôle du missile, et ainsi permettre la génération d'une loi de guidage basée sur cette information (Gupta 1981*b*).

Les frontières des régions atteignables sont d'un intérêt particulier pour notre problème. Considérons par exemple la fonction

$$J[\mathbf{u}] = \int_{t_0}^{T} dt \tag{3.6}$$

qui définit un problème à temps minimum, et un état $\mathbf{x} \in \mathbb{R}^3$. La loi optimale de contrôle en boucle fermée $\bar{\mathbf{u}} = \bar{\mathbf{u}}(t, \mathbf{x})$ définit l'ensemble $\mathcal{X}_T$ des points qui peuvent être rejoints à partir de $\mathbf{x}_0$ dans un temps minimum T. Alors, l'enveloppe de cette région représente l'ensemble des points les plus éloignés qui peuvent être atteints en partant du même point initial dans un temps inférieur à T.

3.1.2 Extension de l'atteignabilité

Bien que le fait de savoir *quels points* peuvent être atteints dans un temps T permette à un robot de planifier des trajectoires admissibles, tel que cela a été montré pour un bras manipulateur dans (Latombe 1991), une meilleure fonction de contrôle peut être générée si le robot a des connaissances sur la *qualité* des trajectoires permettant d'atteindre ces points. En d'autres mots, l'information fournie par les frontières des régions T-atteignables peut être renforcée par l'adjonction d'une estimation de la qualité des points qui les composent.

Nous définissons ainsi une extension à la notion de régions atteignables en introduisant une fonction du temps et de la position dont la forme peut dépendre de la

tâche à accomplir :

$$(\mathbb{R}, \mathbb{R}^3) \longrightarrow \mathbb{R}$$
$$g \ : \ (t, \mathbf{x}) \longmapsto g(t, \mathbf{x}). \tag{3.7}$$

Considérons par exemple un robot dont la trajectoire planifiée peut changer en accord avec les conditions environnementales alors que le but final reste le même. Il serait important, dans ce cas, de connaître le nombre de trajectoires différentes permettant d'atteindre une position similaire —comme une mesure de confiance dans la trajectoire planifiée. Par exemple, si on considère le cas d'un robot mobile qui est poursuivi, il est utile de savoir s'il existe de nombreux contrôles $\mathbf{u}$ permettant d'aller au voisinage d'une position où le robot est en sécurité, parce que cela indiquerait intuitivement que, même s'il y a des perturbations sur le parcours, la destination finale ne sera pas trop affectée. Dans ce cas, la fonction correspondrait à la densité de fonctions de contrôle $\mathbf{u}$ pour un voisinage donné d'un point $\mathbf{x}$ à l'intérieur d'une région T-atteignable.

En fonction des objectifs du robot, d'autres fonctions peuvent être définies. Idéalement, la fonction g devrait prendre en compte des critères importants comme la présence d'obstacles, la consommation d'énergie, la longueur du chemin et la nombre de trajectoires différentes. En effet, les valeurs correspondant aux points appartenant à un obstacle devraient être négatives pour prévenir qu'une trajectoire ne passe par eux. La consommation d'énergie est également un important critère lorsqu'un robot doit par exemple se joindre périodiquement à un autre pour se recharger. Les fonctions peuvent aussi refléter les conditions environnementales, telles que l'humidité de la route, la texture du sol (douce ou dure), la présence de graviers sur la route, *etc.*

Toutes ces fonctions ne sont pas prises en compte pour construire les lois de contrôle optimales $\bar{\mathbf{u}}(t, \mathbf{x})$ et ne sont utilisées que lors de la phase de planification, lorsque le robot utilise sa Carte Dynamique pour générer sa trajectoire.

Pour utiliser des méthodes classiques d'optimisation, la valeur de la fonction devrait préférablement être réelle et scalaire, évitant ainsi les problèmes reliés à la maximisation d'une fonction multidimensionnelle. Il faut donc combiner les différentes valeurs des fonctions, comme par exemple lorsqu'on tient compte du nombre normalisée de trajectoires (fonction d) et de la présence d'un obstacle (fonction o) pour résoudre une tâche spécifique, la valeur de $g(t, \mathbf{x})$ sera $d(\mathbf{x}, t)$, si $o(\mathbf{x}, t) = 0$, $o(\mathbf{x}, t)$ sinon. La fonction combinée finale à valeur réelle pourra ainsi être utilisée dans une représentation de type bitmap.

On introduit alors la fonction de transformation t_g (qui sera dépendante de la tâche à accomplir) pour décrire la façon dont les fonctions individuelles $g_i, i \in \{1, \ldots, m\}$ doivent être combinées pour créer la fonction g de la Carte Dynamique :

$$\begin{aligned} & (\mathbb{R}, \mathbb{R}^3) \longrightarrow \mathbb{R} \\ g \ : \ & (t, \mathbf{x}) \longmapsto g(t, \mathbf{x}) = t_g(g_1(t, \mathbf{x}), \ldots, g_m(t, \mathbf{x})) \end{aligned} \tag{3.8}$$

Les régions T-atteignables du système dynamique modélisant le robot forment avec la fonction g ce que nous appelons la Carte Dynamique.

Un des grands avantages de la Carte Dynamique est qu'elle est construite à l'échelle du robot lui-même. Cela signifie que la Carte Dynamique est une représentation intrinsèque des capacités du robot, et est donc invariante par rapport à sa taille lorsque l'environnement n'est pas considéré. De ce fait la Carte dynamique d'un modèle réduit d'une automobile sera la même (relativement à la taille) que celle de l'automobile en grandeur réelle. Par taille, nous entendons ici la valeur de L par exemple dans le cas de la voiture. L'encombrement du véhicule impliquera un agrandissement des obstacles qui dépendra évidemment de la taille et de l'encombrement du véhicule contredisant l'invariance. La propriété d'invariance reste cependant vrai pour les régions T-atteignables.

Dans la section suivante, nous allons appliquer le concept que nous avons introduit

ici à différents types de robots mobiles.

3.2 Application du concept à différents modèles de robots mobiles

Afin de montrer que la Carte Dynamique est un concept applicable à différents modèles de robots mobiles, nous allons construire les cartes respectives d'une automobile, d'un robot ayant deux roues motrices indépendantes et d'un véhicule articulé. Nous considérons ici, un axe des temps discrétisé et nous considérerons que les robots sont représentés par un seul point (le point de référence). Des techniques d'agrandissement des obstacles (Latombe 1991) pourront être utilisées pour tenir compte de l'encombrement du véhicule.

3.2.1 Robots de type véhicule automobile

La carte va représenter les capacités de manœuvre d'un robot mobile avec un ensemble de courbes emboîtées $C_0, \ldots, C_T$. Chaque courbe C_t entoure l'ensemble des points qui peuvent être atteints à partir de la même position initiale dans le temps T_t. La forme de chaque C_t, et le domaine contenu entre deux courbes consécutives, dépend de la vitesse initiale du robot, de l'orientation initiale de ces roues, et du modèle dynamique du robot.

La construction d'une carte dynamique pour n'importe quel type de véhicule se fait toujours en deux étapes : 1/ la construction des limites externes des régions atteignables et 2/ le calcul de la valeur de la fonction en chaque point de ces régions.

3.2.1.1 Approche exhaustive

Nous avons choisi un modèle cinématique simple d'une voiture pour étudier la construction des Cartes Dynamiques. Le comportement dynamique du véhicule induit

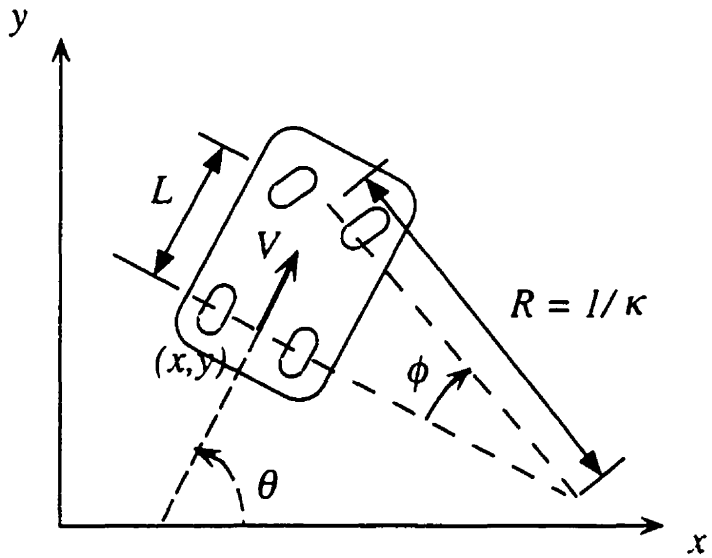


Figure 3.3 : Paramètres d'un modèle simple de voiture

par la masse, les forces de Coriolis, *etc.* est négligé pour l'instant dans la construction de la carte[2]. Ainsi, le modèle dynamique du véhicule (il s'agit d'un modèle cinématique du point de vue de la robotique, cependant ce modèle correspond à un système dynamique représenté par l'équation $\dot{\mathbf{x}} = f(\mathbf{x}, t, \mathbf{u})$) conformément à la figure 3.3 est :

$$\begin{cases} \dot{x} &= V\cos(\theta), \\ \dot{y} &= V\sin(\theta), \\ \dot{\theta} &= \frac{V}{L}\tan(\phi), \\ |\dot{\phi}| &\leq \dot{\phi}_{max}, \\ |\phi| &\leq \phi_{max}, \end{cases} \tag{3.9}$$

où x, y est la position dans le plan du centre de l'essieu arrière. θ est l'orientation de l'axe longitudinal de la voiture. V est la vitesse suivant ce même axe. ϕ est l'orientation des roues et L est la distance entre l'axe des roues arrière et avant. L'utilisation des commandes « bang-bang »nécessitent la discretisation temporelle. Nous utili-

[2]C'est-à-dire qu'on utilise les vitesses directement lors de la construction sans chercher à savoir comment on les obtient

serons ΔT pour désigner le temps entre deux commandes discrètes. La voiture est commandée par l'intermédiaire des accélérations des paramètres ϕ et V. L'orientation initiale des roues est donnée par ϕ_0. Puisque les Cartes Dynamiques sont centrées sur le robot lui-même, les valeurs initiales de x et y sont nulles. pour simplicité de représentation nous avons choisi la valeur initiale de θ à $\pi/2$.

Dans les tâches que devront décrire les robots, nous avons supposé que le profil d'accélération serait connu *a priori* (par exemple un profil d'accélération maximale à partir de la position initiale ou un freinage d'urgence). Toutefois, sans réduire la généralité du modèle nous avons choisi ici de considérer la vitesse des véhicules constante $V = V_{max}$. Bien que ce choix semble restrictif, nous discuterons plus loin d'éventuels relâchement de cette contraire.

La seule variable de contrôle sera $\dot{\phi}$ (parfois notée u), c'est-à-dire le taux de changement de l'angle des roues ϕ.

Dans (Snow 1967), il est précisé que les régions atteignables sont obtenues à partir de trajectoires générées par des contrôles de type "bang-bang". Nous allons tout d'abord calculer quels sont ces contrôles "bang-bang" associés au système décrit par les équations 3.9.

On suppose que l'on cherche une trajectoire entre une position initiale au temps t_0, et une position finale au temps T. Il a été prouvé que les véhicules basées sur le modèle précédent sont complètement contrôlables et qu'il existe ainsi toujours un contrôle admissible pour atteindre n'importe quelle configuration à partir d'une position initiale (Laumond, Jacobs, Taïx et Murray 1994).

La fonction que l'on désire maximiser par rapport au contrôle u correspond à trouver les trajectoires à temps minimum pour atteindre chaque point de la carte. La fonction de coût (au sens du contrôle optimal (Snow 1967)) est de la forme :

$$J[u] = \int_{t_0}^{T} L_0(t, x, u)dt. \tag{3.10}$$

Pour avoir des trajectoires à temps minimum, il suffit de prendre la fonction $L_0(x, u, t)$ identiquement égale à un. La fonction $J[u]$ devient alors :

$$L_0(x, u, t) = 1. \tag{3.11}$$

$$J[u] = T - t_0. \tag{3.12}$$

Transformons maintenant le système 3.9 dans le formalisme de Pontryagin et Snow (Snow 1967) :

$$\begin{cases} \dot{x}_1 = f_1(t, \mathbf{x}, u) = V \cos(x_3), \\ \dot{x}_2 = f_2(t, \mathbf{x}, u) = V \sin(x_3), \\ \dot{x}_3 = f_3(t, \mathbf{x}, u) = \frac{V}{L} \tan(x_4), \\ \dot{x}_4 = f_4(t, \mathbf{x}, u) = \quad u, \\ |\dot{x}_4| \leq \quad \dot{\phi}_{max}, \\ |x_4| \leq \quad \phi_{max}, \end{cases} \tag{3.13}$$

avec la vitesse V considérée comme constante.

L'Hamiltonien de ce système est défini par :

$$H_0(t, x, p, u) = \sum_{i=0}^{4} p_i f_i(t, x, u) - L_0(t, x, u). \tag{3.14}$$

où $p_i, i \in 1, 2, 3, 4$ sont les variables adjointes qui satisfont le système d'équations différentielles suivant :

$$\dot{p}_i = -\frac{\partial H}{\partial x_i}(t, x, p, u). \tag{3.15}$$

$$= -\sum_{j=0}^{4} p_j \frac{\partial f_j}{\partial x_i}(t, x, u) - \frac{\partial L_0}{\partial x_i}(t, x, u). \tag{3.16}$$

avec $\frac{\partial L_0}{\partial x_i}(t, x, u) = 0$ d'après l'équation 3.12.

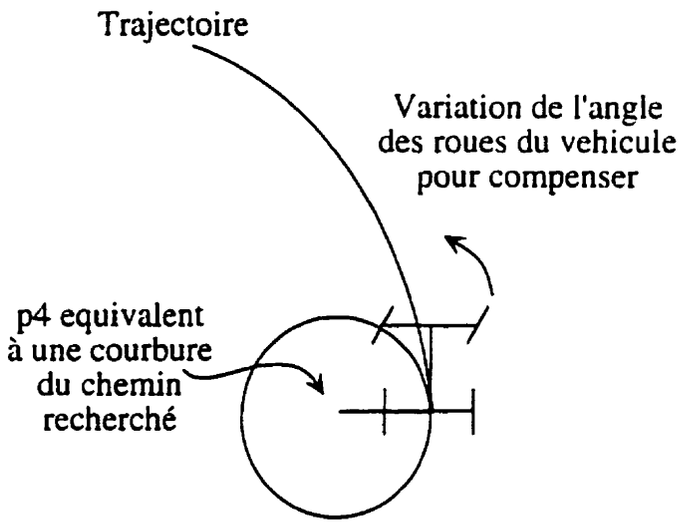


Figure 3.4 : Explication physique de la maximisation de H_0

Par le théorème de maximisation de Pontryagin, les trajectoires optimales sont obtenues pour les valeurs de u qui maximisent l'Hamiltonien H_0.

Le système décrivant les variables adjointes est alors :

$$\begin{cases} \dot{p}_1 = 0, \\ \dot{p}_2 = 0, \\ \dot{p}_3 = -V\frac{\partial \cos(x_3)}{\partial x_3}p_1 - V\frac{\partial \sin(x_3)}{\partial x_3}p_2 = V\sin(x_3)p_1 - V\cos(x_3)p_2, \\ \dot{p}_4 = -\frac{V}{L}\frac{\partial \tan(x_4)}{\partial x_4}p_3 = -\frac{V}{L}(1+\tan^2(x_4))p_3. \end{cases} \tag{3.17}$$

La dernière équation correspondant à $\dot{p}_4$ ressemble à l'équation de la variation de la courbure instantanée du chemin suivante (également présentée dans l'équation 6.2 du chapitre 6):

$$\kappa = \frac{\tan(\phi)}{L}, \tag{3.18}$$

$$\dot{\kappa} = \frac{V}{L}\left(1+\tan^2(\phi)\right)\omega. \tag{3.19}$$

où κ est la courbure instantanée du mouvement du robot et $\omega = \dot{\phi}$ est la vitesse angulaire de l'angle des roues avant.

On obtient ainsi $H_0 = p_1 f_1 + p_2 f_2 + p_3 f_3 + p_4 u - 1$. Puisque l'on cherche a maximiser H_0 par rapport à u, il apparaît que :

- si $p_4 > 0$, le maximum est obtenu pour $u = u_{max}$. Ceci correspondrait à tourner les roues le plus rapidement possible vers la gauche, lorsqu'un équivalent de la courbure p_4 du chemin désiré est positif (voir figure 3.4).
- si $p_4 < 0$, le maximum est obtenu pour $u = -u_{max}$ (inverse de précédemment).
- si $p_4 = 0$, le contrôle est alors singulier et le cas requiert un peu plus d'étude.

Il faut d'abord préciser que, dans notre étude, les valeurs initiales et finales seront fixées de la façon suivante : $x(t_0) = (0, 0, \pi/2, \phi_0)$ (correspondant toujours à la même position initiale mais avec une valeur de l'angle des roues différentes) et $x(T) = (x, y, \theta, \phi_T)$, et que la valeur de l'orientation finale de la trajectoire est libre. De ce fait, par la condition de transversalité de Pontryagin, p_3 est nulle au temps T.

Etudions maintenant le cas où p_4 est nulle sur un intervalle $[t_1, t_2]$. Puisque p_4 est nulle sur cette intervalle, sa dérivée est aussi nulle sur cet intervalle, $\dot{p}_4 = 0$. D'après l'équation 3.17, puisque $(1 + \tan(p_4)) > 0$, cela implique que $p_3 = 0$ sur l'intervalle $[t_1, t_2]$. Ceci est équivalent à :

$$\sin(x_3)p_1 - \cos(x_3)p_2 = 0 \tag{3.20}$$

Quatre cas se présentent alors :

- $p_1 \neq 0$ et $p_2 \neq 0$:
- $p_1 = 0$ et $p_2 \neq 0$;
- $p_1 \neq 0$ et $p_2 = 0$:
- $p_1 = 0$ et $p_2 = 0$.

Les trois premiers cas impliquent que x_3 est une constante ($tan(x_3) = p_2/p_1$ par exemple pour le premier cas), ce qui implique que $\dot{x}_3$ est nul sur l'intervalle $[t_1, t_2]$. D'après l'équation 3.13, si on suppose que $|x_4| \leq \pi/2$ (cette contrainte physique est réaliste, aucune voiture ne pouvant tourner ses roues à angle droit), alors x_4 est nul sur $[t_1, t_2]$, ce qui correspond à un contrôle u identiquement nul.

Le quatrième cas induit que l'Hamiltonien est constant (puisque le système est stationnaire). De plus, puisque qu'il n'y a pas de critère terminal, la valeur finale de l'Hamiltonien est nulle. De ce fait avec $p_1 = p_2 = p_3 = p_4 = 0$ on obtient la contradiction $H_0 = 0 = -1$. Ce cas est donc rejeter.

De cette étude, on retire le fait que toutes les trajectoires optimales au sens où nous l'avons défini sont produites à partir d'une séquences de contrôles "bang-bang", c'est-à-dire u prend sa valeur dans l'ensemble $\{-u_{max}, 0, u_{max}\}$.

Ceci constitue la condition nécessaire que doivent satisfaire les trajectoires optimales, mais ne correspond pas à une condition suffisante d'optimalité.

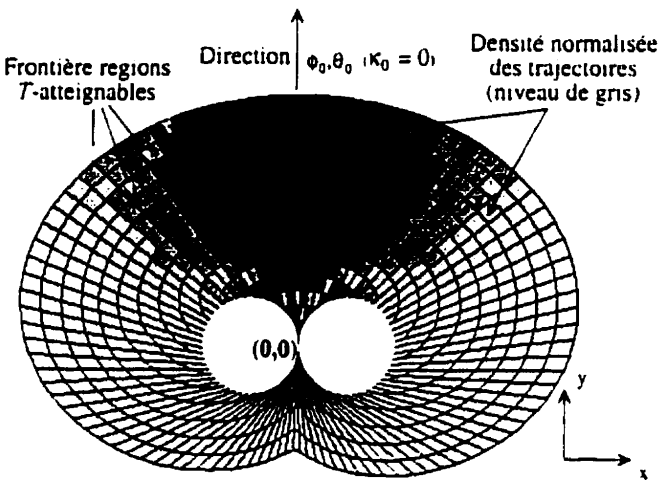
Il est cependant possible de calculer toutes les positions qu'il est possible d'atteindre à partir d'une même position initiale, par un calcul récursif. Ainsi, en chaque position $(x(t), y(t))$ on peut calculer les trois positions atteintes au temps $t + \delta t$ grâce aux trois contrôles $\{-u_{max}, 0, u_{max}\}$. Pour obtenir les régions T-atteignables, il suffit de récupérer les trajectoires finissant à l'instant T et correspondant à la surface externe des régions atteignables.

La figure 3.5 présente deux exemples de carte construites de façon exhaustive.

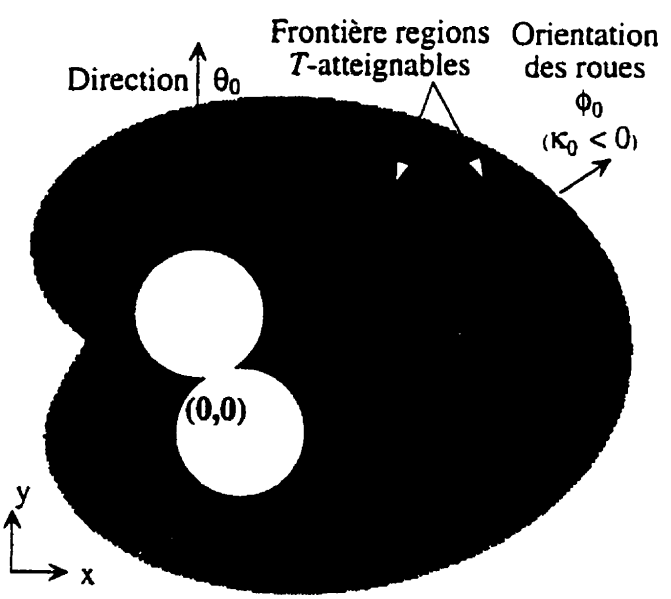
Il est toutefois évident qu'un calcul exhaustif de toutes les positions devient quasi impossible lorsque le temps augmente. Il est donc nécessaire de trouver une approximation de la Carte Dynamique.

3.2.1.2 Approche par courbes limites

La construction des régions atteignables pour un robot ayant une vitesse de braquage $\dot{\omega}_{max}$ infinie est décrite dans (Boissonnat et Bui 1994). La démonstration



(a) $\phi_0 = 0$



(b) $\phi_0 = \phi_{max}$

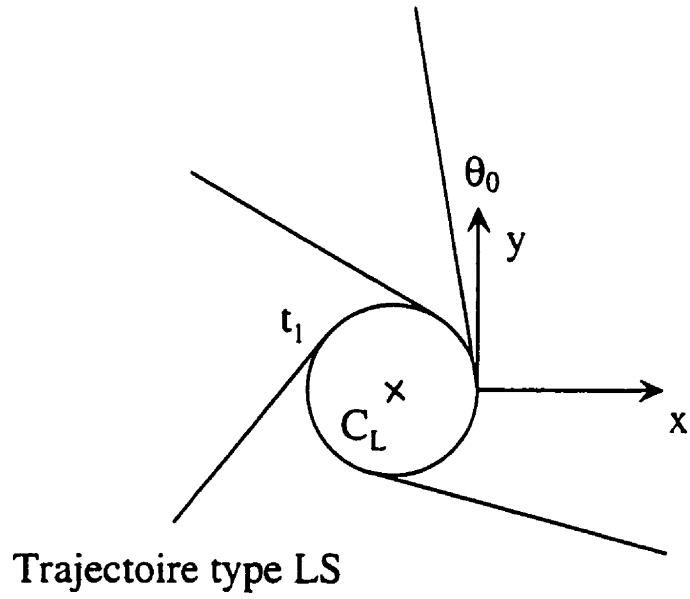
Figure 3.5 : (a) $\phi_0 = 0$ (b) $\phi_0 = \phi_{max}$

repose sur les travaux présentés dans (Reeds et Shepp 1990, Dubins 1957) qui ont classifié toutes les trajectoires possibles pour une voiture possédant cette propriété de variation infinie du taux de changement de l'angle des roues. De (Boissonnat et Bui 1994), il ressort que les trajectoires qui constituent la surface externe des régions T-atteignables sont du type RS ou LS (R = "right", L = "left" et S = "straight"), c'est-à-dire que les roues sont à droite ou à gauche pendant un temps t_1 puis l'angle des roues passe à zéro (voir figure 3.6). Deux autres types de trajectoire sont considérés dans (Boissonnat et Bui 1994), les types RL et LR. Toutefois, nous allons montrer plus loin que ces trajectoires ne sont pas nécessaires pour la construction de la carte dynamique.

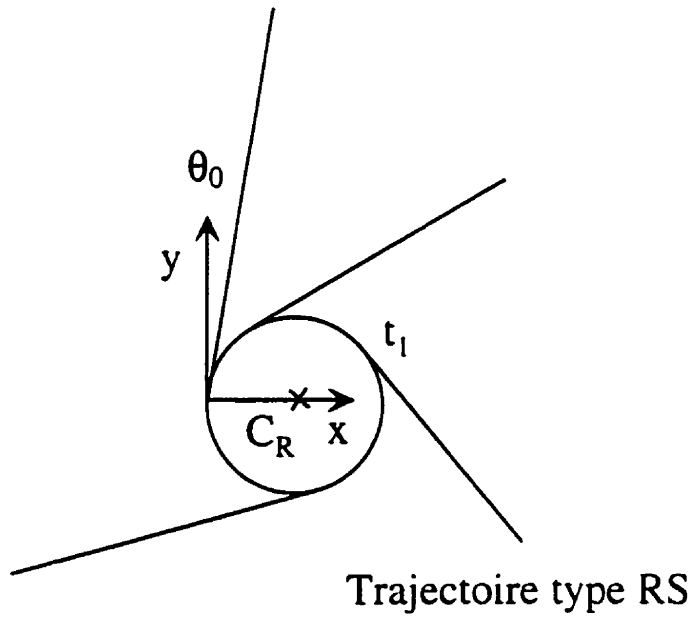
La faiblesse cependant de ce genre de méthode, est de ne pas tenir compte de l'effet (important) que la valeur initiale des roues induit sur l'atteignabilité de l'espace. Ainsi, les courbes formées par les trajectoires LS et RS seront identiques quelle que soit la valeur initiale de ϕ.

Il est possible de faire une analogie avec les études présentées dans (Boissonnat et Bui 1994) et (Reeds et Shepp 1990) en utilisant des courbes de transition. En effet, on peut voir dans la figure 3.6 que les cercles C_L et C_R correspondant à la valeur maximale de l'angle ϕ jouent un rôle important dans la construction des courbes LS et RS. Dans notre modélisation, nous avons tenu compte du fait que les roues ne pouvaient tourner avec une vitesse infinie, ce qui implique une trajectoire de transition entre une valeur initiale de ϕ et l'une de ses valeurs maximales ($-\phi_{max}$ ou $+\phi_{max}$). La situation est inversée lorsque l'on passe de cette valeur maximale à une valeur nulle pour l'angle des roues.

Ainsi, les courbes formant les régions atteignables seront créées de la façon suivante : pour une courbe équivalente à une courbe RS et pour une valeur de $\phi = \phi_0$ au temps t_0, la première courbe de transition (vers la courbe R, voir figure 3.7) est calculée en faisant varier ϕ le plus rapidement possible (cette variation maximale est déterminée par la valeur de $\pm\dot{\phi}_{max}$) de ϕ_0 à ϕ_{max}. L'instant t_1 correspond à l'instant où



(a) *LS*



(b) *RS*

Figure 3.6 : Trajectoires de type *LS* (a) *RS* (b)

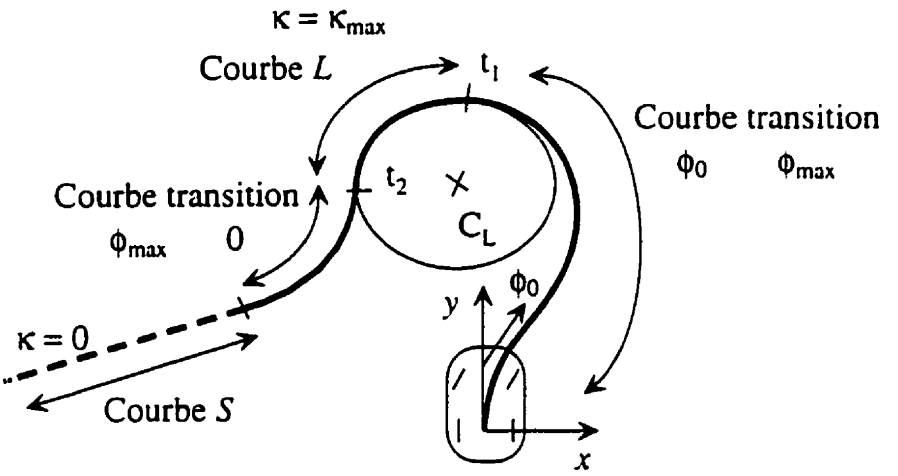


Figure 3.7 : Analogie aux courbes RS à l'aide de courbes de transition

ϕ est égal à ϕ_{max}. Une fois cette valeur obtenue on parcourt le cercle de la même façon que pour une courbe R usuelle. Par la suite pour un temps quelconque t_2 supérieur au temps t_1, on calcule la courbe de transition (vers la courbe S) avec ϕ variant à nouveau le plus rapidement possible vers pour revenir à 0. On laisse ensuite la valeur de l'angle des roues à 0 par obtenir l'équivalent des courbes S. La figure 3.7 illustre le processus expliqué précédemment. En ce qui concerne les courbes de type RL, il s'agit des courbes de longueur minimale permettant d'atteindre l'intérieur des cercles C_R et C_L (voir figure 3.8). Ces trajectoires ne sont pas fondamentales pour créer les cartes dynamiques puisque nous nous attachons plutôt à trouver des trajectoires tentant en quelque sorte de "s'éloigner" du point initial.

L'optimalité des trajectoires de type RS et LS avec les transitions que nous avons introduites a été vérifiée expérimentalement. Ainsi, en calculant toutes les trajectoires finissant dans un temps compris entre $T-1$ et T, il est possible de constater qu'aucune trajectoire ne permet d'atteindre les points finaux des courbes précédentes (mis à part les courbes optimales elles-mêmes).

Une fois toutes les trajectoires RS et LS construites, il ne reste plus qu'à construire la surface externe des régions T-atteignables en liant les points des trajectoires au même temps T. La figure 3.9 montre comment construire les régions T-atteignables

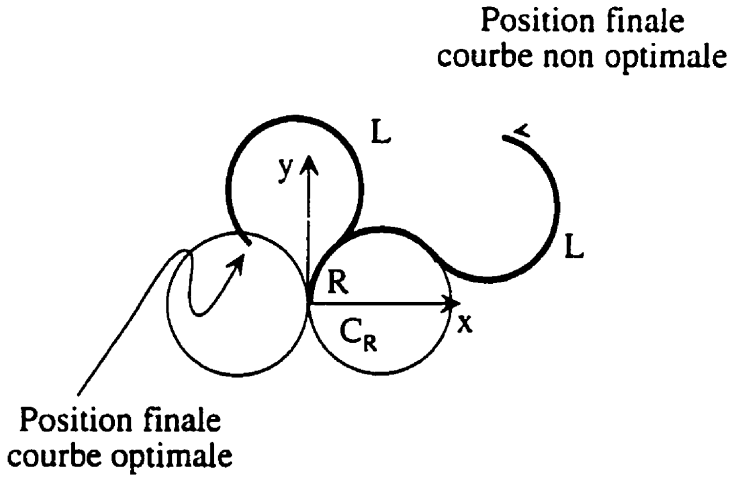


Figure 3.8 : Courbes de type RL optimale ou non

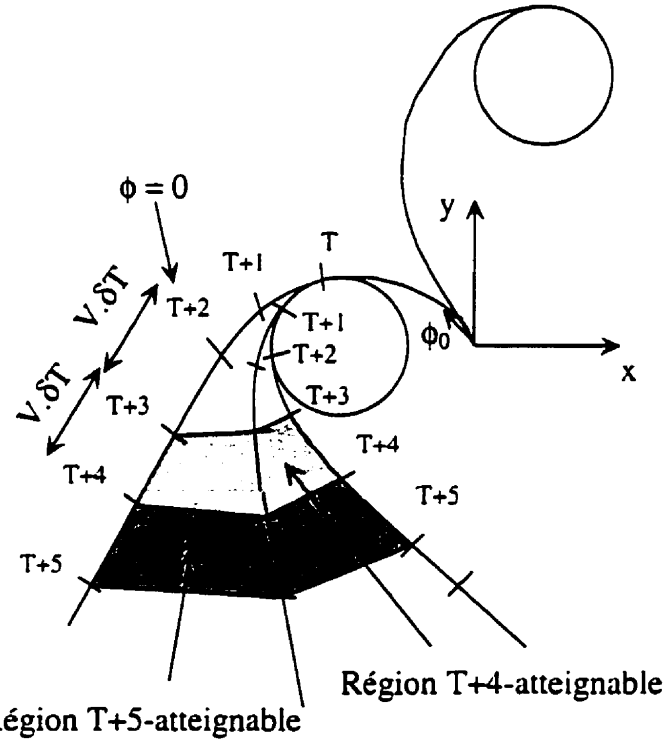


Figure 3.9 : Méthode de construction des régions T-atteignables

à partir des courbes précédentes.

Il est nécessaire cependant de trouver l'intersection (si elle existe) entre les courbes RS et LS au même instant. Il est en effet évident que les courbes se recouvrent à l'arrière du véhicule. Les courbes ne sont alors plus développées au delà de ces

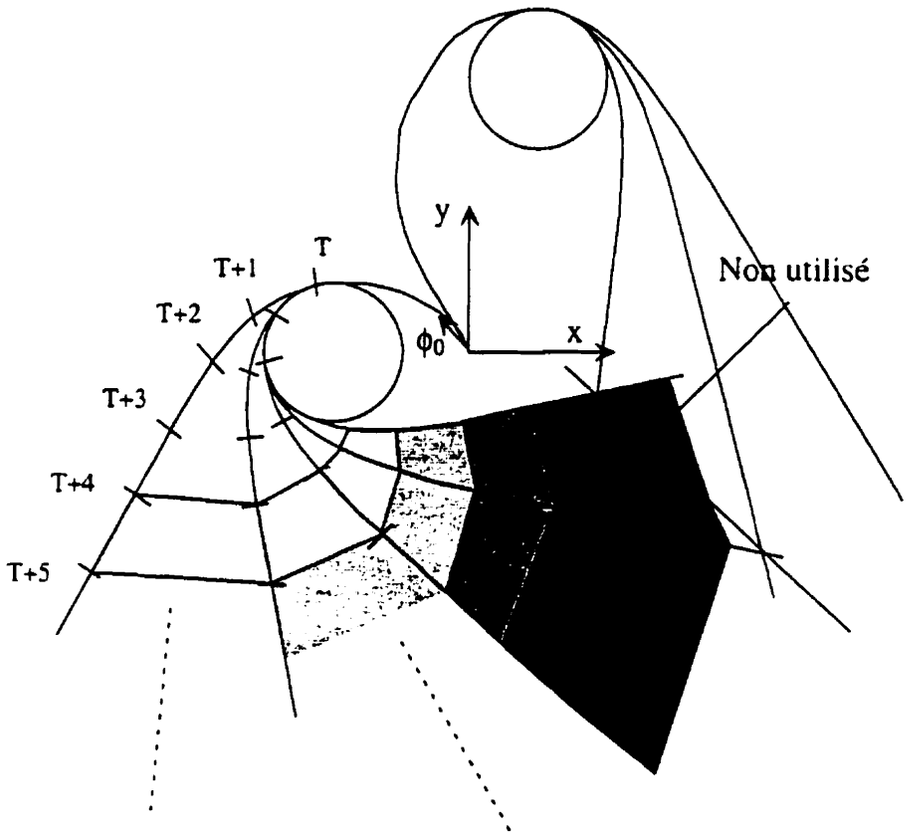


Figure 3.10 : Intersection entre courbes *RS* et *LS* au même instant

intersections, car notre optique est toujours d'obtenir l'enveloppe externe des régions atteignables.

La figure 3.10 présente la situation lorsque les courbes *RS* et *LS* se rejoignent. Nous sommes maintenant en mesure de construire les régions *T*-atteignables. Il s'agit désormais de calculer les fonctions qui seront utiles à un robot lors de la tâche à accomplir.

Les fonctions seront évidemment toujours dépendantes de la tâche à accomplir. Par exemple, nous parlerons dans le chapitre suivant de fonctions permettant à un robot d'éviter un obstacle et d'essayer d'échapper à son adversaire.

Une des fonctions les plus importantes est le nombre de trajectoires finissant dans un temps *T* entre les limites externes des régions *T-1*-atteignables et *T*-atteignables. Dans le cadre des tâches que nous avons considéré, il est préférable d'avoir plusieurs

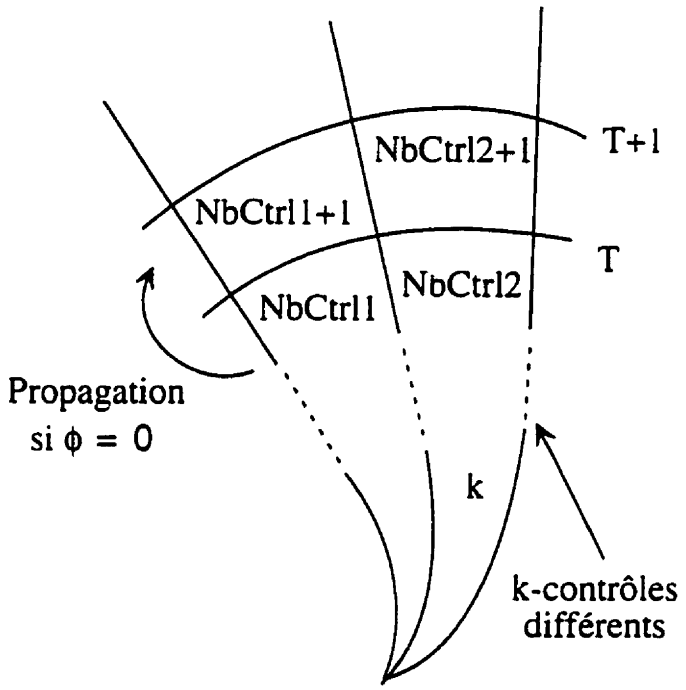


Figure 3.11 : Approximation du nombre de trajectoires à partir des valeurs de contrôles distinctes pour construire les courbes *RS* et *LS*

trajectoires possibles pour atteindre une même région. Le nombre de trajectoires indique alors la facilité d'accès de la zone de l'espace considérée. Par exemple, dans le cas de la recherche de trajectoires d'évasion, un robot doit avoir à sa disposition plusieurs trajectoires possibles pour choisir celle qui lui facilitera son évasion.

Lorsque nous avons calculé les Cartes de façon récursive lors de l'approche exhaustive de construction, le calcul du nombre de trajectoires revenait simplement à utiliser des accumulateurs pour les points finaux de chaque trajectoire et de normaliser ensuite l'ensemble des valeurs. Ce calcul n'est bien entendu pas possible pour la méthode de construction utilisant les courbes *LS* et *RS*. Toutefois, il est possible d'obtenir une approximation des valeurs de la fonction du nombre de trajectoires en utilisant le nombre de contrôles différents nécessaires pour atteindre une position (voir figure 3.11). Nous considérons en effet qu'entre deux courbes optimales, la différence du nombre de contrôles pour les générer évalue grossièrement combien de trajectoires non-optimales aurait pu être insérées entre les deux. Les figures 3.12 et 3.13 donnent

quelques exemples de Cartes Dynamiques pour une voiture avec la fonction de nombre de trajectoires ainsi calculée et avec des variations sur les paramètres de la voiture.

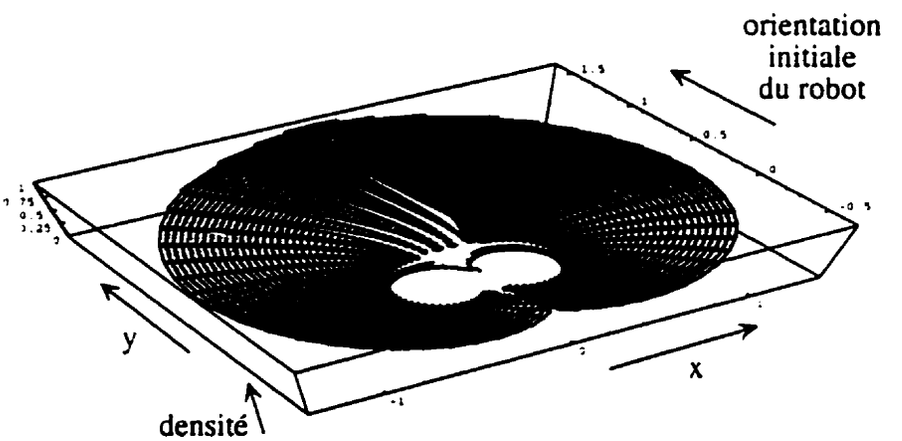
Remarque : Il est important de préciser que certaines figures utilisent un point de vue 3D. La quantification discrète des valeurs d'orientation des roues du véhicule va alors induire dans certains cas des discontinuités dans la représentation 3D des cartes. Ainsi suivant les points de vue, certaines défauts d'affichage (ou sauts) peuvent apparaître. Ce défaut visuel est éliminé sur les représentation « bitmap »des cartes.

Les figures 3.14 et 3.15 présentent deux exemples de Cartes Dynamiques avec des fonctions prenant en compte l'environnement (obstacles et direction de mouvement privilégiée symbolisant par exemple une route et son bas-côté). Les Cartes Dynamiques que nous avons montrées ne sont déterminées que pour des valeurs extrémales de la vitesse. Cependant, si on venait à utiliser un profil d'accélération (ou de vitesse) connu[3], les Cartes pourraient être modifiées en changeant les longueurs des segments R ou L puis S (ainsi que des transitions entre les courbes) permettant de générer les Cartes en fonction de la vitesse. Par exemple, dans la figure 3.16, plusieurs profils $V = V(t)$ sont utilisés pour construire la même courbe RS. Cela serait équivalent à appliquer une déformation non linéaire de l'ensemble des trajectoires constituant la base de la Carte Dynamique et d'obtenir ainsi une Carte Dynamique en accord avec le profil de vitesse.

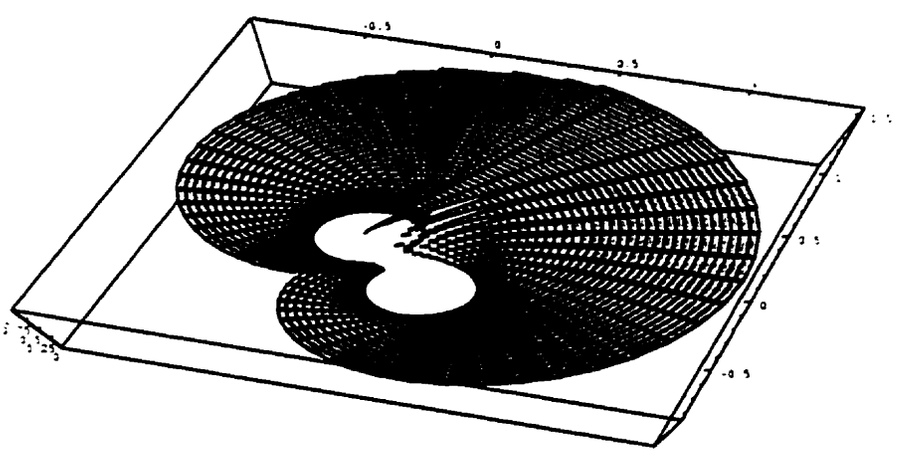
3.2.2 Carte Dynamique d'un robot à deux roues motrices indépendantes

Un autre type de robot communément répandu est celui pour lequel deux roues seulement contribuent aux déplacements (propulsion et direction) du véhicule, les autres roues n'ayant qu'un rôle de soutien d'équilibre. Les moteurs des deux roues actives sont contrôlables de façon indépendante, ce qui permet par exemple au robot

[3]Dans le cadre des taches que nous avons envisagées, cette hypothèse est très raisonnable.



(a) $\phi_0 = 0$



(b) $\phi_0 = -\phi_{max}$

Figure 3.12 : Exemples de Cartes Dynamiques pour une voiture avec $L = 0.22$, $\Delta T = 0.033$, $\phi_{max} = \pi/4.0$, $\dot{\phi}_{max} = \phi_{max}/(5.0 * \Delta T)$, $V = 1.0$ et (a) $\phi_0 = 0$, (b) $\phi_0 = -\phi_{max}$.

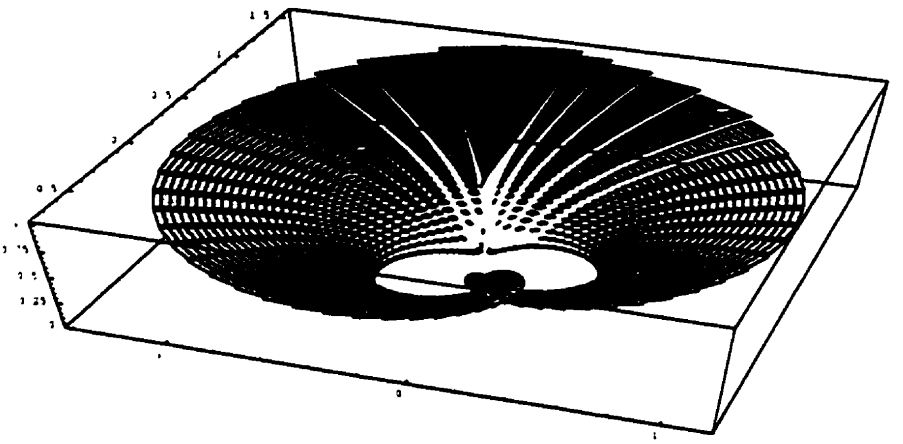


Figure 3.13 : Exemple de Carte Dynamique pour une voiture avec $L = 0.22$. $\Delta T = 0.033$. $\phi_{max} = \pi/4.0$. $\dot{\phi}_{max} = \phi_{max}/(5.0 * \Delta T)$, $V = 1.0$ et $\phi_0 = \phi_{max}/2$

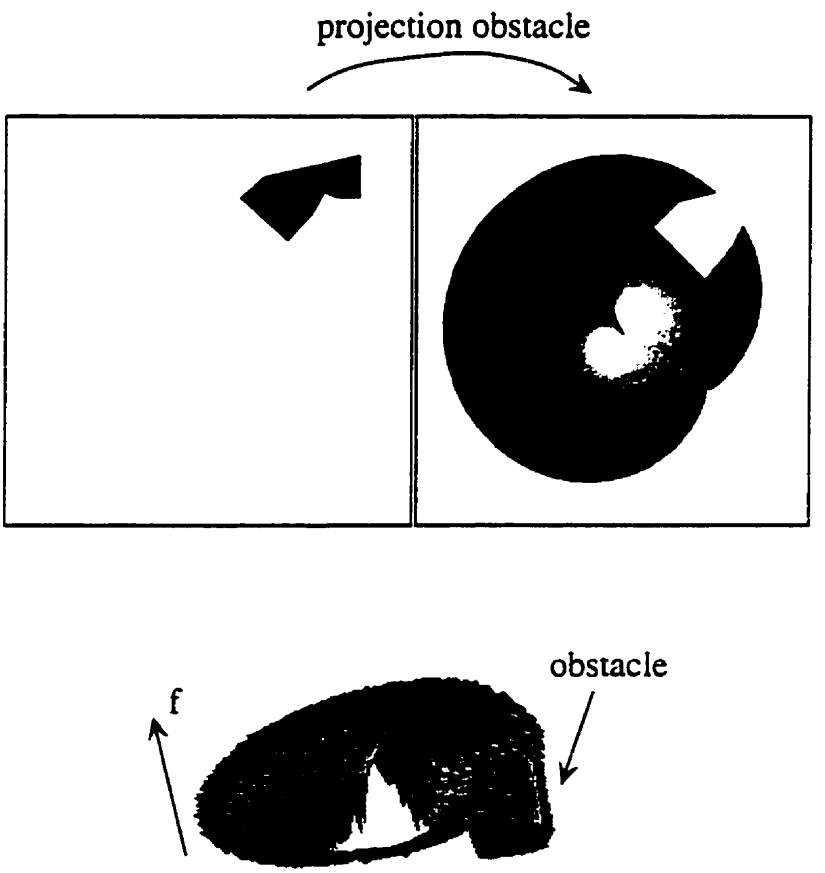


Figure 3.14 : Carte Dynamique pour un robot avec un obstacle

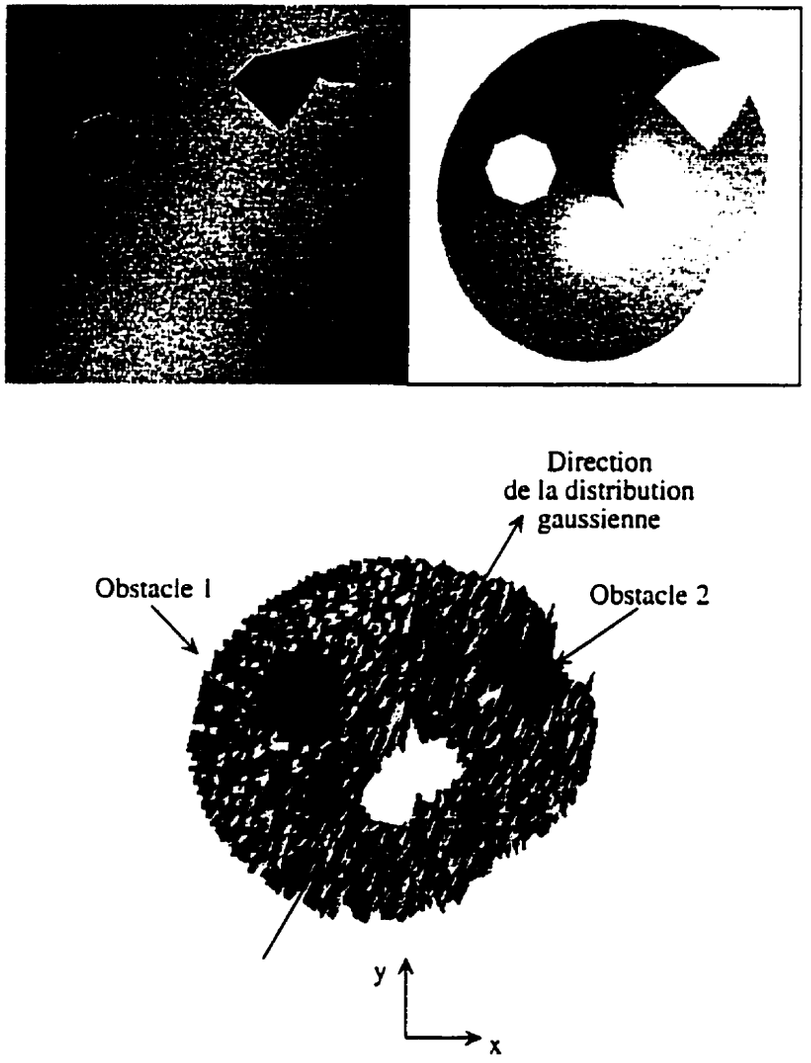


Figure 3.15 : Carte Dynamique d'un robot dans un environnement contenant deux obstacles et une distribution gaussienne symbolisant une direction de mouvement privilégiée

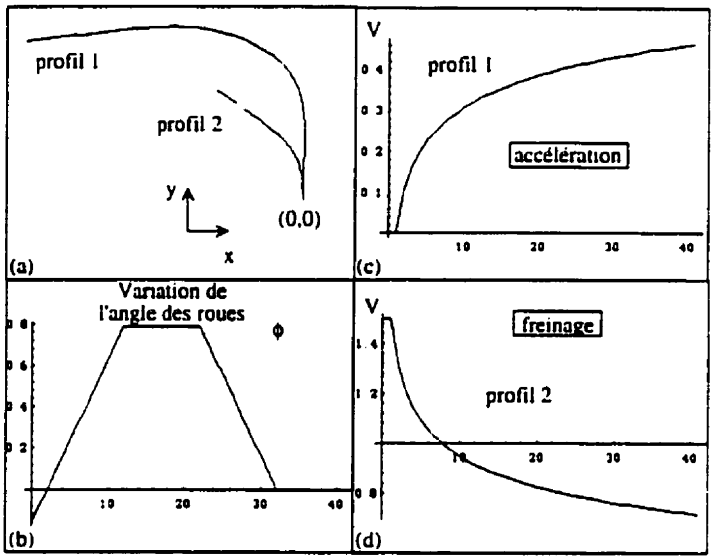


Figure 3.16 : (a) La même courbe RS (définie par la variation de ϕ (b)) pour plusieurs profils de vitesse : (c) accélération et (d) freinage.

de tourner sur lui-même. (Reister et Pin 1994) présente une étude complète sur les régions atteignables de ce type de robot. Leur contexte est toutefois un peu différent par rapport à nos objectifs puisque leur robot est censé toujours partir d'un état statique, faire une trajectoire optimale et s'arrêter complètement.

Le modèle cinématique du robot présenté à la figure 3.17 est :

$$\begin{cases} \dot{\theta} &= \frac{(\omega_D - \omega_G)}{D}, \\ \dot{x} &= \frac{(\omega_D + \omega_G)}{2}\cos(\theta), \\ \dot{y} &= \frac{(\omega_D + \omega_G)}{2}\sin(\theta), \\ \dot{\omega}_D &= u_D,\ \dot{\omega}_G = u_G, \\ \dot{\gamma}_D &= \omega_D,\ \dot{\gamma}_G = \omega_G, \\ |u_G| &\leq u_{max},\ |u_D| \leq u_{max}, \\ |\omega_G| &\leq \omega_{max},\ |\omega_D| \leq \omega_{max}, \end{cases} \tag{3.21}$$

où (x, y) est la position du milieu de l'axe des roues, θ l'orientation du robot. ω_D, ω_G les vitesses des roues droite et gauche en mètres par seconde. γ_D, γ_G les translations des roues droite et gauche (la vitesse angulaire multipliée par le rayon des roues)

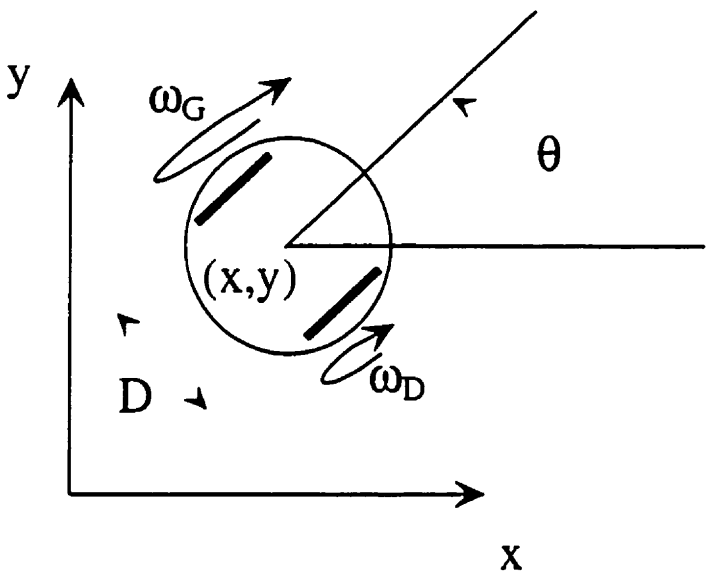
mesurées en mètres, D la distance entre les roues et u_D, u_G les contrôles appliqués au robot. Dans (Reister et Pin 1994), il a été prouvé que les courbes optimales étaient obtenues grâce à des contrôles de types "bang-bang" (c'est-à-dire $u_G = \pm u_{max}$ et $u_D = \pm u_{max}$). Une chose importante n'est cependant pas prise en compte dans cet article : les vitesses maximales des roues ne sont jamais considérées. Il semble en effet que les accélérations maximales (ou minimales) soient appliquées sans donner de bornes supérieures sur les vitesses des roues. Cependant, l'effet de ce défaut n'est pas évident, étant donné que leurs trajectoires sont relativement courtes, et que le robot part à l'arrêt et s'arrête à la fin de chaque trajectoire. Il est nécessaire pour nous de considérer cet effet et d'ajouter une borne sur les vitesse des roues.

Nous allons créer la frontière extérieure des régions atteignables de la façon suivante : pour une valeur initiale de ω_G et ω_D, un contrôle est généré avec $u_G = \pm u_{max}$ et $u_D = \mp u_{max}$, jusqu'à ce que $\omega_G = -\omega_D = \pm \omega_{max}$, ce qui représente une situation où le robot tourne sur lui-même le plus vite possible. A partir de chaque point de la courbe ainsi générée, on applique un contrôle u_{max} sur u_G et u_D. Cela permet ainsi de trouver l'équivalent de la courbe de transition faisant passer de κ_{max} vers zéro. Finalement, de la même façon que pour la voiture, on génère les régions T-atteignables en reliant l'ensemble des points au même temps T.

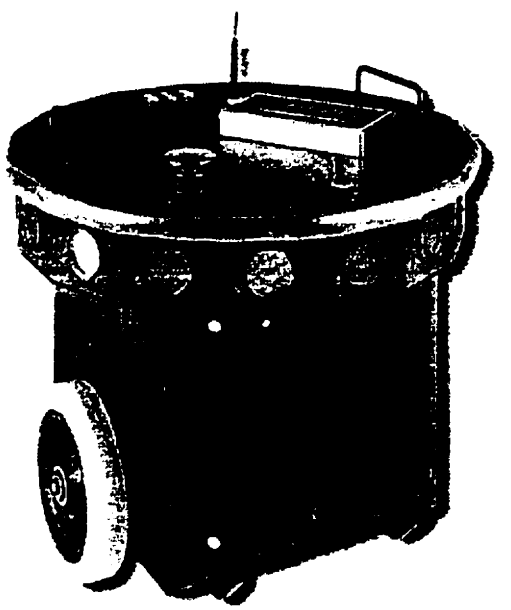
La figure 3.18 résume ce processus de construction. Le calcul des fonctions se fait de la même façon que pour le robot voiture. La figure 3.19 présente quelques exemples de Carte Dynamique d'un robot "skid-steer" pour plusieurs conditions initiales différentes.

3.2.3 Carte Dynamique d'un véhicule articulé

Nous considérons ici un véhicule articulé se composant de deux parties identiques aux robots à deux roues motrices indépendantes de la section précédente, qui sont liées et dont l'orientation relative est contrôlée par un vérin. Le véhicule est donc contrôlé par l'intermédiaire de ce vérin et des vitesses des roues de chaque élément



(a)



(b)

Figure 3.17 : (a) Modèle d'un véhicule à deux roues motrices indépendantes ("skid-steer" platform) (b) Un exemple de véhicule : le robot SCOUT Nomad de la compagnie Nomadic Technologies, Inc.

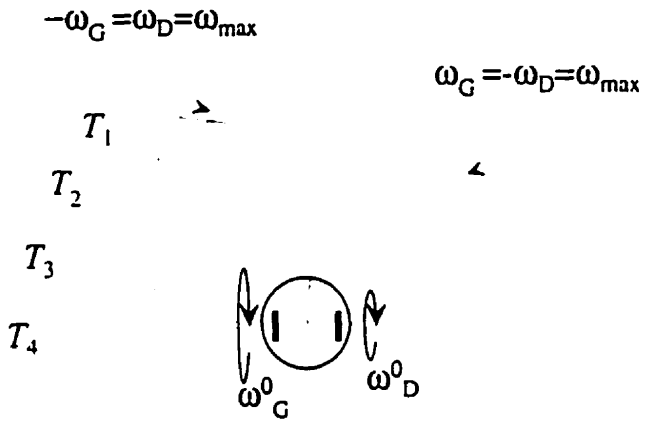


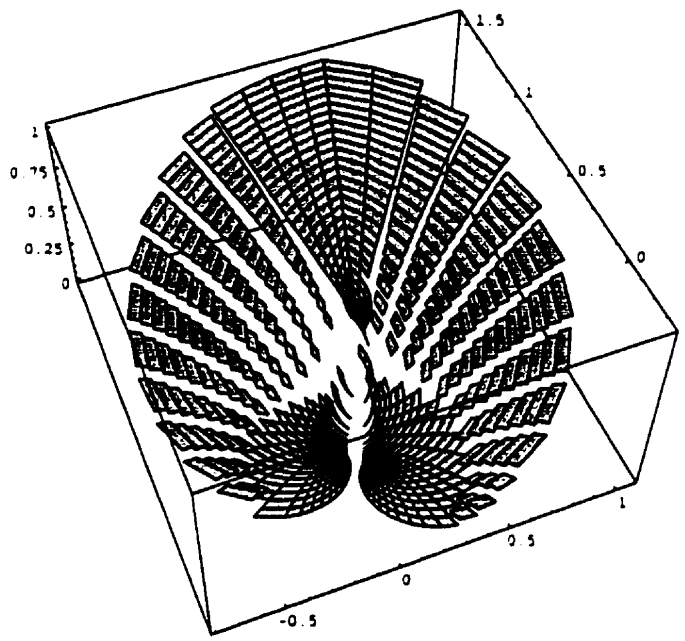
Figure 3.18 : La construction des régions T-atteignables pour un robot de type "Skid-Steer"

(une relation existe toutefois entre les vitesses des roues de chaque élément en fonction de l'angle de pliage (vérin) pour éviter le glissement des roues). Le modèle cinématique du véhicule articulé présenté à la figure 3.20 est (voir (Polotski 1995)) :

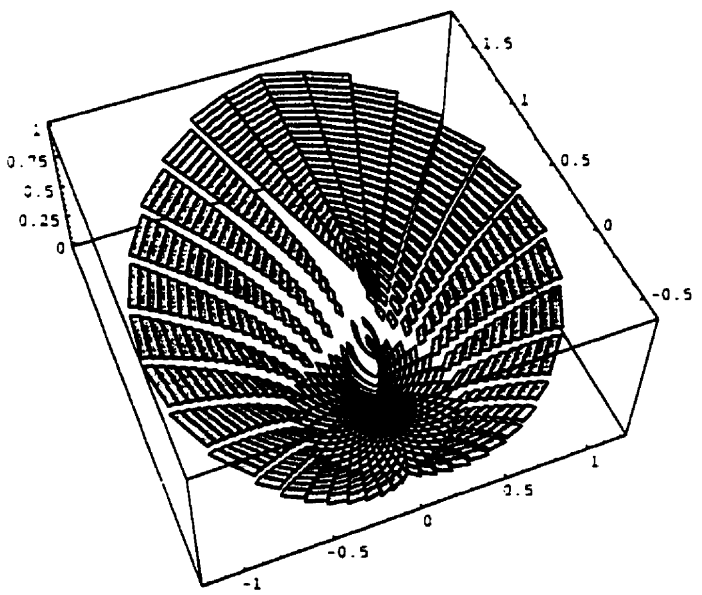
$$\begin{cases} \dot{x} = V\cos(\theta), \\ \dot{y} = V\sin(\theta), \\ \dot{\theta} = \frac{V}{L_1}\tan(\delta), \\ \dot{\phi} = V\left(\left(\frac{1}{L_1} + \frac{\cos(\phi)}{L_2}\right)\tan(\delta) - \sin(\phi)\right). \\ \dot{\phi} = u, \\ |u| \leq u_{max}, \\ |\phi| \leq \phi_{max}, \end{cases} \tag{3.22}$$

où (x, y) sont les coordonnées du premier élément du véhicule, θ est l'orientation du premier élément, δ la direction du mouvement du centre de pliage, ϕ l'angle de pliage et u le contrôle appliqué pour changer l'angle de pliage.

La Carte Dynamique du véhicule est créée de la même façon que pour une voiture : les courbes limites RS et LS sont générées avec les courbes de transition et les posi-

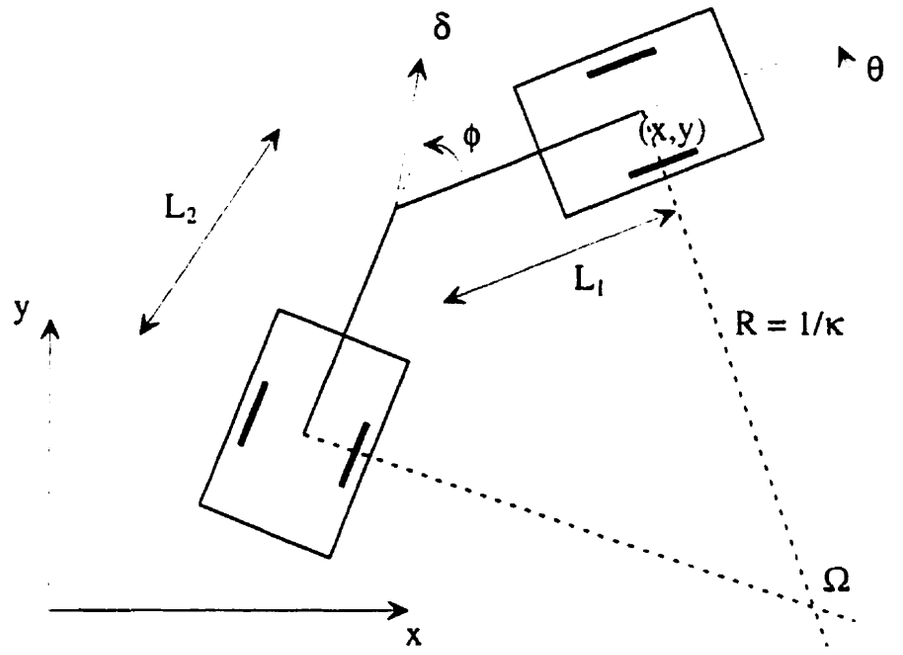


(a) $\omega_G = \omega_{max}/2$ et $\omega_D = 0$

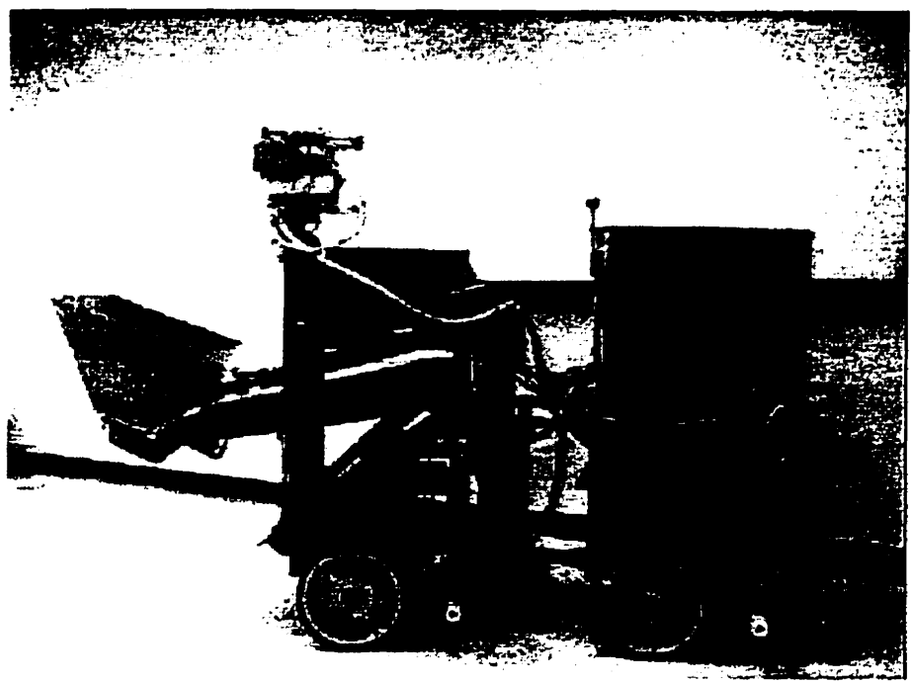


(b) $\omega_G = \omega_D = 0$

Figure 3.19 : Exemples de Carte Dynamique pour un robot à deux roues indépendantes avec $\omega_G = \omega_{max}/2$ et $\omega_D = 0$ (a) et $\omega_G = \omega_D = 0$ (b)



(a)



(b)

Figure 3.20 : (a) Modèle d'un robot articulé (b) Robot articulé développé dans le Groupe de Recherche en Perception et Robotique, Montréal, Canada.

tions atteintes au même instant sont ensuite connectées. Il est nécessaire de préciser qu'aucune étude jusqu'à présent et à notre connaissance ne précisent les trajectoires de temps minimum pour un véhicule articulé. Cependant, dans la situation ou le véhicule est en marche avant seulement, le problème est équivalement à celui d'une voiture.

Les fonctions sont également calculées de la même façon que pour une automobile.

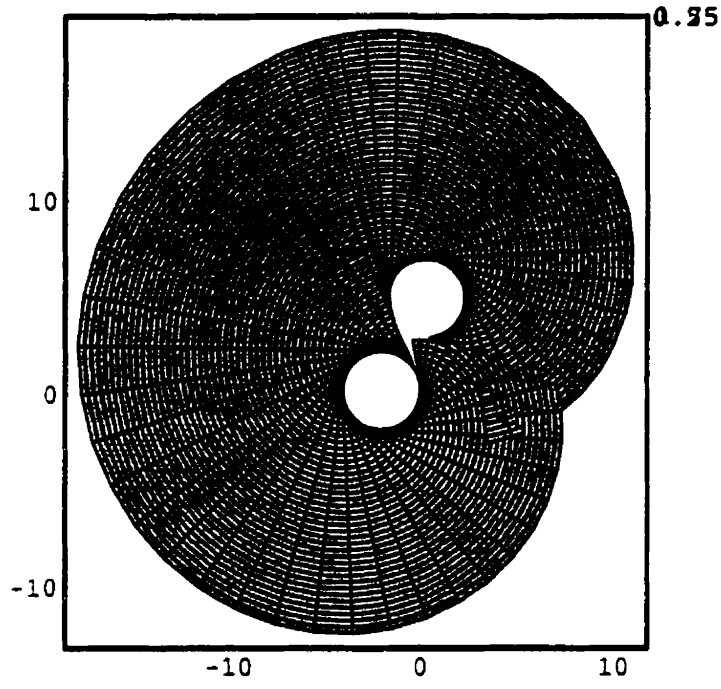
La figure 3.21 présente les exemples de quelques Cartes Dynamiques pour des orientations initiales différentes de ϕ et différentes fonctions.

Bien que nous ayons fait l'étude des Cartes Dynamiques dans le cas de trois modèles cinématiques différents de robots, il serait préférable dans le futur de vérifier la validité de la Carte Dynamique pour toutes les classes de robots mobiles telles qu'elles sont définies dans (Campion, Bastin et D'Andréa-Novel 1996). Des modèles plus complexes pour des robots incluant les forces dynamiques pourraient également être utilisés (Julier et Durrant-Whyte 1995).

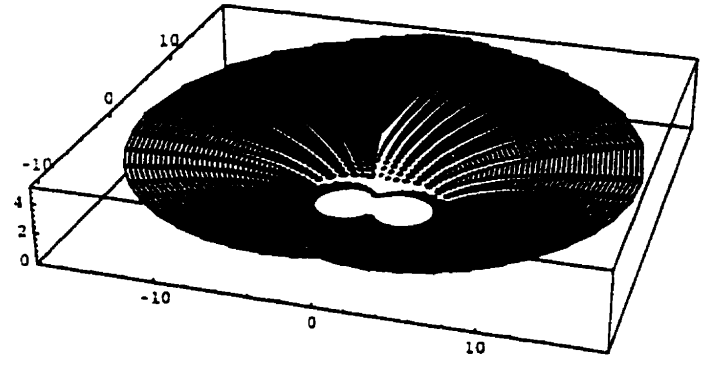
3.3 Représentation des Cartes Dynamiques

Afin de ne pas être liés aux modèles dynamiques des robots et de permettre l'apprentissage des Cartes Dynamiques, nous allons présenter dans cette section une autre façon de représenter les régions atteignables. En effet, l'utilisation du modèle dynamique des robots est contraignante et n'offre pas une grande flexibilité en ce qui concerne les possibles variations par rapport à la réalité. De plus, on peut vouloir apprendre la Carte Dynamique d'un robot sans connaître à l'avance le modèle dynamique sous-jacent.

Comme nous l'avons vu dans les trois exemples de Carte Dynamique d'un automobile, d'un véhicule à deux roues indépendantes et d'un véhicule articulé, les courbes extrêmes correspondant à la variation maximale de l'angle des roues de la voiture, de l'accélération maximale des roues du véhicule à deux roues indépendantes ou de la

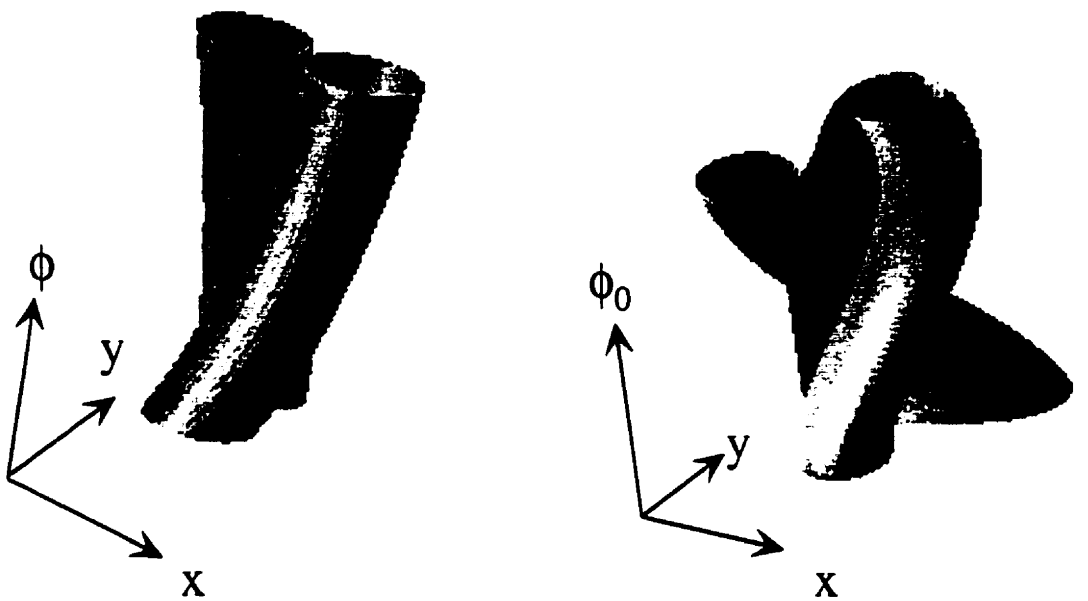


(a) $\phi = \phi_{max}$



(b) $\phi = 0$

Figure 3.21 : Exemples de Carte Dynamique pour un véhicule articulé avec $\phi = \phi_{max}$ (a) et $\phi = 0$ (b)



(a) (b)

Figure 3.22 : Exemples de volume pour différentes valeurs des paramètres (a) $V = 1.0m/s$, $\phi_{max} = \pi/4$, $\Delta T = 33ms$, $\dot{\phi}_{max} = \phi_{max}/(5\Delta T)$, $L = 0.22$ (b) $V = 1.5m/s$, $\phi_{max} = 3\pi/8$, $\Delta T = 33ms$, $\dot{\phi}_{max} = \phi_{max}/(20\Delta T)$, $L = 1.0$

variation maximale de l'angle de pliage du véhicule articulé sont primordiales pour la construction des cartes. Ces courbes dépendent bien sûr des conditions initiales du système, comme par exemple l'angle initial des roues de la voiture.

Par exemple, dans le cas d'une automobile, si nous représentons dans l'espace (x, y, ϕ_0), l'ensemble de ses courbes, nous obtenons un volume qui dépend des caractéristiques du robot. Les figures 3.22 et 3.23 présentent plusieurs volumes créés pour différentes valeurs des paramètres du modèle. Toutefois cette représentation dépend du modèle du véhicule par l'intermédiaire de ϕ et ne pourra pas être utilisée pour apprendre les capacités d'un autre robot sans connaître auparavant son modèle dynamique. Il est plus intéressant de voir ce volume dans l'espace (x, y, κ), où κ représente la courbure instantanée de la trajectoire. En effet, la courbure est une variable différentielle qu'il est possible d'extraire d'un ensemble de données sans con-

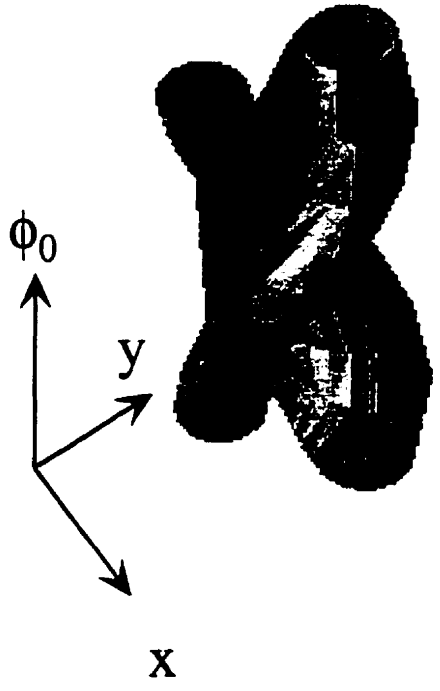
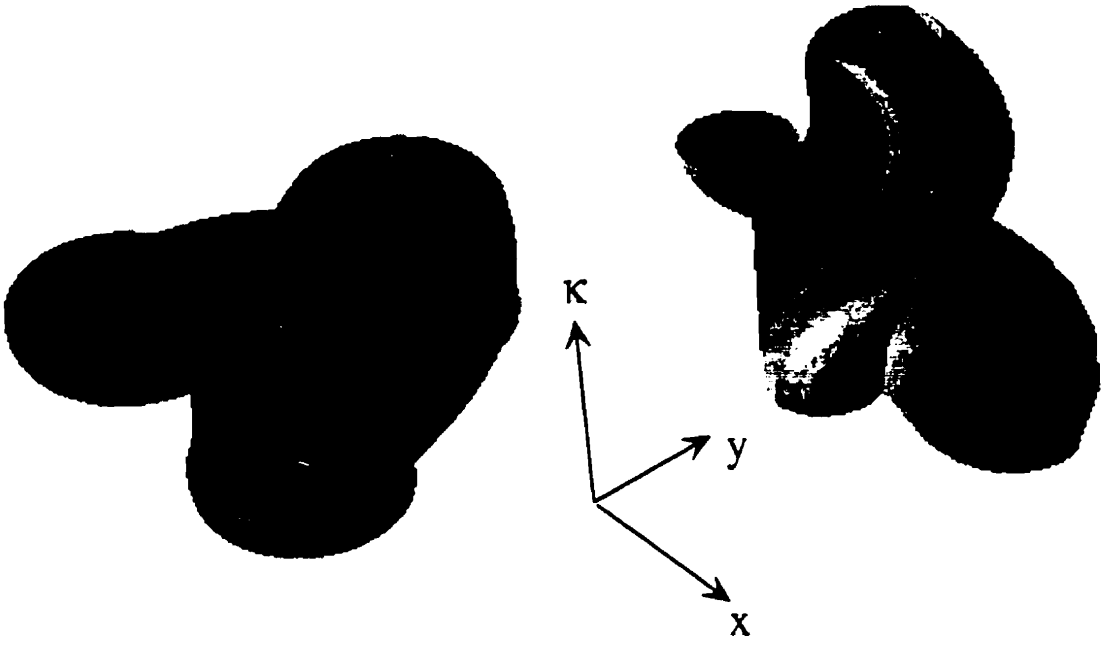


Figure 3.23 : Exemple d'un volume de représentation des Cartes Dynamiques pour les paramètres $V = 1.0m/s$, $\phi_{max} = \pi/4$, $\Delta T = 100ms$, $\dot{\phi}_{max} = \phi_{max}/(5\Delta T)$, $L = 0.22$



(a) (b)

Figure 3.24 : Exemples de volume reconstruit pour le véhicule articulé (a) $V = 10.0m/s$, $\phi_{max} = \pi/3$, $\Delta T = 33ms$, $\dot{\phi}_{max} = \phi_{max}/(10 * \Delta T)$, $L_1 = 1.5$, $L_2 = 1.0$ (b) $V = 30.0m/s$, $\phi_{max} = \pi/4$, $\Delta T = 100ms$, $\dot{\phi}_{max} = \phi_{max}/(10 * \Delta T)$, $L_1 = 5.0$, $L_2 = 3.5$

naître *a priori* le modèle de l'objet considéré. Dans le cas de la voiture, $\kappa = \tan(\phi)/L$, ce qui correspond à une relation bijective sur l'intervalle de définition $]-\pi/2; \pi/2[$ de ϕ. Cette transformation n'induit donc aucune perte d'information d'un espace à l'autre. Il en est de même dans le cas du véhicule articulé, avec $\kappa = \sin(\phi)/(L_2 + L_1 \cos(\phi))$ qui est bijective sur $] - \pi/2; \pi/2[$. Les figures 3.26, 3.24 et 3.25 donnent des exemples de volume de représentation de la carte dynamique dans l'espace (x, y, κ) pour l'automobile et le véhicule articulé.

Pour le véhicule à deux roues motrices, la situation est un peu différente puisqu'il est nécessaire de prendre en compte la vitesse initiale du robot. Toutefois, pour une vitesse V_0 initiale fixe, on a initialement

$$\frac{\omega_D + \omega_G}{2} = V_0 \tag{3.23}$$

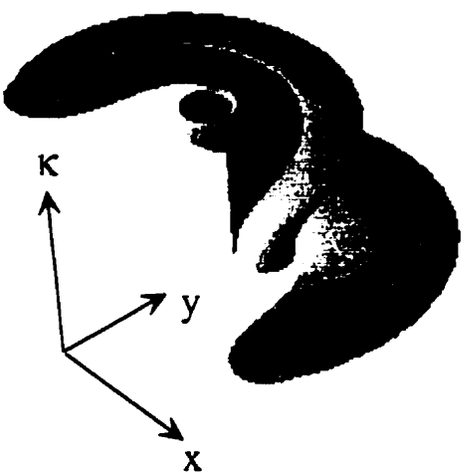


Figure 3.25 : Exemple de volume pour le véhicule articulé. $V = 30.0m/s$, $\phi_{max} = \pi/6$, $\Delta T = 100ms$, $\dot{\phi}_{max} = \phi_{max}/(20 * \Delta T)$, $L_1 = 2.0$, $L_2 = 3.5$

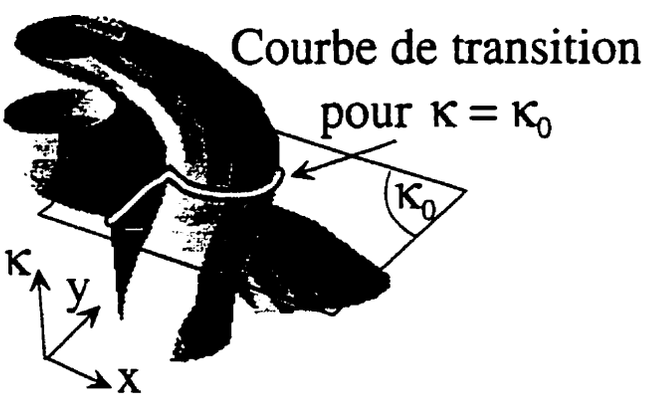
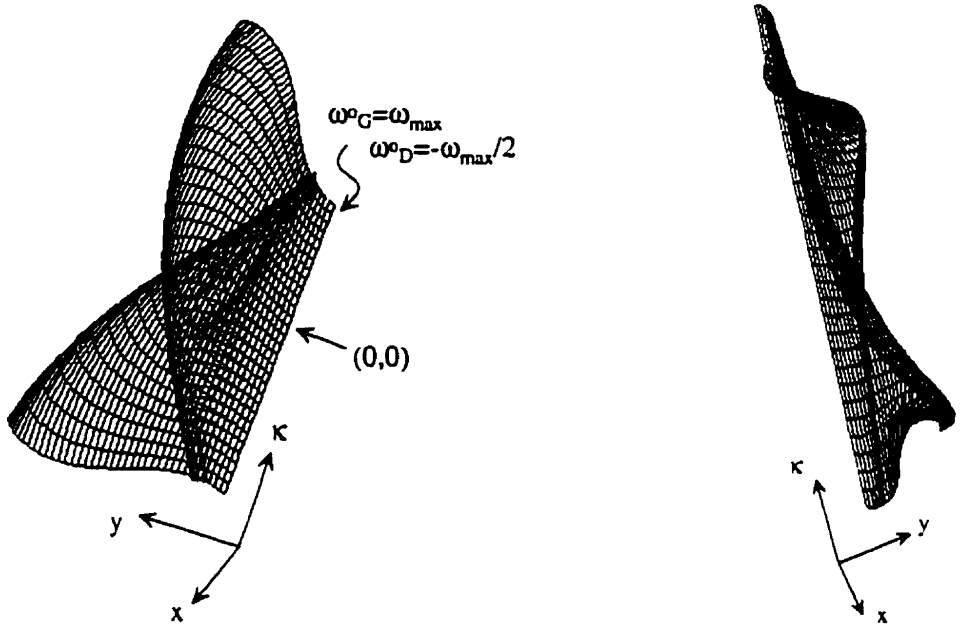


Figure 3.26 : Volume de représentation des Cartes pour une automobile dans l'espace (x, y, κ) et $V = 1.0m/s$, $L = 0.22$, $\phi_{max} = \pi/4$, $\Delta T = 100ms$, $\dot{\phi}_{max} = \phi_{max}/(5 * \Delta T)$



(a) (b)

Figure 3.27 : Exemples de volume de représentation pour le véhicule à deux roues indépendantes (a) $V = \omega_{max}/2$, $D = 1.76$, $\Delta T = 33ms$, $u_{max} = 5.0$, $\omega_{max} = 20 * \Delta T * u_{max}$ (b) $V = \omega_{max}/2$, $D = 0.76$, $\Delta T = 33ms$, $u_{max} = 5.0$, $\omega_{max} = 20 * \Delta T * u_{max}$

et la relation entre la courbure κ et les vitesses initiales des roues sera bijective. En effet, pour le modèle du véhicule à deux roues indépendantes l'équation de la courbure est :

$$\kappa = \frac{(\omega_1 - \omega_2)}{D * (\omega_1 + \omega_2)} \tag{3.24}$$

$$V_0 = \omega_1 + \omega_2 \tag{3.25}$$

Il y aura donc un volume pour chaque valeur de V_0 considérée. Ceci ne restreint pas la généralité de la représentation, puisque la valeur de la vitesse peut également être extraite d'un ensemble de données de la même façon que la courbure. Les figures 3.27 et 3.28 présentent plusieurs volumes pour un même modèle pour des vitesses initiales différentes. On peut également remarquer la forme un peu spéciale des volumes

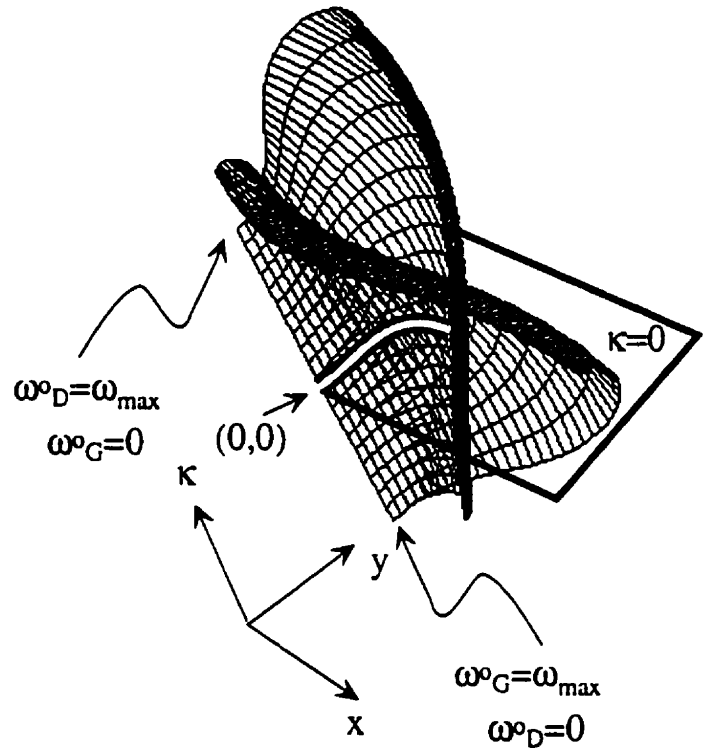


Figure 3.28 : Volume de représentation des Cartes Dynamiques d'un robot à deux roues indépendantes $V = \omega_{max}$, $D = 0.76$, $\Delta T = 33ms$, $u_{max} = 5.0$, $\omega_{max} = 20 * \Delta T * u_{max}$

dans le cas de ce robot. Cela provient de sa capacité à tourner sur lui même, d'où il découle que les cercles C_L ou C_R mentionnés auparavant sont de rayon nul, quelles que soient les bornes sur les vitesses des roues.

Il est nécessaire de montrer que cette représentation est suffisante pour construire la Carte Dynamique du robot. Il est ainsi possible d'obtenir les courbes de transition, mentionnées lors de la phase de construction de la Carte Dynamique d'une voiture, qui font passer d'une courbure maximum (correspondant dans le cas de la voiture à $\tan(\phi_{max})/L$) à une courbure nulle (voir figure 3.7) directement à partir du volume. Par exemple, si l'on prend la tranche du volume pour $\kappa = 0$, on obtient la courbe $\kappa(t)$, avec $\kappa(0) = 0$ et $\kappa(T) = \pm\kappa_{max}$ (correspondant donc à une courbe de transition suivi d'une courbe R ou L). La valeur de T désigne ici le temps au point de contact entre le cercle C_R ou C_L et la courbe de transition. La courbe de transition passant de $\kappa = -\pm\kappa_{max}$ à 0 (correspondant donc à une courbe de transition suivi d'une courbe S), de résoudre le système suivant (voir figure 3.29) :

$$\begin{cases} \dot{x} = V\cos(\theta), \\ \dot{y} = V\sin(\theta), \\ \dot{\theta} = V\kappa(T-t), \end{cases} \tag{3.26}$$

$$t \in [0; T], \tag{3.27}$$

où $x(0), y(0), \theta(0)$ correspondent au point considéré pour créer la courbe de transition, et $\kappa(t)$ est la courbe mentionnée plus haut. Cela permet d'obtenir la courbe de transition, et cela quel que soit le modèle dynamique du véhicule.

Dans l'ensemble des exemples que nous avons présentés pour différents modèles de véhicules, on remarque que la représentation est topologiquement identique et similaire quel que soit le modèle de véhicule, et que la Carte Dynamique permet ainsi de se soustraire à une dépendance structurelle des robots donnée par leurs modèles dynamiques. Si on considère une population de robots hétérogènes, ce genre de représentation permet d'unifier l'approche de planification et d'obtenir alors une

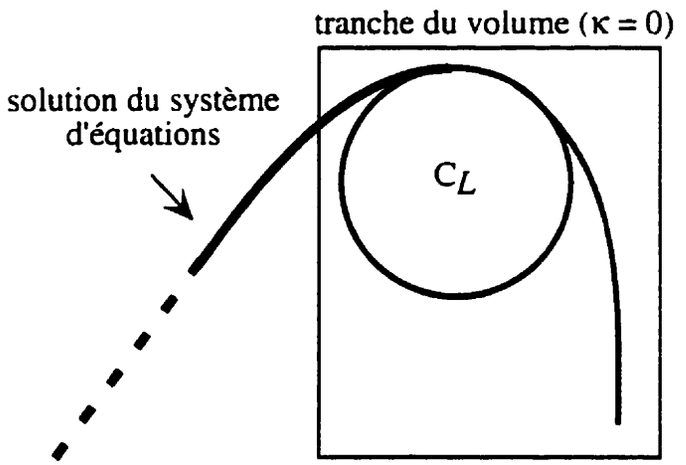


Figure 3.29 : Construction d'une courbe de la Carte Dynamique à partir du volume de représentation

indépendance structurelle intéressante. Cette représentation expose également les capacités limites de déplacement du robot.

3.4 Utilisation de la Carte Dynamique dans le plan image d'un robot

La Carte Dynamique constitue une abstraction (ou représentation) des capacités cinématiques d'un robot. Toutefois, bien que la représentation de la Carte soit intrinsèque au robot, l'aspect sensoriel semble être seulement une source externe d'information. Dans cette section, nous allons tenter de franchir ce fossé pour montrer l'étroite relation qui existe réellement entre la Carte Dynamique et le (ou les) senseur du robot.

Nous allons nous concentrer essentiellement sur les capteurs visuels passifs tels que caméras CCD (dont nos robots sont équipés comme expliqué au chapitre 6), bien que d'autres types senseurs puissent être substitués sans problème apparent.

Le premier aspect intéressant avec une "Carte Dynamique" dans le plan image, réside dans l'utilisation directe qui est faite des données sensorielles en évitant l'utilisation de transformations inverses pour aller du plan image vers la Carte Dynamique

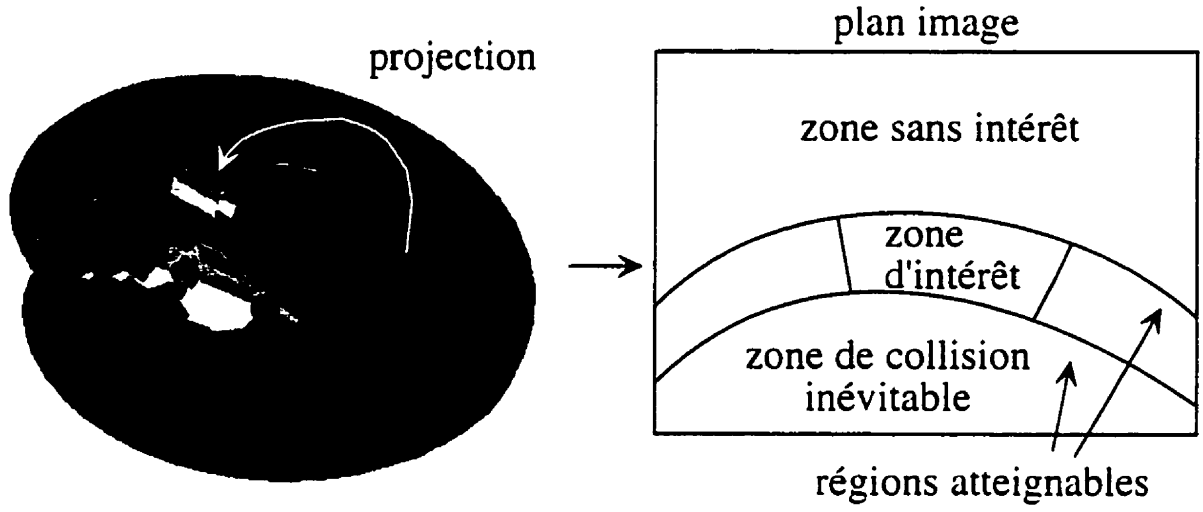


Figure 3.30 : La Carte Dynamique, le robot et son senseur

bidimensionnelle. De plus, un examen de la plupart des techniques actuelles d'analyse d'images révèle qu'une partie non négligeable du temps est consacrée à réduire l'information en provenance du ou des senseurs à sa partie signifiante. Ce point de vue est profondément défendu par les travaux de Dickmanns (Dickmanns et Graefe 1988*b*). L'intérêt de porter alors la Carte Dynamique dans l'image est d'obtenir un senseur "déformé" suivant les capacités de déplacement du robot. Par exemple, un algorithme coûteux d'évitement d'obstacles peut être amélioré en désignant *a priori* les zones de l'image que le robot aura "tendance" ou sera capable d'atteindre. Cette information est fournie par la Carte Dynamique une fois qu'elle est projetée dans le plan image de la caméra (ou n'importe quel espace de senseur pour autant qu'il existe une méthode de projection depuis l'espace de coordonnées du robot vers le senseur). La figure 3.30 montre la Carte Dynamique dans le plan 2D avec la relation avec sa caméra.

Une application possible de la Carte Dynamique dans l'image est également de prévoir en fonction de la trajectoire planifiée, la zone image où se trouvera le robot dans le futur par rapport à l'instant présent et ainsi prédire si cette trajectoire est faisable pour le robot. Dans le cas contraire, une replanification de la trajectoire est demandée pour éviter par exemple de rentrer en contact avec un obstacle. La repro-

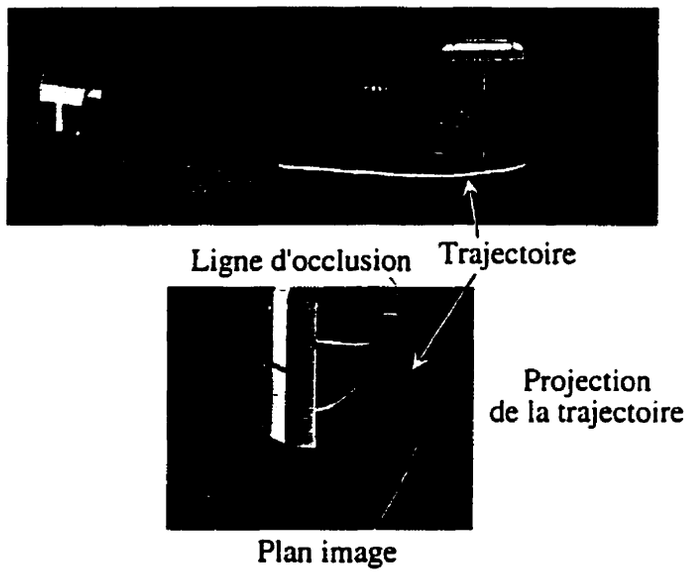


Figure 3.31 : Reprojection de trajectoire planifiée dans le plan image pour vérifier la cohérence des chemins. Par exemple une trajectoire passant derrière un obstacle doit couper une des lignes d'occlusion de cet obstacle

jection dynamique de trajectoire (voir figure 3.31) peut éventuellement être utilisée.

Construction de la Carte Dynamique dans le plan image La construction de la Carte Dynamique dans le plan image est faite à partir de la Carte déjà construite dans l'espace du robot. Pour cela, il est nécessaire de connaître la méthode de projection depuis l'espace du robot vers le plan de la caméra. Cette projection nécessite la connaissances de plusieurs transformations : en premier la relation entre la Carte Dynamique et le référentiel du robot (immédiat), ensuite la transformation entre le référentiel du robot et le repère de la caméra (transformation robot/senseur), finalement la projection dans le plan image de la caméra (voir figure 3.32). La dernière étape est facilement obtenue par une calibration de la caméra à l'aide du même logiciel de calibrage de Tsai-Wilson mentionné dans le chapitre 6. La difficulté se situe au niveau de la transformation robot/caméra. En effet, il est délicat de trouver la relation entre la caméra et le référentiel du robot, comme le prouvent les nombreux travaux existants (Hervé 1993).

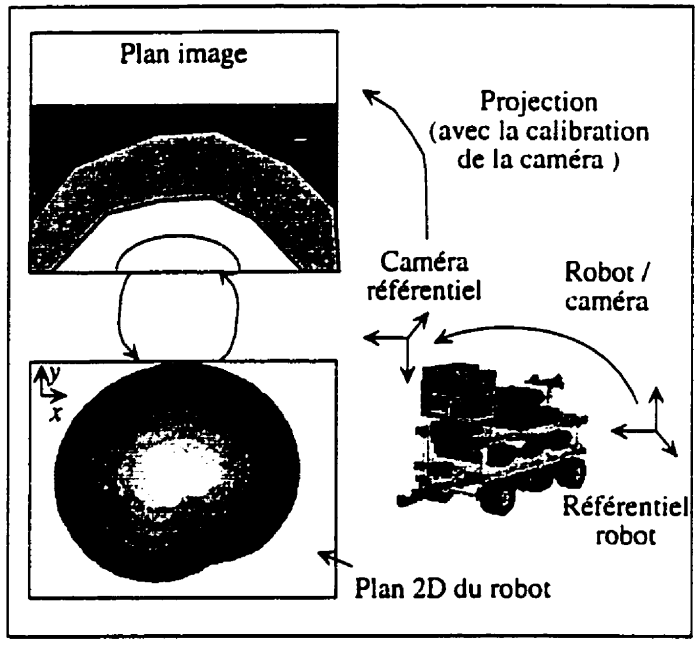


Figure 3.32 : Projection de la Carte Dynamique dans le plan image

Il existe cependant un moyen simple de faire la calibration de la caméra et de déterminer la transformation robot/caméra. La calibrage de la caméra nécessite une grille de calibration disposée sur le sol, afin de connaître la relation entre les points de l'image et les points dans le plan de la grille. A partir de cette relation, une calibration de type Tsai (Tsai et Huang 1984) est utilisée. Il faut savoir que le résultat de la calibration est d'obtenir à la fois les paramètres intrinsèques de la caméra (distance focale, taille des pixels de la caméra, *etc.*) ainsi que les paramètres extrinsèques définissant les transformations entre le repère de la caméra et le repère associé à la grille. L'astuce est donc de créer une grille dont le repère sera volontairement placé au milieu de l'essieu arrière du robot (bien que le placement soit fait à la main, un repérage physique peut être utilisé pour améliorer la précision). De ce fait la relation entre la Carte Dynamique et le plan image est immédiate (pour une caméra rigidement fixée au robot) puisque le repère de la grille se déplace avec le robot et que la configuration de la caméra n'évolue pas.

Fonctions de la Carte Dynamique dans le plan image Par analogie aux Cartes Dynamiques dans l'espace de configuration du robot, les fonctions utilisées pour la planification peuvent être définies dans le plan image pour aider à l'analyse des séquences d'images.

Les possibilités de fonctions sont nombreuses. Elles pourront être choisies suivant la tâche à accomplir. Toutefois, la représentation des fonctions est différente dans le cas de l'image. En effet, une valeur numérique est sans signification dans une image, et la représentation des valeurs doit être cohérente avec la physique de la caméra. Par exemple, la valeur d'une fonction peut correspondre à un masque pour créer du flou ("blurr"), ou à une diminution de la résolution dans la partie considérée.

Citons plusieurs possibilités de fonctions :

- La valeur du temps avec la forme des courbes T-atteignables indique des zones de sécurité. Ainsi, un algorithme d'évitement d'obstacle peut mettre à profit cette information pour savoir quelles zones de l'image doivent être analysées en premier pour détecter un risque de collision (en quelque sorte cela sert de Temps-avant-Collision (TTC) augmenté avec les régions T-atteignables. En effet, chaque région de l'image est étiquetable avec la valeur du temps associée. Chaque étiquette indique alors le temps avant que le robot arrive sur la région désignée. (Kundur et Raviv 1994) introduit également ce genre d'indice visuel (VTC : "Visual Threat Cue")).

- La valeur de la fonction du nombre de trajectoires permet d'indiquer le taux de transparence d'un masque sur l'image. Les régions peu susceptibles d'être visitées par le robot (et ayant ainsi un valeur faible de la fonction) peuvent donc être cachées afin de diminuer les zones de recherches d'algorithme de détection de mouvement indépendant.

- La valeur des fonctions combinées du nombre de trajectoires avec la valeur du temps associée à la région T-atteignable permet de réduire la résolution

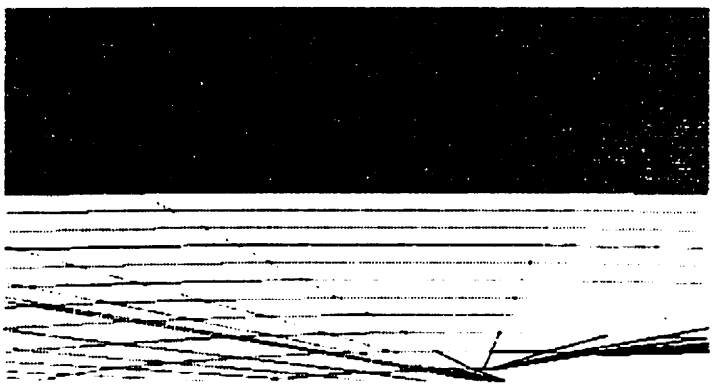


Figure 3.33 : Exemple de courbes T-atteignables projetées dans le plan image avec $\phi_0 > 0$

dans certaines portions de l'image dans le but de réduire l'information de façon cohérente pour d'autres algorithmes d'analyse d'images.

- *etc.*

La combinaison des Cartes comme nous allons le présenter dans le chapitre suivant est également un élément important de la modélisation des interactions entre robots mobiles. Il est ainsi possible de la même façon que pour les fonctions de projeter ces interactions dans le plan image pour faciliter l'étude des images pour la planification. Les interactions à nouveau seront représentées par des masques, ou des régions d'attention (dans (Labonté 1997), les régions d'attention sont utilisées pour des systèmes de codage intelligent. Les Cartes Dynamiques fournissent une méthode de création de ces régions d'attention en se basant sur une réalité physique (les régions atteignables) et des informations supplémentaires en fonction de la tâche) qui seront appliquées sur l'image.

Les figures 3.33 et 3.34 présentent deux exemples de Cartes Dynamiques dans le plan image. Pour l'instant, les techniques de planification et les fonctions dans l'image n'ont pas été implantées.

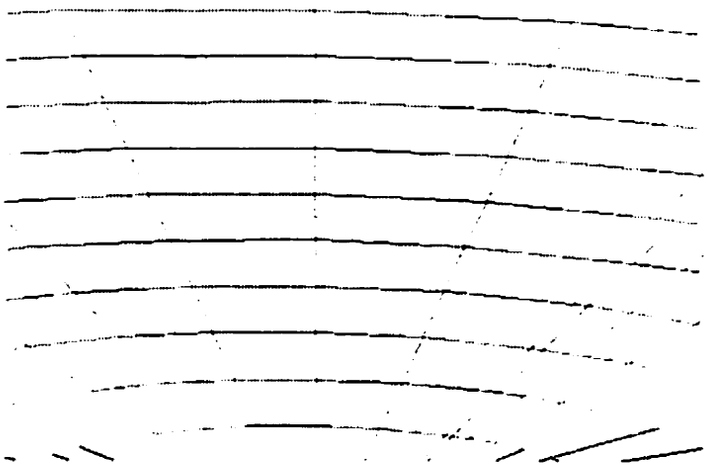


Figure 3.34 : Projection des courbes T-atteignables pour ϕ_0

3.5 Conclusion

Les principaux avantages du concept de Carte Dynamique que nous avons introduit dans ce chapitre sont :

- La Carte Dynamique d'un robot représente une modélisation géométrique des capacités cinématiques du robot. La construction des Cartes Dynamiques à partir des modèles de trois robots mobiles différents a ainsi été présentée.

- La Carte Dynamique est une représentation intrinsèque invariante par rapport à la taille du robot.

- Un volume géométrique de représentation des Cartes Dynamiques a été introduit dans ce chapitre. Cette représentation est générique par rapport aux modèles de robots, fournissant ainsi une indépendance structurelle intéressante pour un système multi-agent.

- Grâce à cette représentation l'apprentissage de la forme des Cartes Dynamiques est également indépendant du modèle du robot considéré. L'apprentissage de la Carte des robots du banc d'essai expérimental est présenté dans le chapitre 5.

- La Carte Dynamique permet de projeter les capacités cinématiques d'un robot dans l'espace du senseur, fournissant ainsi un couplage entre perception et planification.

Nous allons présenter dans le chapitre suivant comment les Cartes Dynamiques de plusieurs robots peuvent être combinées afin de représenter les interactions entre ces robots.

Chapitre 4

Utilisation des Cartes Dynamiques pour la planification

Nous allons analyser dans ce chapitre, la façon d'exploiter les informations contenues dans les Cartes Dynamiques afin de planifier des trajectoires dans le cadre de tâches précises à accomplir (Zanardi, Hervé et Cohen 1997*b*). La première section va présenter la méthode générale de combinaison des Cartes Dynamiques pour représenter les interactions entre robots. Dans la section suivante, nous présenterons différentes techniques d'optimisation pour générer les trajectoires que devra ensuite suivre le robot. Notre technique de représentation des Cartes Dynamiques afin de faire une combinaison efficace est décrite par la suite, avant de présenter des exemples d'utilisation des Cartes Dynamiques pour planifier des trajectoires.

4.1 Principe général de la combinaison de Cartes Dynamiques

Comme nous l'avons vu dans le chapitre précédent, les Cartes Dynamiques représentent à la fois les capacités motrices intrinsèques d'un robot ainsi que d'autres conditions environnementales.

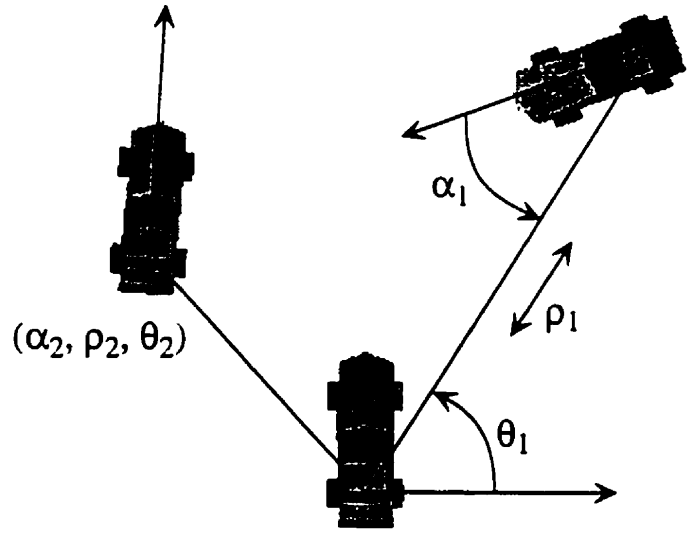


Figure 4.1 : Les paramètres de pose relative de robots

Un robot peut donc exploiter ces connaissances *a priori* sur ses capacités ainsi que celles des autres robots. Ainsi, à partir d'une estimation des poses relatives des robots envers le robot considéré, la Carte Dynamique du robot est combinée avec les estimations des Cartes Dynamiques des autres robots. La figure 4.1 présente des exemples de pose relative. L'apprentissage des Cartes Dynamiques des autres robots sera décrit dans le chapitre 5. Nous supposerons pour l'instant que le robot possède un moyen d'obtenir une estimation des capacités dynamiques des autres robots grâce à ses senseurs, ainsi que des poses relatives des robots.

Une fois que les paramètres de pose sont identifiés, le robot doit seulement appliquer une transformation aux Cartes Dynamiques des autres robots suivant les positions relatives respectives. Les différentes Cartes transformées sont alors combinées sur la Carte Dynamique du robot de façon à refléter les interactions correspondant à la tâche à accomplir.

Le robot a ainsi accès aux capacités dynamiques des autres robots ainsi qu'à des informations pertinentes telles que le nombre normalisée de trajectoires. La figure 4.2 illustre ce processus de transformation des Cartes Dynamiques. Les interactions entre

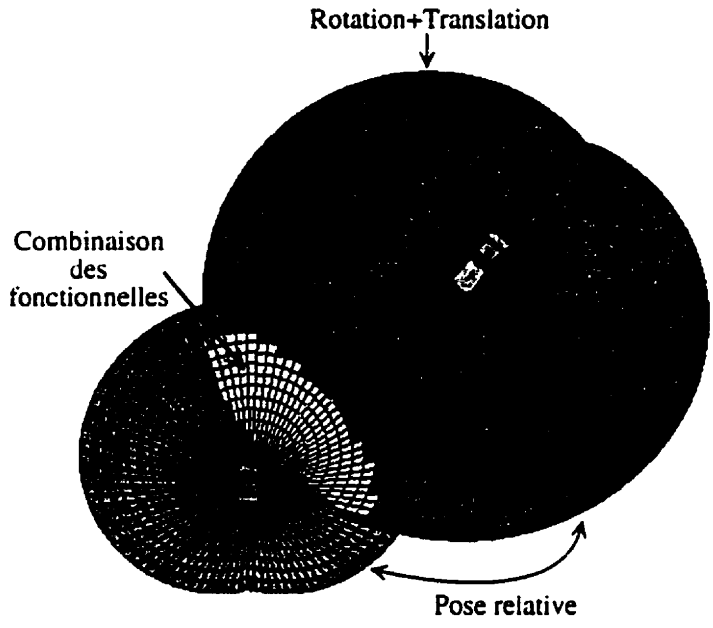


Figure 4.2 : Transformation des Cartes Dynamiques

les robots sont alors rendues explicite dans la manière de combiner les fonctionnelles de chaque robot. Par exemple, considérons le cas d'un robot mobile qui est poursuivi et qui doit donc planifier une trajectoire d'évasion. Intuitivement, le robot va essayer de trouver les zones qu'il peut atteindre avant ses adversaires et avec la meilleure "confiance" associée.

Définissons maintenant le processus de représentation des interactions entre un ensemble de robots $R_i, i \in \{0, \ldots, n\}$, du point de vue du premier robot R_0. Soient $tp_i, i \in \{1, \ldots, n\}$ les transformations de pose relative (translation et rotation) entre le robot R_0 et le robot $R_i, i \neq 0$. Considérons $g_i, i \in \{0, \ldots, n\}$, les valeurs des fonctionnelles respectives des robots $R_i, i \in \{0, \ldots, n\}$, et $T_i, i \in \{0, \ldots, n\}$, une fonction donnant la valeur du temps pour la région T-atteignable dans laquelle se trouve le point, alors les interactions sont représentées par la fonctionnelle G donnée par la fonction de combinaison I_g :

$$G : (t, \mathbf{x}) \longmapsto G(t, \mathbf{x}) = I_g \left[g_0(t, \mathbf{x}), T_0(t, \mathbf{x}), g_1 \left(t, tp_1(\mathbf{x}) \right), T_1(t, tp_1(\mathbf{x})) \ldots, \right.$$

$$g_n\left(t, tp_n(\mathbf{x})\right), T_n(t, tp_n(\mathbf{x}))] \tag{4.1}$$

La fonction G peut ensuite être utilisée dans une procédure d'optimisation numérique afin de trouver une trajectoire admissible. Le problème se formule généralement de la façon suivante :

$$\max_{\mathbf{x}(T), T=\{0,\ldots,T_f\}} \sum_{T=0}^{T_f} G(T, \mathbf{x}(T)), \text{ contraint à } \begin{cases} \mathbf{x}(T) \in \text{régions } T\text{-atteignable}, \\ C(\mathbf{x}(T)) = 0, \\ D(\mathbf{x}(T)) \leq 0, \end{cases} \tag{4.2}$$

avec C et D représentant des contraintes additionnelles sur la trajectoire du robot, comme par exemple les contraintes non-holonomes d'une voiture.

Un avantage intéressant de la Carte Dynamique réside dans la possibilité de faire de la replanification dynamique (au sens donnée par Feddema (Feddema et Lee 1990) pour un bras manipulateur). En effet, considérons un robot qui a planifié sa trajectoire pour les quelques secondes suivantes. Cette trajectoire est optimale au sens de la fonction G mais les conditions environnementales ainsi que les attitudes des autres robots vont forcément s'altérer de plus en plus avec le temps. De ce fait cette trajectoire qui devient sous optimale avec le temps permet au robot d'acquérir et de traiter de nouvelles informations sur l'environnement et les autres robots et de les intégrer à nouveau sur une Carte Dynamique du robot dans le futur (car la trajectoire du robot est connue, les formes intrinsèques des Cartes Dynamiques seront connues à l'avance) qui sera alors utilisée pour faire la replanification d'une nouvelle trajectoire optimale en accord avec une nouvelle fonction G. La figure 4.3 résume cette idée de replanification dynamique.

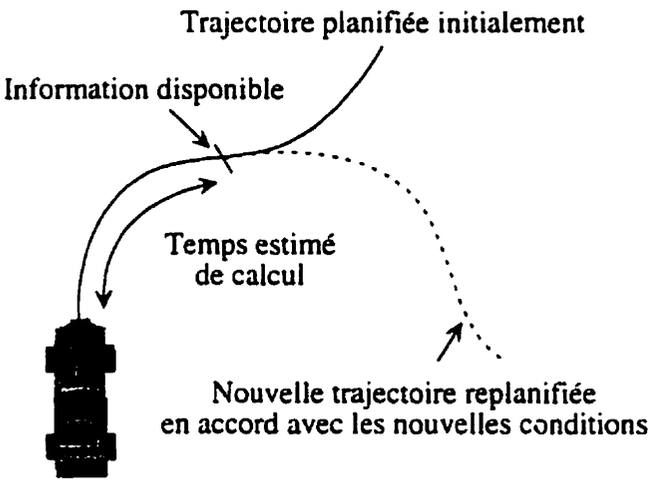


Figure 4.3 : Replanification dynamique de trajectoire par l'intermédiaire des Cartes Dynamiques

4.2 Planification de trajectoires

Une fois que les différentes Cartes Dynamiques des robots ont été combinées et que la fonction G est obtenue, il est nécessaire de planifier la trajectoire du robot. Il est également nécessaire lors de la planification de la trajectoire de prendre en compte les contraintes non-holonomes du robot concerné.

Nous avons utilisé plusieurs techniques d'optimisation numérique pour le calcul de trajectoires. Les algorithmes travaillent à différents niveaux d'abstraction (local global) afin d'étudier les effets sur les trajectoires planifiées. Nous avons assumé dans la plupart des algorithmes, que le modèle cinématique du robot considéré était connu. Cependant en ce qui concerne les autres robots, seulement leurs Cartes Dynamiques sont connues, pas leurs modèles cinématiques. Il est important de préciser que lors de procédure d'optimisation, les trajectoires obtenues ne sont pas forcément optimales. Toutefois, dans les méthodes proposées de calcul des trajectoires, chaque point des trajectoires va se trouver dans la zone entre deux régions T et $T-1$-atteignables. De ce fait, les courbes sous-optimales planifiées bénéficient des informations fournies par la Carte Dynamique bien que celle ci soit construite à partir de courbes optimales.

Le premier algorithme se base sur le plus simple algorithme d'optimisation connu : suivre la plus forte pente sur la surface tridimensionnelle formée par la fonction G dans l'espace $(x, y, \mathbb{R})$. Afin d'inclure les contraintes non-holonomes du véhicule dans la trajectoire planifiée, nous avons utilisé le modèle simple d'une voiture. L'algorithme est alors le suivant : pour la position (x, y, θ, ϕ) du système modélisant le robot, les trois positions qu'il est possible d'atteindre par contrôle "bang-bang" (voir chapitre 3) sont calculées. Parmi ces positions, celle ayant la valeur maximale de G sera choisie afin de recommencer le processus. L'avantage évident de cette technique est la rapidité pour trouver une trajectoire pour le robot. Toutefois, puisque cet algorithme d'optimisation fonctionne de façon locale il est possible de se trouver dans des situations où le robot est bloqué définitivement par un obstacle. Il est possible de remédier à ce problème en appliquant une fonction de potentiel répulsive pour les obstacles, plutôt qu'une simple valeur de présence. Toutefois, la fonction de répulsion n'est pas toujours simple à créer et de plus, beaucoup de travaux ont montré que ce genre de potentiel avait tendance à créer des minimums locaux artificiels.

Il faut cependant préciser que dans le cadre de cette méthode, il est nécessaire d'envisager des trajectoires à très court terme. En effet, la trajectoire peut devenir clairement sous-optimale rapidement puisque cette méthode ne garantie pas l'appartenance d'un point à un temps T à une région T-atteignable. Ce problème pourrait éventuellement être résolu en utilisant la valeur des temps reliés aux zones entre deux courbes T-atteignables : ainsi si le point choisi par la descente de gradient au temps $T + 1$ se trouvent entre les courbes $T - 1$ et T, ce point sera rejeté est un éventuel retour en arrière pourra être utilisé pour garantir que chaque point planifié par la descente de gradient est un point entre deux courbes optimales pour la bonne valeur de T.

Une autre façon de voir le problème est d'envisager l'optimisation globale de la trajectoire. Un premier algorithme que nous avons utilisé est basé sur une heuristique originale.

La procédure d'optimisation suppose qu'il existe un point appartenant à la trajectoire recherchée sur chaque limite extérieure des régions T-atteignables. L'algorithme fonctionne ensuite de façon heuristique en déplaçant les points sur les frontières pour obtenir une trajectoire optimale. Le but est ici d'obtenir une valeur maximale pour la sommation des valeurs de la fonction G suivant la trajectoire. L'heuristique fonctionne de la façon suivante : soit un ensemble initial de point $(x_0, y_0), \ldots, (x_{T_f}, y_{T_f})$ avec (x_i, y_i) appartenant à la frontière de la région T_i-atteignable. Soit i tel que $G(t, (x_i, y_i)) = \min_j G(t, (x_j, y_j))$. Le point (x_i, y_i) est alors déplacé à gauche et à droite sur la limite externe de la région T_i-atteignable. Les autres valeurs des points $(x_j, y_j), j \neq i$ sont également modifiées de manière à respecter les contraintes de courbure maximale le long de la trajectoire. Si le déplacement provoque une augmentation de la somme $\sum_j G(t, (x_j, y_j))$ la nouvelle suite de point est prise comme point de départ de la nouvelle itération. La maximisation s'arrête lorsqu'il n'existe plus de déplacement permettant d'obtenir une augmentation de cette somme.

Du fait de l'aspect itératif de la méthode, il est impossible de prévoir le temps nécessaire pour obtenir une trajectoire. De plus, les trajectoires obtenues maximisent la somme des valeurs de la fonction G le long du parcours, mais il se peut qu'en certains points le robot ait de grandes chances d'être attrapé, ce qui contredit le but des Cartes Dynamiques. Un exemple de trajectoires planifiées par cet algorithme sera donnée dans la section 4.4.3. La figure 4.4 illustre le fonctionnement de cet algorithme.

Les deux dernières méthodes d'optimisation que nous proposons se basent sur le même principe : un ensemble de trajectoires est testé sur la Carte Dynamique combinée et la trajectoire optimale au sens désiré est choisie. Nous avons mentionné que dans le cas de l'algorithme d'optimisation globale, c'était la somme des valeurs de la fonction G suivant la trajectoire considérée qui constituait la fonctionnelle à optimiser. Il est clair qu'il peut y avoir des cas où le robot a de grandes chances de se faire attraper le long du parcours. Pour pallier ce défaut, il est possible de faire

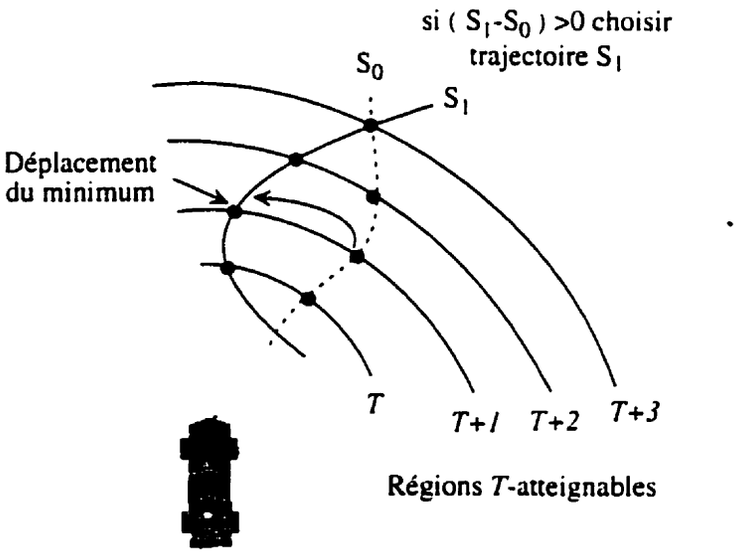


Figure 4.4 : Procédure d'optimisation heuristique pour déterminer une trajectoire

appel à une optimisation de type Minimax. En effet, avec $Traj_i, i \in \{1.....V\}$, les trajectoires testées sur la Carte, il suffit ici de choisir la trajectoire ayant sa valeur minimale égale à :

$$\max_{i=0,...,N} \left[\min_{(x,y)\in Traj_i} G(t,(x,y)) \right]$$

Dans la première des deux méthodes, nous avons choisi d'utiliser les courbes de type RS et LS qui nous ont permis initialement de construire la Carte Dynamique. Le nombre de ces courbes étant relativement faible, il suffit de les projeter dans l'espace de la Carte combinée et d'utiliser les valeurs correspondantes de la fonction G suivant le parcours pour trouver la trajectoire minimax. Le problème réside cependant dans le manque de flexibilité qu'offrent ces courbes. En effet, l'orientation finale du robot ne pourra plus varier suivant la courbe S et cela limite grandement les possibilités de replanification, de plus la valeur de l'angle des roues sera souvent maintenue à zéro, ce qui est parfois un net désavantage.

Nous avons alors proposé dans une deuxième méthode d'utiliser un réseau de courbes non-optimales, mais qui présentent l'intérêt de couvrir un spectre plus large

de valeur d'orientation et d'angle des roues possible lors de la planification. Les courbes sont formées de deux parties de même durée. La première courbe passe de la valeur initiale ϕ_0 à une valeur ϕ_1, puis dans la seconde de ϕ_1 à ϕ_2, avec $\phi_1, \phi_2 \in [-\phi_{max}; +\phi_{max}]$. Il faut supposer que le temps donné à chaque courbe est suffisant pour tourner les roues de $-\phi_{max}$ à ϕ_{max} (afin de couvrir un espace plus important). Comme dans le cas de la descente de gradient, il est impératif de ne pas prendre des trajectoires trop longues. Ainsi, les trajectoires obtenues dans le cas des réseaux de courbes sont donc sous-optimales au sens qu'elles ne sont pas formées de points appartenant aux limites des régions T-atteignables. Elles conservent cependant les critères qui sont définis entre deux régions T-atteignables. Forcément si la longueur des trajectoires du réseaux est augmentée de façon trop importante. la sous-optimalité de ces trajectoires est alors de plus en plus prononcée.

L'objectif de cette méthode est de proposer un ensemble suffisamment significatif de trajectoires dont la pertinence à la résolution de la tâche sera comparée. Pour construire cet ensemble de courbes, nous avons donc proposé deux arcs de clotoïde dont les courbures initiale et finale décrivent le spectre complet des valeurs possibles. La courbure initiale de la première clotoïde est fixée par l'orientation initiale des roues du robot. L'algorithme suivant permet de faire la construction de ces courbes :

Lg = longueur de chaque courbe

pour $\phi_1 \in [-\phi_{max}; \phi_{max}]$ **faire**

  pour $\phi_2 \in [-\phi_{max}; \phi_{max}]$ **faire**

$$\Delta\phi_1 = \frac{(\phi_1 - \phi_0)}{Lg * \Delta T}$$

$$\Delta\phi_2 = \frac{(\phi_2 - \phi_1)}{Lg * \Delta T}$$

$x = y = 0; theta = \pi/2; \phi = \phi_0;$

pour $T = 0; Lg - 1$ **faire**

$$\theta = \theta + \frac{V}{L}\tan(\phi) * \Delta T$$

$$x = x + V * \cos(\theta) * \Delta T$$
$$y = y + V * \sin(\theta) * \Delta T$$
$$\phi = \phi + \Delta\phi_1 * \Delta T$$

fin pour

pour $T = Lg; 2 * Lg$ **faire**

$$\theta = \theta + \frac{V}{L} \tan(\phi) * \Delta T$$
$$x = x + V * \cos(\theta) * \Delta T$$
$$y = y + V * \sin(\theta) * \Delta T$$
$$\phi = \phi + \Delta\phi_2 * \Delta T$$

fin pour

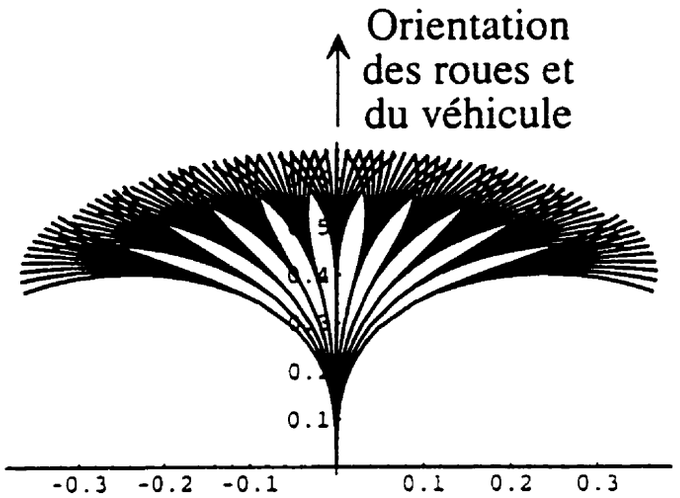
fin pour

fin pour

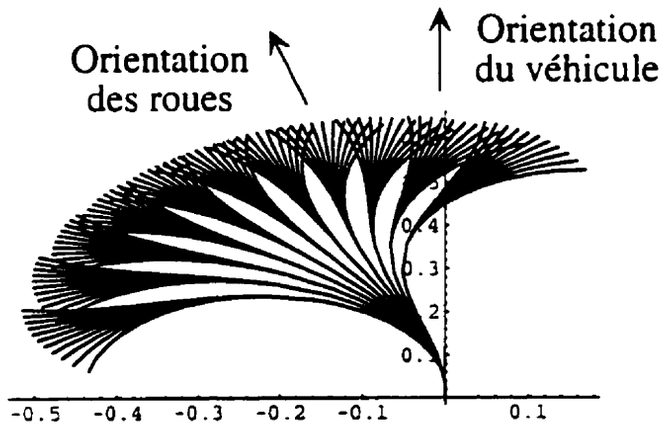
La figure 4.5 présente deux exemples de réseaux de courbes générés par cet algorithme. Sur cette figure, on constate que l'orientation finale du robot est très variable, bien que les positions finales obtenues soient similaires. Cela présente l'avantage certain de produire un ensemble de courbes plus intéressantes et plus représentatives de ce que peut faire le robot. On remarque toutefois que certaines portions de l'espace ne sont pas couvertes. Ce problème pourrait être résolu en supposant que la vraie trajectoire optimale se situe aux environs de la trajectoire appartenant au réseau qui a été choisie, et que de ce fait, il serait possible de générer un nouveau réseau de trajectoires autour de celle déjà obtenue.

4.3 Représentation pour une combinaison efficace

Nous avons choisi de représenter les Cartes Dynamiques pour la combinaison, par des "bitmaps", ou images en niveaux de gris. Le choix a été motivé à la fois par la rapidité potentielle de combinaison des bitmaps par des architectures dédiées et également par leur simplicité de construction.



(a)



(b)

Figure 4.5 : Exemple de réseau de courbes sous-optimales avec (a) $\phi_0 = 0$ et (b) $\phi_0 = \phi_{max}$ pour la génération de trajectoires sur la Carte Dynamique

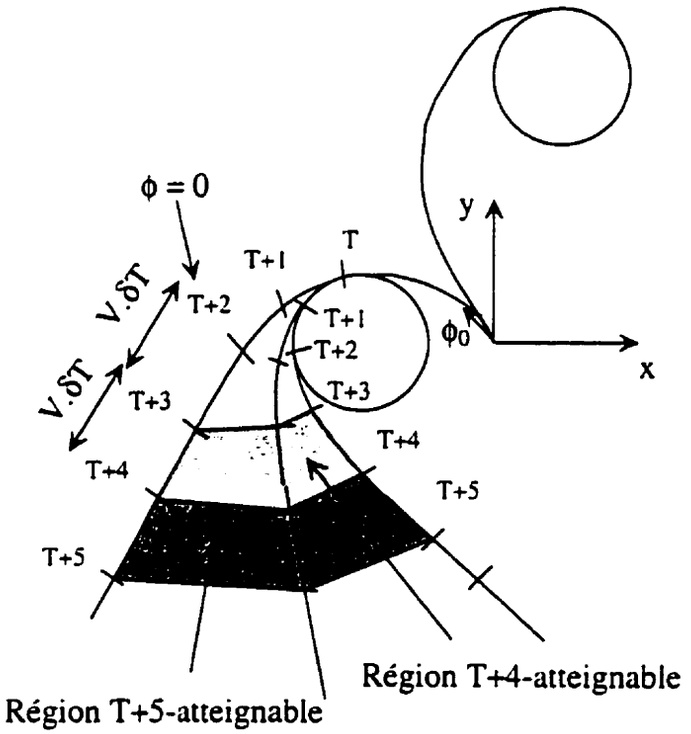


Figure 4.6 : Méthode de construction des régions T-atteignables

Ainsi, un bitmap de taille fixe est créé, puis chaque Carte Dynamique, qui correspond à une valeur initiale de ϕ (nous avons discretisé cette valeur), est construite. Chaque quadrilatère composant la carte (voir figure 4.6, déjà présentée dans le chapitre précédent) est construit à l'aide de simples routines graphiques de remplissage de polygone (la valeur de la fonctionnelle à l'intérieur du quadrilatère indique la valeur des points correspondant sur le bitmap). La figure 4.7 présente un exemple de bitmap créé de cette façon. Les conditions environnementales sont également représentées par un bitmap, qui est combiné avec la Carte Dynamique pour créer la nouvelle valeur de la fonction sur la Carte du robot pour être par la suite utilisée par les fonctions d'optimisation. Les obstacles sont créés également avec les routines de remplissage de polygone avec une valeur constante maximale (lors de la combinaison le bitmap est inversé).

Pour rajouter un autre type de condition, nous avons utilisé une droite avec une distribution normale orthogonalement. Cela permet de représenter par exemple une

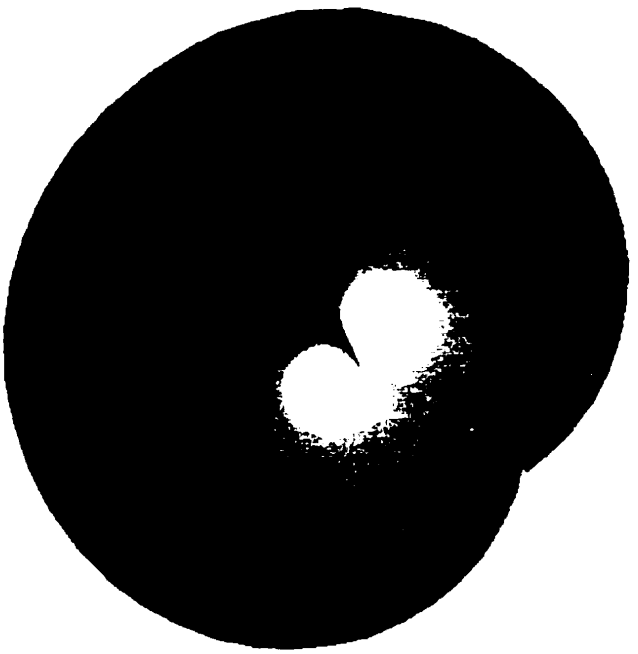


Figure 4.7 : Exemple de bitmap représentant une Carte Dynamique. les valeurs de la fonctionnelle sont données par les niveaux de gris avec la couleur noire indiquant la valeur maximale

direction privilégiée ou une route et son bas-côté. Un exemple de bitmap pour les conditions environnementales avec un obstacle est présenté dans la figure 4.8. Nous essayerons en général d'utiliser des fonctions de combinaisons séparables, comme par exemple une simple différence entre les valeurs des temps pour les régions T-atteignables. Cela permet ainsi des opérations simple sur les bitmaps pour combiner les Cartes Dynamiques. Les quatre opérations élémentaires suivantes sont nécessaires pour faire la combinaison : rotation, translation (offset), mise à l'échelle et addition/soustraction (voir figure 4.9). La combinaison des Cartes de deux robots permet d'obtenir une Carte de situation instantanée.

Si plusieurs robots sont envisagés, la méthode est similaire mais requiert toutefois deux étapes : une combinaison de chaque Carte des robots $R_i, i \neq 0$ pour obtenir les Cartes de situation instantanée $ISM_i, i \neq 0$, avec celle du robot R_0 : puis les $ISM_i, i \neq 0$ sont combinées en un Carte de situation instantanée combinée avec un

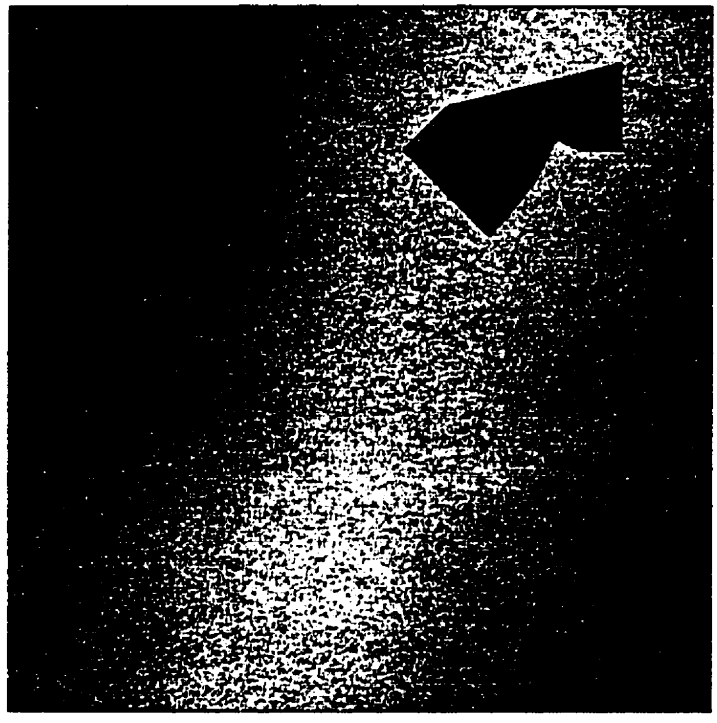


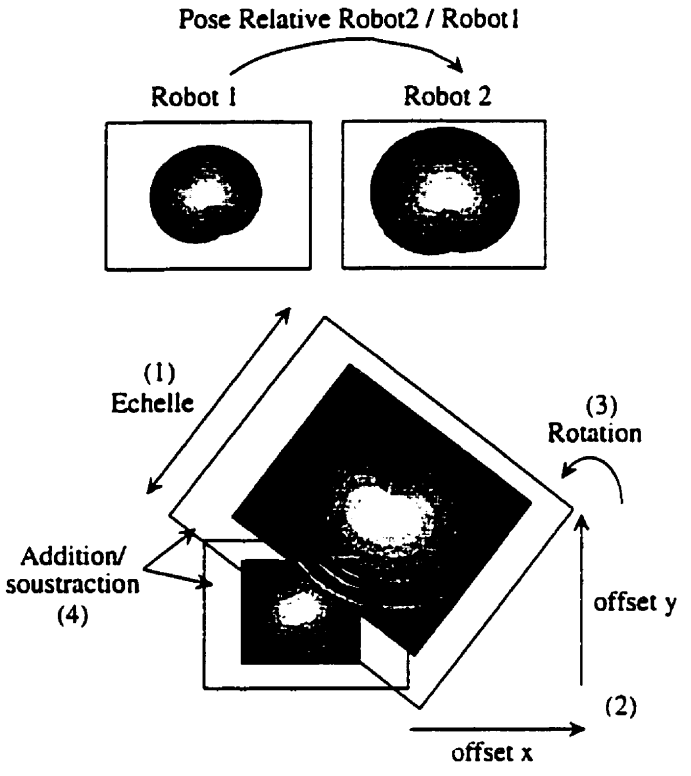
Figure 4.8 : Un bitmap avec un obstacle et une droite privilégiée

choix de la valeur à considérer en fonction de toutes les Cartes Dynamiques combinées obtenues. L'ensemble des calculs sont faisables en parallèle. La figure 4.10 présente la méthode de combinaison parallèle de plusieurs Cartes Dynamiques.

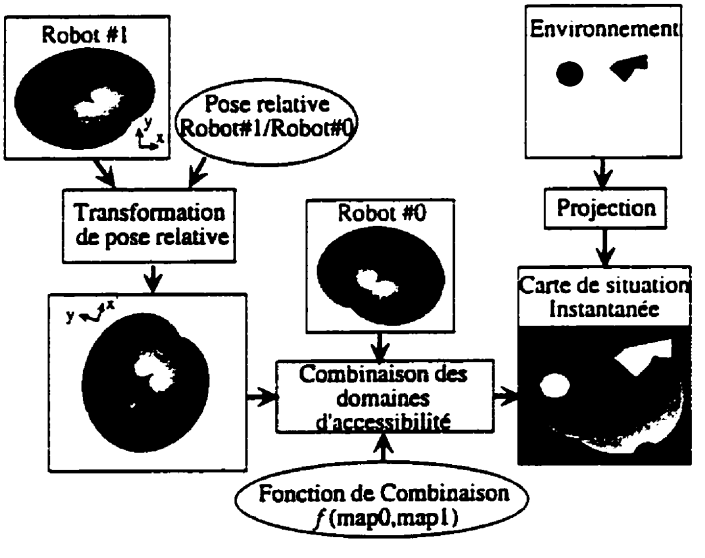
4.4 Exemple

Dans cette section, nous allons utiliser la Carte Dynamique pour résoudre le problème de l'évasion d'un robot mobile qui est poursuivi par un ou plusieurs autres robots. Ce problème est également analysé à partir des jeux différentiels au travers de travaux de Isaacs (Isaacs 1965) dans l'annexe C.

Dans un premier temps, le contexte du problème sera introduit, ensuite la méthode à partir des Cartes Dynamiques sera présentée.



(a)



(b)

Figure 4.9 : (a) Principe de la combinaison des cartes à partir des bitmaps (b) Combinaison des cartes entre deux robots pour obtenir la Carte de situation instantanée.

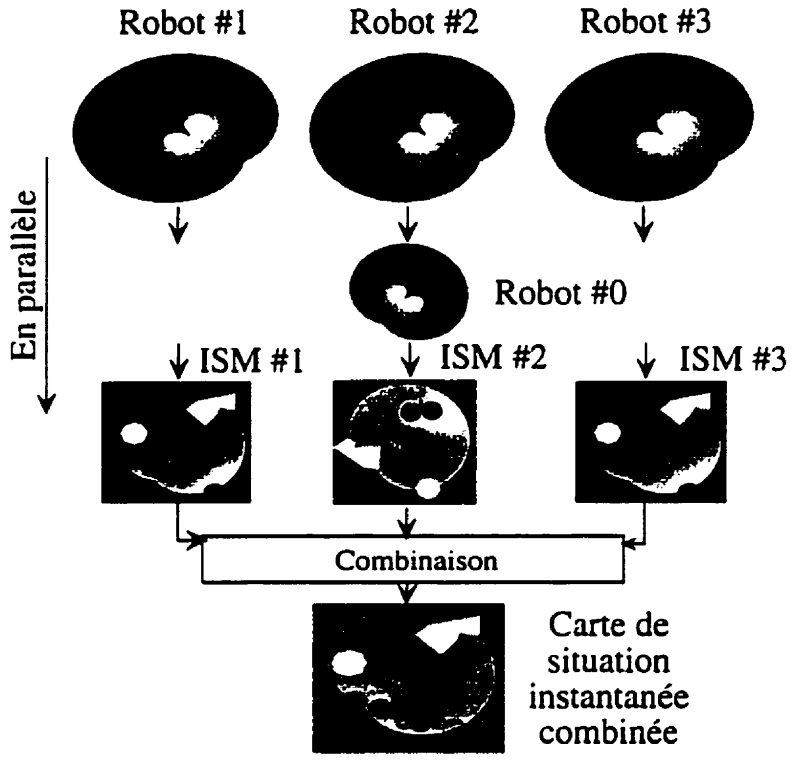


Figure 4.10 : Combinaison en parallèle de plusieurs Cartes Dynamiques

4.4.1 Le contexte

Nous considérons un ou plusieurs robots mobiles R (pour *Renard*) qui doivent essayer de capturer un ou plusieurs robots mobiles L (pour *Lapin*). Ces derniers tentent bien sûr d'échapper à leurs adversaires.

Plusieurs types de scénarios sont envisageables :

- Plusieurs robots R poursuivant un L :
  - Les R doivent prédire et analyser le comportement de leur proie et collaborer pour l'attraper ;
  - L doit éviter de se faire capturer en s'éloignant ou en se cachant.
- Un seul robot R à la poursuite de plusieurs robots L :
  - R doit optimiser ces décisions suivant qu'il désire attraper un seul L (le plus proche de lui) ou le plus grand nombre possible de L, l'un après l'autre ;

- Les L peuvent avoir une stratégie de collaboration pour troubler leur adversaire en s'enfuyant dans des directions opposées par exemple.

- Plusieurs R face à plusieurs L : les interactions sont complexes et multiples. Les situations seront des instances des deux précédents types de scénario.

Dans ce contexte, nous allons présenter, suivant les situations, le comportement que l'on attend des robots et la problématique soulevée.

4.4.1.1 Tactique d'un L face à un R

Nous envisagerons que les robots sont capables de se percevoir l'un l'autre. Intuitivement, deux tactiques existent pour L :

1. Sortir du champ de vision de R qui peut être limité en se cachant derrière un obstacle, ou en utilisant les défauts des capacités visuelles du R (par exemple on peut imaginer qu'un robot ne perçoit l'environnement que dans un domaine limité).

2. Si le robot L a un moyen d'estimer les capacités dynamiques de R (Sa courbure maximale, sa vitesse maximale, etc.), il peut utiliser cette connaissance pour trouver une trajectoire qui va lui permettre de se soustraire à son ennemi.

Devenir invisible pour son adversaire Si on n'envisage pas les techniques des camouflages que certains animaux utilisent, un L peut se soustraire visuellement à un autre robot en disparaissant derrière un obstacle ou en utilisant les défauts de vision du robot R.

- **Utilisation d'un obstacle pour se cacher.**

  Supposons toujours la situation d'un robot R pris en chasse par un robot F, mais avec un obstacle à proximité.

  L doit alors pour utiliser l'obstacle comme protection être capable de :

- Reconnaître l'obstacle ;
- Déterminer la meilleure trajectoire pour aller derrière l'obstacle et être invisible ;
- Choisir entre rester caché (au risque de se faire attraper facilement), continuer sa route (en espérant que R ne pourra pas prédire sa trajectoire), ou changer complètement de trajectoire.

- **Utilisation des défauts d'un système de vision.**

  Un R capable d'estimer l'orientation de l'axe optique de son adversaire pourra utiliser cette information pour planifier une trajectoire au delà des limites supposées du champ de vision d'un F.

Tactique dynamique d'évasion Une bonne tactique d'échappement pour L se baserait sur une connaissance complète de la dynamique de son adversaire (vitesse maximale et minimale, sa courbure maximale, *etc.*).

Ainsi, L serait plus en sécurité dans des régions que R peut difficilement atteindre à cause de ses capacités dynamiques. Cela peut par exemple se définir en termes de nombre de trajectoires que peut suivre R pour aller dans une certaine région.

La tactique de L sera alors de planifier des trajectoires qui passent par des régions que R n'est pas capable d'atteindre avant L ou au pire atteindre difficilement. Il s'agit de problème qui sont typiquement résolus à l'aide des Cartes Dynamiques.

4.4.1.2 Tactique d'un R face à un L

Le robot mobile R doit être capable de :

- Reconnaître le robot L :
- Prédire la trajectoire de sa proie :
- Contrôler ses mouvements à partir des données recueillies.

4.4.2 Tactique d'évasion à l'aide des Cartes Dynamiques

Parce que le problème d'évasion/poursuite est très complexe, nous allons nous concentrer seulement sur les stratégies d'évasion d'un robot poursuivi par un ou plusieurs autres robots à l'aide des Cartes Dynamiques. Notre problème est similaire à ceux évoqués dans la théorie des jeux différentiels (Isaacs 1965).

Les points de la trajectoire que devra suivre L afin de s'échapper peuvent être facilement trouvés à partir de l'heuristique suivante : choisir des points qui seront atteints d'abord par L et avec le maximum de délai avant l'arrivée de R à la même position. Comme nous l'avons vu auparavant, la Carte Dynamique est parfaitement adaptée pour résoudre ce genre de problème.

Ainsi, L ayant une estimation à la fois de la pose relative avec son adversaire (disponible par analyse du mouvement indépendant dans l'image 6) et de la Carte Dynamique de son adversaire, il suffit alors de combiner les deux cartes comme nous l'avons vu précédemment. La Carte indique alors les zones de sécurité pour L par les points où le temps T_L (temps du L pour aller au point considéré) est inférieur au temps T_R (temps du robot R pour atteindre le même point).

Pour utiliser les connaissances associées aux Cartes Dynamiques, les fonctionnelles peuvent être combinées de façon à refléter les meilleures conditions pour le robot L. Nous allons tester plusieurs combinaisons de fonctionnelles afin de planifier les trajectoires. Les fonctionnelles utilisées dans le cadre de cette tâche sont seulement le nombre de trajectoire et les conditions environnementales.

Bien que nous n'ayons envisagé pour l'instant qu'un adversaire, si le nombre de R venait à augmenter, la stratégie de L resterait la même, il n'aurait qu'à superposer la carte de tous ces adversaires sur une même carte en accord avec les poses relatives envers L. Il faut cependant faire attention à ce que les effets positifs d'un robot R n'occultent pas les effets négatifs d'un autre. Pour éviter ce genre de problème, le minimum des valeurs combinées des fonctionnelles entre le robot L et chaque robot R est utilisée pour construire la Carte Dynamique combinée.

Différents fonctions de combinaison Nous adoptons les notations suivantes : $T_{Robot}(x, y)$ désigne le temps associé à la région T-atteignable du robot dans laquelle se trouve le point (x, y). $d_{Robot}(x, y)$ indique de la même façon la distribution au point (x, y) de la fonctionnelle. $R_i, i \in \{1, \ldots, n\}$ représentent les différents robots renards.

La première fonction de combinaison simple correspond à la différence des temps entre les régions T-atteignables :

$$G(x, y) = \min_{i \in \{1,\ldots,n\}} (T_{R_i} - T_L) \tag{4.3}$$

La combinaison suivante utilise la valeur du nombre de trajectoires dans la Carte Dynamique :

$$G(x, y) = \min_{i \in \{1,\ldots,n\}} d_L - d_{R_i} \times (T_L - T_{R_i}) \tag{4.4}$$

La combinaison exprime l'interaction en diminuant (ou augmentant) la valeur du nombre de trajectoires du robot R en fonction de la différence de temps pour atteindre la zone considérée et la valeur du nombre de trajectoires relatif à l'adversaire. La valeur de la fonction reflète deux choses :

- lorsque $T_{R_i} > T_L$, il est préférable que le nombre de trajectoires d_{R_i} soit grand (laissant ainsi beaucoup de mauvaises trajectoires pour l'adversaire) et cela augmente donc la valeur de la fonction de L (car $T_L - T_{R_i}$ est négatif) :
- lorsque $T_L > T_{R_i}$, il est préférable que peu de trajectoires permettent à R_i d'atteindre cette zone potentiellement dangereuse pour L. De ce fait la valeur de la fonction de L est diminué (car $T_L - T_{R_i}$ est positif).

La non-séparabilité de la fonction de combinaison pose cependant un problème lors de l'utilisation de la représentation par bitmap présentée à la section précédente.

Afin de rendre la troisième fonction de combinaison séparable, la valeur utilisée sur le bitmap est T_{Robot}/d_{Robot} qui fournit une information sur le nombre de trajectoires

relativement au temps. En faisant ensuite une différence simple on obtient :

$$G(x,y) = \min_{i \in \{1,\ldots,n\}} \frac{T_{R_i}(x,y)}{d_{R_i}(x,y)} - \frac{T_L(x,y)}{d_L(x,y)} \tag{4.5}$$

avec $T_{R_i}(x,y)/d_{R_i}(x,y)$ une récompense pour le robot L et $T_L(x,y)/d_L(x,y)$ une punition. En effet, le robot L ne doit pas aller dans des régions où le robot adversaire a un nombre de trajectoires importante (de ce fait le terme inversement proportionnel) et avec un temps faible (de ce fait le terme proportionnel au temps).

4.4.3 Expériences

Dans cette section, nous allons présenter les résultats sur la planification de trajectoires d'évasion dans le cas d'un face-à-face et dans le cas où un robot est poursuivi par trois autres. L'ensemble des résultats obtenus dans cette section ont été réalisés en simulation. Les robots poursuivants ont une tactique simple mais efficace de poursuite en utilisant la position réelle du robot L. Pour l'instant, l'évitement de collision pour les robots poursuivants n'a pas été implanté. Il faut donc savoir que les résultats sont susceptibles d'être meilleurs lorsque les robots poursuivant seront eux aussi gênés par les obstacles. La figure 4.11 illustre la planification de trajectoires avec les différents robots en présence.

4.4.3.1 Face-à-face

Considérons d'abord le cas d'un face-à-face entre deux robots. Pour chaque méthode de calcul de la trajectoire, nous avons utilisé les trois types de combinaisons de fonctions présentées dans la section précédente. Nous utiliserons les termes DT, DD1 et DD2 pour désigner respectivement la fonction de combinaison de différence des temps de l'équation 4.3 et les deux fonctions de combinaison avec le nombre de trajectoires des équations 4.4 et 4.5.

Nous allons présenter maintenant, une étude comparative de la planification de

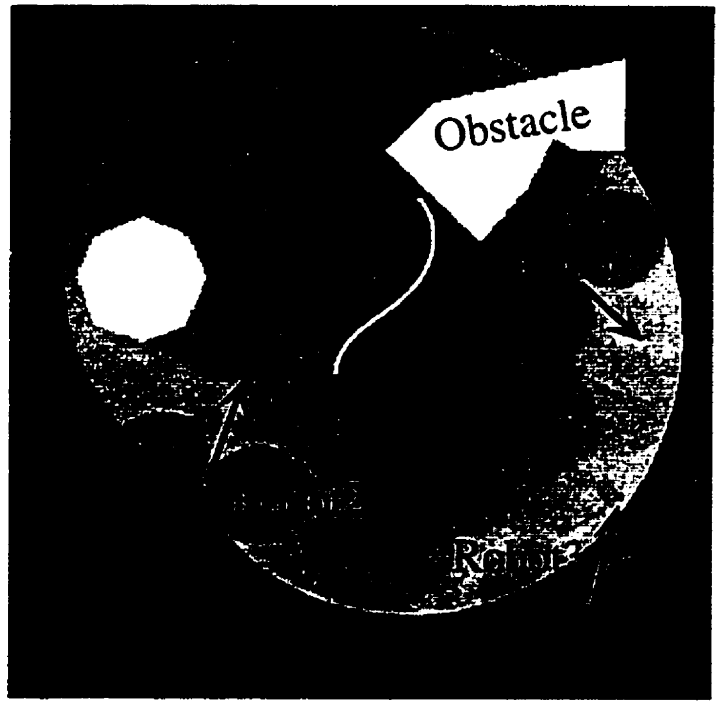


Figure 4.11 : Illustration de la planification d'une trajectoire avec plusieurs robots

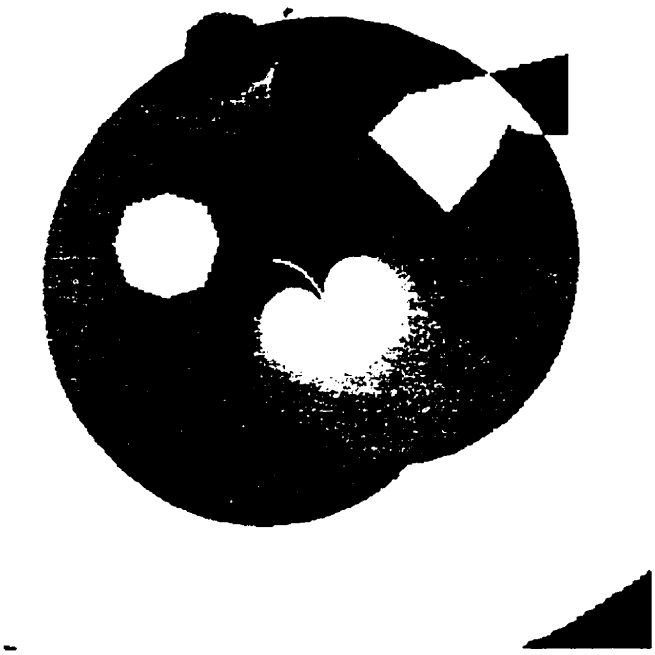
trajectoires à l'aide de la technique par descente de gradient et par réseau de courbes (méthodes présentées dans la section précédente). L'étude s'est faite dans deux situations : (1) les deux robots sont identiques; (2) le robot poursuivant est plus rapide (environ 50% plus rapide) mais à un rayon de braquage plus faible. Les tests ont été effectués sur un même ensemble de valeurs aléatoires des positions initiales du robot poursuivant et de l'angle des roues de L. Dans toutes les figures, le robot L est orienté suivant l'axe vertical. De plus, pour les figures en niveaux de gris la couleur blanche correspond au minimum de la fonction et la couleur noire indique le maximum de la fonction.

Descente de gradient. Les figures 4.12, 4.13, 4.14 et 4.15 présentent des exemples de trajectoires planifiées par la méthode de descente de gradient pour les deux types de robots et les trois fonctions de combinaison[1].

On peut constater dans les figure 4.12 et 4.13 que les trajectoires planifiées pour

[1]Les figures correspondant à des mêmes cas ont été séparées pour être plus visible

(a) DT



(b) DD1

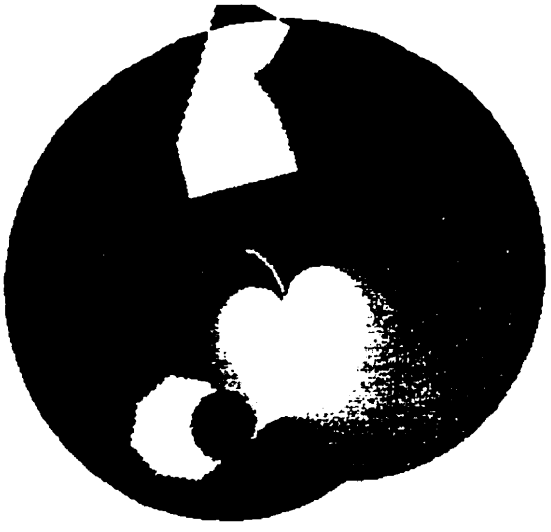
Figure 4.12 : Trajectoires obtenues avec la méthode de descente de gradient avec des robots adversaires plus rapides, pour la même position initiale et les deux fonctions de combinaison DT (a) et DD1 (b)

Figure 4.13 : Trajectoire obtenue avec la méthode de descente de gradient avec des robots adversaires plus rapides, pour la même position initiale et la fonction de combinaison DD2

les trois fonctions de combinaisons sont pratiquement identiques. La forme de la fonction combinée dans la figure 4.12(a) exprime bien l'effet de la fonction de l'autre robot sur le robot L. On peut voir en effet que les zones maximales se sont déplacées vers l'arrière du robot. Dans les figures 4.14 et 4.15 les mêmes effets sont observables.

Les tableaux 4.1 et 4.2 présentent les résultats numériques reliés à la méthode de descente de gradient pour les deux cas de robots. Ces résultats correspondent à des simulations effectuées de la façon suivante : le robot planifie en premier sa trajectoire par l'algorithme de descente de gradient puis, à la fin de sa trajectoire, une nouvelle combinaison des Cartes est effectuée dans la nouvelle situation (les autres robots se sont bien entendu déplacés) pour planifier à nouveau une trajectoire et ainsi de suite. L'expérience est faite dans un temps fini identique pour toutes les expériences. Une capture est déterminée lorsque la distance entre le robot poursuivant et L est inférieure à un seuil prédéfini. Dans les tableaux, "Minimum" indique la valeur moyenne de la distance minimale entre les deux robots dans chaque expérience.

(a) DT



(b) DD1

Figure 4.14 : Trajectoires obtenues avec la méthode de descente de gradient pour des robots identiques, pour la même position initiale et les deux fonctions de combinaison DT (a) et DD1 (b)

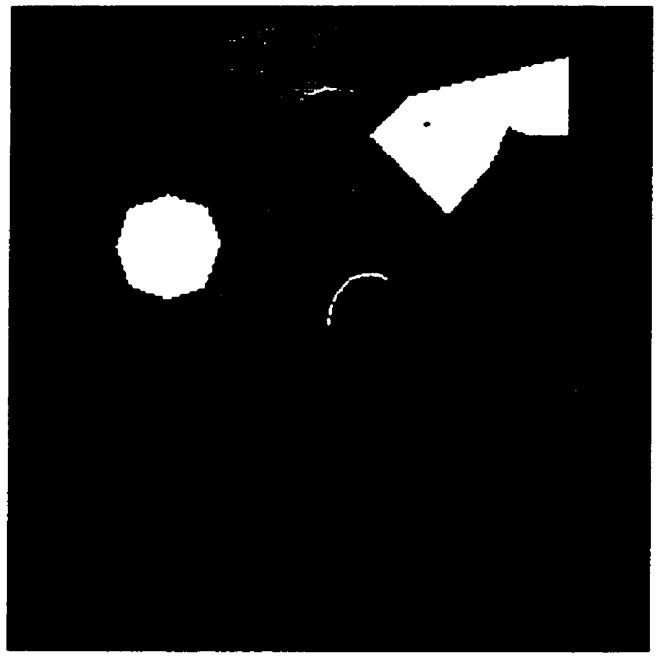


Figure 4.15 : Trajectoire obtenue avec la méthode de descente de gradient pour des robots identiques, pour la même position initiale et la fonction de combinaison DD2

"Maximum" indique la valeur moyenne de la distance maximale entre les deux robots dans chaque expérience, "Temps moyen avant capture" désigne en moyenne le temps nécessaire au robot pour être capturé et "% capture" correspond au pourcentage des expériences qui s'achèvent par la capture du robot.

D'après les valeurs données dans les tableaux, la deuxième fonction DD1 semble permettre au robot de s'échapper le plus souvent (avec également une valeur moyenne de minimale de distance plus importante). Toutefois, le temps moyen avant capture semble indiquer que dans le cas de la première fonction même s'il y a moins d'évasion, la capture est souvent plus retardée dans le temps. La fonction DD2 est assez décevante dans ce cas.

Dans le tableau 4.2, il apparaît que les résultats sont similaires au cas précédent. Cependant, la fonction DD1 a ici des résultats nettement supérieures aux deux autres fonctions. Les résultats sont évidemment meilleurs pour toutes les fonctions puisque les deux robots sont identiques et ont donc la même vitesse.

Tableau 4.1 : Résultats de l'algorithme de descente de gradient avec le robot poursuivant plus rapide

Fonction	Minimum	Maximum	Temps moyen avant capture	taux de capture
DT	0.095	2.18	70.4	83%
DD1	0.17	2.16	64.6	70%
DD2	0.14	2.17	64.1	80%

Tableau 4.2 : Résultats de l'algorithme de descente de gradient avec les deux robots identiques

Fonction	Minimum	Maximum	Temps moyen avant capture	taux de capture
DT	0.3	2.16	76.3	53%
DD1	0.5	2.12	69	36%
DD2	0.33	2.15	72.2	50%

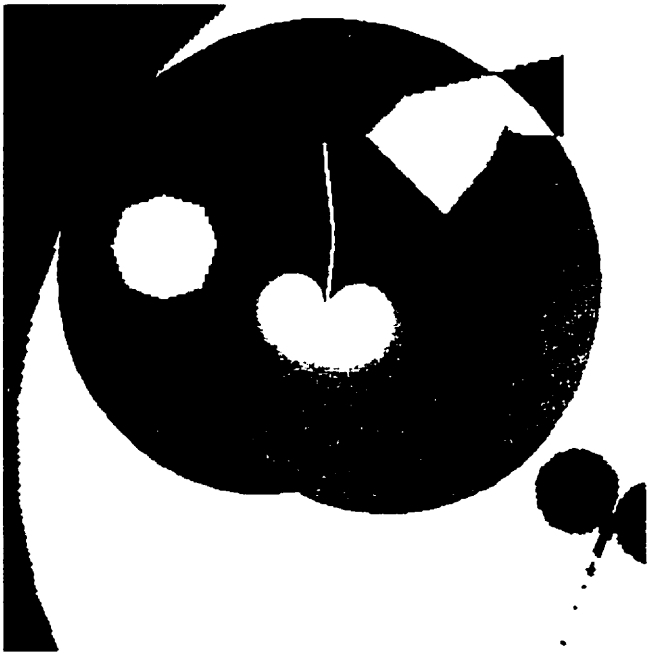
Réseau de courbes. Les trajectoires planifiées dans le cas de la méthode de réseaux de courbes sont présentées dans les figures 4.16, 4.17, 4.18 et 4.19.

Contrairement à la première méthode par descente de gradient, certaines des trajectoires dans la même situation initiale sont qualitativement différentes suivant la fonction de combinaison utilisée (voir figures 4.16 et 4.17).

Les trajectoires obtenues dans les différentes situations initiales des figures 4.18 et 4.19 exprime bien la diversité des trajectoires que le réseau de courbes permet de créer.

Les tableaux 4.3 et 4.4 présentent les résultats numériques d'expériences effectuées à nouveau en simulation. La planification est un peu différente par rapport à la descente de gradient. En effet, la trajectoire est planifiée pour un temps deux fois supérieur au temps des trajectoires planifiées par la descente de gradient. Cependant le robot ne se déplace que dans la moitié de la trajectoire planifiée (cela permet aussi de faciliter la comparaison des deux algorithmes). Dans le cas des réseaux, les trajectoires planifiées sont plus longues pour prendre en compte plus d'effet à long

(a) DT



(b) DD1

Figure 4.16 : Trajectoires obtenues avec la méthode des réseaux pour la même position initiale et les deux fonctions de combinaison DT (a) et DD1 (b)

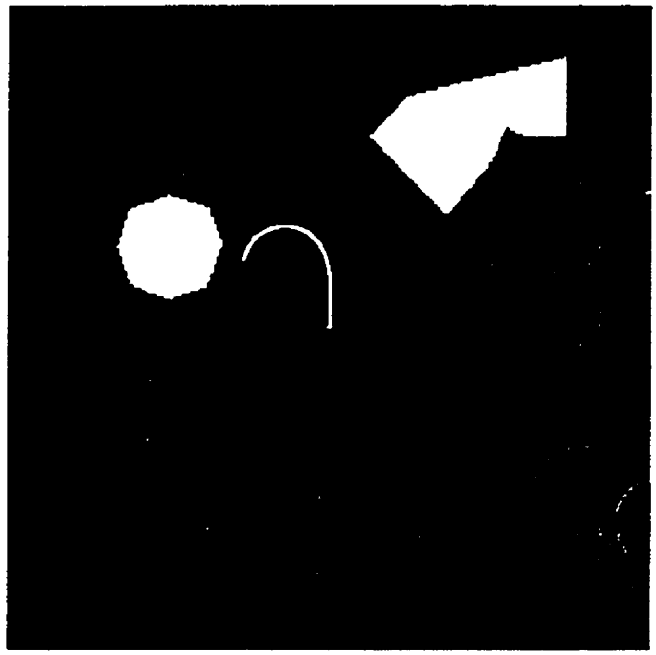
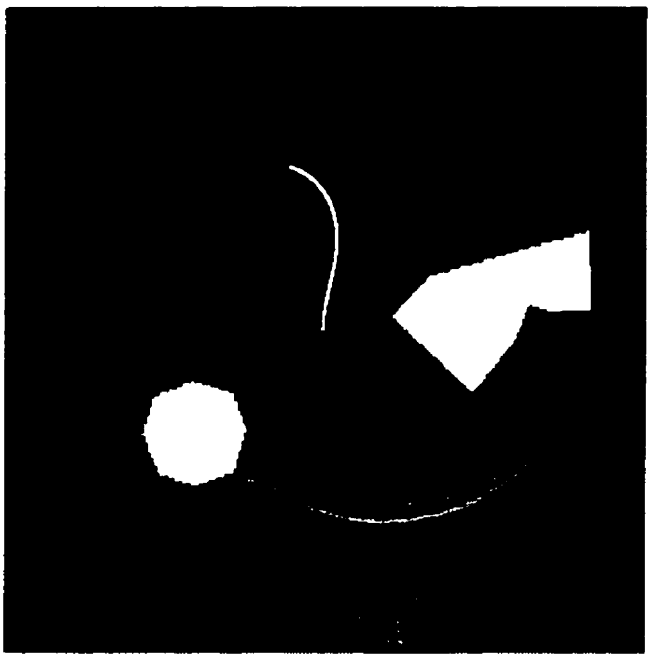


Figure 4.17 : Trajectoire obtenue avec la méthode des réseaux pour la même position initiale et la fonction de combinaison DD2

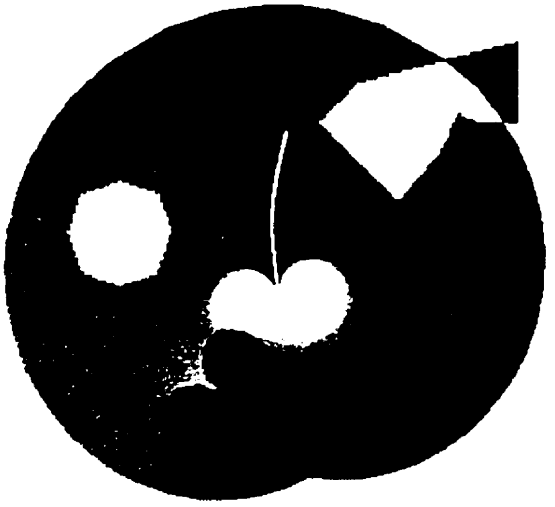
terme (tel que les obstacles). Les valeurs entre parenthèse correspondent aux valeurs obtenues par l'algorithme de descente de gradient.

Dans le tableau 4.3, on peut voir facilement que, dans ce cas, c'est la fonction DD2 qui semble nettement supérieure aux autres. Le même effet sur les temps moyens avant capture que dans le cas de la descente de gradient est observé ici. On remarque toutefois la grande différence d'efficacité entre les deux méthodes de planification. Ainsi. pour la fonction DD2, il y a une diminution de presque 50% du taux de capture. Cet effet était prévisible puisque la descente est essentiellement une méthode locale, tandis que les réseaux de courbes permettent de faire un compromis entre planification locale et globale.

Les mêmes effets observés dans le tableau 4.3 sont également obtenus dans le cas des deux robots identiques (tableau 4.4). Il faut remarquer ici que, dans le cas de l'algorithme des réseaux de courbes, les résultats correspondant à la fonction DD1 ont subi une augmentation de l'efficacité (taux de capture diminuant de 36% à 26%) plus faible en comparaison de celle des deux autres fonctions (de 53% à 20% et 50% à



(a) DT



(b) DD1

Figure 4.18 : Trajectoires obtenues avec la méthode des réseaux pour la même position initiale et les deux fonctions de combinaison DT (a) et DD1 (b)

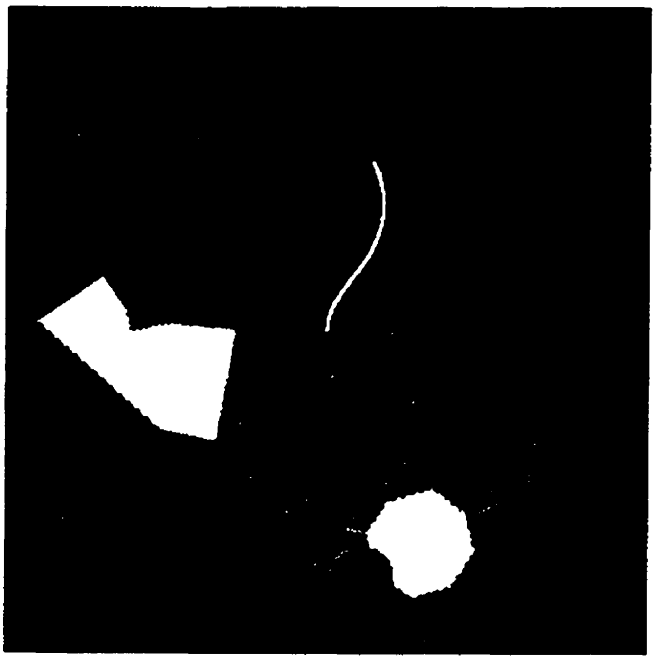


Figure 4.19 : Trajectoire obtenue avec la méthode des réseaux pour la même position initiale et la fonction de combinaison DD2

Tableau 4.3 : Résultats de l'algorithme par réseau de courbes avec le robot poursuivant plus rapide

Fonction	Minimum	Maximum	Temps moyen avant capture	taux de capture
DT	0.32 (0.09)	2.14 (2.18)	72.8 (70.4)	63% (83%)
DD1	0.26 (0.17)	2.17 (2.16)	58 (64.6)	66% (70%
DD2	0.45 (0.14)	2.17 (2.17)	61.6 (64.1)	36% (80%)

Tableau 4.4 : Résultats de l'algorithme par réseau de courbes avec les deux robots identiques

Fonction	Minimum	Maximum	Temps moyen avant capture	taux de capture
DT	1.19 (0.3)	2.14 (2.16)	85.6 (76.3)	20% (53%)
DD1	1.04 (0.5)	2.15 (2.12)	60 (69)	26% (36%)
DD2	1.46 (0.33)	2.2 (2.15)	73.5 (72.2)	13% (50%)

13%). Ceci aurait tendance à suggérer que la fonction DD1 exprime mieux de façon locale les interactions entre les robots, alors que les fonctions DT et DD2 sont plus représentatives globalement.

Les résultats obtenus dans cette section sont encourageants. La tactique d'évasion est assez efficace dans le cas d'un adversaire plus rapide. La méthode par réseau de courbes est ici nettement supérieure à la technique par descente de gradient.

4.4.3.2 Trois contre un

Dans cette section, nous allons étendre l'étude de planification de trajectoires d'évasion pour un robot ayant trois adversaires.

Dans un premier temps, nous avons utilisé l'algorithme heuristique pour obtenir les trajectoires d'évasion du robot. Un exemple de trajectoire obtenue par cet algorithme est présenté à la figure 4.20.

Nous avons toutefois constaté que cet algorithme ne parvenait pas à trouver une solution dans un temps acceptable (dans certains cas, aucune solution satisfaisante n'était même obtenue). Il est ainsi impossible de prévoir à l'avance le temps nécessaire à la recherche d'une trajectoire optimale.

De la même façon que dans le cas du face-à-face, nous avons procédé à une étude comparative des deux techniques de planification de trajectoires par descente de gradient et par réseau de courbes. Les tests ont été effectués sur un même ensemble de valeurs aléatoires des positions initiales et des valeurs de l'angle des roues pour les

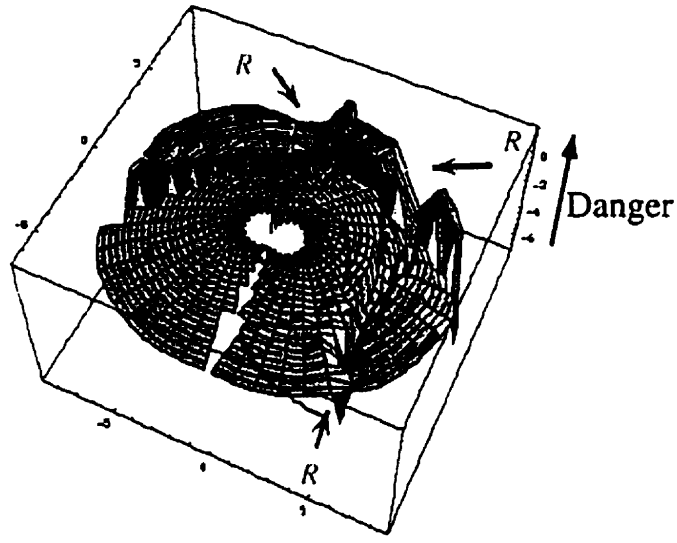


Figure 4.20 : Trajectoire obtenue par l'algorithme heuristique de planification avec trois robots à éviter

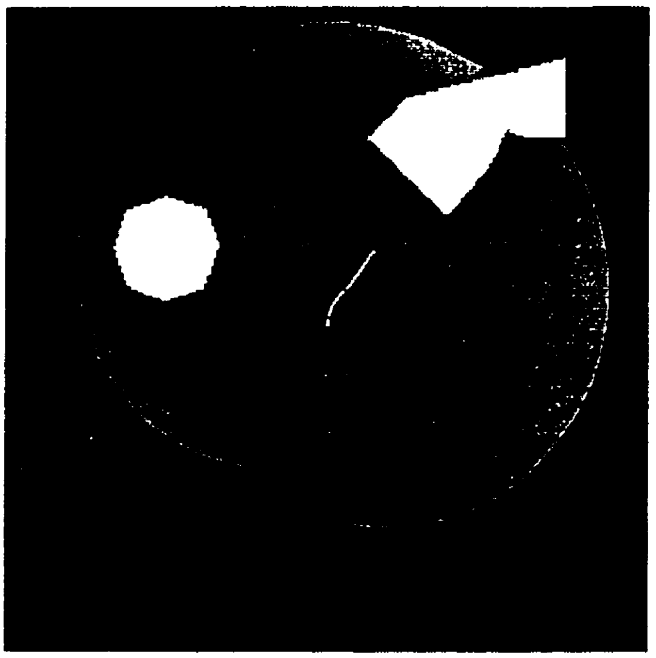
quatre robots présents dans l'environnement.

Descente de gradient. Les figures 4.21 et 4.22 présentent des exemples de trajectoires à court terme planifié par l'algorithme de descente de gradient avec les trois différentes combinaisons de fonctions lorsque les robots poursuivant sont plus rapides.

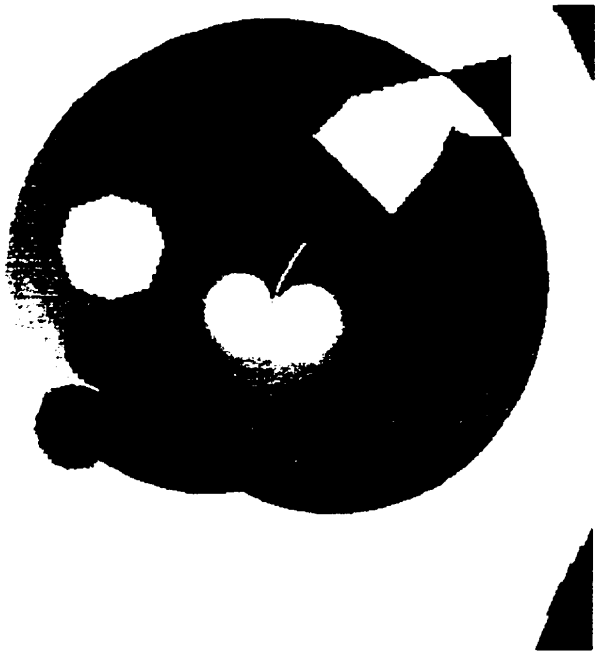
Des exemples de trajectoires obtenues par le même algorithme dans le cas où tous les robots sont identiques sont présentés dans les figures 4.23 et 4.24.

On peut observer dans la figure 4.21(a) que l'algorithme de descente de gradient va se diriger dans la zone où le robot a du mal à tourner (à l'intérieur du cercle C_R). On doit remarquer également la complexité de la fonction combinée pour les quatre robots.

Comme nous l'avons mentionné auparavant, l'avantage de l'algorithme de descente de gradient est sa rapidité, cependant il arrive parfois que le robot soit bloqué par un obstacle lors d'une planification. La figure 4.25 propose un exemple de trajectoire planifiée pour laquelle le robot se trouvera bloqué par l'obstacle à la fin de sa trajectoire sans espoir de pouvoir tourner assez vite pour l'éviter.



(a) DT



(b) DD1

Figure 4.21 : Trajectoires obtenues par l'algorithme de descente de gradient lorsque les robots adversaires sont plus rapides et avec les deux fonctions de combinaison DT (a) et DD1 (b)

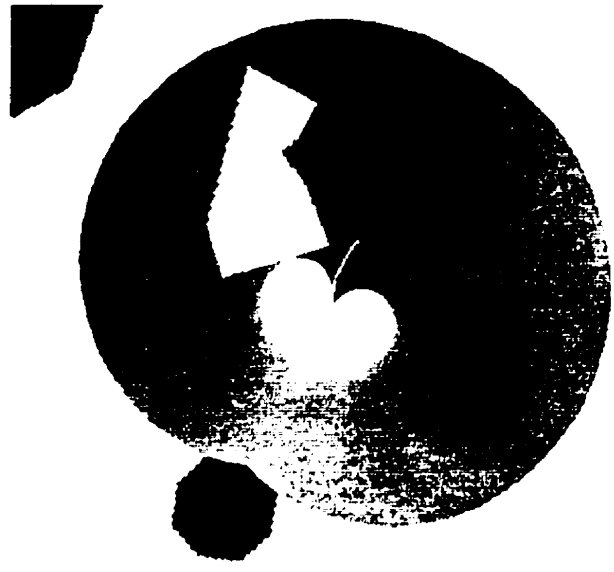
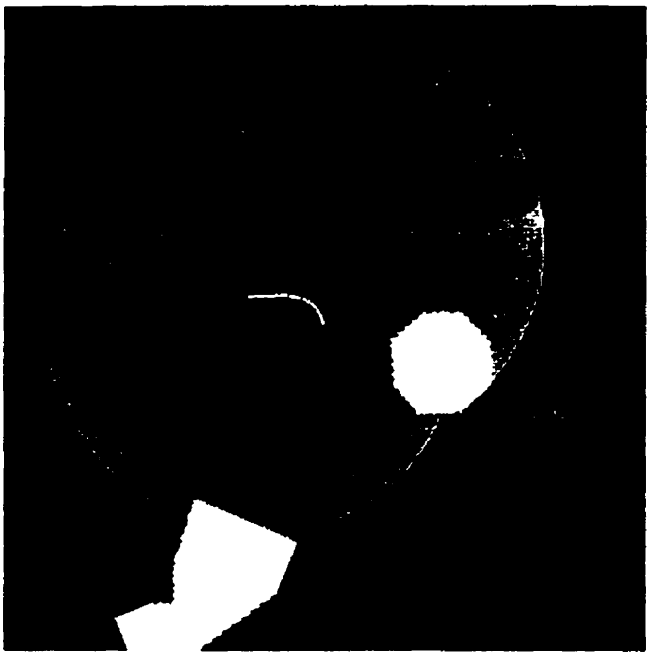


Figure 4.22 : Trajectoire obtenue par l'algorithme de descente de gradient lorsque les robots adversaires sont plus rapides et avec la fonction de combinaison DD2

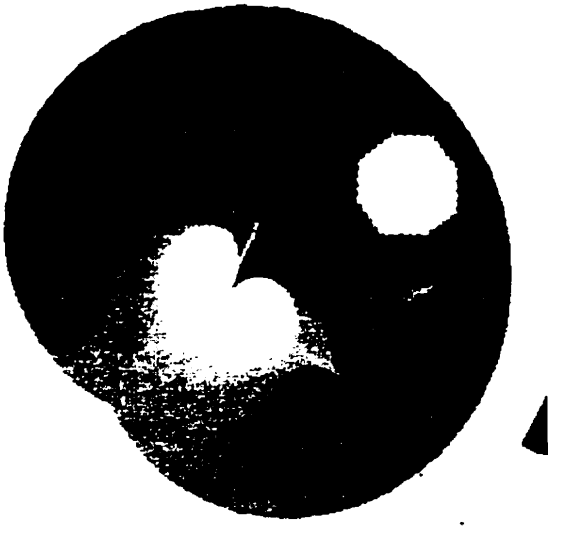
Les tableaux 4.5 et 4.6 donnent les différents résultats numériques associés aux calculs des trajectoires par descente de gradient pour les deux configurations de robots considérées. "Somme Distances" désigne la valeur moyenne de la somme des distances du robot poursuivi par rapport aux autres (il s'agit d'un indice sur l'espace libre qui existe en moyenne entre le robot et ses poursuivants. Une forte valeur indique que le robot a réussi à maintenir un espace libre entre lui et ses adversaires le plus large possible). Les autres éléments sont identiques à ceux données dans la section précédentes.

Le résultat le plus frappant du tableau 4.5 concerne le taux de réussite de la fonction DD1. En effet, les autres fonctions ont un taux de capture de 100%. Ceci aurait donc tendance à renforcer que la fonction DD2 exprime bien les interactions localement par rapport aux robots ainsi bien adapté à la descente de gradient.

Cette hypothèse est cependant contrariée par les résultats concernant la fonction DD1 dans le tableau 4.6 qui sont assez mauvais. Cependant, il faut noter ici la valeur moyenne du temps avant capture qui semblerait indiquer que dans le cas où le robot



(a) DT



(b) DD1

Figure 4.23 : Trajectoires obtenues par l'algorithme de descente de gradient lorsque les robots adversaires sont plus rapides et avec les deux fonctions de combinaison DT (a) et DD1 (b)

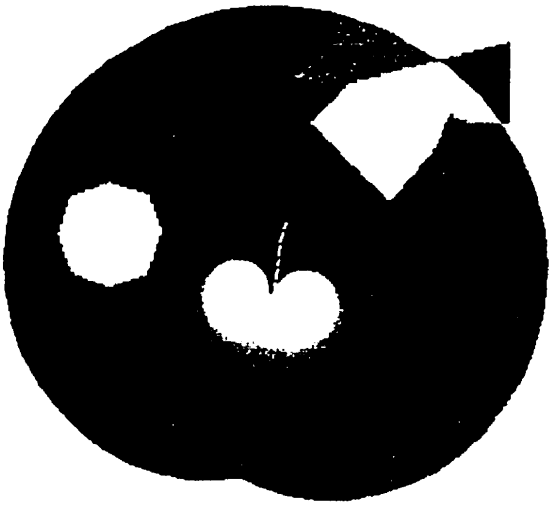


Figure 4.24 : Trajectoire obtenue par l'algorithme de descente de gradient lorsque les robots adversaires sont plus rapides et avec la fonction de combinaison DD2

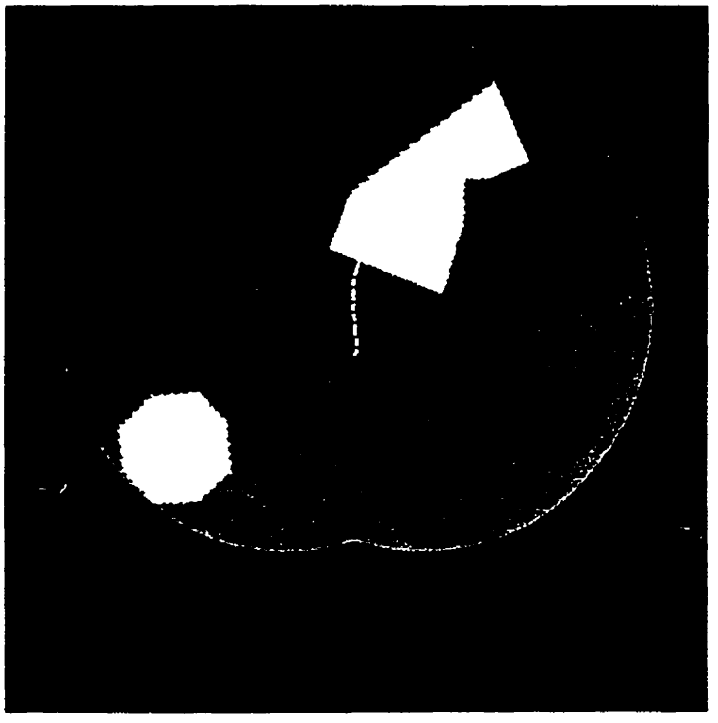


Figure 4.25 : Problème lors de la planification par descente de gradient : le robot se trouve bloqué par l'obstacle à la fin de sa trajectoire

Tableau 4.5 : Résultats de l'algorithme de descente de gradient avec les robots poursuivant plus rapides

Fonction	Somme Distances	Minimum	Maximum	Temps moyen avant capture	taux de capture
DT	1.15	0.05	2.84	53.13	100%
DD1	1.32	0.09	2.85	51.30	86%
DD2	1.2	0.05	2.82	53.3	100%

Tableau 4.6 : Résultats de l'algorithme de descente de gradient avec des robots identiques

Fonction	Somme Distances	Minimum	Maximum	Temps moyen avant capture	taux de capture
DT	1.7	0.2	2.8	69	70%
DD1	1.25	0.13	2.8	74	80%
DD2	1.47	0.13	2.8	62	73%

est capturé avec l'aide de la fonction DD1 c'est avec un temps plus important.

Les résultats obtenus dans cette section sont à nouveau encourageants mais semblent poser un dilemme quant à la fonction qu'il faut choisir pour accommoder à la fois le cas où les robots poursuivants sont plus rapides et celui où ils sont identiques au robot poursuivi.

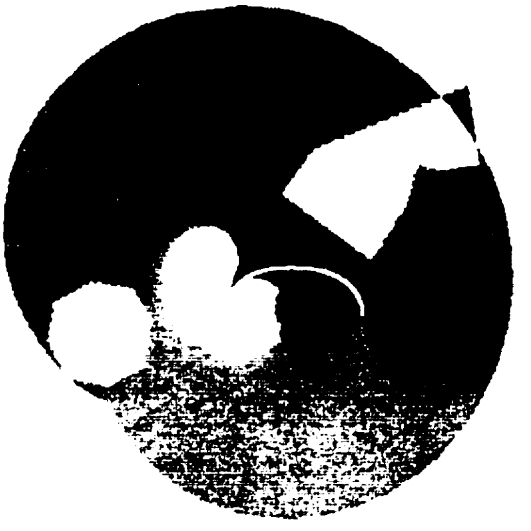
Réseau de courbes. Les résultats produits par la méthode des réseaux de courbes pour les deux types de robots et pour les différentes fonctions de combinaison sont proposés dans les figures 4.26, 4.27, 4.28 et 4.29.

Les deux figures 4.26 et 4.27 présentent la diversité des trajectoires qu'il est possible d'obtenir par l'intermédiaire des réseaux de courbes. On peut voir par exemple dans la figure 4.26(a) que le robot profite de la zone morte du cercle C_L (définie dans le chapitre 3) pour échapper à son adversaire. Dans la figure 4.27(b) on peut voir le "couloir" formé par les différentes combinaisons des fonctions des robots dans lequel le robot se dirige directement.

(a) DT

(b) DD1

Figure 4.26 : Trajectoires obtenues par l'algorithme réseau de courbe lorsque les robots adversaires sont plus rapides et avec les deux fonctions de combinaison DT (a) et DD1 (b)

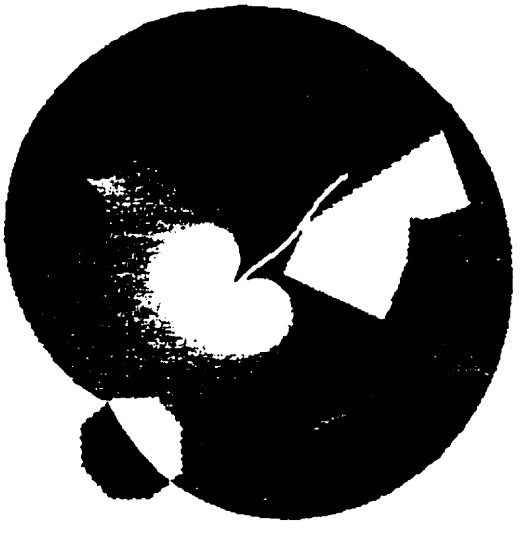


(a) DD1

(b) DD2

Figure 4.27 : Trajectoires obtenues par l'algorithme réseau de courbe lorsque les robots adversaires sont plus rapides et avec les deux fonctions de combinaison DD1 (a) et DD2 (b)

(a) DT



(b) DD1

Figure 4.28 : Trajectoires obtenues par l'algorithme réseau de courbe lorsque les robots adversaires sont plus rapides et avec les deux fonctions de combinaison DT (a) et DD1 (b)

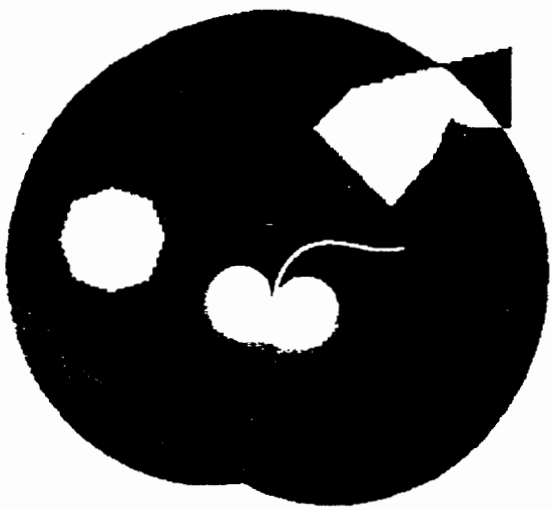
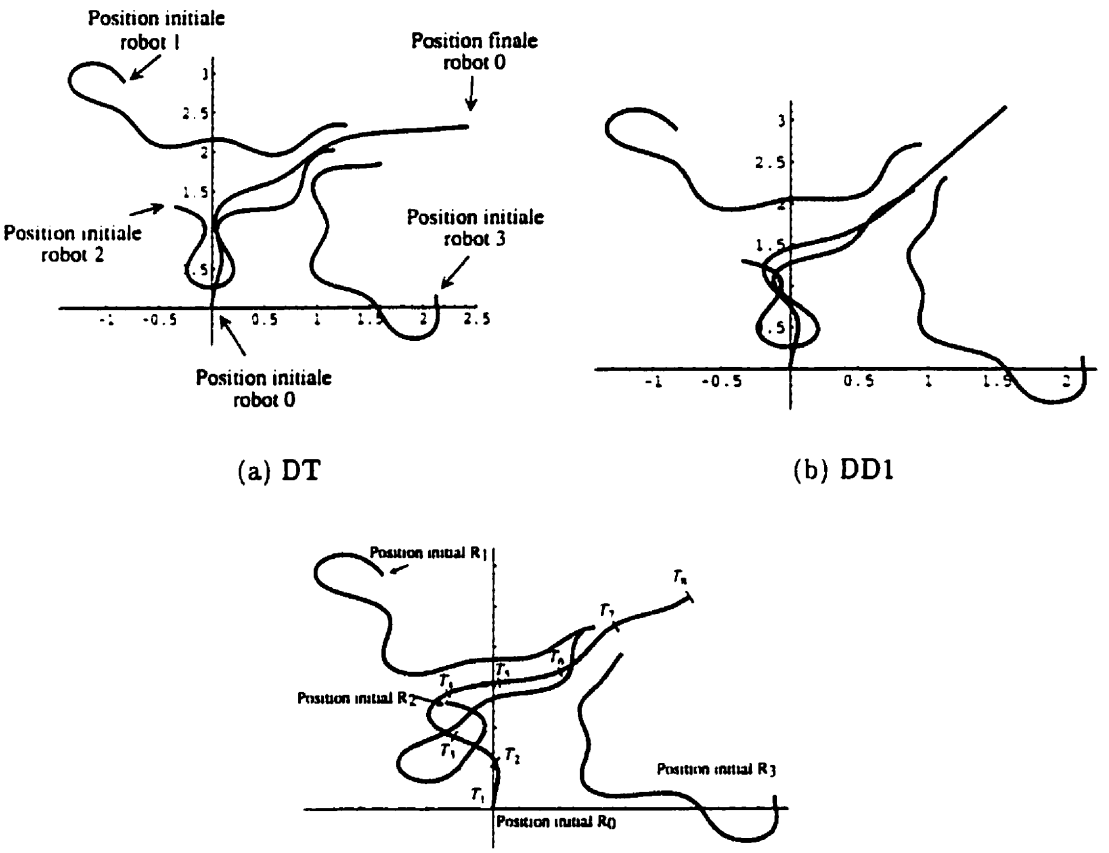


Figure 4.29 : Trajectoire obtenue par l'algorithme réseau de courbe lorsque les robots adversaires sont plus rapides et avec la fonction de combinaison DD2

Dans les figures 4.28 et 4.29, on peut remarquer de nouveau la formation de ces couloirs dans les différentes Cartes. Il sera intéressant dans de futurs travaux d'augmenter le nombre de robots pour observer la modification de la forme des fonctions sur les Cartes combinées. On peut également remarquer dans la figure 4.28 que le robot évite à la fois d'être capturé et de rentrer en contact avec l'obstacle.

Dans la figure 4.30, des trajectoires complètes des quatre robots identiques sont présentées dans le même repère pour des mêmes conditions initiales. La figure 4.31 présente la succession des Cartes Dynamiques obtenues pour les trajectoires de la figure 4.30(b). Les figures 4.32 et 4.33 présentent en simulation 3D les positions relatives des robots qui correspondent à ces mêmes Cartes Dynamiques. Les trajectoires du robot sont obtenues par la méthode des réseaux de courbes et les trois différentes fonctions de combinaison. On peut voir clairement que les trajectoires sont qualitativement identiques. Toutefois, il apparaît dans le cas de la fonction DT par exemple que la trajectoire du robot est moins chaotique, alors que dans le cas des deux autres fonctions, le robot semble pousser ses poursuivants aux limites de leurs capacités (voir



(a) DT

(b) DD1

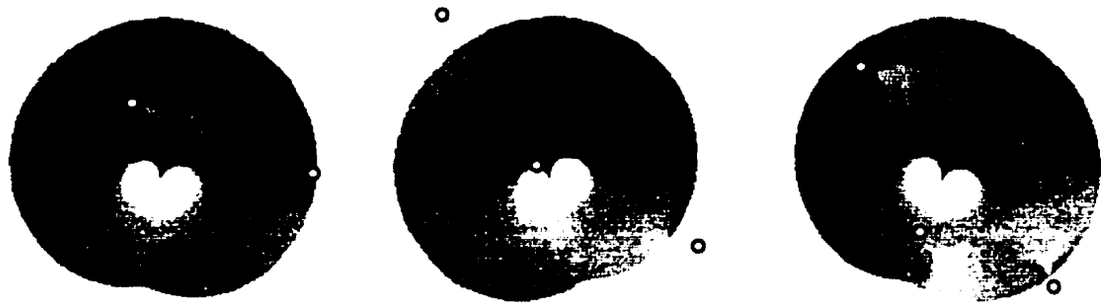
(c) DD2

Figure 4.30 : Trajectoires complètes des quatre robots identiques, avec la tactique d'évasion fournit par les trois fonctions de combinaisons DT (a), DD1 (b) et DD2 (c)

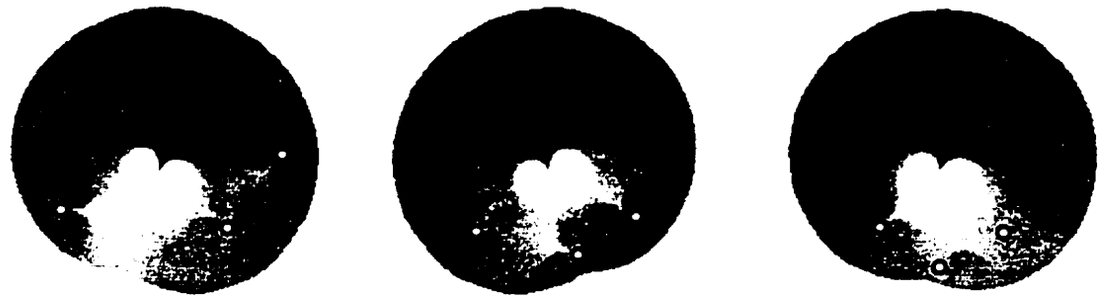
trajectoire du deuxième robot) de déplacement. Par exemple dans la figure 4.30(c), le deuxième robot est obligé de faire un tour complet pour atteindre son adversaire, perdant ainsi du terrain sur son objectif.

Les résultats numériques des expériences réalisées de la même façon que dans le cas d'un seul robot sont présentées dans les tableaux 4.7 et 4.8. Entre parenthèses les résultats obtenus dans le cas de la descente de gradient sont présentés à nouveau. On peut voir de nouveau que la méthode de descente de gradient est inférieure en terme d'efficacité à la méthode des réseaux de courbes. Toutefois, à nouveau, l'amélioration des résultats est moins significatives pour la fonction DD1 (tableau 4.7).

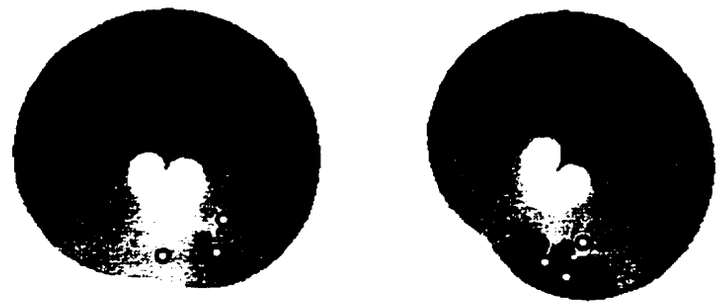
Dans le cas des robots identiques, les trois fonctions sont équivalentes. Notons



(a) T_1 (b) T_2 (c) T_3

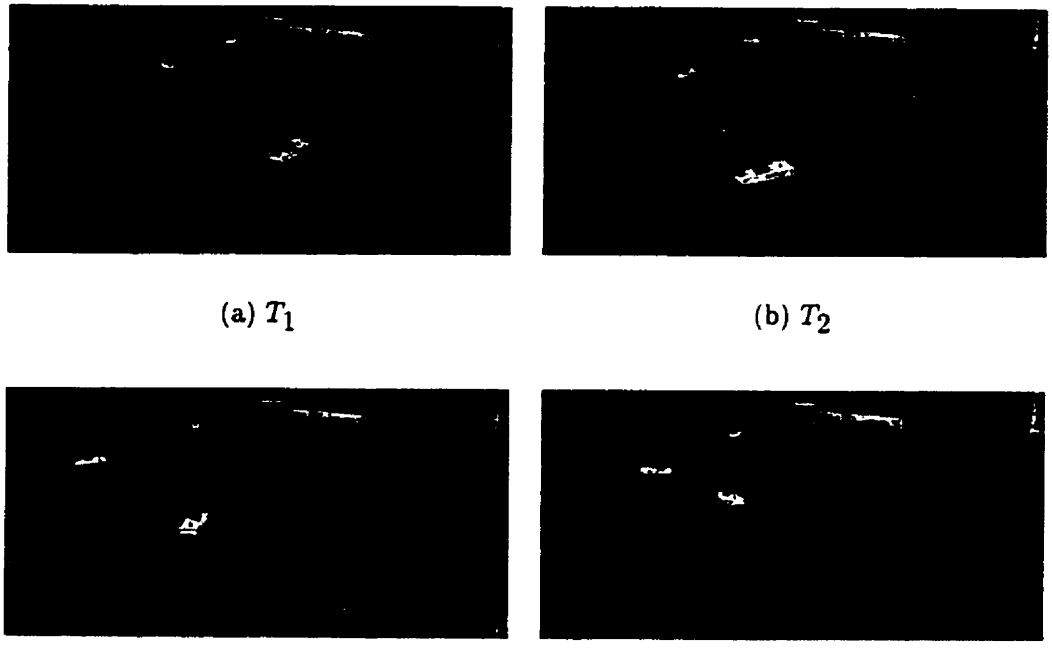


(d) T_4 (e) T_5 (f) T_6



(g) T_7 (h) T_8

Figure 4.31 : (a)-(h) Séquence des Cartes Dynamiques en correspondance avec les trajectoires des robots dans la figure 4.30(c)



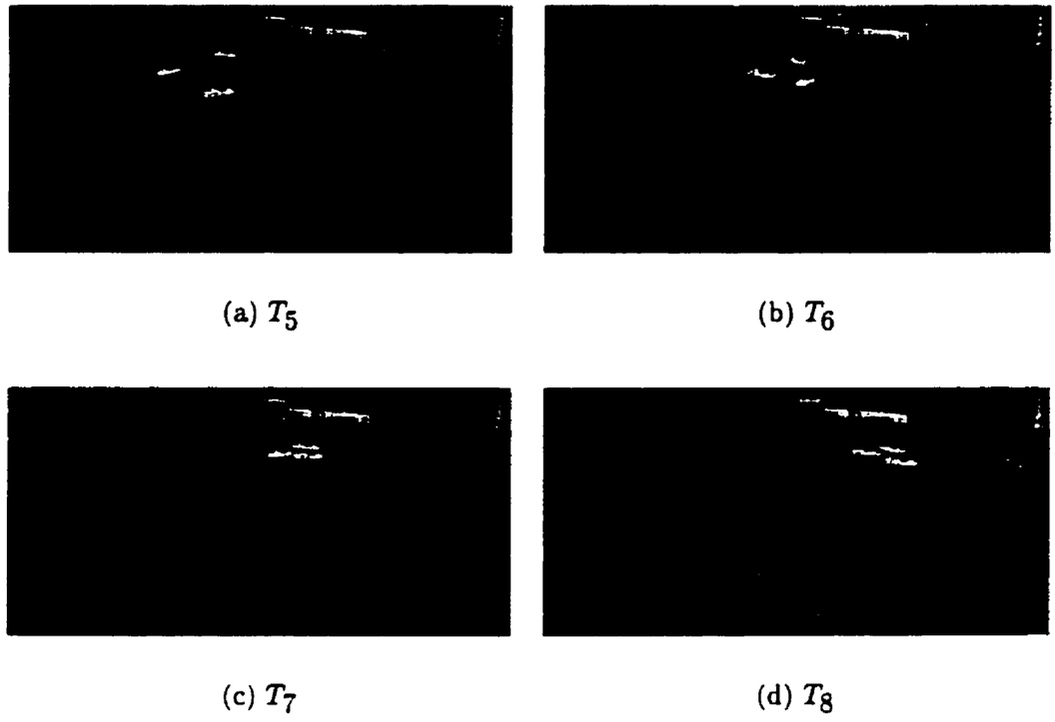
(a) T_1 (b) T_2

(c) T_3 (d) T_4

Figure 4.32 : (a)-(d) Simulation des positions relatives des robots dans l'environnement correspondant aux Cartes (a),(b),(c) et (d) de la figure 4.31

Tableau 4.7 : Résultats de l'algorithme par réseau de courbes avec les robots poursuivants plus rapides

Fct	Somme Distances	Minimum	Maximum	Temps moyen avant capture	taux de capture
DT	1.55 (1.15)	0.13 (0.05)	2.87 (2.84)	62.2 (53.1)	70% (100%)
DD1	1.61 (1.32)	0.12 (0.09)	2.86 (2.85)	63.3 (51.3)	80% (86%)
DD2	1.65 (1.2)	0.13 (0.05)	2.84 (2.82)	52.3 (53.3)	86% (100%)



(a) T_5 (b) T_6

(c) T_7 (d) T_8

Figure 4.33 : (a)-(d) Simulation des positions relatives des robots dans l'environnement correspondant aux Cartes (e),(f),(g) et (h) de la figure 4.31

Tableau 4.8 : Résultats avec réseau de courbes et des robots identiques

Fct	Somme Distances	Minimum	Maximum	Temps moyen avant capture	taux de capture
DT	3.07 (1.7)	0.45 (0.2)	2.83 (2.8)	77.3 (69)	20% (70%)
DD1	3.46 (1.25)	0.56 (0.13)	2.83 (2.8)	44.3 (74)	26% (80%)
DD2	3.12 (1.47)	0.46 (0.13)	2.78 (2.8)	52.4 (62)	23% (73%)

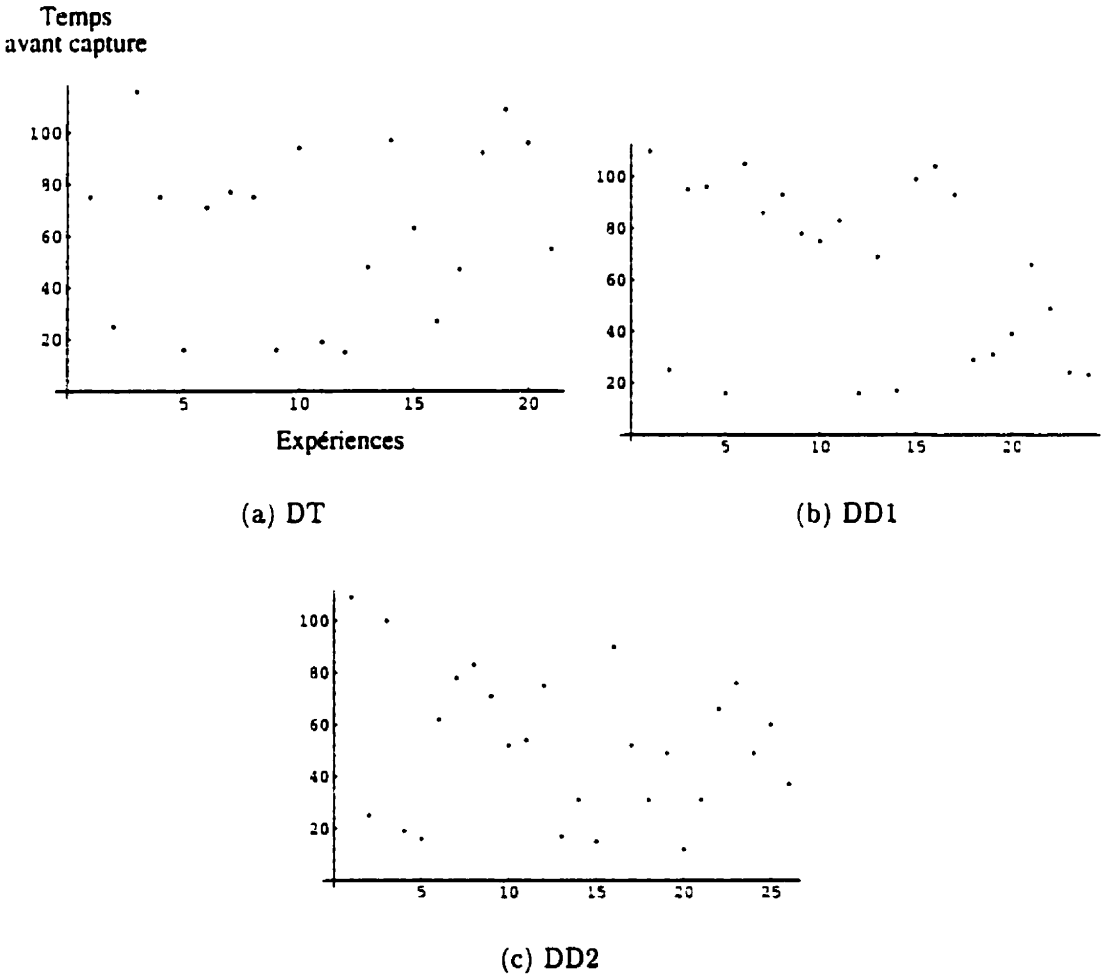
cependant l'amélioration des valeurs de "Somme Distances" dans le cas des réseaux de courbes. Ceci signifie que la méthode par réseaux de courbes (lorsque les robots sont identiques) utilise l'espace libre créé par les capacités motrices des robots poursuivants. En moyenne, l'espace autour du robot est deux fois plus important dans ce cas.

La figure 4.34 montre la valeur du temps avant capture pour chacune des expériences. On peut voir, dans le cas de la fonction DD1, par exemple, qu'il existe une séparation très nette entre les valeurs faibles de capture et les valeurs élevées (figure 4.34(b)). Cette séparation est moins nette dans le cas de la fonction DT et quasi inexistante dans le cas de la fonction DD2. Ce résultat suggère que la fonction DD1 est plus efficace pour pousser la valeur du temps avant capture à ses limites.

4.4.3.3 Conclusions sur les expériences

La méthode par réseau de courbes est de façon évidente la plus efficace des deux méthodes de planification proposées. Cependant, la descente de gradient présente l'avantage d'être rapide et simple à utiliser. Suivant l'application recherchée, il sera utile de regarder plus en détail le rapport qualité/coût des deux méthodes.

En ce qui concerne les fonctions de combinaison, il apparaît que la fonction DD1 est supérieure aux autres en conjonctions avec la descente de gradient. Il reste cependant que cette fonction est non séparable, rendant ainsi le processus de combinaison plus coûteux à implanter. Le choix d'une fonction de combinaison est ici assez délicat. De nouveau, en fonction du but désiré, le compromis entre complexité et efficacité devra



(a) DT

(b) DD1

(c) DD2

Figure 4.34 : Distribution des valeurs de temps avant capture pour les expériences avec les fonctions de combinaison DT (a), DD1 (b) et DD2 (c)

être pris en compte.

4.4.4 Et la poursuite?

Le problème de la poursuite à partir des Cartes Dynamiques n'a pas été envisagé dans cette thèse. Cependant, nous devons faire remarquer que cela semble facilement envisageable à partir des Cartes Dynamiques. Par exemple, dans la figure 4.35, une Carte Dynamique combinée dans le but de capturer trois robots est présenté. Les zones noires correspondent aux positions pour lesquelles le temps des des régions atteignables du robot poursuivant et d'un des robots poursuivis est identique. La fonction de combinaison devra ainsi mettre en relief ce genre de position. Il s'agit ici de l'exemple le plus simple de fonction de combinaison. Il sera intéressant par la suite d'étudier quelle fonction est la plus appropriée dans le cas du problème de poursuite. Il faut noter que dans le cas du problème de capture, l'objectif est un peu différent et que la Carte Dynamique va permettre non seulement de planifier une trajectoire de capture mais également de choisir la cible qui semble être optimalement la plus simple à capturer.

Ce problème de capture nous a également amené à réfléchir à la coopération de robot à partir des Cartes Dynamiques. Considérons une situation de 2 robots devant en intercepter un autre. Grâce aux techniques présentée dans ce chapitre, chaque robot poursuivant (R) peut obtenir une Carte Dynamique combinée, similaire à celle présentée dans la figure 4.35), représentant leurs interactions avec le robot poursuivi (L). Assumons maintenant que soit les robots F sont en mesure de partager leurs Cartes Dynamiques combinées, soit sont chacun capable d'extrapoler la Carte Dynamique combinée de leur partenaire. Grâce à ce partage explicite ou implicite, chaque robot F va pouvoir utiliser la Carte Dynamique combinée de son partenaire pour savoir les positions où il est plus en mesure d'intercepter le R que son partenaire (La valeur sur la Carte Dynamique devant être ainsi augmentée) ou inversement les zones où son partenaire aurait tendance à plus facile attraper R (Correspondant à

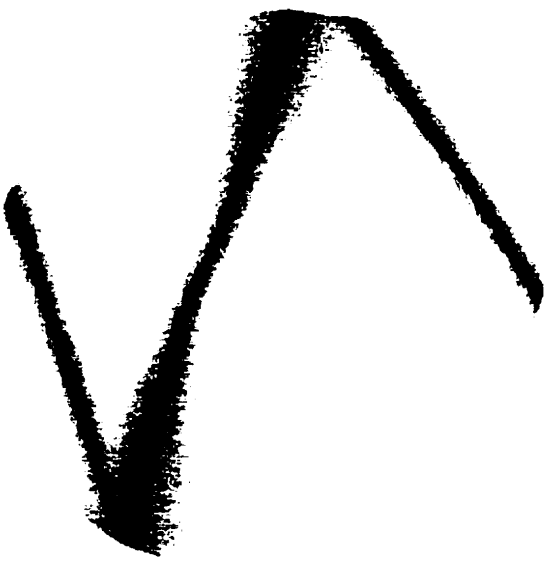


Figure 4.35 : La Carte Dynamique combinée dans le cadre d'une tâche de capture

une diminution de la valeur sur la Carte Dynamique). Cette opération est faite à nouveau en combinant les Carte Dynamiques des deux partenaires.

La figure 4.36 présente un exemple simple de ce processus de coopération avec les trois robots. Les figures 4.36(a) and 4.36(b) présentent deux exemples de Carte Dynamique combinée pour l'interception. Les régions en noir indiquent les positions où le robot R pourrait être intercepté. Dans la figure 4.36(d) le second F a combinée sa Carte avec celui de l'autre F. Comme indiqué sur la figure, certaines zones ont subi une diminution de leur valeur alors que d'autres ont été augmentées. Cet effet sera additionnel au fur et à mesure que le nombre de robot poursuivant augmente. Cela aurait pour conséquence de réduire plus les rôles de chaque robot à l'intérieur de la tâche de capture.

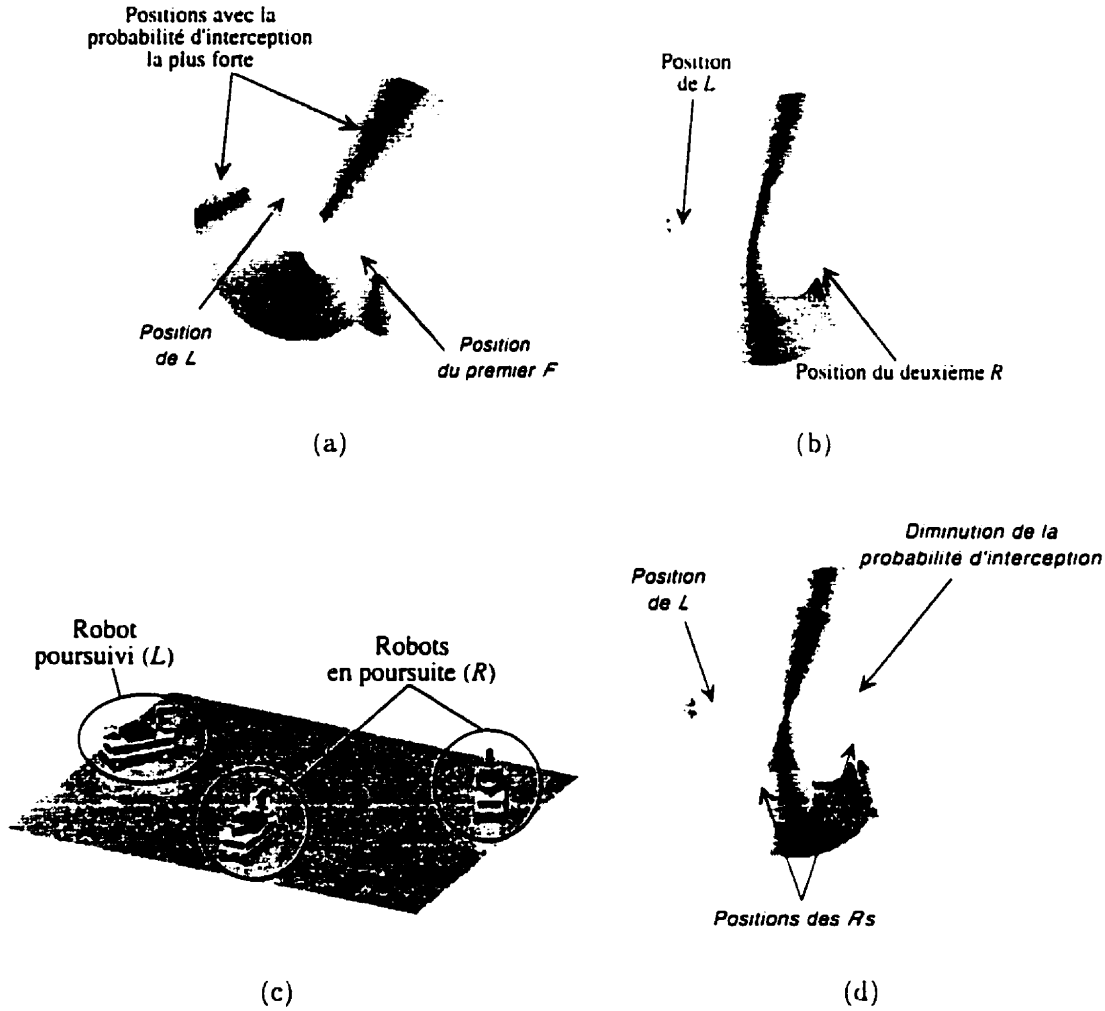


Figure 4.36 : (a) Carte Dynamique combinée entre le premier F et R (b) Carte Dynamique combinée entre le second F et R (c) Pose relative du robot R avec les 2 robots Fs (d) Carte Dynamique combinée du deuxième F avec la coopération du premier F

Chapitre 5

Apprentissage des Cartes Dynamiques

Dans ce chapitre, nous allons étudier trois types d'apprentissage reliés aux Cartes Dynamiques : tout d'abord, la section 5.1 a pour but d'explorer l'apprentissage par renforcement de trajectoire d'interception. Nous verrons également que cette technique permet d'apprendre les régions accessibles d'un robot. Après avoir mis en avant les qualités et les défauts de cette méthode d'apprentissage, nous allons envisager deux techniques d'apprentissage qui se base sur le volume de représentation des Cartes Dynamiques. Nous envisagerons en premier l'auto-apprentissage de la Carte d'un robot (Section 5.2). Finalement, nous étudions, dans la Section 5.3, la possibilité d'apprendre les Cartes Dynamiques des autres robots par l'intermédiaire de la vision.

5.1 Apprentissage par renforcement d'une stratégie de poursuite et des régions atteignables d'un robot

Cette section a pour but d'explorer une solution alternative aux Cartes Dynamiques pour faire de la planification réactive (Zanardi 1995, Zanardi 1996). Nous verrons cependant, que le comportement de l'algorithme décrit dans ce chapitre était prévisible par l'intermédiaire des Cartes Dynamiques et qu'il est également possible d'utiliser ces résultats pour générer les régions accessibles d'un robot.

5.1.1 Introduction

Nous étudions dans ce chapitre, comment un robot R (pour Renard) peut apprendre à capturer des robots L (pour Lapin), en s'adaptant à sa stratégie d'évasion à l'aide d'un réseau neuronal. Il serait utile également que R apprenne les situations pour lesquelles ses actions n'auront aucune influence sur la suite des événements. Cette faculté permettrait ainsi à un robot de regarder dans des zones qui sont intéressantes *a priori* et ainsi prendre de meilleures décisions sur d'éventuels multiples proies ou épargner ses ressources calculatoires (qui sont grandement limitées) afin de se concentrer sur l'essentiel. Ceci revient donc à déterminer les régions accessibles du robot.

Apprendre à capturer est défini ici comme l'abilité à trouver la bonne association entre un état du système (par exemple l'état interne de R et la pose relative de L) et le contrôle que R doit appliquer dans le but de suivre une trajectoire interceptant L.

En faisant l'hypothèse que R n'a aucune information *a priori* sur les capacités dynamiques de son adversaire, la seule façon pour R d'apprendre est d'essayer des stratégies (des séquences de contrôle) et d'analyser ensuite les succès et les défaites. Le paradigme d'apprentissage par renforcement est ainsi bien adapté à ce genre de

situation. De plus, l'algorithme d'apprentissage se doit d'être autonome et l'association entre les états possibles du système et les actions (ou contrôles) devrait être continue.

Nous avons donc choisi de représenter cette association par un réseau de neurones, qui non seulement a la capacité de réduire la dimension de l'espace de correspondance mais qui présente également des aptitudes à la généralisation.

Nous allons tout d'abord présenter l'algorithme d'apprentissage, dans deux situations : premièrement, avec R ayant accès directement aux informations de pose relative des deux robots, deuxièmement dans la situation où le robot poursuivant utilise la vision comme seule source de renseignements.

5.1.2 Apprendre à capturer par renforcement

Comme notre objectif est de vouloir implanter ce système en temps réel de façon logicielle, nous avons utilisé un réseau de type *feed-forward* avec $N_C > 1$ couches et des fonctions d'activation de type sigmoïde. En effet le calcul d'une réponse pour ce genre de réseau est quasi immédiat.

Les entrées sont les valeurs normalisées de l'état du système dynamique X. Il est à noter que les robots R et L sont engagés dans un jeu différentiel représentable par un système dynamique. Chaque neurone de la couche de sortie indique un contrôle différent et exclusif qui peut être appliqué au système.

La figure 5.1 présente l'architecture générale du système.

L'algorithme consiste en deux phases qui sont répétées indéfiniment. Dans la première phase, partant d'une position initiale aléatoire, le réseau procède de façon à produire une séquence d'états et de contrôles avec leurs valeurs de renforcement associées (sans modification de la valeur de ses poids). L'état initial, $X(0)$, est donné en entrée du réseau de neurones, et produit en sortie le vecteur $A(0)$. Soit $a(0)$ l'index dans le vecteur $A(0)$ du neurone ayant le taux d'activation maximum. Le contrôle correspondant est alors appliqué au système qui évolue de l'état $X(0)$ vers

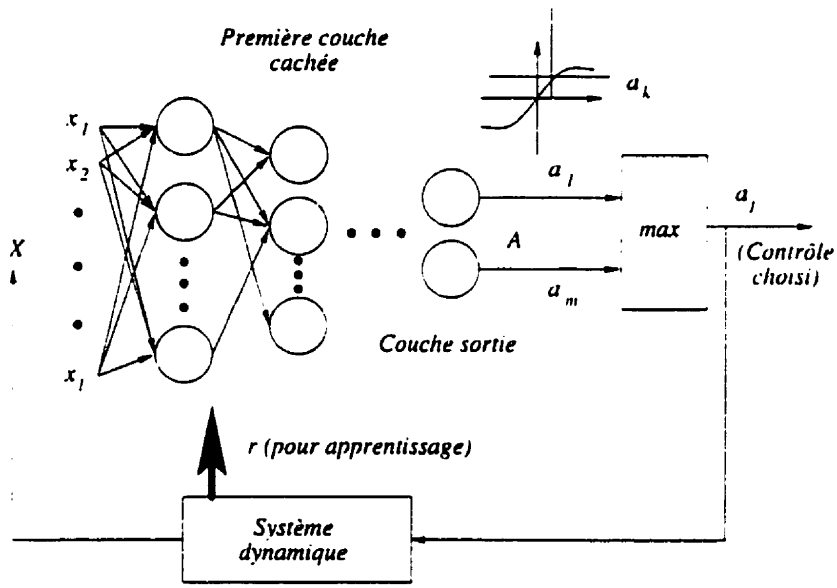


Figure 5.1 : Structure du système d'apprentissage

l'état $X(1)$, avec le renforcement $r(0)$ qui lui est associé. Ces opérations sont répétées jusqu'à atteindre un état $X(n)$ ayant un renforcement $r(n-1)$ qui indique soit un succès (valeur +1) ou un échec (valeur -1).

Dans la seconde phase, le réseau doit apprendre si les choix des actions correspondant aux états en entrée effectués dans la première phase étaient judicieux.

Il est d'abord nécessaire de calculer un renforcement rétropropagé $\tilde{r}$ qui correspond à l'influence du dernier choix (bon ou mauvais) sur l'ensemble de la séquence des vecteurs de contrôle :

$$\tilde{r}_j = r(j) + \alpha \cdot \tilde{r}_{j+1} \qquad (5.1)$$
$$j = n-2, n-1, \cdots, 0$$

α est un paramètre d'influence entre les états successifs. Les valeurs des vecteurs de sortie, $A(0), \cdots, A(n-1)$, sont également modifiées pour refléter la pertinence des contrôles choisis, $a(0), \ldots, a(n-1)$, associés aux états $X(0), \cdots, X(n-1)$. Ainsi avec

$A^k(j)$ la valeur du neurone k de sortie à l'itération j, est modifiée suivant la formule :

$$\tilde{A}_j^k = \begin{cases} A^k(j) + \beta \tilde{r}_j, \text{ si } k = a(j) \\ A^k(j) - \gamma \tilde{r}_j, \text{ si } k \neq a(j) \\ \beta, \gamma \in [0, 1]. \end{cases} \tag{5.2}$$

β et γ correspondent à des paramètres d'importance de la modification des valeurs des neurones gagnants et/ou perdants. La valeur d'activation du neurone vainqueur est ainsi augmentée si la propagation arrière du renforcement est positive, ou diminuée si ce n'était pas l'action appropriée. Dépendant de la valeur de γ, les neurones perdants sont modifiés pour amplifier les effets de correction sur des bonnes ou mauvaises actions.

Finalement, le réseau apprend à associer les valeurs d'entrées $X(0), \cdots, X(n-1)$ avec les valeurs $\tilde{A}_0, \cdots, \tilde{A}_{n-1}$, en utilisant un algorithme bien connu de rétro-propagation (Hinton 1989). Toutefois, du fait de la complexité de notre réseau et pour accélérer la convergence, nous avons utilisé un algorithme légèrement modifié de rétro-propagation incluant des momentums (ou moment d'inertie) pour éviter une descente dans un des multiples minimums locaux. La figure 5.2 présente une vue d'ensemble de l'algorithme.

Comme nous l'avons déjà mentionné, le système dynamique que nous étudions ici est composé de deux robots mobiles engagés dans une poursuite. Le modèle cinématique de chaque robot est donné par (voir figure 5.3) :

$$\begin{cases} \dot{x} = V \cos\theta \\ \dot{y} = V \sin\theta \\ \dot{\theta} = V/D \tan\phi \\ |\phi| \leq \phi_{max} \\ |\dot{\phi}| \leq \dot{\phi}_{max} \end{cases} \tag{5.3}$$

Dans le but de prouver la validité de l'algorithme, nous avons considéré deux

répéter:
  Initialiser $X(0)$
  répéter tant que $r = -1$ or $r = 1$
    $A(t)$ = sortie du réseau ($X(t)$)
    $a(t) = \max_j A^j(t)$
    Appliquer contrôle $a(t)$ pour obtenir $X(t+1)$
    $r(t)$ = renforcement ($X(t), X(t+1)$)
  fin répéter
  $\tilde{r}$ = rétro-propagation de r
  $\tilde{A}$ = réponse modifiée ($A, \tilde{r}$)
  Apprentissage de l'association $X, \tilde{A}$
fin répéter

Figure 5.2 : Survol de l'algorithme d'apprentissage

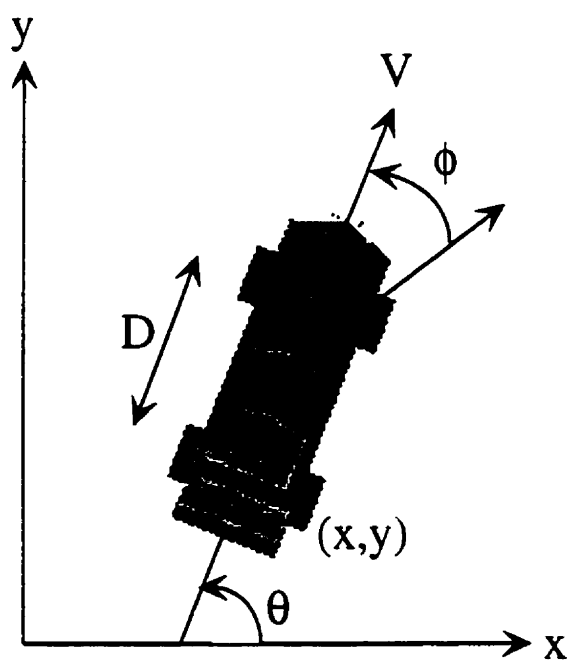


Figure 5.3 : Robot de type voiture

scénarios :

1. R a accès à la pose relative et la vitesse de L;

2. R a seulement accès à des informations acquises par son système visuel.

Accès direct aux informations relatives à L Dans le premier scénario, nous utilisons le vecteur d'état suivant (voir figure 5.4) :

$$X = \begin{bmatrix} V & \text{vitesse de } R \\ \phi & \text{angle des roues de } R \\ \rho, \omega & \text{coordonnées polaires de } L \\ \psi & \text{orientation relative de } L \\ \delta & \text{vitesse de } L \text{ suivant } \psi \end{bmatrix} \tag{5.4}$$

Nous avons choisi de contrôler seulement l'angle des roues du robot R puisqu'on peut se dire que le poursuivant et le poursuivi seront à leurs vitesses maximums respectives. Il n'y a donc que 3 neurones de sortie correspondant à une augmentation de l'angle des roues, une diminution, ou au maintien de la valeur précédente de ϕ. On dira que R a réussi à attraper son adversaire quand la distance entre les deux robots est inférieure à une marge de sécurité prédéfinie.

Un échec survient quand soit L est trop loin, ou derrière R (on considère un champ de vision limité pour un robot et qu'un robot à l'extérieur de ce champ de vision est impossible à attraper). Bien qu'il soit possible de supposer que le robot puisse tenter d'extrapoler la trajectoire de son adversaire, compte-tenu des capacités dynamiques d'un robot, il est quasiment impossible de rattraper un adversaire qui est sorti du champ de vision du robot. Pour accélérer la convergence, le renforcement indique si R se rapproche de L (récompense) ou non (punition). Cette information peut d'ailleurs être obtenue à l'aide des senseurs.

L'algorithme doit non seulement fournir des séquences de contrôle pour attraper l'adversaire, mais aussi déterminer les régions où L ne pourra être rejoint quelles que

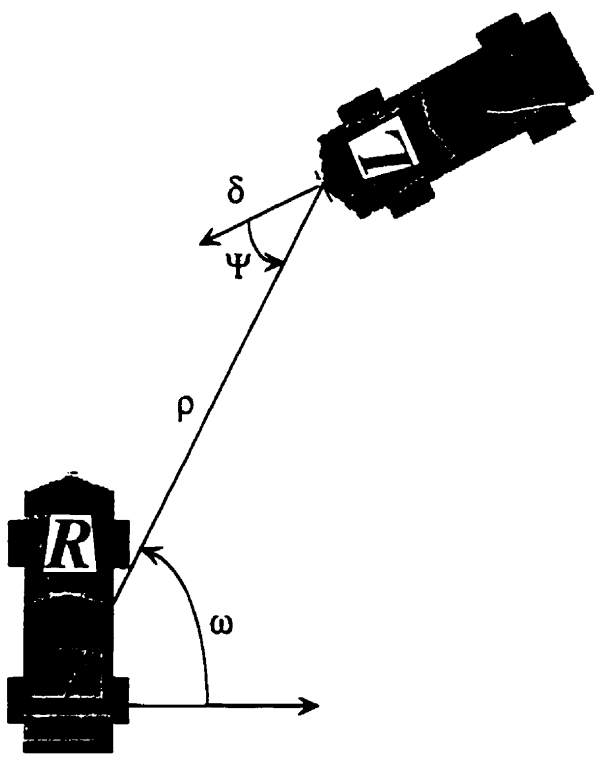


Figure 5.4 : Paramètres de pose relative entre R et L

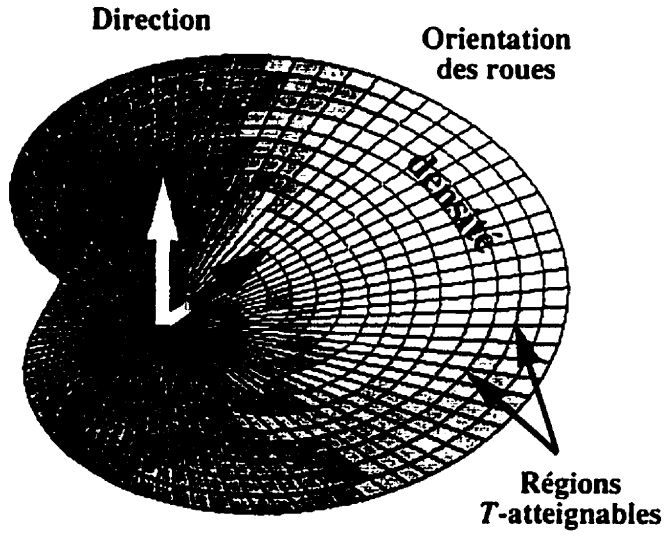


Figure 5.5 : Régions atteignables pour un robot mobile (les niveaux de gris correspondent à la distribution des régions atteignables)

soient les actions de R. Le lien est évident avec les Cartes Dynamiques que nous avons décrites dans les chapitres précédents. En effet, ces cartes nous ont permis de montrer que la distribution des régions atteignables pour un robot mobile n'était pas uniforme (voir figure 5.5) et que certaines portions de l'espace ne sont pas atteignables dans un certain laps de temps. Donc, nous nous attendions à un même comportement de la part de notre algorithme d'apprentissage. Nous allons discuter plus en profondeur des résultats dans la section 5.1.3.

Accès à des informations visuelles sur L. Dans un second scénario, un senseur de vision simulé est utilisé. Ici, R a accès à une information partielle sur son adversaire. Puisque le système devra être implanté en temps réel, on ne peut pas se fier à une analyse basée sur la reconstruction. Nous avons fait le choix de travailler directement sur l'image. Le vecteur d'état du système est alors composé de l'état interne du robot R (vitesse V et angle des roues ϕ), de la projection dans l'image d'un point connu sur L (x_i, y_i) et de la vitesse instantanée de ce point dans l'image (u, v) (voir figure 5.6).

Puisque la projection sur l'image élimine l'information de profondeur, nous avons

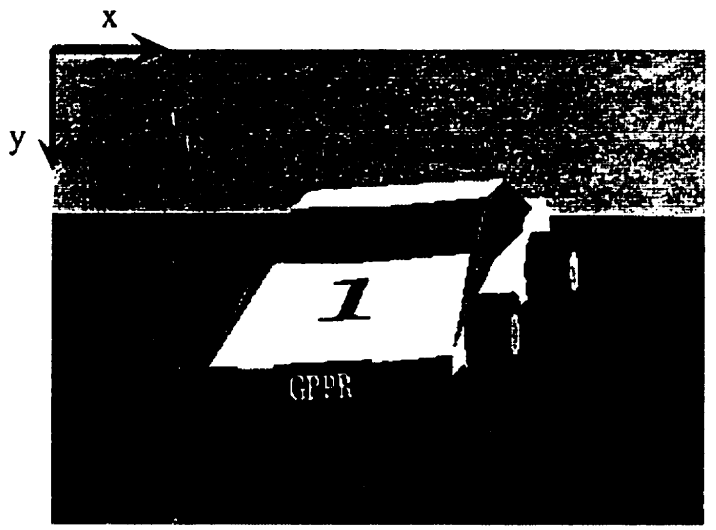


Figure 5.6 : Paramètres visuels pour la poursuite

ajouté un autre élément au vecteur d'état, la valeur du temps-avant-collision (TTC voir (Nelson et Aloimonos 1989) pour une définition plus formelle), qui mesure le temps restant avant que la cible ne rentre en contact avec le plan image.

Une remarque importante est que, puisque R connaît son propre mouvement, il est possible d'extraire les composantes de rotation des vitesses images et de calculer le TTC facilement, comme présenté dans (Burger et Bhanu 1992).

Egalement, on peut remarquer que la caméra devrait être préférablement placée entre les roues arrières du véhicule car, dans cette position, la composante de translation dans l'image due à la rotation de la voiture coïncide avec le vecteur de translation instantané.

Afin de calculer la valeur du renforcement, on fait l'hypothèse que R connaît la hauteur de la caméra H et la hauteur h du point traqué (x_i, y_i) sur le robot L par rapport au sol (comme par exemple une lampe rouge sur le robot facilement repérable). Il est ainsi possible de déterminer *a priori* dans l'image les positions indiquant que L est proche d'être capturé (récompense) ou qu'il est trop loin (punition) (voir figure 5.7).

Une punition est également produite si L sort de l'image. D'une façon similaire au premier scénario, nous utilisons une valeur de renforcement qui indique si L se

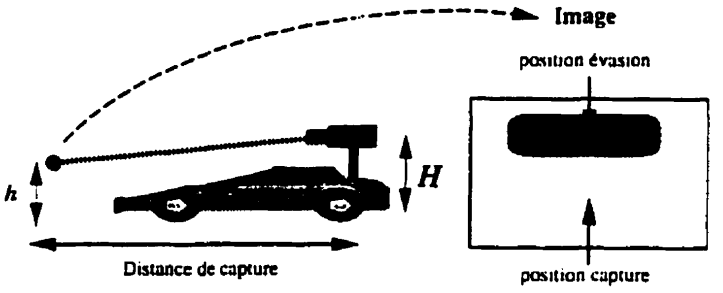


Figure 5.7 : Positions dans l'image qui indique la capture ou une évasion

rapproche et si la valeur du TTC diminue pour accélérer la convergence.

Il est important de noter que l'atteignabilité est directement accessible dans le plan image. En effet, les positions images correspondant à des zones non atteignables sont identifiables et ainsi R n'a pas besoin de porter son attention, c'est à dire, sa puissance de calcul, à de telles positions, afin d'épargner ses ressources pour d'autres tâches.

5.1.3 Résultats

Dans cette section, nous allons présenter les résultats pour le système d'apprentissage décrit dans ce chapitre. Le cas où le robot a un accès direct aux informations concernant l'autre robot sera d'abord envisagé. Par la suite, nous étudierons les résultats lorsque le robot ne possède qu'une information sensorielle. Dans l'ensemble des deux sections suivantes, le taux de réussite est obtenu en faisant des tentatives à partir de paramètres initiaux aléatoires et ensuite en divisant le nombre de succès par le nombre total d'essais. Les positions citées comme succès (ou échec), dans l'espace du robot ou le plan image, correspondent aux positions initiales du robot L pour lesquelles le réseau produit une séquence de contrôles permettant de l'intercepter (ou au robot L de s'échapper).

5.1.3.1 Apprentissage avec accès direct aux informations de pose relative

Nous avons envisagé ici trois cas :

- L est immobile ;
- L a une trajectoire rectiligne et un mouvement uniforme ;
- le robot adversaire est actif et possède une méthode efficace d'évasion.

Dans tous les cas, le réseau est à trois couches, avec 80 neurones sur les deux premières couches cachées et 3 neurones sur la couche de sortie. Ce choix s'est fait par expérience, puisqu'il n'existe pas vraiment d'autre moyen à notre connaissance de choisir optimalement le nombre de neurones sur les couches cachées d'un réseau de neurones.

5.1.3.2 Robot immobile

Lorsque le robot adversaire est immobile, la dimension de l'espace de représentation est ainsi diminuée puisque les valeurs d'états de l'adversaire correspondant aux mouvements ψ et δ sont inutiles. Nous avons cependant pensé qu'il pouvait être utile de garder la même structure de réseau afin de pouvoir utiliser le réseau appris dans ce cas comme une valeur initiale pour les autres cas. Environ 150 itérations de la boucle tentative-apprentissage ont été nécessaires pour obtenir un taux de succès de 45% (voir figure 5.8). Ce résultat peut paraître décevant si on considère que le robot L est immobile. Toutefois, en approfondissant l'étude des résultats, nous avons constaté que le phénomène des régions atteignables permettait de les expliquer. L'angle des roues initial implique en effet que portion non négligeable de l'espace devienne non atteignable quelles que soient les actions du robot.

La figure 5.9 montre, par exemple, la limitation qui est imposée sur les positions que le robot est capable d'atteindre lorsque l'angle de ses roues est initialement au maximum. Dans cette figure, on peut également considérer que les succès et échecs de la figure 5.8 représentent désormais des zones accessibles ou inaccessibles. Il faut aussi garder à l'esprit que, dès que le robot à capturer disparaît, cela est considéré comme

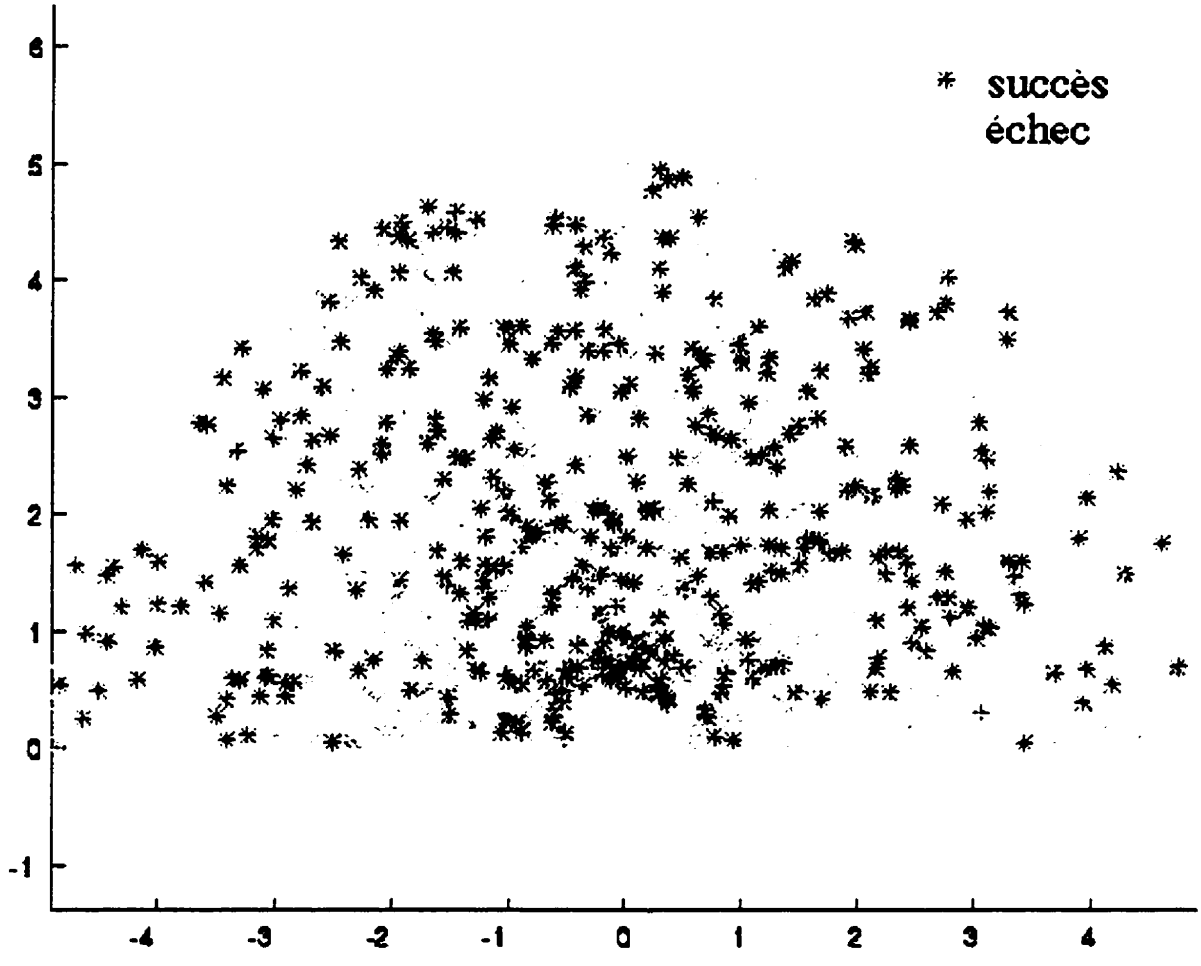


Figure 5.8 : Les succès et les échecs du réseau de neurones avec pour toutes les valeurs de $\varphi(0)$

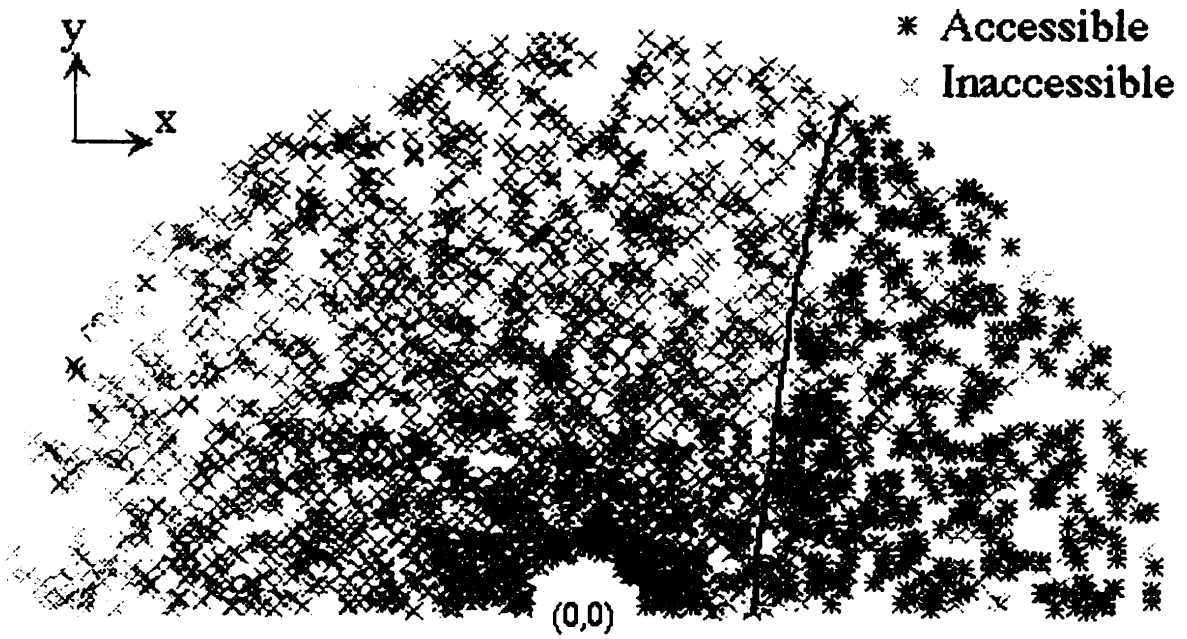


Figure 5.9 : Succès et échecs lorsque $\phi_0 = -\phi_{max}$ (taux de succès de 12%)

un échec. De ce fait, le robot ayant les roues tournées au départ, son changement d'orientation va directement rendre une partie de l'espace "non-atteignable".

Une des principales faiblesse de notre approche réside dans le type de réseau qui a été choisi. En effet, il est notoire que les réseaux de type feed-forward avec backpropagation ont tendance à oublier des configurations déjà apprises auparavant. Pour pallier ce problème, une phase de réapprentissage des bonnes configurations est incluse dans l'algorithme. Comme le montre la figure 5.10, lors du réapprentissage, des oscillations se produisent en ce qui concernent le taux de réussite de l'algorithme (ici pour une valeur de ϕ donnée). Ce phénomène peut s'expliquer par "l'élasticité" du réseau insuffisante ou trop importante pour faire cohabiter les différentes configurations. De plus, il est certain que, dans certaines situations, plusieurs trajectoires permettent d'atteindre le même point, imposant éventuellement des configurations contradictoires au réseau. Il serait important par la suite de considérer ces deux problèmes en analysant les résultats lors de la variation des nombres de neurones sur les couches cachées et en rajoutant une étape d'élimination des configurations

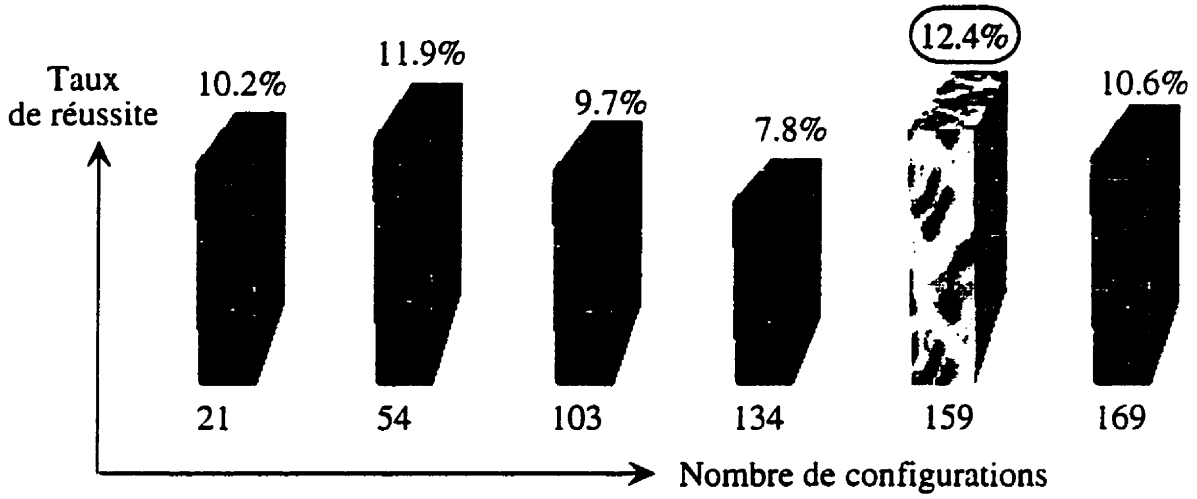


Figure 5.10 : Variation du taux de réussite du réseau pour $\phi(0) = \phi_{max}$

contradictoires ou non-optimales lors de la phase d'apprentissage.

5.1.3.3 Robot avec une trajectoire rectiligne et un mouvement uniforme

Dans le deuxième cas que nous avons étudié, le robot qui est poursuivi se déplace suivant une ligne droite. Le réseau initial fourni à notre système est celui obtenu dans le cas précédent (sous-section 5.1.3.2). Les résultats obtenus ici correspondent à un apprentissage avec 143 cycles tentative/apprentissage. Le taux de réussite dans ce cas là est d'environ 30%. Cette diminution était à prévoir par rapport au cas précédent. Le robot poursuivi étant en mouvement, le poursuivant est restreint à la fois par sa propre atteignabilité (définie lors de la première expérience avec le robot immobile) et par le mouvement de l'autre robot qui sortira donc plus facilement des régions atteignables du poursuivant

Si la valeur de $\phi(0)$ est fixée à zéro. le taux de réussite augmente jusqu'à 45% (voir figure 5.11). On obtient ainsi le même genre de résultats que dans le cas du robot immobile. c'est-à-dire des régions atteignables en fonction des paramètres initiaux du système. Dans ce cas, la forme des régions atteignables dépend à la fois de l'orientation initiales des roues du robot mais également de l'orientation et de la vitesse initiale du robot à poursuivre.

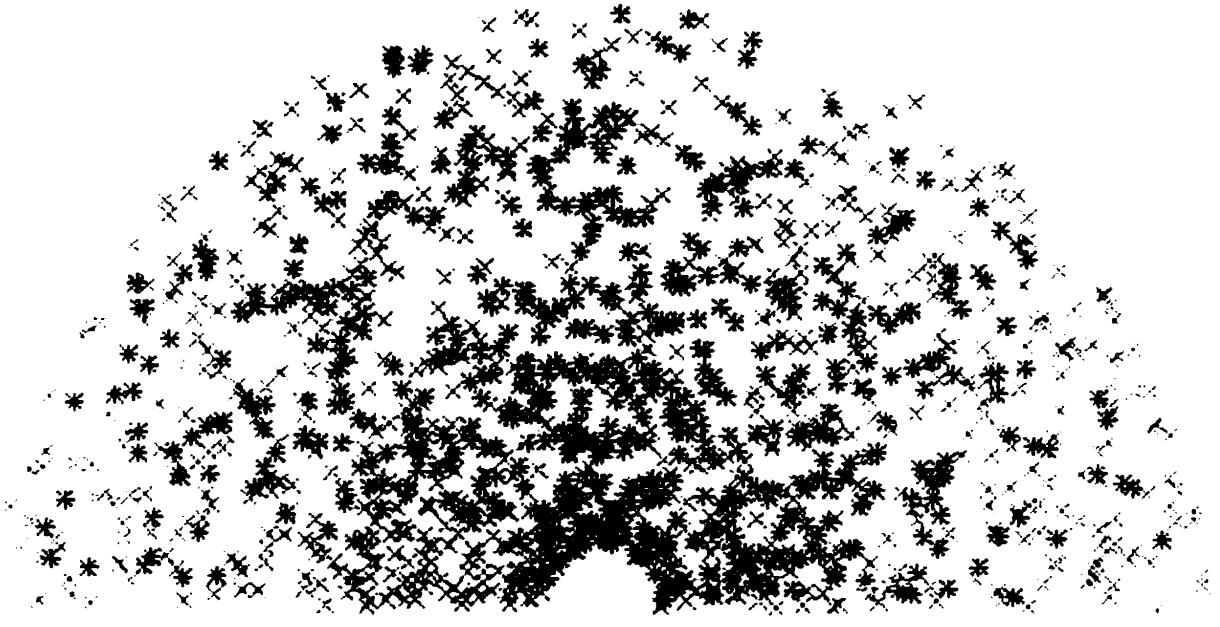


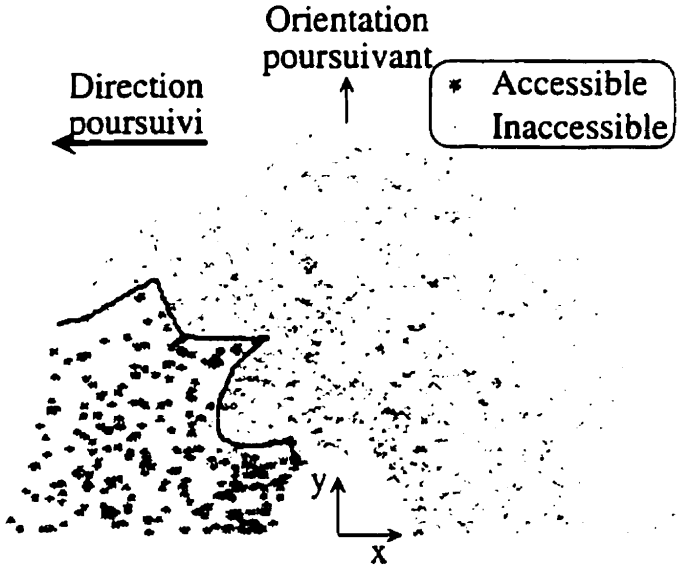
Figure 5.11 : Résultat du réseau dans le cas du robot en déplacement rectiligne et $\phi(0) = 0$

En effet, dans la figure 5.12(a), il est clair que la zone atteignable avec l'orientation initiale du robot poursuivi indiquée est plus grande que celle présentée dans la section précédente. Le phénomène est inversé dans la figure 5.12(b) avec un taux de réussite quasiment nul.

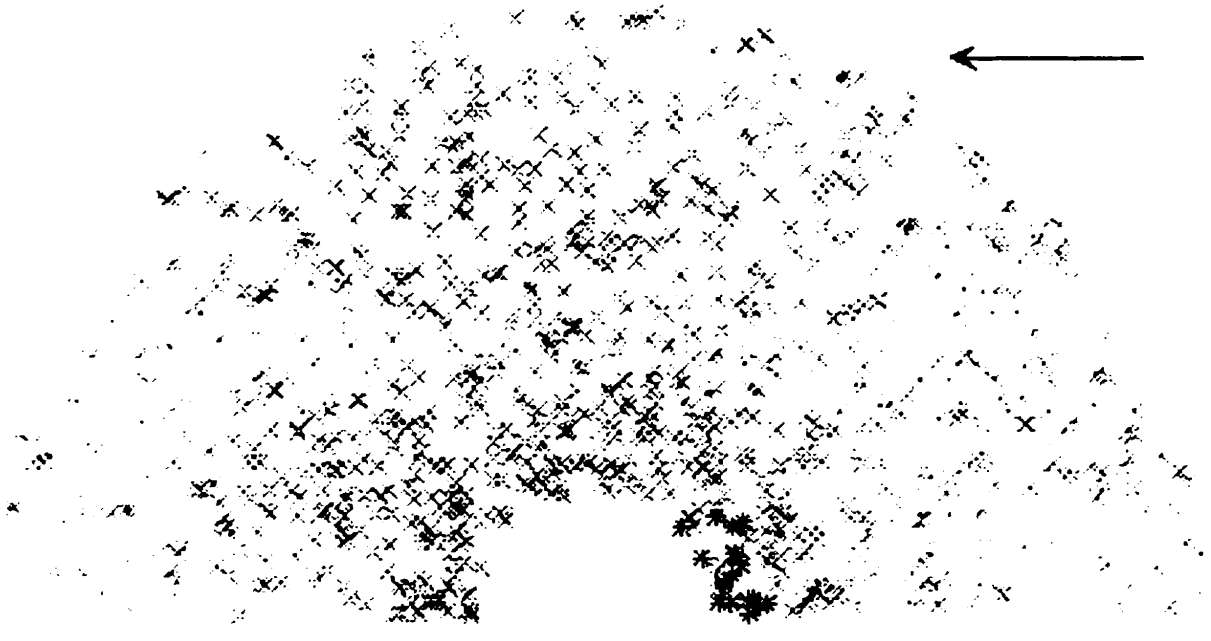
La figure 5.13 présente deux exemples de résultats obtenus avec des orientations initiales différentes du robot poursuivi et deux angles des roues initiaux. Ces résultats montrent bien la disparité des résultats que l'on peut obtenir avec des paramètres initiaux différents.

La figure 5.13(b) présente toutefois une autre limitation de l'algorithme puisque d'après les valeurs initiales utilisées, les zones atteignables devraient être symétrique par rapport à l'axe des y. Une solution (coûteuse) à ce léger problème serait d'imposer au réseau de fournir des résultats symétriques. Finalement, la figure 5.14 présente quelques exemples de trajectoires planifiées par le réseau en fonction de position initiale de L.

Nous avons également utilisé des valeurs de paramètres correspondant à un des

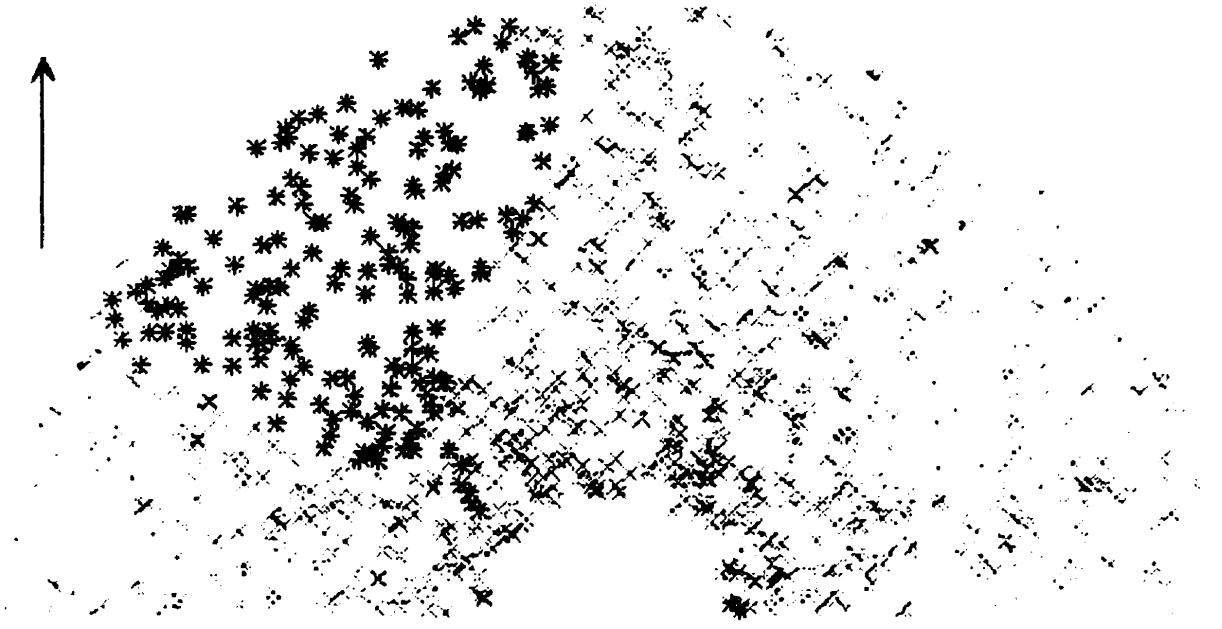


(a) $\phi(0) = \phi_{max}$ et $\psi(0) = \pi$



(b) $\phi(0) = -\phi_{max}$ et $\psi(0) = \pi$

Figure 5.12 : Succès et échecs du réseau lorsque le robot à capturer se déplace de façon rectiligne suivant la direction $\psi(0) = \pi$ et avec $\phi(0) = \phi_{max}$ (a) et $\phi(0) = -\phi_{max}$(b)



(a) $\phi(0) = \phi_{max}/2$ et $\psi(0) = \pi/2$

(b) $\phi(0) = 0$ et $\psi(0) = -\pi/2$

Figure 5.13 : Exemples de résultats du réseau avec $\phi(0) = \phi_{max}$, $\psi(0) = \pi/2$ (a) et $\phi(0) = 0$, $\psi(0) = -\pi/2$ (b)

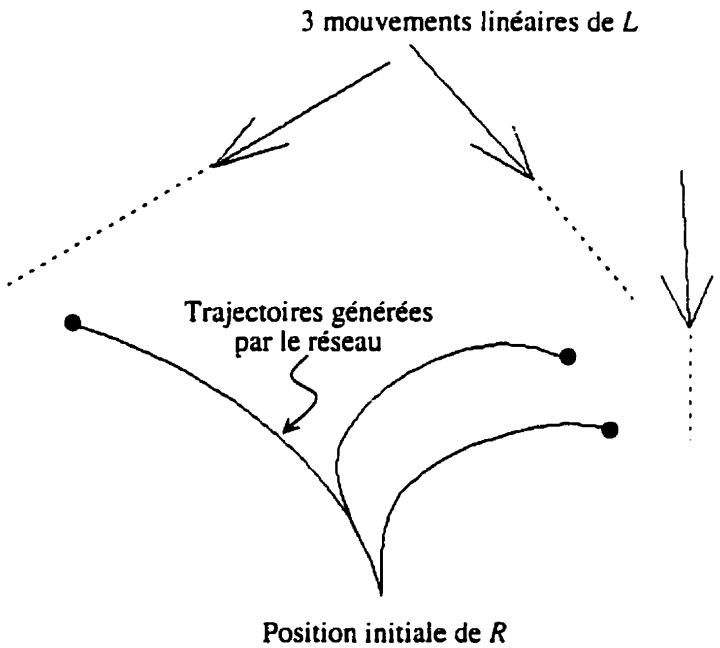


Figure 5.14 : Exemples de trajectoires effectivement planifiées par le réseau de neurones pour plusieurs positions initiales de L

deux robots de notre banc d'essai expérimental (voir chapitre 6), afin de planifier par la suite des trajectoires directement sur les robots. Toutefois, la contre-réaction odométrique n'étant pas disponible au moment des expériences, nous avons utiliser le réseau sans contre-réaction de position. Un exemple de trajectoire réalisé par le robot est présenté dans la figure 5.15.

Dès que les données odométriques seront utilisables en temps réel, il sera possible de planifier directement les trajectoires pour les robots. Il faut préciser que les réseaux seront appris en simulation avec des paramètres les plus proches possibles des valeurs réelles. Il serait en effet impraticable de procéder à l'apprentissage en situation réelle.

5.1.3.4 Robot actif

Dans ce cas, nous avons doté le robot adversaire d'une tactique d'évasion active (malheureusement, pour l'instant, il n'était pas possible d'utiliser l'algorithme basé sur les Cartes Dynamiques), se basant sur la position du robot qui tente de le poursuivre. La tactique est très simple mais efficace : le robot tourne ses roues pour avoir

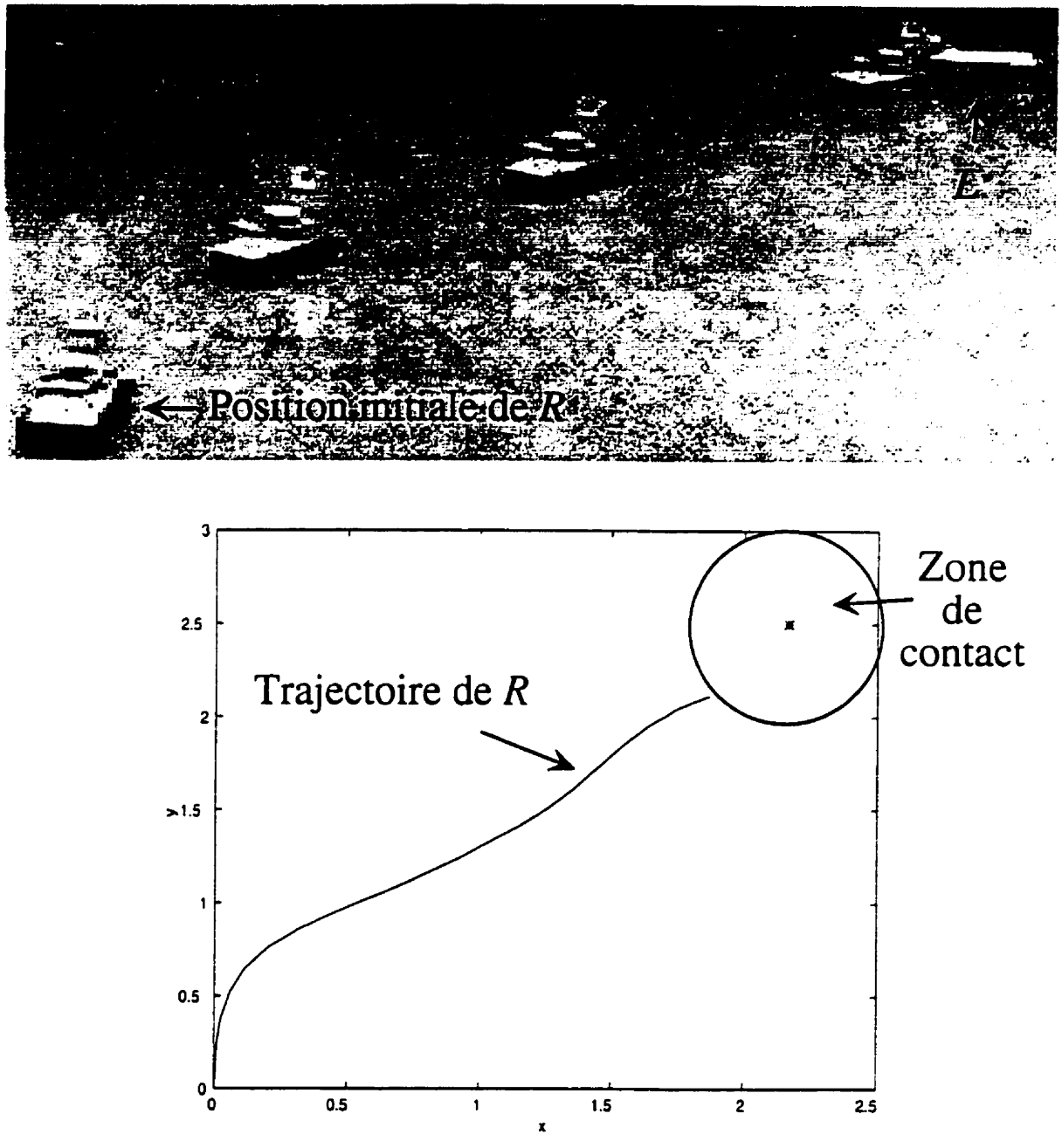


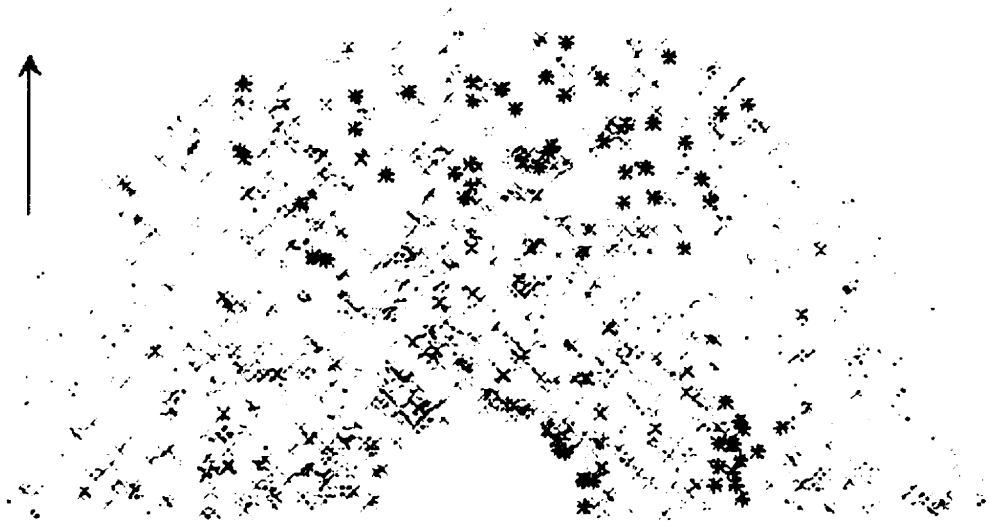
Figure 5.15 : Exemple de trajectoire planifiée pour un des robots de notre banc d'essai expérimental

son adversaire dans son dos, puis part en ligne droite. Par analogie avec le cas précédent, nous avons utilisé comme valeur initiale du réseau, le résultat de l'algorithme du cas précédent. Logiquement, le taux de réussite diminue dans ce cas. Toutefois les résultats sont un peu décevants puisque le robot arrive à capturer son adversaire dans moins de 10% des cas après 200 itérations. Ces résultats peuvent s'expliquer par la complexité que l'adversaire actif induit sur l'apprentissage. En effet, la tactique du robot étant active quand une mauvaise configuration est obtenue et que le réseau est alors modifié, l'adversaire va s'adapter de nouveau et augmenter ainsi la probabilité d'obtenir une mauvaise séquence de contrôles. La figure 5.16 présente les résultats pour deux valeurs différentes de l'angle des roues ϕ.

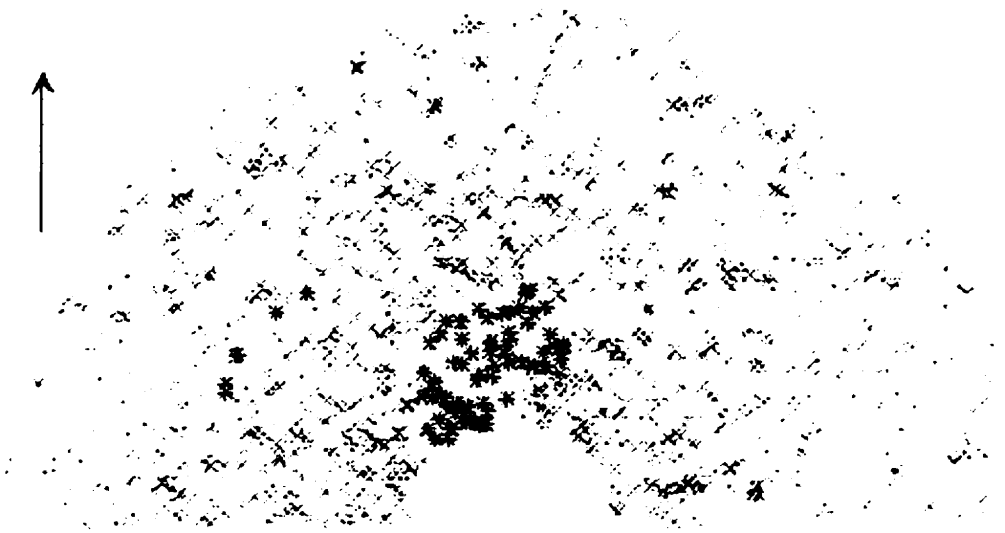
De façon surprenante, lorsque nous avons testé le réseau appris précédemment contre la tactique active d'évasion, les résultats ont augmenté de façon significative (jusqu'à un taux de réussite de 30%). La figure 5.17 présente un exemple de résultat avec le réseau appris dans le cas précédent après 143 itérations de l'algorithme. L'apprentissage d'une stratégie de capture face à une tactique complexe d'évasion semble ainsi superflu. Les capacités d'anticipation du réseau appris dans la section 5.1.3.3 sont apparemment suffisantes pour appréhender les mouvements d'un robot. Le système fonctionnant continuellement, le robot va anticiper chaque variation du mouvement de l'autre robot comme un mouvement rectiligne. D'après les Cartes Dynamiques, ce genre de mouvement est parmi les plus probables (voir la forme des Cartes Dynamiques par rapport à $\phi = 0$) pour un robot.

5.1.3.5 Apprentissage avec données sensorielles

Dans ce cas, le robot a accès seulement à des informations acquises par son senseur visuel. L'apprentissage est toujours fait en simulation, en choisissant au hasard des positions et des orientations initiales dans l'espace du robot et en les projetant ensuite dans le plan image de la caméra. La caméra étant inclinée vers le sol, la distribution des positions initiales dans le plan image n'est pas aussi uniforme que dans les exem-



(a) $\phi(0) = 0$ et $\theta = \pi/2$



(b) $\phi(0) = \phi_{max}/2$ et $\theta = \pi/2$

Figure 5.16 : Résultats de l'algorithme avec un adversaire ayant une tactique d'évasion active $\phi(0) = 0$ (taux de réussite 6.1%) (a) $\phi = \phi_{max}/2$ (taux de réussite 8.1%) (b)

Figure 5.17 : Exemple de résultat avec le réseau appris dans le cas précédent pour une adversaire actif $\phi(0) = 0$ et $\psi(0) = 3\pi/2$ (taux de réussite 30%)

ples précédents. Cela ne gène cependant pas l'analyse des résultats. La figure 5.18 présente les succès et les échecs pour des positions et paramètres quelconques du système pour 200 itérations du cycle d'apprentissage.

Comme nous l'avons observé dans les trois types d'exemples de la section précédente, le robot est soumis à des régions atteignables dont les formes dépendent des valeurs initiales des paramètres du système. Nous avons élaboré dans le chapitre 3 la relation qui existe entre les capacités motrices d'un robot et ses capacités sensorielles. Les résultats obtenus par notre système d'apprentissage sont en parfait accord avec les prédictions faites par les Cartes Dynamiques. En effet, dans les figures 5.19 et 5.20 les régions atteignables apparaissent clairement en fonction à la fois de l'orientation initiale des roues du robot, ainsi que de l'orientation initiale du robot à capturer.

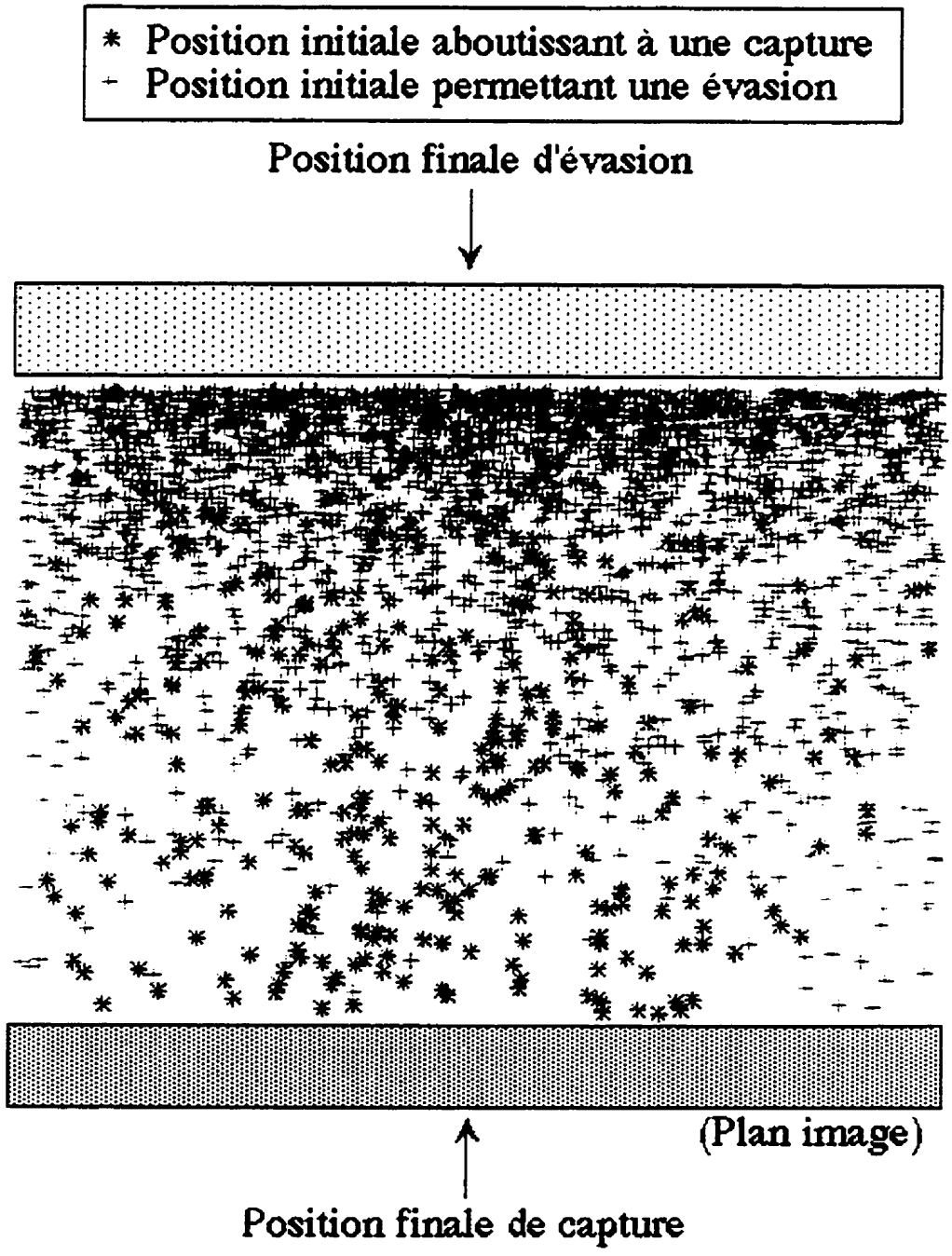
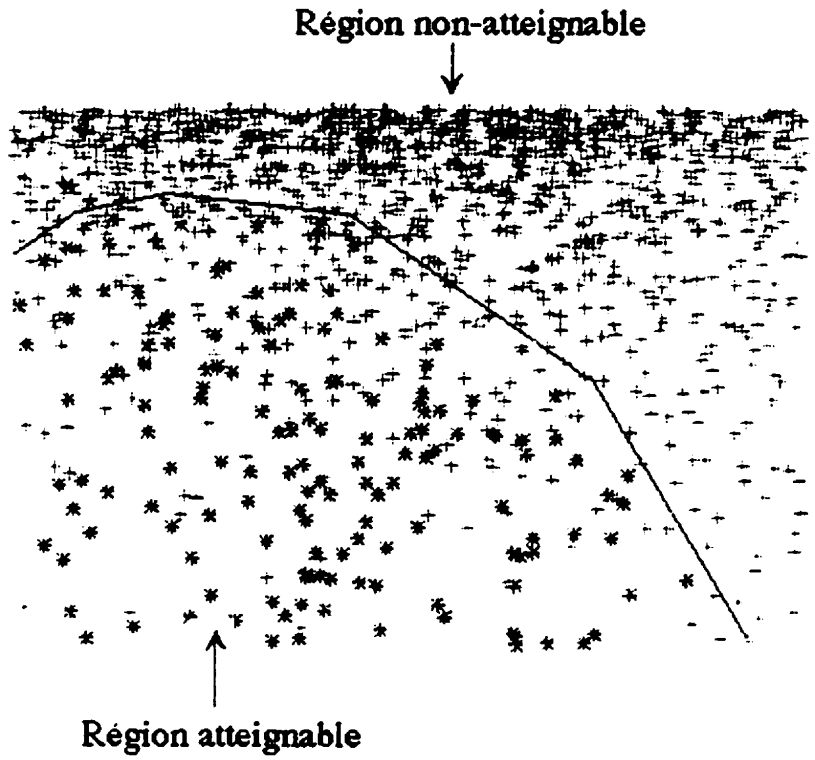
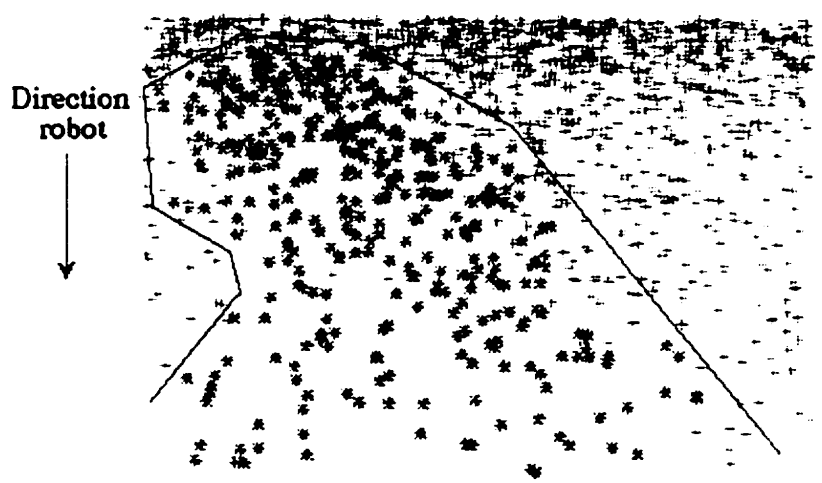


Figure 5.18 : Succès et échecs pour les positions initiales du robot à capturer dans le plan image (taux de réussite 10%)



(a) $\phi(0) = 4/5\phi_{max}$



(b) $\phi(0) = \phi_{max}/2$ et direction du robot $3\pi/2$

Figure 5.19 : Régions atteignables obtenues dans le plan image avec $\phi(0) = 4/5\phi_{max}$ (taux de réussite 13.2%) (a) et $\phi(0) = \phi_{max}/2$, direction du robot $3\pi/2$ (taux de réussite 26.4%) (b)

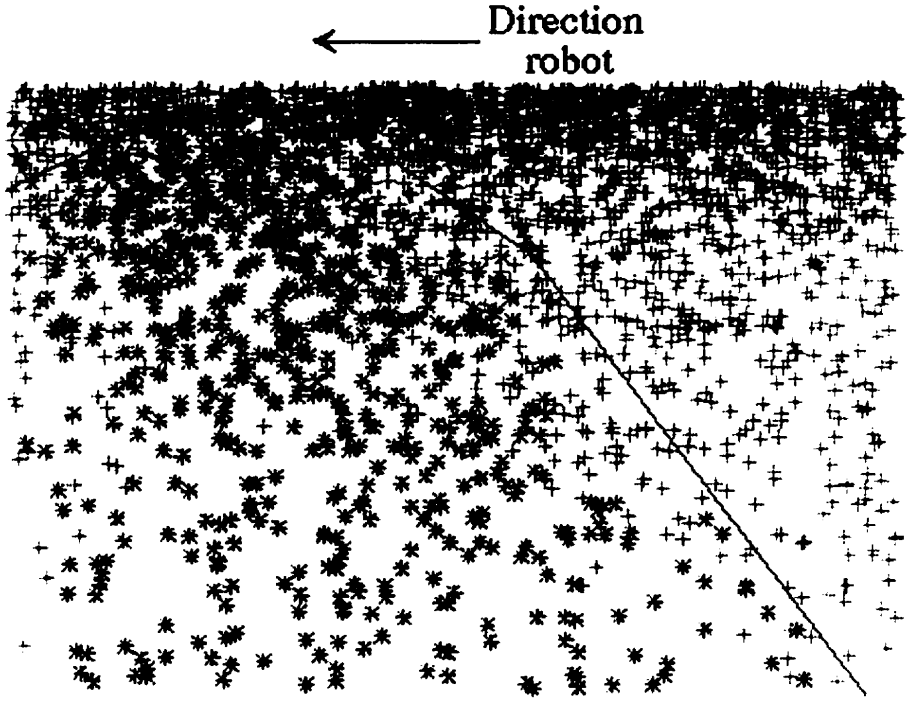


Figure 5.20 : Régions atteignables obtenues dans le plan image avec $\phi(0) = 0$ et la direction du robot L $\delta(0) = \pi$ (taux de réussite 16.3%)

5.1.4 Avantages et inconvénients de l'apprentissage par renforcement

Dans l'ensemble des résultats obtenus, la notion de régions atteignables a fortement influencé l'interprétation des résultats. Dans la littérature, peu d'attention est généralement consacrée à l'analyse des échecs d'un algorithme. Nous pensons qu'il s'agit ici d'un élément important d'une quelconque analyse. En effet, comme nous l'avons vu, dans certains cas, quelles que soient les actions du robot, il ne parviendra pas à capturer son adversaire. Le robot peut ainsi savoir à l'avance si ces actions peuvent permettre la capture de son adversaire. Dans le cas contraire, le robot peut consacrer ses ressources de calcul à d'autres tâches plus importantes. Le robot a donc appris ses régions accessibles par rapport aux paramètres actifs de la tâche et peut à l'avance déterminer les actions à prendre. Dans un environnement avec plusieurs robots, ce genre d'analyse peut être utile lorsqu'un choix doit se faire entre plusieurs

robots à intercepter.

L'avantage essentiel de cette technique d'apprentissage réside dans sa capacité à apprendre avec un minimum de connaissance *a priori* sur les robots. Le système va ainsi apprendre les trajectoires d'interception en ayant à sa seule disposition le résultat final de chaque expérience. De plus, l'apprentissage prodigue avantageusement à la fois les trajectoires d'interception et les régions atteignables du robot.

Les défauts de la méthode d'apprentissage par renforcement sont les suivants :

1. l'apprentissage dépend de la stratégie du poursuivi : En effet, le système se base sur la supposition que la stratégie du poursuivi est toujours la même, afin que la relation états/actions soit unique. Si le poursuivi change de stratégie, il est nécessaire de refaire l'apprentissage au complet car à chaque état pourra correspondre une autre action.

2. l'apprentissage ne renseigne pas vraiment sur les capacités du poursuivant : ce défaut découle du premier point. Si l'apprentissage dépend des capacités du robot à s'évader, les régions accessibles vont refléter la capacité du poursuivant à attraper le poursuivi avec sa stratégie d'évasion. La Carte Dynamique nous fournit, elle, une information sur les capacités **intrinsèques** de déplacement du robot dans son environnement.

3. la relation états/actions peut être longue à obtenir : plusieurs centaines de tentatives sont parfois nécessaire pour obtenir la correspondance entre les états et les actions. C'est le problème généralement rencontré par les systèmes d'apprentissage par renforcement. Le robot est donc « aveugle »durant cette période.

4. l'apprentissage est basé sur des systèmes neuronaux et hérite donc des avantages et inconvénients inhérents à ce genre de système. Nous avons mentionné les principaux problèmes rencontrés dans l'ensemble de la section sur l'apprentissage par renforcement.

Les deux techniques d'apprentissage que nous allons présenter maintenant ont l'avantage de résoudre les défauts de l'apprentissage par renforcement ainsi que d'apporter d'autres propriétés intéressantes :

1. l'apprentissage fournit une représentation intrinsèque des capacités de déplacement du robot : l'apprentissage n'est pas dépendant de la stratégie du robot que l'on observe.

2. chaque observation fournit des renseignements immédiat au robot sur les formes des Cartes Dynamiques. En effet, chaque trajectoire du robot que l'on observe contient des informations sur ces capacités de déplacement que l'algorithme exploite directement.

3. le résultat est directement utilisable par le robot : le robot peut exploiter le résultat de l'apprentissage en générant la Carte Dynamique du robot observé et ainsi tirer avantage de l'éventail des possibilités de planification en compétition et coopération (ce qui n'était pas le cas dans la technique par renforcement).

5.2 Auto-Apprentissage de la Carte Dynamique par un robot

Nous envisagerons dans le cadre de la section suivante l'apprentissage des capacités motrices des autres robots à partir de la vision. Nous allons étudier dans cette partie une méthode simple d'apprentissage par un robot de sa propre Carte Dynamique. Nous allons montrer dans cette section comment il est possible de calculer la Carte Dynamique d'un robot par apprentissage.

Dans l'auto-apprentissage des cartes, la supposition est faite que le robot connait les commandes qu'il envoie à ses moteurs et donc qu'il peut effectuer une séquence lui permettant d'exposer ses capacités limites. Ainsi, si le robot, partant d'une position initiale quelconque, tourne ses roues au maximum à droite ou à gauche et maintient cette valeur, le robot décrira une tranche du volume de représentation.

Il est bien sûr possible d'utiliser un système d'estimation de paramètres pour un modèle décrivant le robot, mais cela ne correspond pas à l'optique des Cartes Dynamiques. Nous avons choisi d'utiliser pour l'apprentissage une des capacités du banc d'essai que nous présenterons au chapitre 6. En effet, nous disposons d'un système d'observation central qui permet de connaître la position absolue des robots dans l'espace de travail. Comme nous le verrons, ce système est également pourvu d'un filtre de Kalman qui va filtrer les données bruitées du système d'odométrie pour obtenir les valeurs différentielles de la courbure κ du chemin. Nous devons faire remarquer ici que, malgré le fait que les données odométriques soient centralisées, notre système demeure conceptuellement décentralisé. La méthode d'acquisition des données odométriques constitue seulement la solution d'un problème technologique (comme nous le montrerons lors de la présentation du banc d'essai expérimental du chapitre 6). Chaque robot aura donc seulement accès aux données le concernant.

Ainsi, le robot exécute une trajectoire correspondant à une séquence de contrôle limite. Les valeurs des positions absolues sont obtenues à partir du système central d'observation, et le filtre de Kalman permet alors de récupérer la séquence de positions x, y, d'orientation θ et de courbure κ en chaque point.

Le temps de réaction du robot est imposé par le taux de traitement de l'information visuelle fournie par l'observateur. Nous supposons que les capacités dynamiques du robot sont symétriques par rapport à l'angle de ses roues (c'est-à-dire que le robot a la même capacité de tourner à gauche qu'à droite). Il est à noter que cela n'est pas nécessairement vrai dans le cas d'un robot subissant une usure non uniforme des engrenages ou d'une distribution de forces non uniforme dans le cas du vérin de

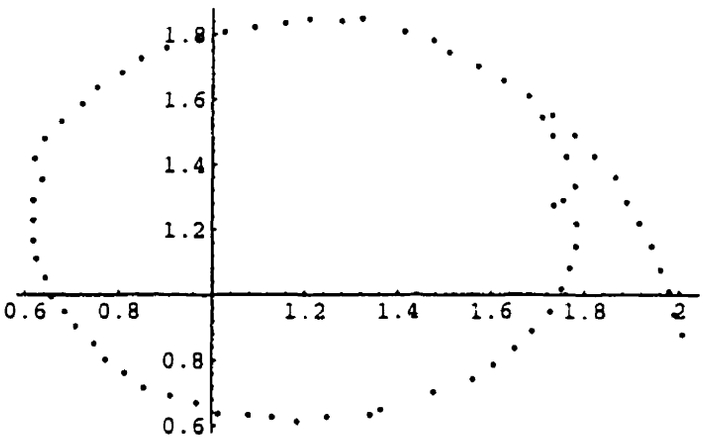


Figure 5.21 : Trajectoire obtenue par l'odomètre central (voir chapitre 6 pour le robot Tom

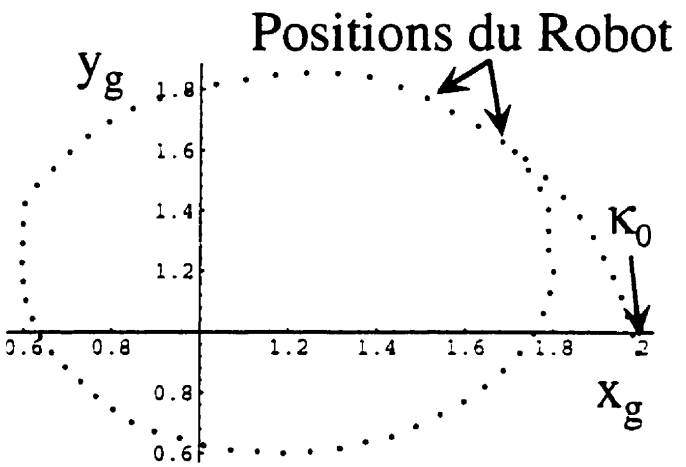
contrôle d'un véhicule articulé.

Afin ne pas avoir à multiplier les expériences pour chaque valeur initiale de κ, il est possible d'utiliser des trajectoires déjà obtenues pour créer une autre tranche du volume. En effet, considérons la trajectoire limite $x_l(t), y_l(t), \theta_l(t), \kappa_l(t)$ (position,orientation et courbure instantanée associée) avec $x_l(0) = y_l(0) = 0, \theta_l(0) = \pi/2, \kappa_l(0) = \kappa_0$. Pour $t = t_1$, une nouvelle trajectoire limite est calculée en utilisant la transformation $\mathcal{T}_{t_1}$ composée d'une translation de vecteur $(-x_l(t_1), -y_l(t_1))$ et d'une rotation d'angle $\theta_l(0) - \theta_l(t_1)$, pour ainsi recaler la trajectoire sur la même position et orientation initiale. La nouvelle trajectoire est alors :

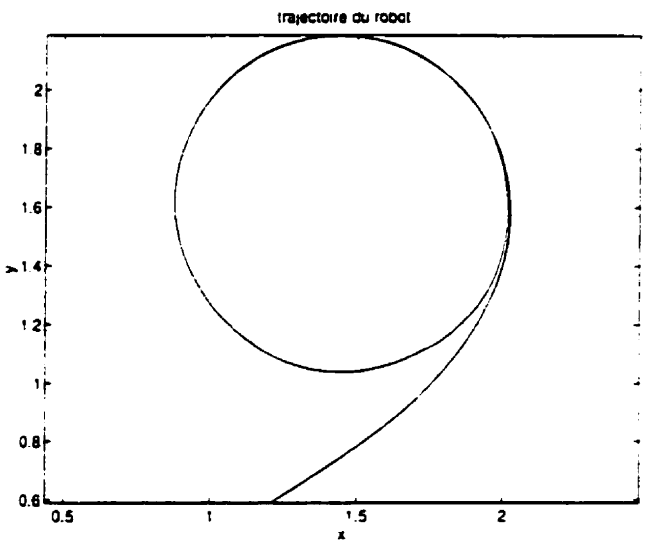
$$\begin{cases} x_{l1}(t) = \mathcal{T}_{t_1}\left[x_l(t+t_1), y_l(t+t_1), \theta_l(t+t_1)\right], \\ y_{l1}(t) = \mathcal{T}_{t_1}\left[x_l(t+t_1), y_l(t+t_1), \theta_l(t+t_1)\right], \\ \theta_{l1}(t) = \mathcal{T}_{t_1}\left[x_l(t+t_1), y_l(t+t_1), \theta_l(t+t_1)\right], \\ \kappa_{l1}(t) = \kappa_l(t+t_1). \end{cases} \tag{5.5}$$

Un exemple de trajectoire acquise pour le robot Tom par l'odomètre central est présenté à la figure 5.21. Le résultat obtenu par filtrage de Kalman sur cette trajectoire est donnée dans la figure 5.22.

Les volumes recréés pour les deux robots du banc d'essai expérimental sont don-



(a)



(b)

Figure 5.22 : (a) Trajectoire pour le robot Tom de la figure 5.21 lissée par un filtrage de Kalman (b) Trajectoire pour l'autre robot (Jerry).

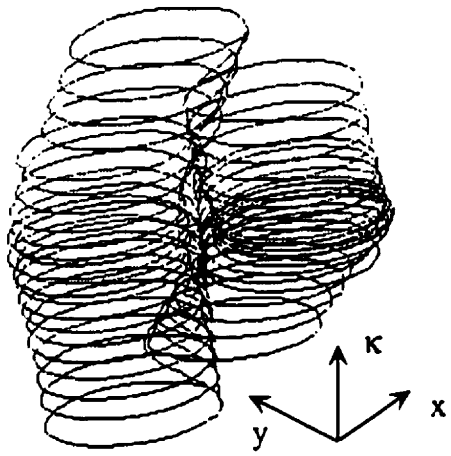


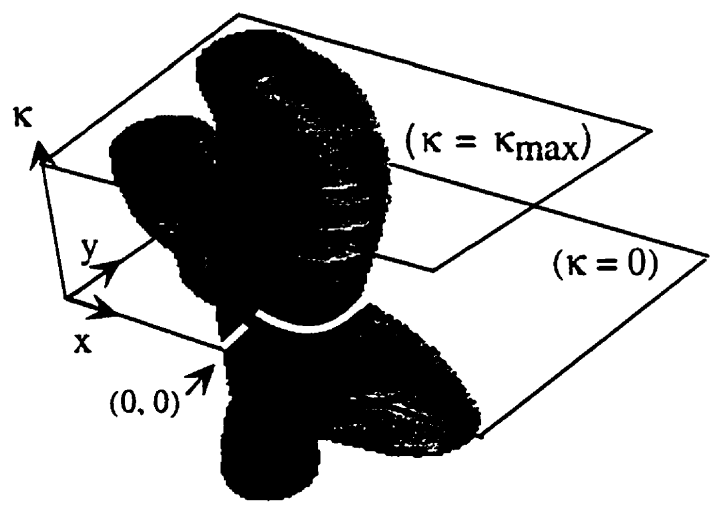
Figure 5.23 : Le volume reconstruit couche par couche pour le robot Tom

nés dans les figures 5.23 et 5.24. L'apprentissage des Cartes n'était réalisable pour l'instant que sur les véhicules automobiles présentés dans le chapitre 6.

5.3 Apprentissage des Cartes Dynamiques des autres robots par la vision

L'objectif de cette section est de démontrer qu'il est possible d'apprendre les Cartes Dynamiques des autres robots se trouvant dans l'environnement par l'intermédiaire du système visuel d'un robot (ici une caméra CCD embarquée, voir le chapitre 6 pour plus de détails). Il est à noter cependant que certains éléments de l'analyse ont été simplifiés. Par exemple, la détection du mouvement du robot dans l'image a été simplifiée en utilisant les lampes rouges (et donc détectable par une caméra munie d'un filtre infra-rouge) se trouvant sur les robots de notre banc d'essai expérimental. De plus, le robot qui utilise son système visuel pour apprendre ne bouge pas.

(a)



(b)

Figure 5.24 : Le volume reconstruit pour le robot Tom (a) et Jerry (b).

Comme dans le cas de l'auto-apprentissage, l'objectif final de l'apprentissage est d'obtenir le volume de représentation des Cartes Dynamiques tel que décrit dans le chapitre 3. Une fois le volume caractéristique du robot est obtenu, il est alors simple d'obtenir la Carte Dynamique du robot (voir chapitre 3). L'apprentissage se fait en trois étapes : 1) Les mouvements du robot dont on cherche à obtenir la Carte Dynamique sont acquis par l'intermédiaire du système visuel (section 5.3.1) 2) Une approximation des trajectoires est utilisée pour extraire les données différentielles nécessaire à la génération du volume (section 5.3.2) 3) On génère le volume qui correspond au mieux à l'ensemble des données acquises (section 5.3.3).

5.3.1 Acquisition des mouvements du robot

Cette section est dédiée essentiellement à l'étude du problème d'acquisition des déplacements du robot dont on cherche à apprendre la Carte Dynamique à partir de la caméra embarquée d'un autre robot. Nous avons décomposé ce problème en deux tâches : tout d'abord, il est important d'obtenir une technique fiable d'extraction et par la suite de procéder à l'extraction en temps réel des données.

Technique d'extraction des données : Nous nous sommes heurtés dès le début au problème d'instabilité numérique pour reconstruction tridimensionnelle à partir d'une seule caméra. Notre technique initiale était de calibrer la caméra embarquée du robot par rapport au sol en utilisant une grille posée à terre. La figure 5.25 présente un exemple de grille de calibration utilisée. Les positions des cercles sont extraites dans l'image puis à partir de la connaissance des positions réelles dans le plan de ces cercles, l'algorithme de calibration de Tsai-Wilson (Willson 1994) est utilisé pour obtenir les données de calibration de la caméra. La hauteur des lampes que nous désirons détecter étant connue, il est alors simple d'obtenir la position du robot adverse dans le plan, en utilisant la position image de la lampe, sa hauteur et

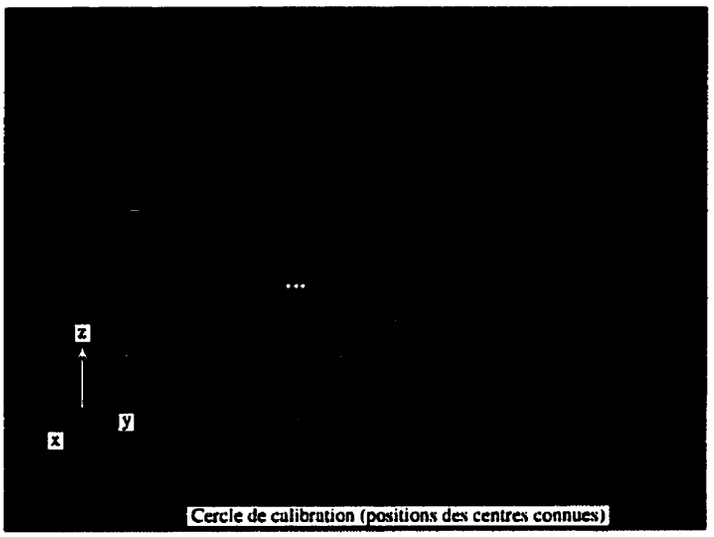


Figure 5.25 : La grille de calibration utilisée pour la caméra embarquée du robot

la calibration inverse de la caméra[1].

L'instabilité numérique de cette approche a été cependant trop importante pour obtenir un quelconque résultat significatif sur le mouvement du robot que l'on observe.

Afin de pallier ce problème, nous avons décidé d'utiliser l'odomètre central (voir chapitre 6) et qui permet d'obtenir la position (x, y) d'un robot dans l'espace de travail défini par la caméra. Ainsi, le robot observé est déplacé dans l'espace de travail et chaque position (x, y) dans cet espace est extraite précisément à partir de l'odomètre. Pour cette même position (x, y) dans l'espace de travail, la position image (x_i, y_i) aux lampes est extraite de l'image acquise par la caméra embarquée du robot. La figure 5.26 illustre ce processus d'appariement. Quand un nombre suffisant de couples (x, y, x_i, y_i) est obtenu l'algorithme de Tsai-Wilson est de nouveau utilisé pour obtenir les données de calibrage de la caméra. Cette technique a le double avantage d'utiliser une calibration "en situation" de la caméra, à partir des positions réelles du robot, et également de permettre une vérification des résultats obtenus grâce aux données fournies par l'odomètre. L'ensemble des expériences sont faites dans ces conditions.

[1] La calibration inverse fournit la position (x, y) d'un point dans le repère de la grille de calibration à partir de la hauteur z de ce point et sa position (x_i, y_i) dans le plan image

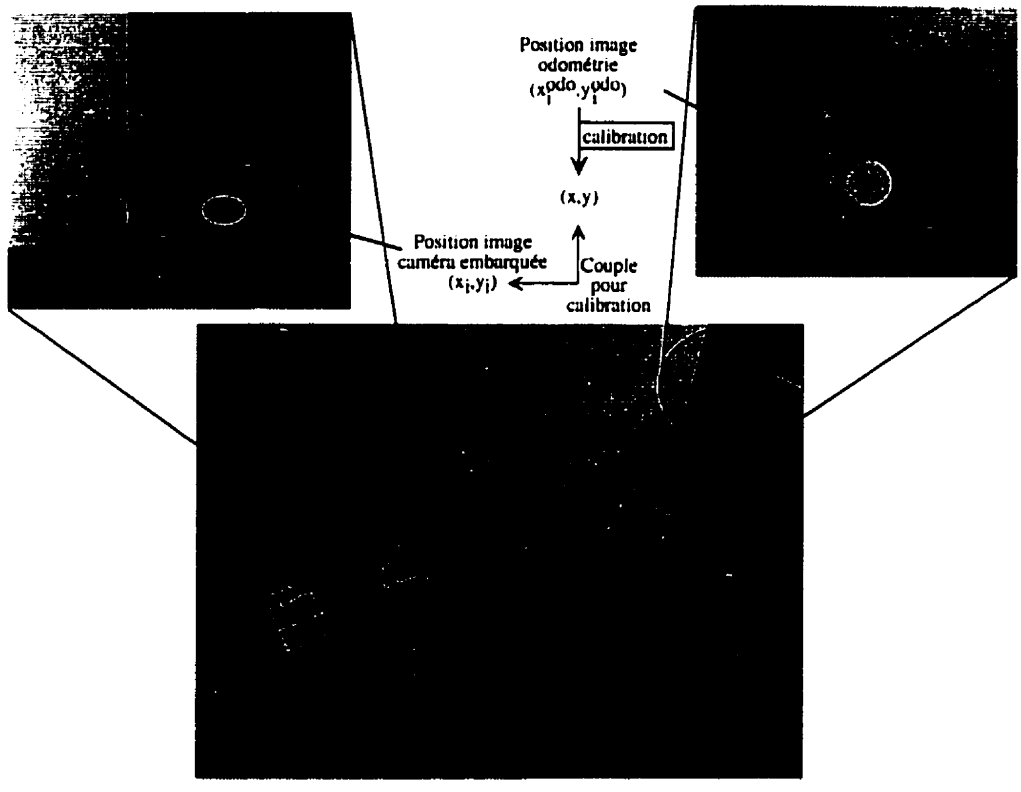
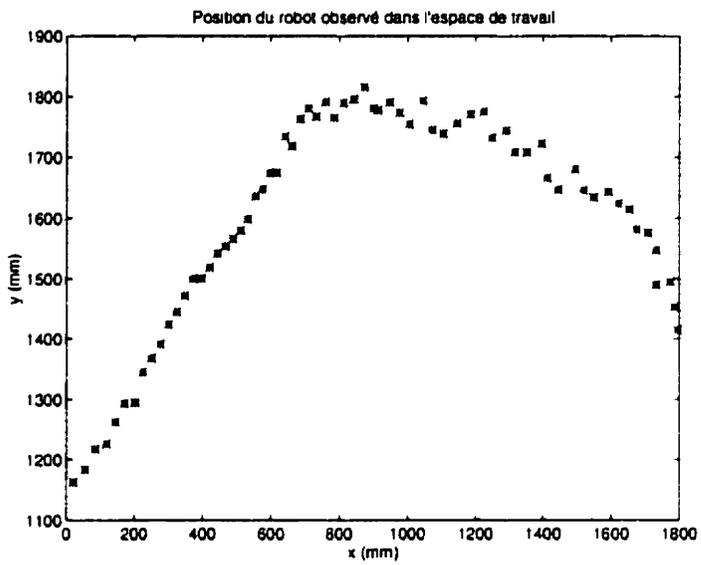


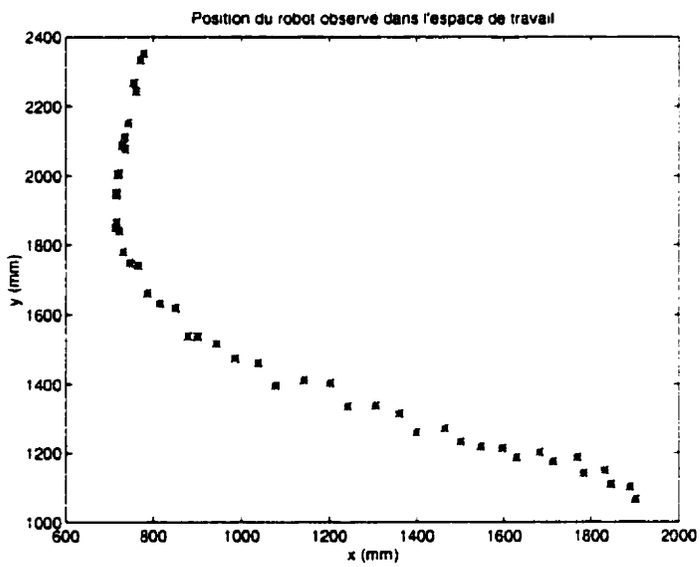
Figure 5.26 : Processus de calibration "en situation" : correspondance entre le point (x, y) dans l'espace de travail (odométrie) et le point image (x_i, y_i) de la caméra du robot embarquée.

avec une calibration préalable de la caméra du robot.

Acquisition des trajectoires du robot en mouvement : Une fois que la calibration de la caméra a été réalisée par l'intermédiaire de la technique décrite précédemment, il est relativement simple d'acquérir les positions du robot en mouvement. Pour cela, nous avons utilisé un algorithme similaire à celui développé pour l'odomètre central (voir chapitre 6) pour obtenir la position du robot dans l'espace de travail. Le filtrage de Kalman n'est cependant pas nécessaire dans ce cas. Le traitement est fait à 15 images/seconde. Chaque position dans l'image (x_i, y_i) est immédiatement convertie en une position (x, y) dans l'espace de travail grâce à la calibration de la caméra. La figure 5.27 présente deux exemples de trajectoires acquises. Ces données sont alors disponibles pour la suite de l'apprentissage : l'extraction des données différentielles



(a)



(b)

Figure 5.27 : Deux trajectoires représentant le mouvement du robot observé

des trajectoires.

Nous devons faire remarquer que les mouvements du robot que l'on observe étaient dirigés par un opérateur humain à l'aide de la télécommande. Nous verrons que

cela a une influence sur les résultats. Nos ressources étant limitées, les mouvements générés par l'intermédiaire de la télécommande ont fortement diminué le lourd travail d'acquisition des nombreuses données nécessaire à l'apprentissage.

5.3.2 Approximation des trajectoires

Dans de cas de l'auto-apprentissage des Cartes Dynamiques, nous avions besoin d'extraire des données de la trajectoire, tel que la courbure (afin de choisir la tranche du volume à représenter) et l'orientation du véhicule (afin de pouvoir aligner la trajectoire suivant la même orientation initiale). Lors de cet auto-apprentissage, les données sont fournies par le filtrage de Kalman des positions successives des robots dans l'espace de travail. A nouveau, nous allons devoir obtenir ces données différentielles des trajectoires du robot. Il n'est cependant pas possible d'utiliser le filtrage de Kalman puisque, par exemple, les contrôles envoyés au robot que l'on observe ne sont pas disponibles. De plus, comme on peut le voir dans la figure 5.27, les trajectoires obtenues ne sont pas suffisamment lisses pour que des opérations de différentiation puissent être utiliser sans provoquer une instabilité numérique (la courbure est en effet une composition de dérivée du deuxième ordre de la trajectoire).

L'approximation des courbes obtenues est une intéressante alternative à ces calculs. Les fonctions servant d'approximation peuvent en effet être dérivées de façon explicite, ce qui permet de réduire les effets d'instabilité numérique du cas discret. De plus, cela permettra également de lisser les données et d'obtenir ainsi des trajectoires qui seront moins sujettes au bruit introduit lors de la procédure d'acquisition des données.

L'approximation des trajectoires est faite en trois étapes que nous allons détailler dans la suite de cette section : la première étape consiste à transformer les données dans un espace plus adéquat à la procédure d'optimisation. Lors de la deuxième étape, des fonctions polynomiales sont utilisées pour approximer les données dans leur nouvel espace. Finalement, les données différentielles sont extraites à partir des

approximations polynomiales des données extraites.

Transformation des données : Il est en général difficile de faire l'approximation d'une courbe quelconque dans l'espace 2D. Nous avons choisi de procéder à une paramétrisation des données que nous avons obtenu afin de permettre une approximation dans un espace à une dimension. Ainsi, chaque ensemble de points $(X_i, Y_i), i \in \{1, \ldots, m\}$ obtenus à l'étape précédente est transformée en deux couples de points $(s, X(s))$ et $(s, Y(s))$ avec :

$$D_m = \sum_{j=1}^{m} \sqrt{(X_{j-1} - X_j)^2 + (Y_{j-1} - Y_j)^2} \tag{5.6}$$

$$s(i) = \sum_{j=1}^{i} \frac{\sqrt{(X_{j-1} - X_j)^2 + (Y_{j-1} - Y_j)^2}}{D_m} \tag{5.7}$$

$$X(s) = X_i \tag{5.8}$$

$$Y(s) = Y_i \tag{5.9}$$

Les deux nouveaux ensembles de données ainsi obtenues vont permettre de faire une approximation de type $f(x') = y'$ puisque la transformation a permis d'obtenir une suite monotone croissante de valeurs de s.

Approximation par des fonctions polynomiales : L'objectif est désormais de trouver la fonction polynomiale $y = f(x)$ qui est au sens des moindres carrées la plus proche des couples de données $(s, X(s))$ ou $(s, Y(s))$. Nous avons utilisé pour faire ces approximations les outils fournis avec le logiciel de simulation mathématique *Mathematica*. Une des fonctions permet ainsi de faire l'approximation avec un polynôme de degré n d'un ensemble de couple de points (x', y'). Un avantage de cette fonction est de permettre une stabilité numérique des polynômes même lorsque le degré du polynôme est élevé. On obtient ainsi des fonctions d'approximation du type :

$$X(t) = \sum_{k=0}^{n} a_k t^k \tag{5.10}$$

$$Y(t)=\sum_{k=0}^{n} b_k t^k \tag{5.11}$$

$$t\in[0;1] \tag{5.12}$$

Expérimentalement nous avons déterminé que la valeur $n = 6$ était un bon compromis entre stabilité et précision. Les figures 5.28 et 5.29 présentent deux exemples d'approximation sur des ensembles de données $(s, X(s))$ ou $(s, Y(s))$ et les trajectoires résultats $(X(t), Y(t))$ avec $t \in [0; 1]$.

Calcul des données différentielles : Comme nous l'avons déjà mentionné, afin de procéder à l'apprentissage des Cartes Dynamiques il est nécessaire d'obtenir les valeurs de la courbure κ (kappa) ainsi que l'orientation du robot θ (theta).

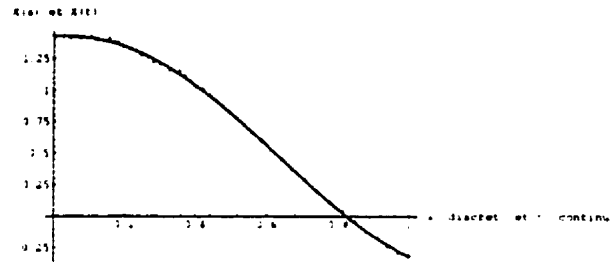
La formule générale pour obtenir l'orientation d'une courbe paramétrique de type $(X(t), Y(t))$ est :

$$\theta(t)=\text{atan}\left(\frac{\frac{dY}{dt}(t)}{\frac{dX}{dt}(t)}\right) \tag{5.13}$$

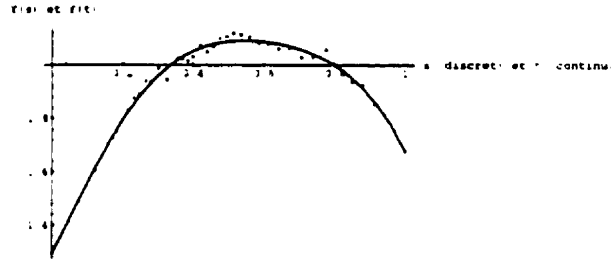
La courbure (qui correspond à la variation de l'angle θ avec le temps) est donnée par la formule :

$$\kappa(t)=\frac{\frac{dX}{dt}(t)\cdot\frac{d^2Y}{dt^2}(t)-\frac{dY}{dt}(t)\cdot\frac{d^2X}{dt^2}(t)}{\left[\left(\frac{dX}{dt}\right)^2(t)+\left(\frac{dY}{dt}\right)^2(t)\right]^{\frac{3}{2}}} \tag{5.14}$$

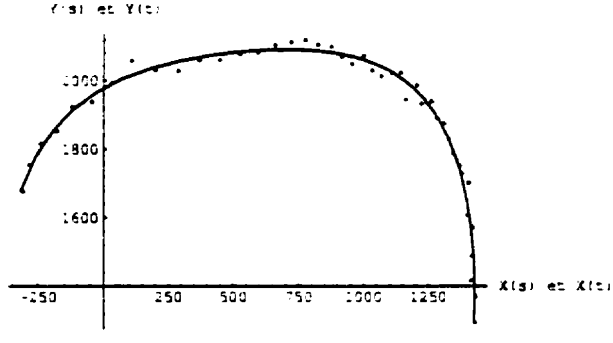
Le logiciel de simulation *Mathematica* nous a ainsi permis de générer les expressions de θ et κ pour les listes de données. La figure 5.30 présente les valeurs de κ et θ pour les deux trajectoires décrites dans les figures 5.28 et 5.29.



(a)

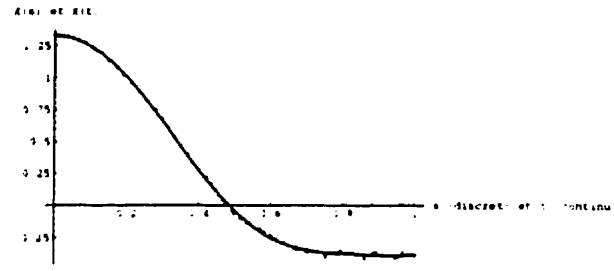


(b)

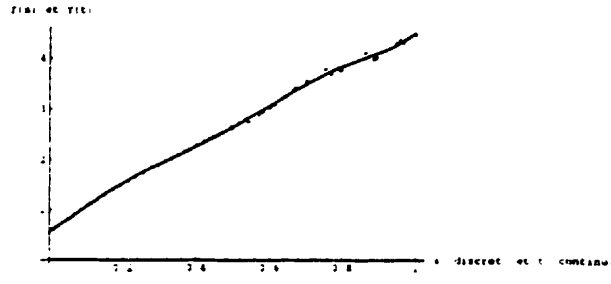


(c)

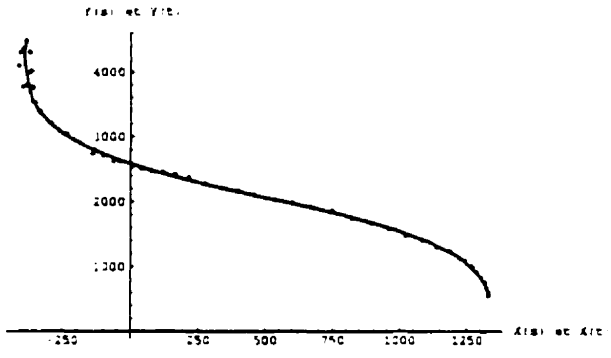
Figure 5.28 : (a) Approximation des données $(s, X(s))$ par $X(t), t \in [0; 1]$ (b) Approximation des données $(s, Y(s))$ par $Y(t), t \in [0; 1]$ (c) Trajectoire $(X(t), Y(t)), t \in [0; 1]$ correspondante dans l'espace (X, Y)



(a)

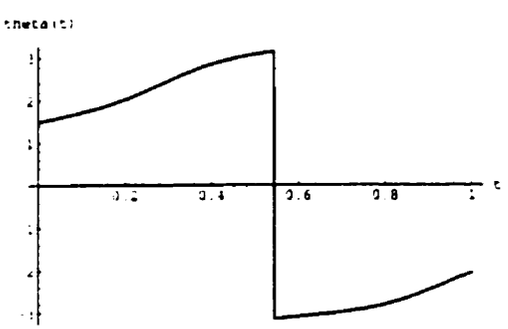


(b)

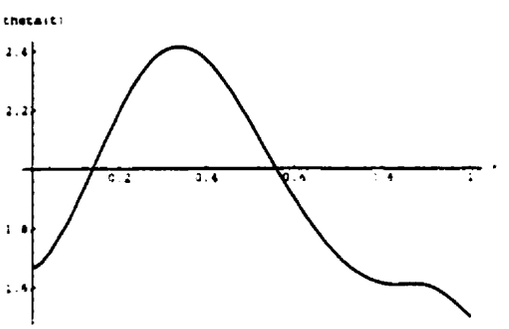


(c)

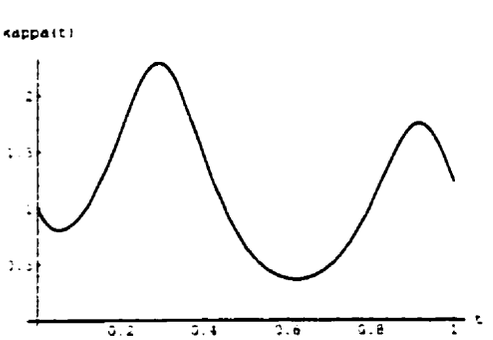
Figure 5.29 : (a) Approximation des données $(s, X(s))$ par $X(t). t \in [0; 1]$ (b) Approximation des données $(s, Y(s))$ par $Y(t), t \in [0: 1]$ (c) Trajectoire $(X(t). Y(t)). t \in [0: 1]$ correspondante dans l'espace (X, Y)



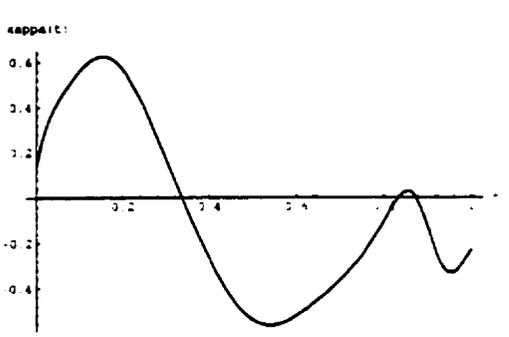
(a) θ (figure 5.28)



(b) θ (figure 5.29)



(c) κ (figure 5.28)



(d) κ (figure 5.29)

Figure 5.30 : (a),(b) orientation (θ) et (c),(d) courbure (κ) pour les deux trajectoires des figures 5.28 et 5.29

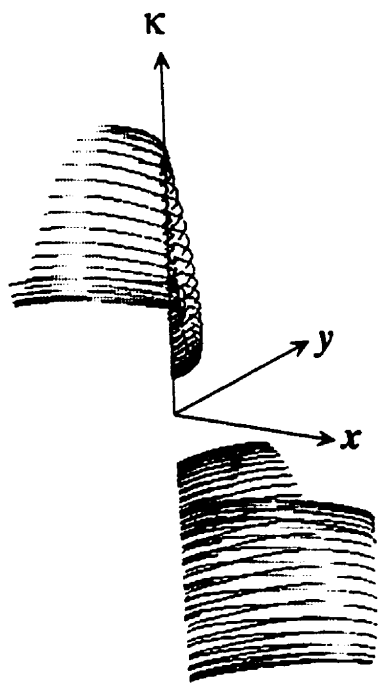
5.3.3 Recherche du volume de représentation des Cartes Dynamiques

Au niveau de cette étape, l'objectif est de pouvoir trouver le volume de représentation de la Carte Dynamique qui correspond aux données qui ont été acquises par la caméra du robot. Nous allons procéder en deux nouvelles étapes : 1) les trajectoires et leurs données différentielles sont transformées dans l'espace (x, y, κ) que nous avons mentionné dans le chapitre 3 ; 2) la recherche du volume est faite dans cette espace de façon itérative.

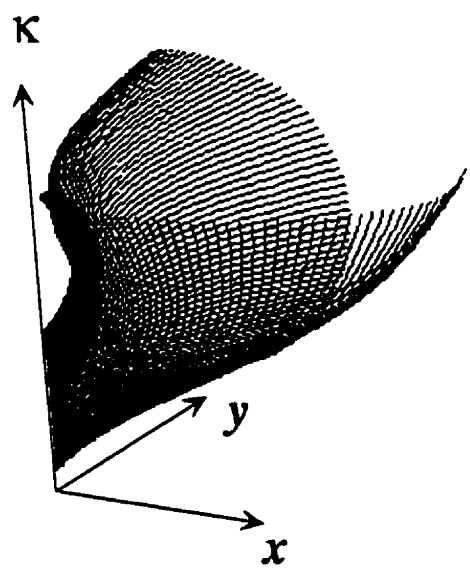
Transformation des trajectoires dans l'espace (x, y, κ) : Afin de transformer la trajectoire en une surface dans l'espace (x, y, κ), nous avons utilisé la technique décrite dans la section 5.2 afin de générer le volume de représentation des véhicules du banc d'essai expérimental. Les équations 5.5 permettent ainsi de transformer les valeurs de positions $x(t), y(t)$ et les valeurs différentielles $\theta(t)$ et κ dans l'espace (x, y, κ). Finalement les valeurs contiguës sont liées afin de former une surface à partir de la trajectoire extraite des senseurs.

La figure 5.31 présente deux exemples de trajectoires qui sont transformées en des surfaces dans l'espace (x, y, κ). Cette transformation permet ainsi de limiter la croissance des volumes de représentation lors de la phase suivante de recherche. La surface devient donc une surface de contrainte à l'intérieur de laquelle le volume de représentation doit se trouver. Nous allons exploiter cette propriété afin de trouver le volume de représentation.

Recherche du volume dans (x, y, κ) : Pour trouver le volume de représentation, nous allons générer un volume paramétré par la vitesse de changement de la courbure $\dot{\kappa}_{max}$. La valeur de la courbure maximum κ_{max} du chemin est extraite de l'ensemble des données différentielles calculées sur les trajectoires. Initialement la valeur de $\dot{\kappa}_{max}$ est infinie, ce qui permet de passer instantanément de $-\kappa_{max}$ à $+\kappa_{max}$. Le volume



(a)



(b)

Figure 5.31 : (a),(b) Surfaces obtenues à partir des deux trajectoires des figures 5.28 et 5.29

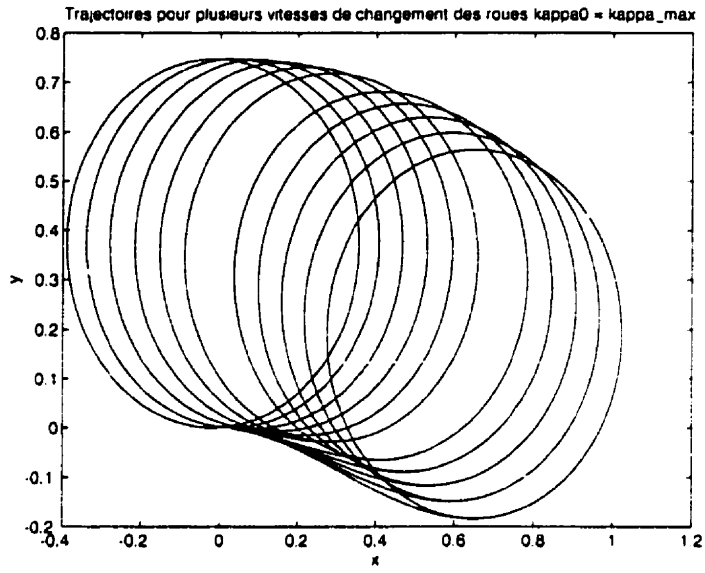


Figure 5.32 : Plusieurs trajectoires correspondantes à un ensemble de valeurs de $\dot{\kappa}_{max}$ avec $\kappa_0 = \kappa_{max}$

formé est alors deux cylindres rectilignes de rayon $1/\kappa_{max}$.

La valeur de $\dot{\kappa}_{max}$ est alors diminuée petit à petit. Le volume croit ainsi à l'intérieur de la surface de contrainte obtenue à l'étape précédente. Dès que le volume de représentation généré dépasse la frontière de la surface de contrainte, la procédure de recherche est terminée et le volume ainsi obtenu est utilisé pour représenter les capacités dynamiques du robot observé.

La figure 5.32 présente plusieurs trajectoires correspondant à un ensemble de valeurs de $\dot{\kappa}_{max}$. Dans la figure 5.33, le volume correspondant à une valeur de $\dot{\kappa}_{max}$ est comparé à l'ensemble des trajectoires constituant la surface de contrainte. Finalement la figure 5.34 donne la forme du volume de représentation, ainsi que celle d'une trajectoire transformée dans l'espace (x, y, κ).

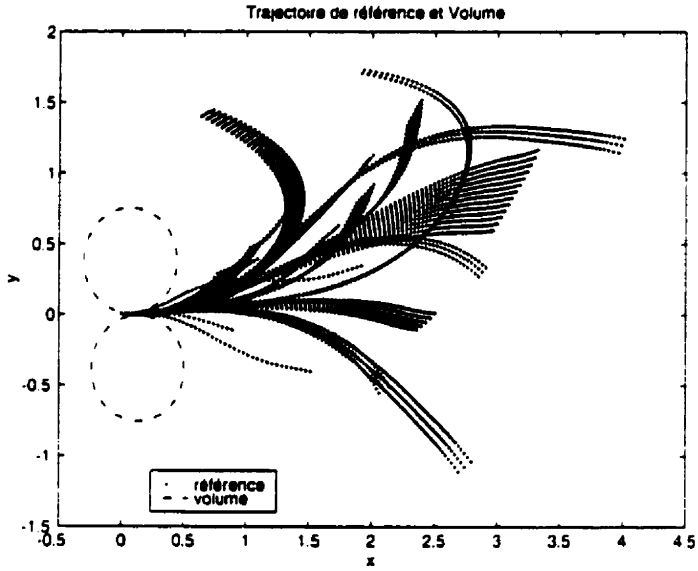


Figure 5.33 : Le volume correspondant à une valeur de $\dot{\kappa}_{max}$ est comparée à l'ensemble des trajectoires constituant la surface de contrainte.

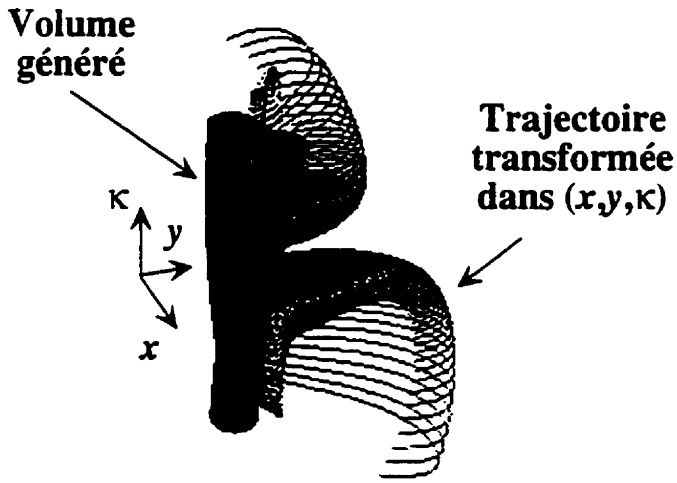


Figure 5.34 : Forme du volume de représentation appris avec une trajectoire transformée dans l'espace (x, y, κ)

5.3.4 Conclusion sur l'apprentissage des Cartes Dynamiques d'un autre robot

Comme on peut le voir dans la figure 5.34, la forme du volume de représentation est différente de celle obtenue lors de l'auto-apprentissage des Cartes Dynamiques. En effet, le volume s'étend plus suivant l'axe κ et les cercles correspondant aux valeurs de κ_{max} sont plus petits. L'explication de cette différence est très simple : nous avons utilisé directement la télécommande dans le deuxième cas d'apprentissage. Nous avons en effet pensé qu'il était préférable d'utiliser directement les télécommandes afin de générer l'ensemble des trajectoires du robot qui est observé pour permettre l'apprentissage de ces Cartes. Le nombre d'expériences à réaliser étant important, cela a grandement facilité l'acquisition des données. Ceci a eu cependant pour conséquence de changer les caractéristiques dynamiques du robot de deux façons : 1/ le taux de changement des roues n'est pas contraint par la conversion numérique-analogique, ainsi que le transfert des commandes depuis la station de travail vers le contrôleur de bas niveau du robot 2/ l'angle de braquage maximal a été contraint de façon logicielle, ce qui n'est pas le cas lors de l'utilisation de la télécommande.

On serait tenté de dire qu'il s'agit ici de problèmes pour la réalisation d'un système expérimental puisque cela démontre qu'un système ne peut être considéré comme une somme de morceaux attachés, mais qu'il est nécessaire de le considérer dans son ensemble. Ceci met en avant à nouveau l'atout fondamental des Cartes Dynamiques de pouvoir appréhender les capacités intrinsèques d'un robot qui est composé d'une station de travail, d'un convertisseur digital/analogique, d'une liaison sans fil, *etc.* La modélisation de chaque élément afin d'obtenir le modèle du robot serait un travail sans fin. La Carte Dynamique décrit finalement le résultat : son mouvement dans l'espace de travail en fonction de ses capacités.

Chapitre 6

Banc d'essai expérimental

Ce chapitre présente en détail le banc d'expérimentation développé dans notre laboratoire pour l'étude de problème avec plusieurs robots (Zanardi, Hervé et Cohen 1997*a*). Nous donnons d'abord un aperçu global du système, avec une description de son architecture fonctionnelle. Une description des robots et de leur système d'odométrie simulé est donnée par la suite. Le problème de l'étalonnage des différents robots du système sera décrit dans l'annexe D. Le protocole de communication entre stations de travail nécessaire pour l'architecture distribuée de notre système est ensuite décrit. Par la suite, un bref aperçu du système de vision active des robots est présenté. Ce chapitre finit sur la description de l'implantation sur nos robots du système de planification réactive basé sur les Cartes Dynamiques dans le cadre d'une tâche de poursuite/évasion.

6.1 Idée globale du système

Le banc d'essai expérimental se compose, pour l'instant, de deux robots mobiles construits à partir de véhicules de loisir radiocommandés (figure 6.1). Ces véhicules manquent malheureusement de codeurs de position. Un système basé sur une caméra calibrée en situation et un filtrage de Kalman opérant sur les informations de position

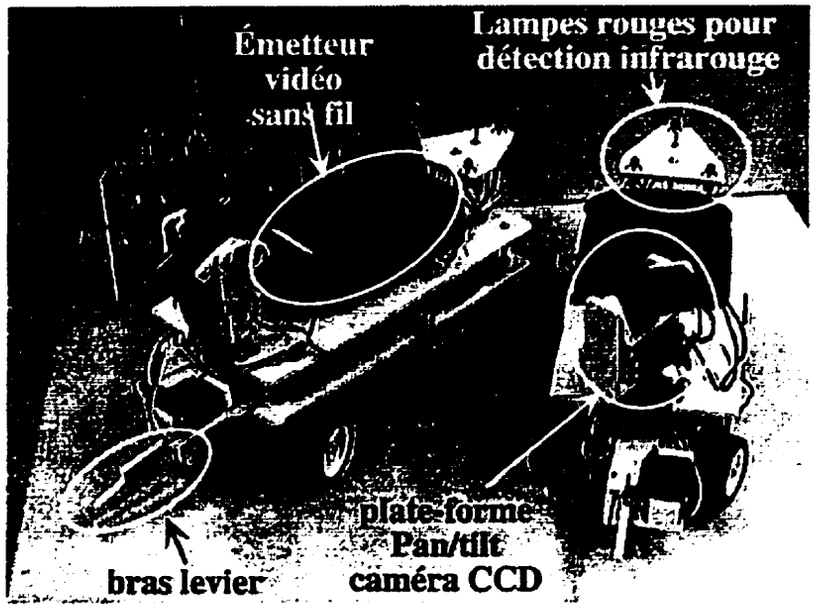


Figure 6.1 : Les deux premiers robots du système expérimental

a été créé afin de simuler l'odométrie interne du robot et fournir ainsi des valeurs de retour dans les boucles de contrôle visuel.

Le système d'estimation d'état (un exemple d'un système d'estimation d'états par observation est donnée dans (Ishikawa et Sampei 1995)) et les caméras sur les robots forment un système visuel distribué qui nous à amenés à utiliser un ensemble de stations de travail en réseau pour répartir la charge de calcul. La figure 6.2 présente une vue d'ensemble du banc d'expérimentation.

Architecture fonctionnelle Les différents blocs de la figure 6.2 décrivent les éléments fonctionnels du projet. Chaque véhicule est contrôlable par l'intermédiaire d'une interface graphique (voir figure 6.3) qui permet également l'affichage du signal vidéo provenant du robot (bloc 1). L'interface permet ainsi à un utilisateur d'entrer des commandes et d'observer les états courants des différents robots. L'ensemble des commandes fournies par l'opérateur sont ensuite transmisses à une boucle de contrôle (bloc 2).

Le système d'odométrie (bloc 4) fournit en retour une position qui est utilisée

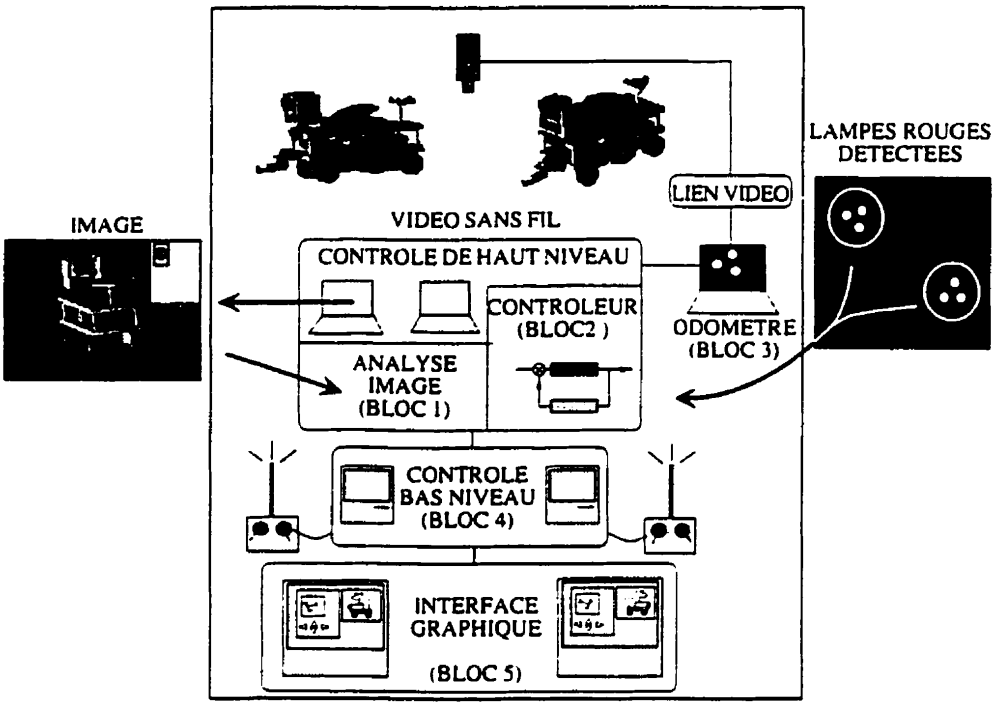


Figure 6.2 : Vue d'ensemble du banc d'essai expérimental

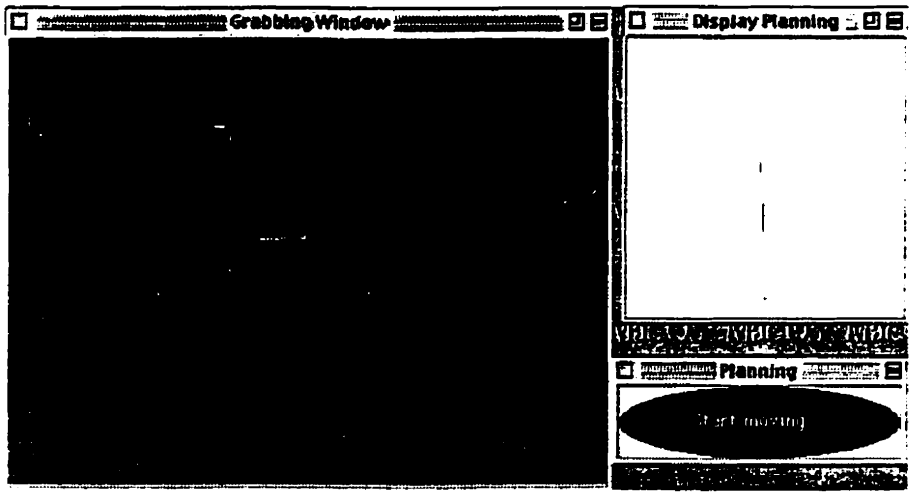


Figure 6.3 : Interface graphique pour le contrôle des robots

dans la boucle de contrôle. Les valeurs de contrôle générées sont envoyés au port série d'une station de travail afin d'être transmisses au véhicule par l'intermédiaire de la télécommande (bloc 3).

Ce type d'architecture s'est montrée très efficace pour des problèmes typiques de coordination de robots en Intelligence Artificielle (Sahota 1994). L'information de

position fournie par l'odométrie simulée peut être par exemple utilisée pour manœuvrer les véhicules afin d'éviter qu'ils n'entrent en collision durant l'exécution de leurs tâches respectives, ou pour transporter de concert une longue barre droite (Lacroix, Polotski, Cohen et Hervé 1997). Ce genre d'architecture est également bien adapté à l'étude de planification de trajectoires — centralisée ou non — en présence d'incertitude, puisque le système central permet d'obtenir des mesures précises sur l'état courant du robot, et le degré d'incertitude ainsi fourni au module de planification peut alors être ajusté.

Il est également possible de développer des stratégies de coordination décentralisées pour de multiples robots mobiles autonomes équipés d'un système visuel. Dans ce cas, l'odométrie simulée ne fournit à chaque robot qu'une estimation de son propre vecteur d'état. Les informations sur les autres robots et l'environnement sont obtenues par le système de vision autonome monté sur le robot. Le bloc 5 sert ainsi à regrouper les informations sur l'environnement et sur les autres robots à partir de la détection de mouvement indépendant et/ou d'obstacles.

Les stations de travail concernées sont : deux Silicon Graphics (type Indy et Indigo 2) pour l'affichage de l'interface graphique et l'envoi de commandes (bloc 1), 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, et finalement deux Macintosh IIci qui sont utilisés pour fournir les valeurs de tension directement aux télécommandes des véhicules (Bloc 3).

6.2 Description des robots mobiles

Nos robots sont basés sur des véhicules radiocommandés miniatures disponible dans des centres de modélisme. La grande variété de fabricants et le coût modique de ces véhicules ont motivés leur achat. Le comportement dynamique d'un robot est également facilement modifiable par le changement d'un moteur ou le remplacement

de la structure de base. Les véhicules ont été modifiés afin de limiter leur vitesse maximale à 1 m/s et leur accélération à environ 0.5 m/s^2.

Les télécommandes permettant de contrôler les servo-moteurs possèdent quatre voies. Deux voies sont utilisées 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.

Afin de permettre aux robots de soulever une barre en collaboration, un petit bras a été rajouté à l'avant du véhicule. Le contrôle du bras est substitué au contrôle de l'inclinaison de la caméra. Le schéma technique de la partie supérieure des robots est présenté à la figure 6.4.

Pour réduire la zone morte des moteurs et ainsi avoir un meilleur contrôle sur la vitesse, un contrôleur de vitesse a été rajouté.

Le principal désavantage de ce genre de véhicule provient de l'absence d'odométrie interne. En effet, pour utiliser directement les valeurs de retour sur la tension des moteurs il est nécessaire de transmettre l'information en sens inverse vers la télécommande, ce qui n'est malheureusement pas possible.

Pour pallier à ce défaut, nous utilisons un observateur central (caméra CCD muni d'un filtre infrarouge) pour détecter la présence de lampes rouges montées sur le robot et fournir les données d'odométrie. De plus amples détails sur le système d'odométrie simulée sont donnés dans la section 6.3.

Contrôle des robots à partir d'une station de travail. Chaque voie de la télécommande correspond à un potentiomètre dont la valeur oscille entre 0 et 5 volts. Il est élémentaire de simuler ce genre de contrôle et d'éliminer ainsi le contrôle transmis par les potentiomètres.

Les tensions sont envoyées *via* un générateur de tension contrôlable par l'intermédiaire d'un langage de commande rudimentaire et d'un port d'accès série ayant un protocole RS232.

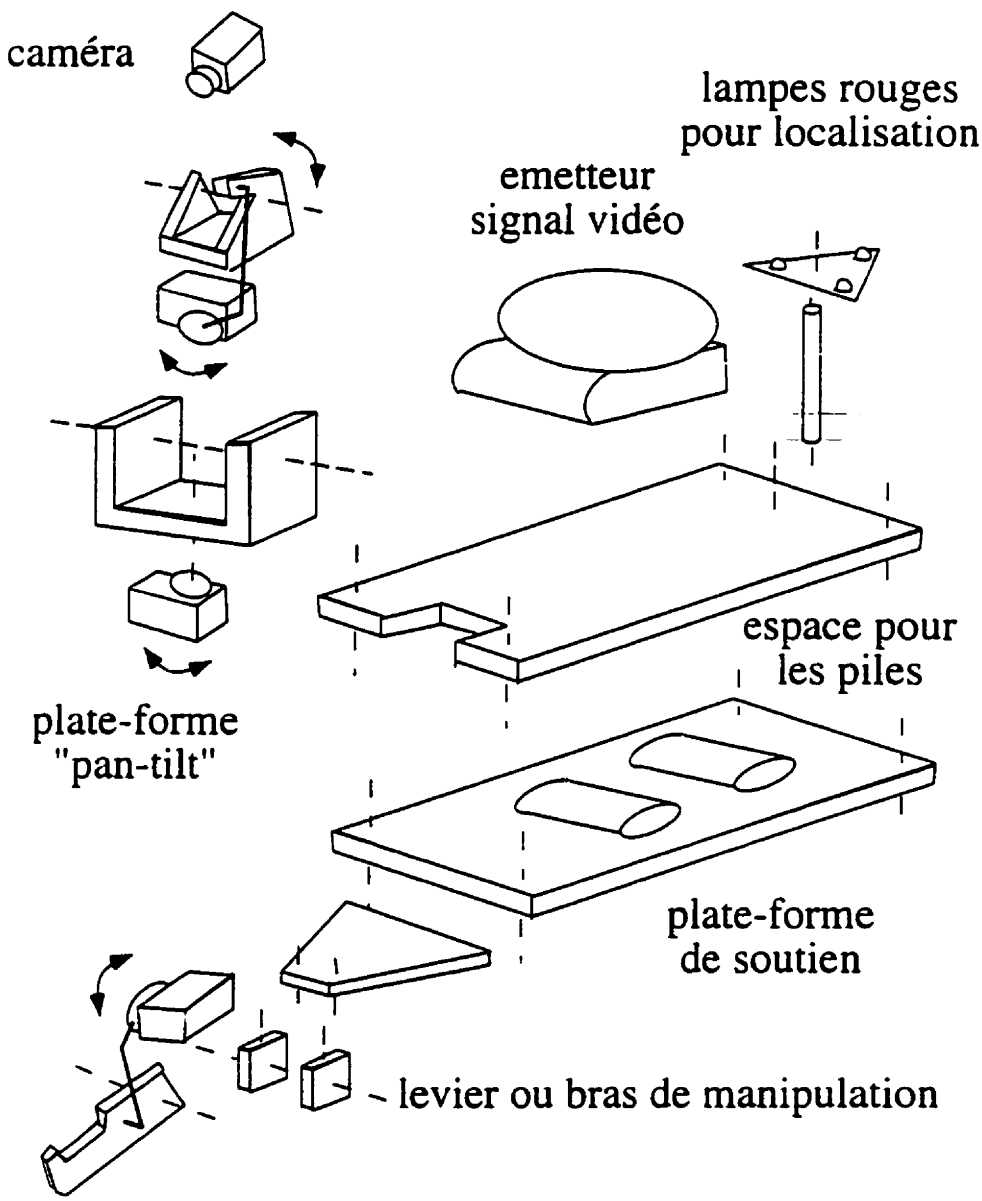


Figure 6.4 : Schéma technique de la partie supérieure des robots

Par ailleurs, les valeurs des tensions, qui sont appliquées à la télécommande sont stockées dans des registres dont l'évolution est seulement contrôlée par une nouvelle commande. Une tension reste donc active sur la télécommande tant qu'une nouvelle valeur n'est pas venue la remplacer.

Transmission du signal vidéo en provenance des caméras des robots. La place sur le robot étant limitée, le traitement des images provenant de la caméra ne peut se faire directement sur le robot. Il est alors nécessaire de transmettre le signal NTSC en sortie des caméras vers les stations de travail sans utiliser de cable vidéo.

Afin de ne pas provoquer d'interférence entre les signaux vidéo provenant des deux voitures, nous avons choisi deux systèmes sans fil opérant dans des gammes de fréquence séparées.

L'alimentation des transmetteurs (ainsi que des caméras et des lampes) se fait par l'intermédiaire de batteries qui sont placées entre la base du robot et la partie supérieure où se trouve les lampes et la caméra (figure 6.4).

6.3 Système d'odométrie simulée

Comme nous l'avons déjà mentionné, les véhicules constituant la base de nos robots ne sont pas équipés d'un système d'odométrie interne. Des lampes rouges sont disposées sur le dessus de la voiture afin d'être facilement repérables par une caméra équipée d'un filtre infrarouge. Le vecteur d'état $\mathbf{x}$ du système pour le filtre de Kalman étendu diffère légèrement de celui du modèle cinématique que nous avons utilisé jusqu'à présent. Le vecteur d'état du système est :

$$\mathbf{x}(k)=(X_k,\ Y_k,\ \theta_k,\ V_k,\ \kappa_k)^T, \tag{6.1}$$

avec k l'index de temps ($t_k = k\Delta T$), X_k et Y_k les coordonnées du véhicule dans le repère de travail, θ_k et V_k l'orientation et la vitesse du véhicule, et $\kappa_k = \tan\phi_k/L$ la

courbure instantanée de la trajectoire (voir figure 6.5). L'équation d'état du véhicule

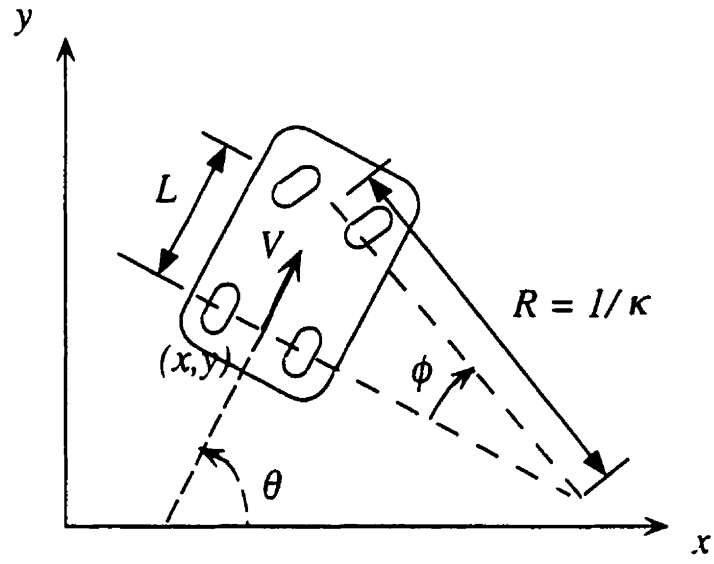


Figure 6.5 : Paramètres du modèle cinématique simple d'un robot de type voiture

est alors :

$$\dot{\mathbf{x}}(k) = \begin{pmatrix} \dot{X}_k \\ \dot{Y}_k \\ \dot{\theta}_k \\ \dot{V}_k \\ \dot{\kappa}_k \end{pmatrix} = \begin{pmatrix} V_k \cos(\theta_k) \\ V_k \sin(\theta_k) \\ V_k \kappa \\ \gamma_y \\ \left(\frac{1}{L} + L\kappa_k^2\right)\omega \end{pmatrix}. \tag{6.2}$$

pour que :

$$\mathbf{x}(k+1) = \mathbf{x}(k) + \dot{\mathbf{x}}(k)\Delta T + \zeta_k$$

$$
\begin{aligned}
&= \begin{pmatrix} X_k + V_k \cos(\theta_k)\Delta T \\ Y_k + V_k \sin(\theta_k)\Delta T \\ \theta_k + V_k \kappa \Delta T \\ V_k + \gamma_k \Delta T \\ \kappa_k + \left(\frac{1}{L} + L\kappa_k^2\right)\omega \Delta T \end{pmatrix} + \zeta_k \\
&= \mathbf{f}(\mathbf{x})(k) + \zeta_k,
\end{aligned}
$$

avec γ l'accélération tangentielle du véhicule et ω la vitesse angulaire des roues directrices, les deux paramètres de contrôle des robots. Le terme ζ correspond à un bruit blanc gaussien non corrélé de dimension 5 qui indique l'erreur de déviation entre le modèle et la réalité (modèle non exact, glissement des roues, *etc.*).

Le filtre de Kalman étendu utilise une forme linéarisée de cette équation, $\mathbf{x}(k+1) = \mathbf{A}_k \cdots \mathbf{x}(k) + \mathbf{u}_k + \zeta$, obtenue par :

$$
\mathbf{A}_k = \frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}}(k) = \begin{pmatrix} 1 & 0 & -V_k \sin\theta_k \Delta T \cos\theta_k \Delta T & 0 \\ 0 & 1 & V_k \cos\theta_k \Delta T & \sin\theta_k \Delta T & 0 \\ 0 & 0 & 1 & \kappa_k \Delta T & V_k \Delta T \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 + 2L\kappa_k \omega \Delta T \end{pmatrix}.
$$

avec

$$
\begin{aligned}
\mathbf{u}_k &= \mathbf{f}(\mathbf{x})(k) - \mathbf{A}_k \cdots \mathbf{x}_k \\
&= \begin{pmatrix} V_k \theta_k \sin\theta_k \\ -V_k \theta_k \cos\theta_k \\ \kappa_k V_k \\ \gamma_t \\ \frac{1}{L}(1 - \kappa_k^2 L^2)\omega \end{pmatrix}.
\end{aligned}
$$

Afin de connaître les paramètres intrinsèques et extrinsèques de la caméra (nécessaire pour relier les coordonnées images aux coordonnées tridimensionnelles), nous avons utilisé l'algorithme de calibrage de Tsai-Wilson (Willson 1994) dont le code source était disponible.

L'étalonnage de la caméra est effectué en disposant des cibles sur le sol suivant une disposition connue, et en extrayant précisément les positions dans l'image de ces cibles. Le logiciel d'étalonnage fournit alors les paramètres qui permettent de relier une position dans l'image à une position dans l'espace de travail et inversement (la hauteur du point considéré doit être connue car la projection perspective affaisse la dimension de l'espace).

La localisation des lumières sur la voiture est facilitée à la fois par le filtre infra-rouge sur l'objectif de la caméra (qui les fait ainsi apparaître comme des points blancs sur un fond noir), mais également par la sortie du filtre de Kalman qui va indiquer les positions prédites des lumières dans l'image.

L'ensemble du processus d'odométrie se passe donc de la façon suivante : pour une position initiale X_0, Y_0, θ_0, à chaque instant k, une prédiction de l'état (et en particulier de sa position) du robot pour l'instant suivant, $\tilde{\mathbf{x}}_{k+1}$, est calculée, et va permettre de guider le repérage des lampes dans l'image. En effet, en reprojetant cette position dans le plan image (grâce aux paramètres intrinsèques et extrinsèques de la caméra), on obtient une position image dans laquelle devraient se trouver les lampes du robot. Un carré de détection variable est alors utilisé pour faire la recherche des points blancs constituant les lampes. Par la suite, à partir de la position et de l'orientation donnée par la disposition des lampes dans l'image, $\hat{X}_k, \hat{Y}_k, \hat{\theta}_k$, une correction est appliquée pour obtenir $\hat{\mathbf{x}}_k$ qui constitue la mesure d'odométrie simulée (voir en annexe A pour plus de précision sur le filtrage de Kalman).

Pour obtenir la position et l'orientation du robot à partir des mesures des positions des N lampes du robot $(\hat{x}_i, \hat{y}_i), i \in \{1, \ldots, \mathcal{N}\}$, et des positions de référence $(\bar{x}_i, \bar{y}_i), i \in$

$\{1,\ldots,\mathcal{N}\}$, nous utilisons la formulation de projection inverse suivante[1] :

$$
\begin{aligned}
S_{\hat{X}} &= \sum_{i=1}^{\mathcal{N}} \hat{x}_i \ , \ S_{\hat{Y}} = \sum_{i=1}^{\mathcal{N}} \hat{y}_i \\
S_{\bar{X}} &= \sum_{i=1}^{\mathcal{N}} \bar{x}_i \ , \ S_{\bar{Y}} = \sum_{i=1}^{\mathcal{N}} \bar{y}_i \\
P_{\hat{X}\bar{Y}-\hat{Y}\bar{X}} &= \sum_{i=1}^{\mathcal{N}} \hat{x}_i\bar{y}_i - \hat{y}_i\bar{x}_i \\
P_{\hat{X}\bar{X}+\hat{Y}\bar{Y}} &= \sum_{i=1}^{\mathcal{N}} \hat{x}_i\bar{x}_i + \hat{y}_i\bar{y}_i \\
A &= \frac{S_{\hat{X}}S_{\bar{Y}} - S_{\hat{Y}}S_{\bar{X}}}{\mathcal{N}^2} - \frac{P_{\hat{X}\bar{Y}-\hat{Y}\bar{X}}}{\mathcal{N}} \\
B &= \frac{S_{\hat{X}}S_{\bar{X}} + S_{\hat{Y}}S_{\bar{Y}}}{\mathcal{N}^2} - \frac{P_{\hat{X}\bar{X}+\hat{Y}\bar{Y}}}{\mathcal{N}} \\
ratio &= \frac{1}{\sqrt{A*A + B*B}} \\
\cos(\theta) &= -B * ratio \\
\sin(\theta) &= A * ratio,
\end{aligned}
$$

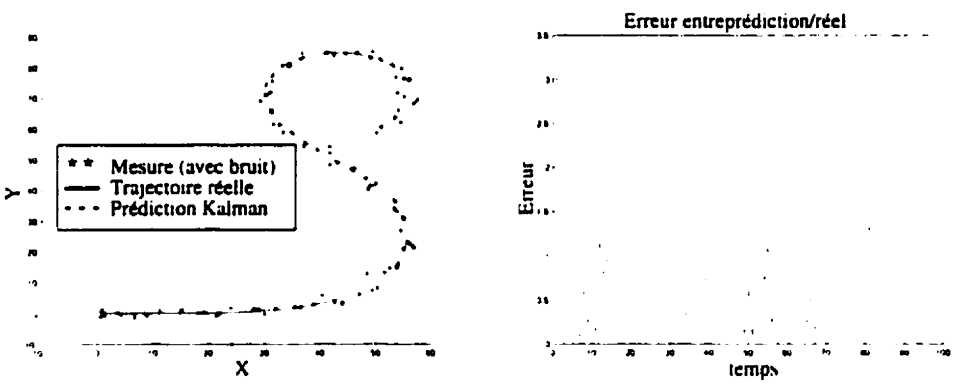
ce qui permet d'obtenir la position et l'orientation par les équations :

$$
\begin{aligned}
x &= \frac{(S_{\hat{X}} - \cos(\theta) * S_{\bar{X}} + \sin(\theta) * S_{\bar{Y}})}{\mathcal{N}} \\
y &= \frac{(S_{\hat{Y}} - \sin(\theta) * S_{\bar{X}} - \cos(\theta) * S_{\bar{Y}})}{\mathcal{N}} \\
\theta &= \text{atan2}(\sin(\theta), \cos(\theta)).
\end{aligned}
$$

Le filtre de Kalman a été conçu de manière à accélérer le calcul de la prédiction et de la correction. Ces améliorations sont basées sur des techniques présentées dans (Chui et Chen 1991) et sont données en annexe A.

La figure 6.6 présente un exemple de simulation de trajectoire estimée par le filtre

[1] $S_{\hat{X}}, S_{\hat{Y}}, S_{\bar{X}}$ et $S_{\bar{Y}}$ répresentent les sommes des coordonnées pour les mesures (désignées par le chapeau) et les références (désignées par la barre) ; $P_{\hat{X}\bar{Y}-\hat{Y}\bar{X}}$ et $P_{\hat{X}\bar{X}+\hat{Y}\bar{Y}}$ répresentent les produits scalaire et vectoriel des vecteurs de mesure et de référence



(a) (b)

Figure 6.6 : Une trajectoire du robot avec les positions estimées par le filtrage de Kalman (a) et l'erreur résiduelle du processus (b).

de Kalman. Il est ainsi possible de voir que l'erreur de positionnement diminue au fur et à mesure que le système procéde à la suite des prédiction et correction malgré les fortes erreurs de positionnement.

6.4 Protocole de communication entre les stations de travail

Comme nous l'avons suggéré dans la première section, plusieurs ordinateurs, de différents types, sont nécessaires pour pouvoir analyser de façon efficace les informations provenant du système de vision, ou de l'odométrie. L'ensemble des stations de travail étant connectées sur un réseau de type *Ethernet*, un protocole de communication entre les machines pour transmettre les informations pertinentes est nécessaire.

Ce protocole consiste tout d'abord en un langage de haut niveau qui permet de coder les informations nécessaires, comme par exemple la position courante du robot ou les commandes à appliquer aux actionneurs du robot, et d'un protocole de communication bas niveau basé sur des *sockets Ethernet*. Nous avons choisi d'utiliser

des sockets de type *stream* qui ont l'avantage d'avoir un protocole sécuritaire de transmission incluant le renvoie des message perdus et la garantie de séquencement correct des messages (le dernier message envoyé ne peut arriver avant le message précédente).

Le langage de haut niveau transmet une instruction par commande à envoyer aux robots. Chaque instruction est formée de trois parties : la partie préfixe indique le type de commande (s'il s'agit d'une commande d'accélération des roues, la valeur l'orientation des roues, des valeurs directes de voltage à appliquer, orientation du pan/tilt *etc.*) avec un dictionnaire associé connu. La seconde partie de l'instruction indique la valeur numérique associée au type de commande (voltage, accélération en m/s, *etc.*). Finalement, la durée d'application de la commande (en millisecondes) constitue la dernière partie de l'instruction.

Il existe toutefois quelques exceptions : par exemple, la commande d'arrêt d'urgence "Stop" ne nécessite pas de valeur numérique explicite puisqu'il s'agit d'une remise à zéro immédiate des valeurs de voltage envoyées au robot et d'une élimination des commandes toujours en attente. Comme le système expérimental doit servir aussi à faire de la replanification dynamique (au sens donnée dans (Feddema et Mitchell 1989)) il doit être possible d'écraser les commandes précédemment envoyées. Certaines instructions permettent ainsi d'indiquer que pour un certain intervalle de temps, il faut extraire les valeurs des files d'attente.

En ce qui concerne la transmission des valeurs d'odométrie, le format étant toujours le même (5 valeurs à virgule flottante), l'instruction est formée de ces cinq valeurs à la suite.

Finalement, la valeur des commandes à transmettre à l'odomètre (qui sont toujours les valeurs de l'accélération tangentielle de la voiture et la vitesse de rotation de l'angle des roues avants) sont transmisses directement.

La figure 6.7 présente une vue d'ensemble des communications dans le système final. Puisque la valeur de la tension à appliquer pour changer la vitesse du robot

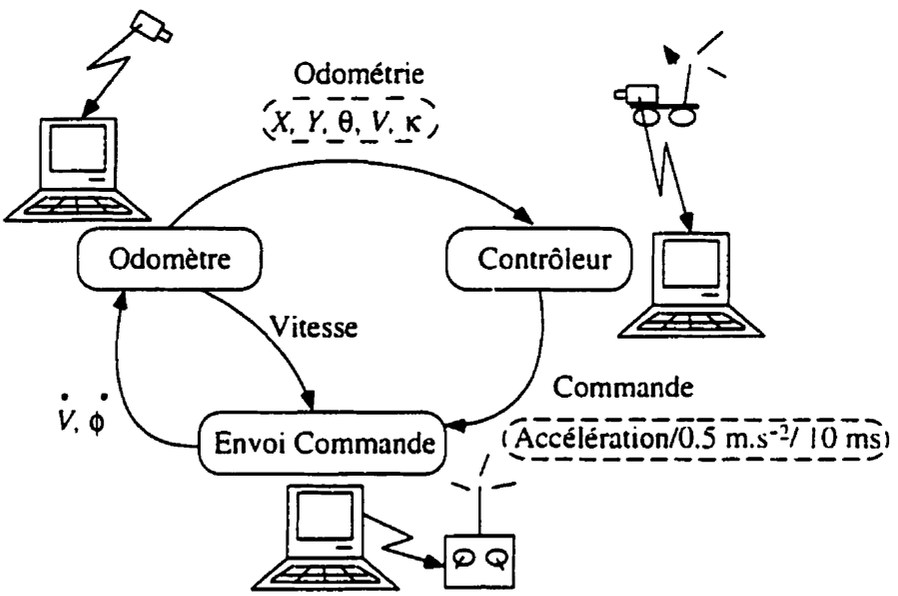


Figure 6.7 : Les communications entre les machines

dépend à la fois de la vitesse courante du robot et de l'accélération appliquée. il est nécessaire d'obtenir de la part de l'odomètre la vitesse actuelle du robot.

Les instructions de commande générées par le contrôleur sont mises dans des files d'attente à l'envoi et à la réception. Une fois les commandes reçues et placées dans les files d'attente correspondant à un servo-moteur (roues. vitesse. pan. tilt. bras). la première instruction de chaque file est traduite périodiquement en voltage. par l'intermédiaire de tables de conversion. et est ensuite envoyée au port série en correspondance avec le servo-moteur. La commande alors effectivement envoyée est transmise à l'odomètre pour être intégrée dans le cycle de prédiction et de correction et fournir les valeurs d'odométrie au contrôleur.

Toutefois. comme le montre le diagramme de la figure 6.8. le délai obligatoire entre réception et traitement oblige à effectuer une correction sur la vitesse courante du robot en fonction de la valeur fournie par l'odomètre.

Communication par Sockets. Afin d'utiliser un protocole de communication entre des machines de différents types et architectures. nous avons utilisé un système

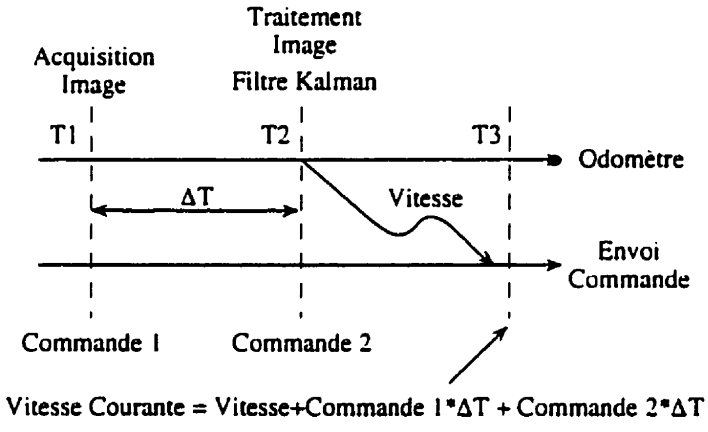


Figure 6.8 : Compensation du temps de traitement des images pour la vitesse

de communication bas-niveau basé sur les sockets *Ethernet*.

Sur les ordinateurs de type Macintosh, une libraire de fonction GUSI permet de simuler les fonctionnalités usuelles des sockets. Lors de l'établissement d'un protocole de socket entre deux ordinateurs (pour une communication point à point), il est nécessaire de désigner un serveur et un client.

Le serveur va créer en premier lieu le descripteur de socket (identique à un descripteur de fichier en UNIX). La socket ainsi créée, est liée à l'adresse IP du serveur (par exemple `peridot.ai.polymtl.ca`) pour attendre d'éventuelles connections de client. Un serveur peut recevoir plusieurs connections de clients différents. Il suffit de prévoir le même nombre de réception aux serveurs que le nombre de clients prévus.

Le client, pour sa part, crée de la même façon le descripteur de socket, mais ensuite il se connecte à l'adresse IP spécifique au serveur. La communication se fait alors dans les deux sens par l'intermédiaire de fonctions d'écriture et de lecture de chaînes de caractères. Un avantage notable des sockets réside dans le fait qu'un message n'est jamais écrasé par un autre message s'il n'a pas déjà été lu.

Dans le cas de notre système, l'odomètre simulé sert de serveur, sur lequel se connectent les contrôleurs (pour recevoir les valeurs d'odométrie) et les machines générant les tensions (pour recevoir la vitesse et envoyer les commandes). Ces dernières font

également office de serveurs où les contrôleurs se connectent afin de transmettre les séquences de commandes à l'aide du langage de commande de haut-niveau (voir figure 6.7).

6.5 Système de vision active

Dans cette section, nous allons décrire l'équipement sensoriel des robots.

Tout d'abord, il a été décidé de ne pas faire le traitement d'images directement à bord des voitures, mais de transmettre plutôt le signal NTSC des caméras jusqu'à une station de travail pour y être analysé. Cette approche au problème d'analyse d'images pour des robots autonomes a été également envisagée dans (Inaba, Kagami, Ishikawa, Kanehiro, Takeda et Inoue 1994).

Un problème d'interférence entre les signaux des caméras se pose cependant avec deux robots dans l'aire de jeux. Les systèmes de sécurité à distance offrent plusieurs solutions à ce genre de problème, mais à des prix prohibitifs. Nous avons opté pour des systèmes grand public dans deux gammes de fréquences afin d'éviter les interférences. Le premier système Recoton opère dans la bande de fréquence 440-450MHz, alors que le système Wavecom utilise les fréquences autour de 1400MHz.

Le modèle Recoton nécessite toutefois un système de réception pour traduire le signal NTSC provenant des caméras, avant de le digitaliser. L'appareil Wavecom fournit un signal vidéo de type NTSC utilisable directement.

Le signal transmis peut alors être digitalisé par une station de travail, pour pouvoir être subséquemment analysé. Les Macintosh 7500 ont une capacité de digitalisation intégrée à cadence vidéo de 30 images par seconde. La carte de digitalisation fonctionne de façon asynchrone et en mode d'accès mémoire direct. Une image peut donc être digitalisée alors que le CPU analyse les images précédemment acquises. En utilisant plusieurs tampons, il est donc possible d'utiliser des algorithmes simples de détection de mouvement indépendant.

La figure 6.9 présente une vue d'ensemble du processus de digitalisation avec le système d'analyse d'image.

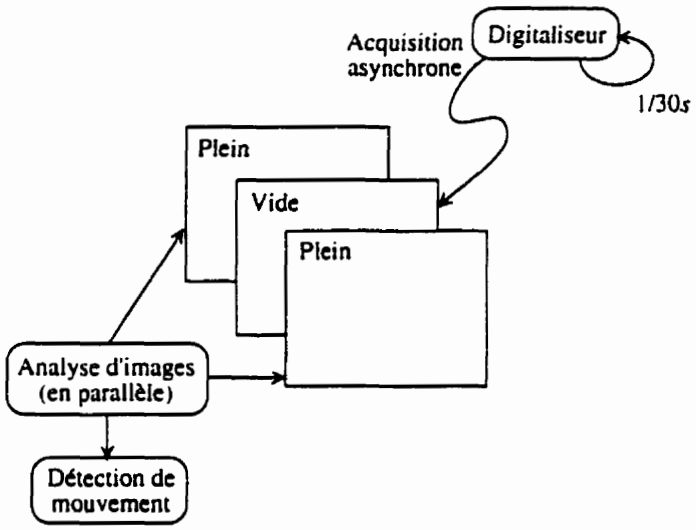


Figure 6.9 : L'acquisition et l'analyse d'images

Le tilt et pan de la caméra permettent également l'utilisation d'algorithmes de suivi de cible actif ou de perception active usuels.

6.6 Implantation du système de planification basée sur les Cartes Dynamiques sur le banc d'essai expérimental

6.6.1 Introduction et généralités

Dans l'ensemble de cette thèse, nous avons indiqué à plusieurs reprises qu'un avantage essentiel du concept de Carte Dynamique était de permettre une planification réactive prenant en compte les capacités dynamiques des agents (ou robots) en présence. Cette partie de la thèse est dédiée à l'application de ces principes à notre banc d'essai expérimental décrit dans les sections précédentes. Il est évident que nous

avons du faire certains choix en fonction des possibilités matérielles. Nous allons au fur et à mesure de cette section décrire ces choix et leur impact sur les conditions expérimentales.

Le premier choix a porté sur le nombre de robots que nous avons pu employer. Alors qu'en simulation, il était possible d'augmenter le nombre de robots, en expérimentation le nombre a été limité à 2. Ce nombre restreint de robots ne permet pas de refléter pleinement l'avantage de la Carte Dynamique par rapport à d'autres méthodes de planification lorsque le nombre de robots augmente.

Toutefois, l'implantation complète du système de planification à partir des Cartes Dynamiques permet de montrer que le premier pas (le plus dur) peut être fait et que cette technique peut être utilisée dans des situations plus complexes dans lesquelles le matériel nécessaire est disponible ouvrant ainsi la voie à d'autres travaux.

Un autre choix difficile à faire était de savoir à quel point la vision embarquée pouvait être utilisée. De nouveau un problème technique se posait : nous avions besoin de trois unités de calcul capables de faire de l'acquisition et traitement (l'une devant calculer l'odométrie, les deux autres l'acquisition des images des caméras embarquées sur les robots), alors que seulement deux unités étaient disponibles dans le cadre de ce projet. Comme nous allons le voir, nous avons implanté deux scénarios avec dans un premier temps aucune vision embarquée (et donc seulement l'odomètre central) et dans un deuxième temps, l'utilisation de la vision embarquée pour le robot poursuivant (qui n'utilise pas l'algorithme de planification basé sur les Cartes Dynamiques).

Avant de passer aux détails de l'implantation qui seront donnés dans les sections 6.6.1 et 6.6.3, nous allons décrire de façon globale les deux scénarios que nous avons envisagé pour nos expériences.

La figure 6.10 présente un schéma global de l'expérimentation dans le cadre du premier scénario. Dans le cadre de ce scénario, nous nous sommes placés dans la même situation que lors des simulations, c'est-à-dire que tous les robots ont accès à toutes les informations (comme position/orientation relative des robots). Le premier

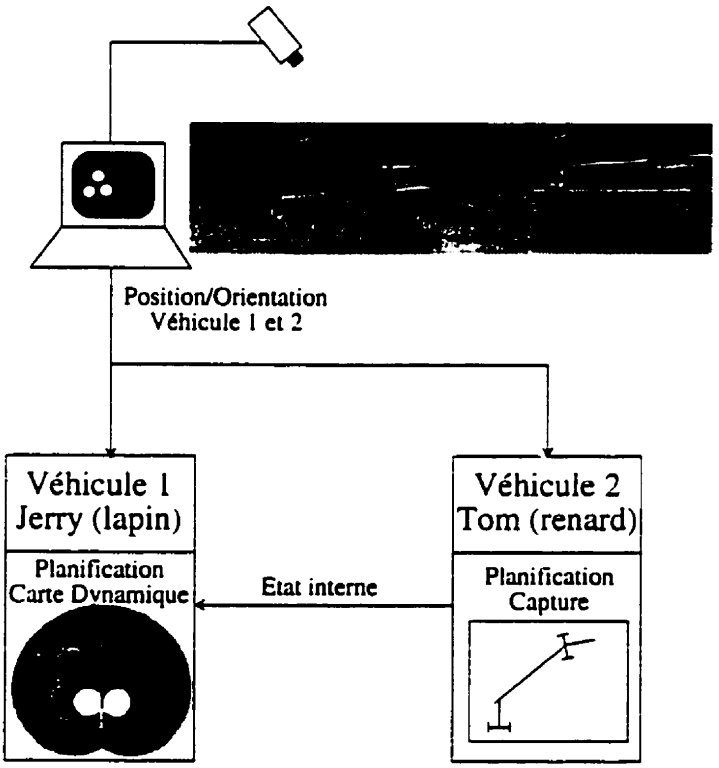


Figure 6.10 : Premier scénario des expériences de planification à partir de la Carte Dynamique

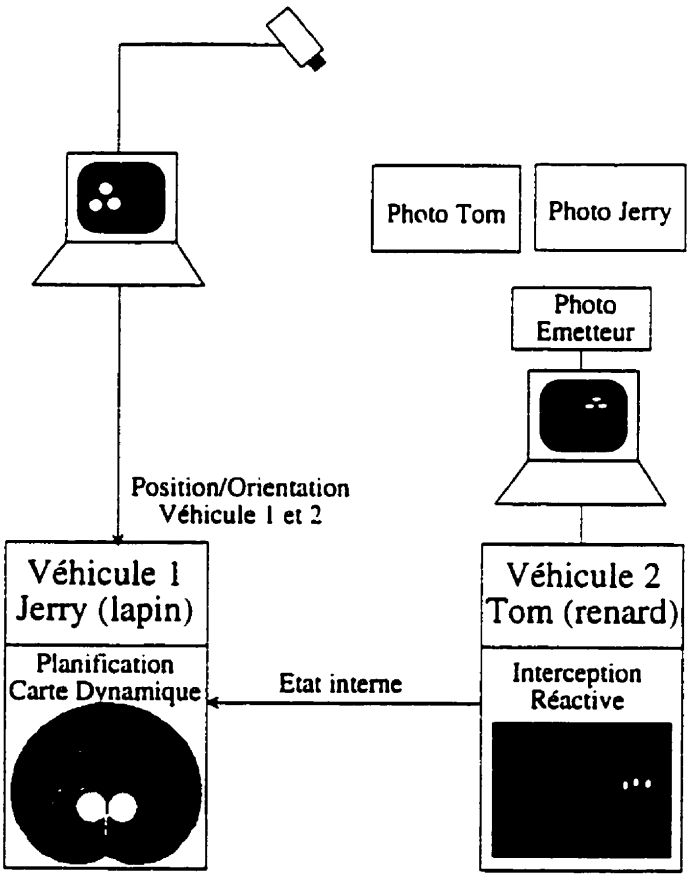


Figure 6.11 : Second scénario utilisant la vision embarquée du robot pour intercepter de façon réactive

robot (Jerry, le poursuivi) utilise donc les informations de position/orientation qui lui sont fournies par l'odomètre central, ainsi que les données sur l'état interne du robot qui le poursuit (Tom, le poursuivant) afin de générer la Carte Dynamique de la situation et planifier la trajectoire qu'il va suivre pour les quelques instants qui vont suivre. Le second robot utilise les mêmes informations de position/orientation relative pour planifier sa trajectoire d'interception.

Pour le scénario 2, nous avons utiliser la vision embarquée du robot poursuivant (Tom, le poursuivant) afin d'utiliser un algorithme de vision réactive pour générer les contrôles d'interception. La figure 6.11 résume le deuxième scénario proposé. Dans ce scénario, le poursuivant utilise son système de vision embarquée pour poursuivre le robot Jerry. La technique d'interception se basant sur la vision sera décrite plus loin.

Le plus sérieux problème de ce système est que le robot poursuivant (Tom) a un champ de vision réduit (celui de la caméra) et peut parfois perdre trace de l'autre robot. Le poursuivi (Jerry), a en effet un très important avantage sur son adversaire en ayant un champ de vision quasi illimité. Notre objectif est ici de montrer qu'une étape supplémentaire peut être franchie. L'étape suivante logique serait l'utilisation de la vision pour trouver les données de position/orientation relative entre les robots afin de planifier à l'aide des Cartes Dynamiques. Dans le chapitre 3 nous avons approché le problème de planification dans le plan image, il reste cependant un important travail de développement avant que l'on puisse directement utiliser les Cartes Dynamiques.

Nous allons maintenant détailler dans la section 6.6.2 l'implantation de la procédure de planification par les Cartes Dynamiques, ainsi que les sacrifices qui ont dû être faits. La section 6.6.3 s'attache à présenter la procédure de planification d'interception dans le cas des scénarios 1 et 2.

6.6.2 Planification à l'aide des Cartes Dynamiques en temps réel

D'après les résultats que nous avons obtenus dans le chapitre 4, il semble préférable d'utiliser la technique de planification utilisant les réseaux de clotoïdes. Nous avons aussi adopté la procédure de combinaison de l'équation 4.4. Le problème est désormais de planifier suffisamment vite afin de permettre au robot de souvent replanifier en fonction des changements de comportement du robot poursuivant. Dans le cadre des expériences que nous avons faites en simulation, le temps de calcul d'une Carte Dynamique combinée n'était pas important puisque les robots était fictifs. A plusieurs reprises, nous avons suggéré comment la combinaison des Cartes Dynamiques pouvait être faite en temps réel à l'aide de cartes possédant des accélérateurs graphiques pour la combinaison rapide de bitmaps. Nous n'avons malheureusement pas eu accès à ce genre de matériel pour l'instant et le défi était donc de faire les combinaisons au niveau logiciel en un temps le plus faible possible.

Pour cela, nous avons utilisé les propriétés de l'algorithme de planification par réseaux de courbes se basant sur les Cartes Dynamiques combinées. En effet, lorsque l'on observe plus en détail les courbes générées par cet algorithme, on remarque qu'elles couvrent en général une surface relativement faible de la Carte Dynamique combinée. De plus, de par leur construction, certaine courbes ont en commun plusieurs portions de la Carte Dynamique. Grâce à ces observations, nous avons pensé qu'il était alors préférable de ne procéder au calcul de la Carte Dynamique que dans les zones utilisées par l'algorithme de planification et en plus d'éviter que certains calculs soient faits en double. L'ensemble de ces améliorations nous ont permis de réduire le temps de calcul bien en dessous de 0.1 seconde, autorisant ainsi un système fonctionnant à 10 images par seconde.

Par la suite, le calcul des Cartes Dynamiques et le système de planification ont été intégrés à l'intérieur d'un système de repérage des véhicules dans l'espace de travail. Ce système est similaire à celui utilisé dans le cas de l'odomètre central du chapitre 6. Il permet de détecter la position et l'orientation de chaque véhicules et ainsi de pouvoir les transmettre au second robot.

Afin de permettre au système d'appréhender des changements significatifs dans l'environnement, la planification par les Cartes Dynamiques est faite 1 fois toutes les 5 unités de temps (dans nos expériences, nous avons utilisé une unité de temps de 0.1 s). Entre deux planifications, la courbe fournie par la procédure de planification est utilisée pour diriger la voiture en boucle ouverte (c'est également pour cette raison que l'intervalle entre deux planifications est gardé assez bas). Pendant cet intervalle, le système de détection traque les deux robots dans l'environnement et transmet par l'intermédiaire de *sockets* ces informations à l'autre robot (Tom, le poursuivant), qui en échange transmet son état interne.

La figure 6.12 présente le diagramme fonctionnel correspondant à la planification par les Cartes Dynamiques et le repérage des robots.

Nous allons maintenant présenter rapidement la technique d'interception du pour-

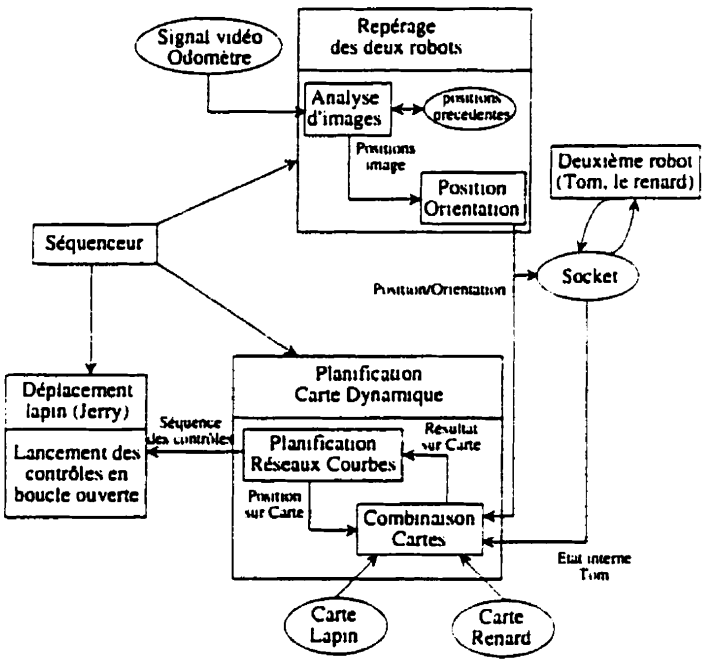


Figure 6.12 : Schéma fonctionnel de la planification à l'aide des Cartes Dynamiques et du repérage des robots

suivant.

6.6.3 Planification d'interception simple en temps-réel

Nous avons proposé deux simples algorithmes de planification d'interception : sans vision et avec la vision embarquée.

Le premier algorithme fonctionne sur un principe similaire au système de poursuite réalisé par la stagiaire Fabienne Lathuilière dans le cadre de son projet de fin d'étude (Lathuilière 1997). Ainsi dans son système, l'objectif était de se positionner directement en arrière de l'autre robot en orientant les roues à l'aide de simple relations géométriques. Dans le cas de notre système, nous allons simplement orienter les roues dans la direction du robot en envisageant le déplacement de Jerry en ligne droite. Ce type de planification est bien adaptée à notre situation, où la position réelle du poursuivi sera connue au fur et à mesure dans le temps, permettant ainsi de compenser pour le manque de connaissance sur son mouvement.

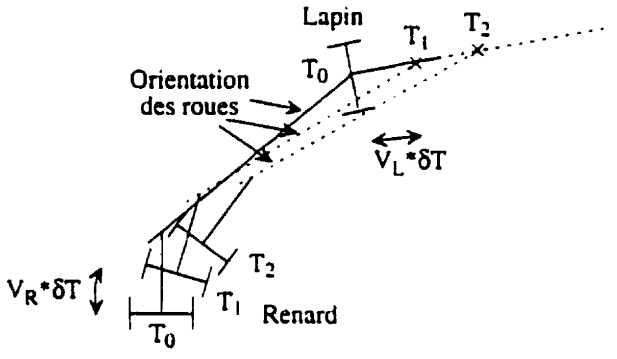


Figure 6.13 : L'angle des roues désiré au temps $T_0.T_1.T_2,\ldots$ est donnée par l'orientation de la droite entre la position des roues avants du poursuivant et les roues arrières de Jerry (poursuivi) en prenant en compte son déplacement en ligne droite.

Nous nous sommes pour l'instant intéressés seulement au point de vue du poursuivi, mais comme nous en discuterons rapidement dans la conclusion, l'introduction d'un système de poursuite basé sur les Cartes Dynamiques est une des ouvertures possibles de notre travail.

La figure 6.13 présente le calcul simple permettant d'obtenir l'angle des roues désiré.

Le principe du deuxième algorithme de planification de poursuite se base sur le principe de réactivité. En effet, l'orientation des roues du robot poursuivant est changée à chaque fois qu'une analyse d'images est faite, s'adaptant ainsi au changement de comportement de Jerry. Le changement est fait en fonction de la position dans l'image des lampes rouges du poursuivi. Ainsi, si les lampes se trouvent à droite (respectivement à gauche) dans l'image, l'angle des roues est graduellement diminué (respectivement augmenté). La force de ce système réside dans sa simplicité. Afin d'accélérer le traitement, un filtre infra-rouge est à nouveau utilisé. Par la suite, un simple contrôle linéaire de la forme $\phi = G.d$, où G est un gain constant et d la distance des lampes par rapport au centre de l'image. La figure 6.14 résume le principe de l'algorithme. Il est evident que ce contrôle pourrait être amélioré si la distance des lampes par rapport au robot était connue. Cependant, même si cela peut être fait, il s'agit d'un effort coûteux en temps (recalibration à chaque expérience) pour une

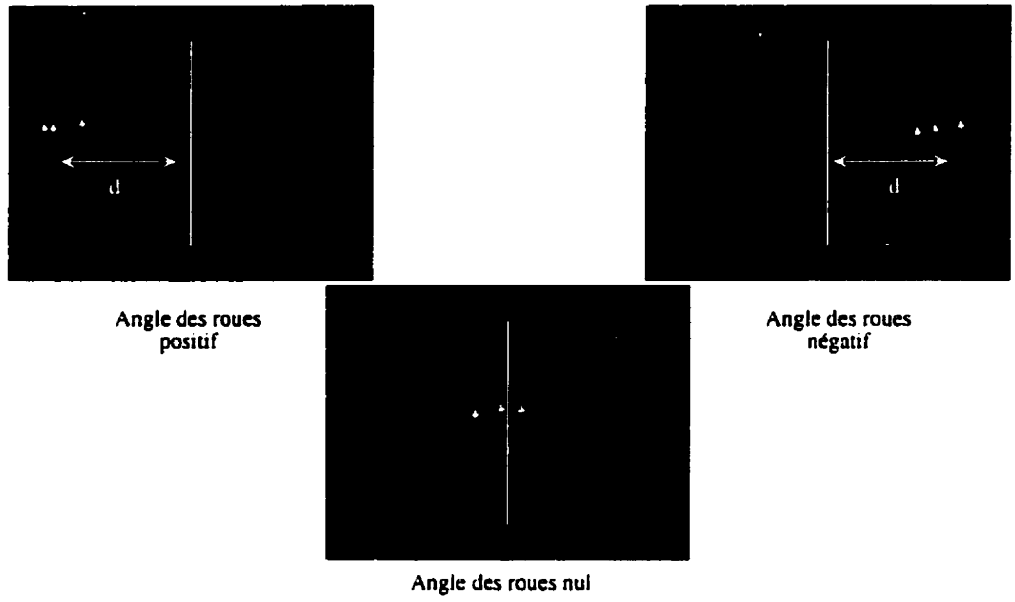


Figure 6.14 : Valeur de l'angle des roues en fonction de la position des lampes dans le plan image

amélioration qui serait à mon avis minime.

6.6.4 Résultats des expériences

Dans cette section, nous allons présenter les résultats obtenus avec le banc d'essai expérimental. Ces résultats concernent essentiellement le premier scénario car il permet d'obtenir les résultats les plus significatifs. Comme nous l'avons déjà mentionné le deuxième scénario est fait pour montrer que l'on peut franchir une étape supplémentaire dans l'implantation d'un système de planification basé sur les Cartes Dynamiques.

Les résultats sont présentés de la façon suivante : tout d'abord, un ensemble de trajectoires réalisées par les deux robots sera présenté avec leurs Cartes Dynamiques associées. Ensuite, nous allons nous intéresser à la stabilité de nos expériences en refaissant la même expérience plusieurs fois. En plus de refaire ces expériences. nous appliquerons une séquence en boucle ouverte au robot poursuivant pour observer la répétabilité de l'algorithme de planification basé sur les Cartes Dynamiques.

6.6.4.1 Exemples de trajectoires pour les deux robots

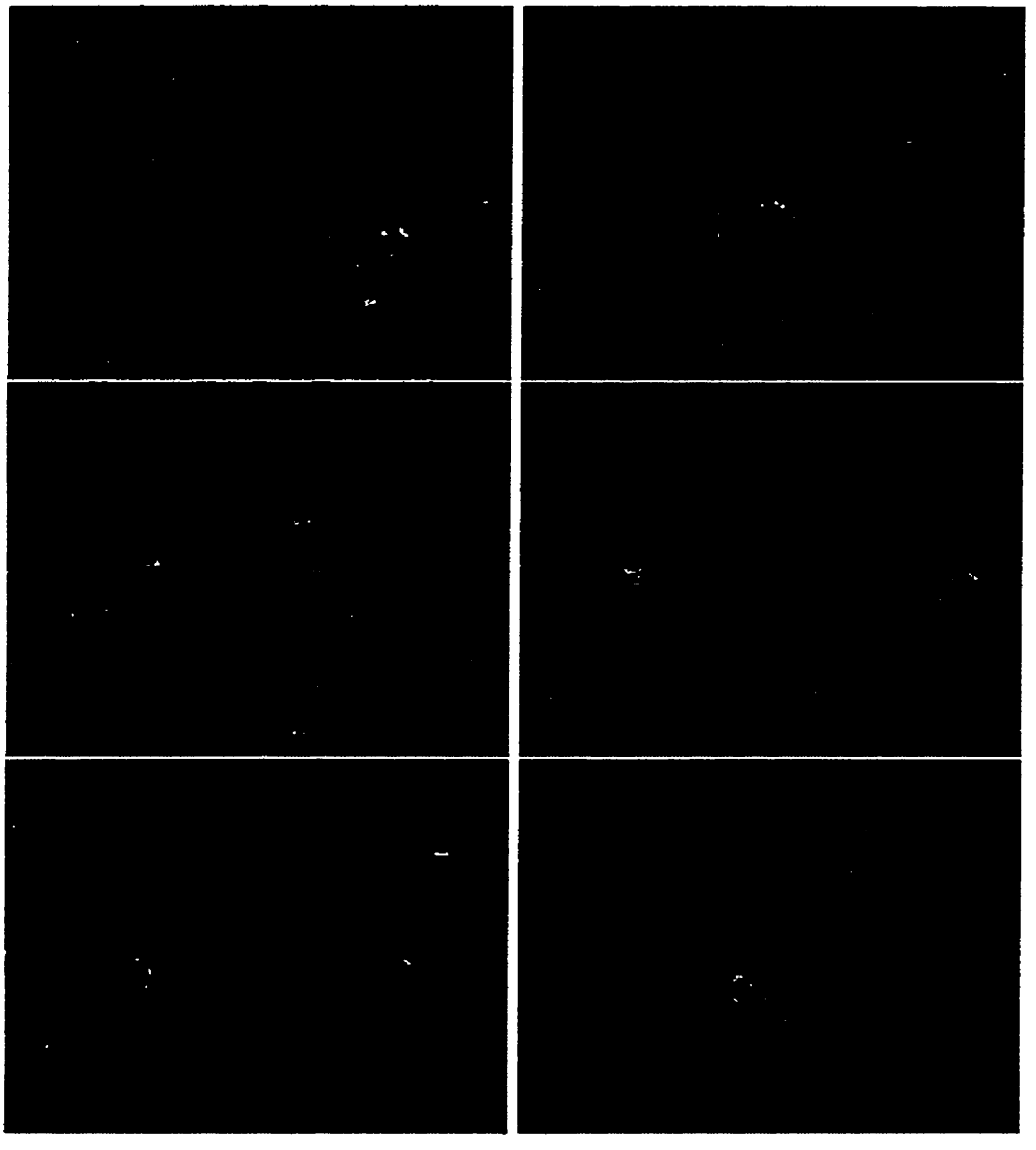


Figure 6.15 : Exemple de trajectoire des deux robots : le poursuivi est en bas à gauche sur la première image

Dans les figures 6.15 et 6.16 présentent deux séquences d'images correspondant à deux expèriences réalisées. On peut voir dans ces deux séquences que le robot poursuivi force le poursuivant à être aux limites de ces capacités de déplacement. Le poursuivant doit faire, par exemple, demi-tour dans une expérience en perdant ainsi beaucoup de temps lors de la poursuite. Le poursuivi prends ainsi avantage des

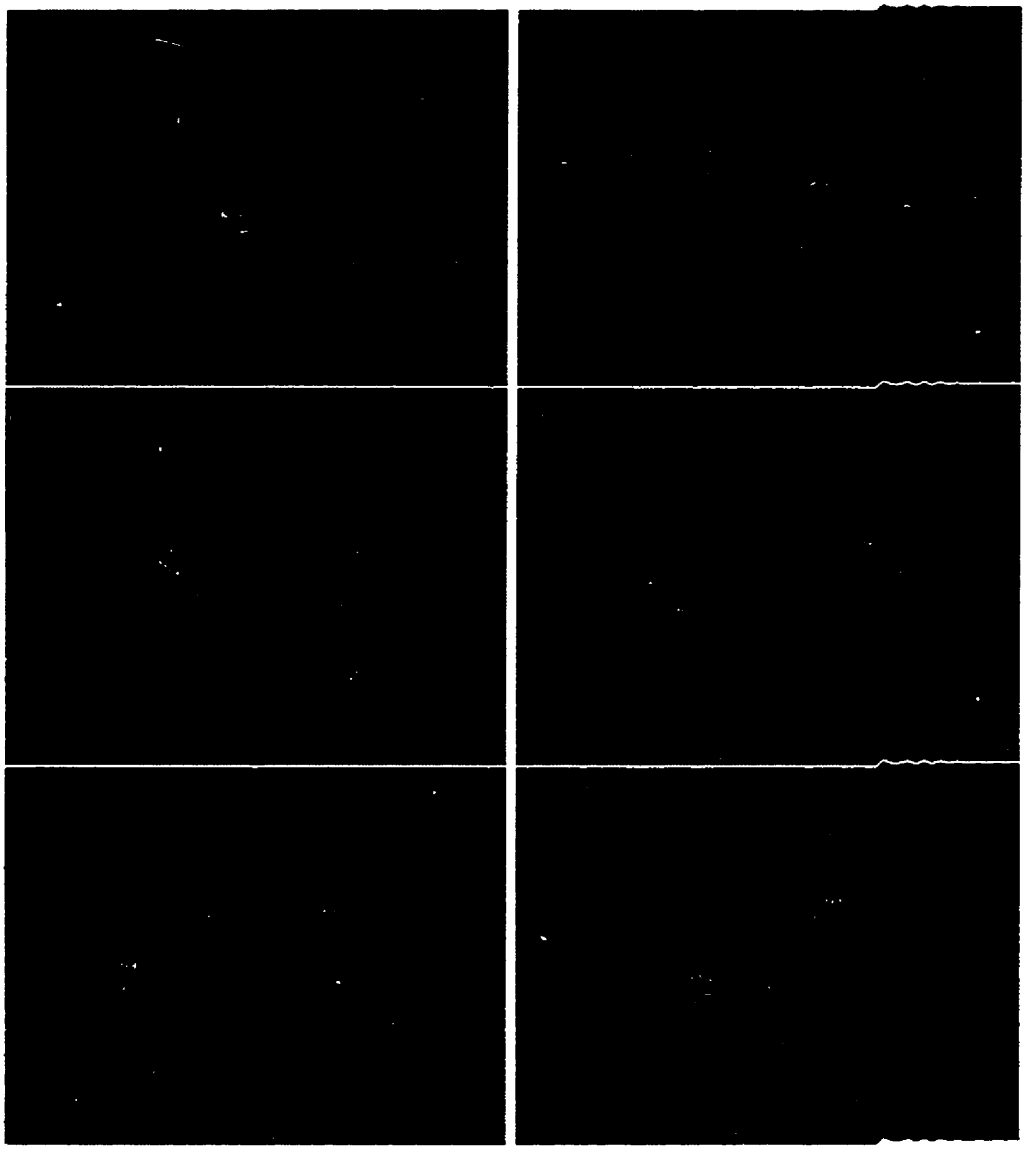
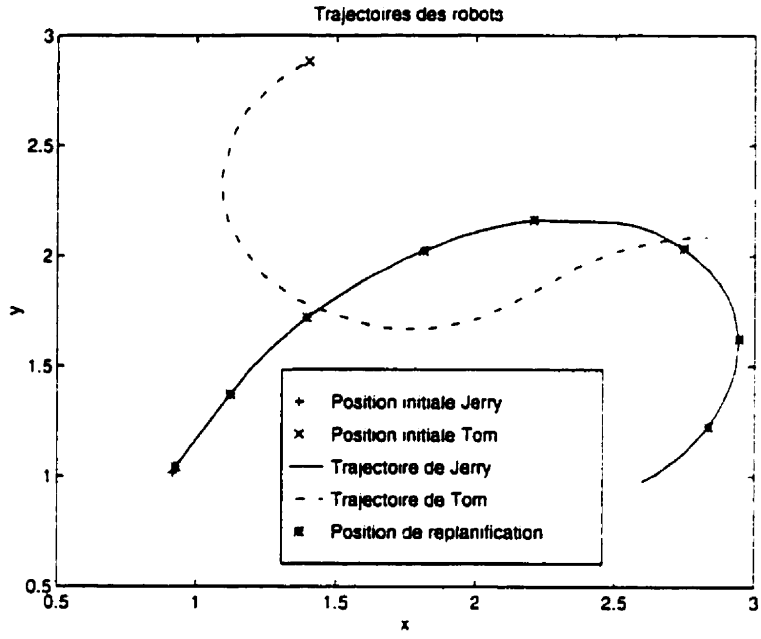


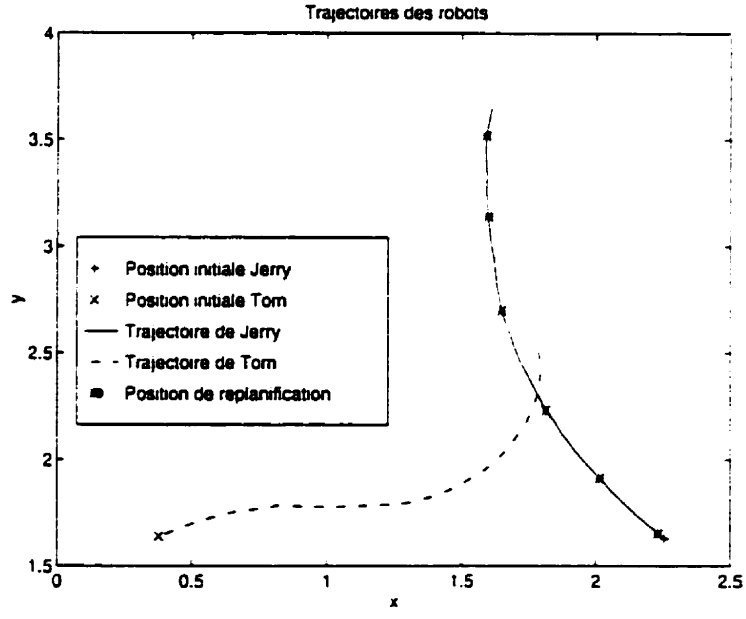
Figure 6.16 : Exemple de trajectoire des deux robots : le poursuivi est en bas de l'image au début

défauts du poursuivant pour s'enfuir. Les figures 6.17, 6.18 et 6.19 présentent plusieurs exemples de trajectoires qui ont été réalisées par les deux robots pour plusieurs situations initiales. La figure 6.20 présente les trajectoires des robots ainsi que les Cartes Dynamiques associées aux situations.

Dans la figure 6.18, on peut observer des erreurs dans le positionnement des robots

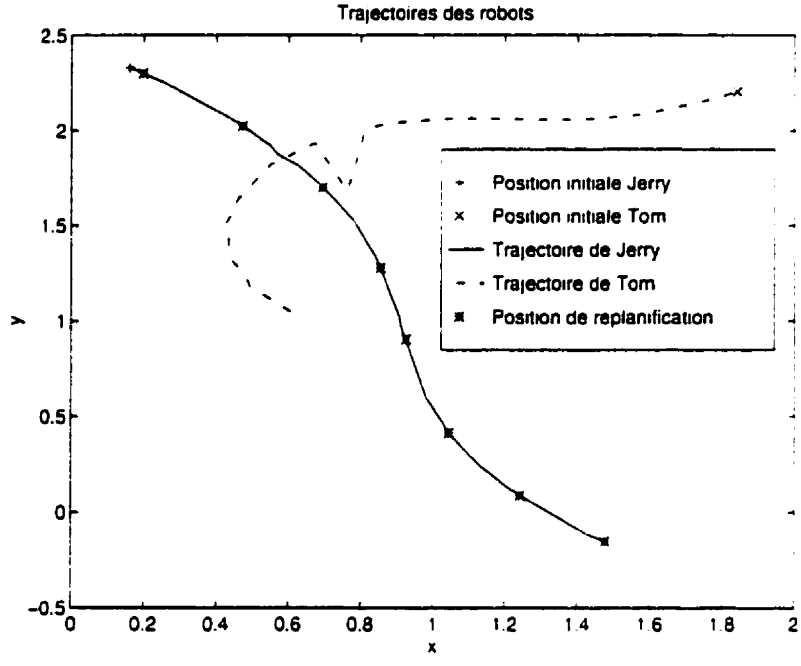


(a)

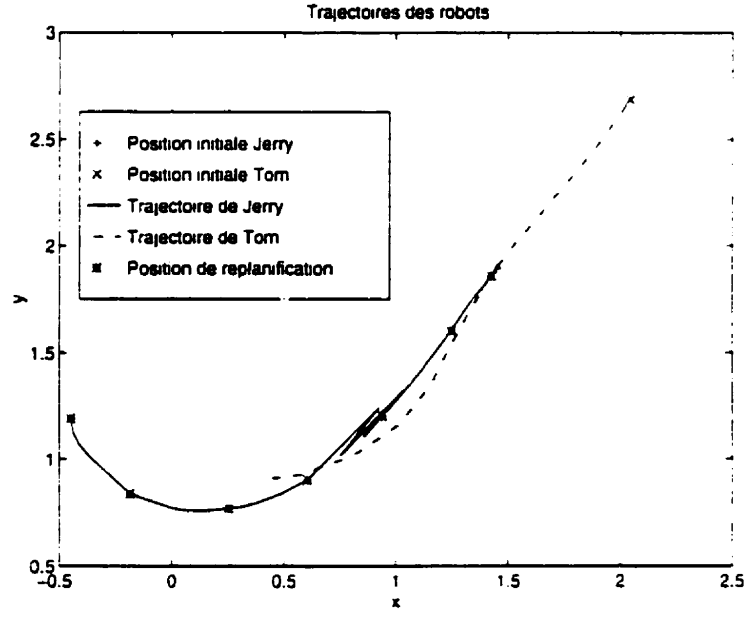


(b)

Figure 6.17 : Deux exemples de trajectoires réalisées par les deux robots



(a)



(b)

Figure 6.18 : Deux exemples de trajectoires réalisées par les deux robots

(erreur supérieure à la moyenne usuelle). Bien que nous n'ayons pas utilisé de filtrage de Kalman, on peut voir que les deux robots ont pu compenser dans les deux cas malgré ces fortes erreurs.

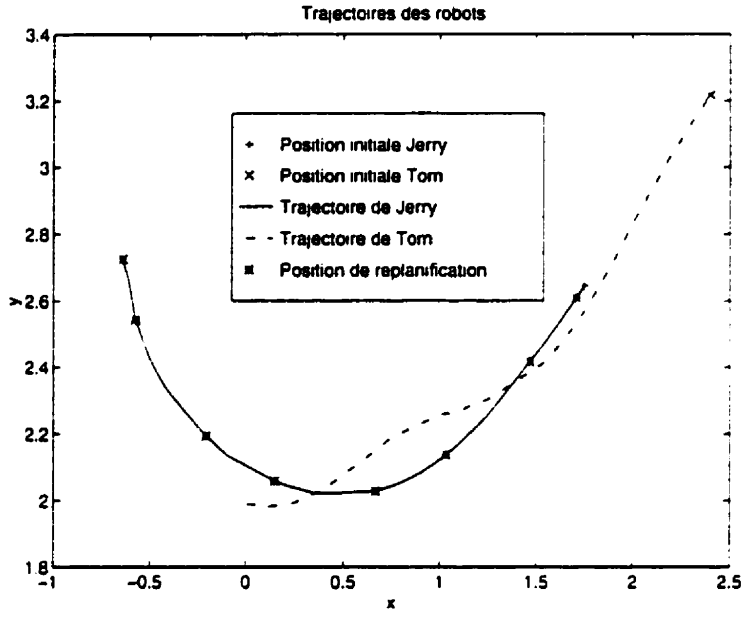
Pour les trajectoires (a) de la figure 6.19, on peut observer que, malgré le fait que les deux robots soient partis l'un derrière l'autre, la trajectoire idéale n'est pas rectiligne pour le poursuivi. En effet, celui-ci finit par tourner ses roues pour tenter de rentrer dans la zone morte de Tom (le poursuivant) et cette tactique semble finalement payer. La trajectoire (b) de la même figure est également intéressante, avec un zigzagement qui force peu à peu le poursuivant à lui même osciller et finalement à s'éloigner d'un bonne trajectoire d'interception.

Dans la figure 6.20, on peut voir que le poursuivi s'est bien adapté aux changements de comportement du poursuivant.

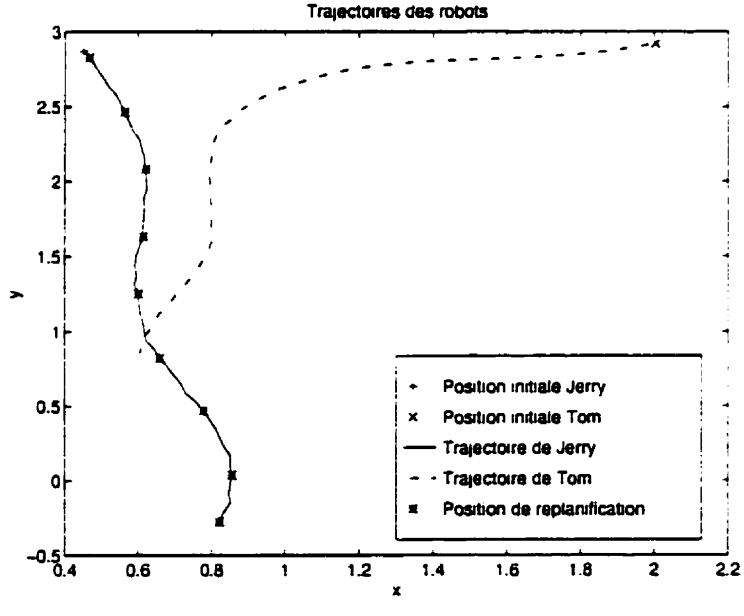
6.6.4.2 Stabilité et répétabilité

La figure 6.21 illustre la stabilité des trajectoires obtenues. Les trajectoires sont obtenues en répétant les expériences à partir des mêmes positions et orientations des robots. On constate facilement que les trajectoires sont fortement similaires pour les deux robots. On constate egalement que, malgré une forte erreur lors de la détection de Jerry (le poursuivi), il est capable de s'adapter aux mouvements du Tom.

Dans la figure 6.22, l'objectif est de montrer la répétabilité des résultats en effectuant les expériences en partant de mêmes positions et orientations des robots, mais aussi en faisant subir au poursuivant une suite de contrôles identiques exécutées en boucle ouverte. A nouveau, on peut remarquer la forte similarité entre les diverses trajectoires de Jerry en fonction de la trajectoire en boucle ouverte du poursuivant (Tom) et cela malgré plusieurs erreurs de position par le système d'odométrie. Il est important aussi de préciser que le repositionnement aux mêmes positions initiales pour les robots est fait à la main, ce qui influence fortement la précision des résultats.

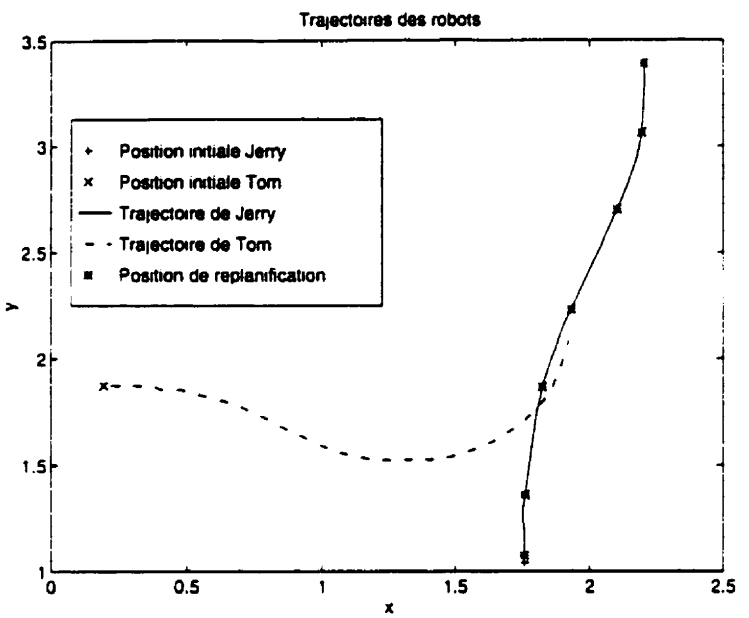


(a)

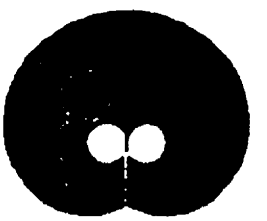


(b)

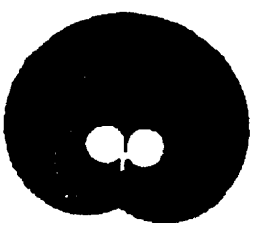
Figure 6.19 : Deux exemples de trajectoires réalisées par les deux robots



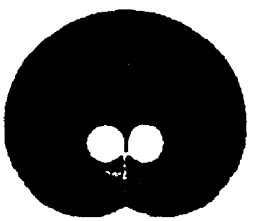
(a)



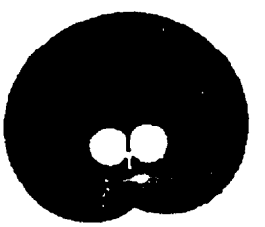
(b)



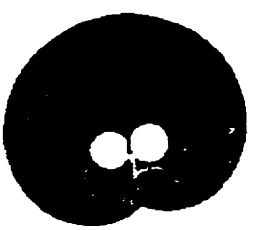
(c)



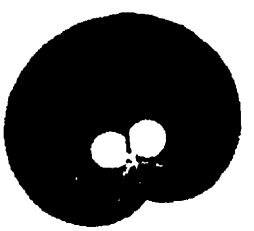
(d)



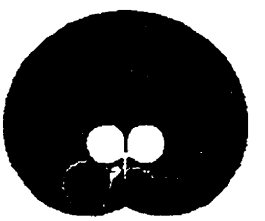
(e)



(f)



(g)



(h)

Figure 6.20 : (a) Trajectoires réalisées par les deux robots (b)-(h) Carte Dynamique et trajectoire planifiée aux instants indiqués dans la figure (a)

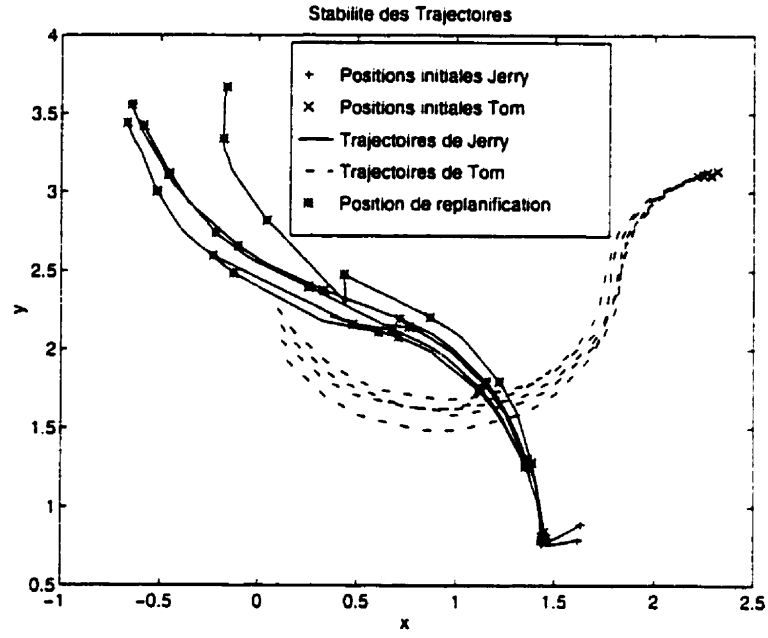


Figure 6.21 : Stabilité de l'exécution des trajectoires

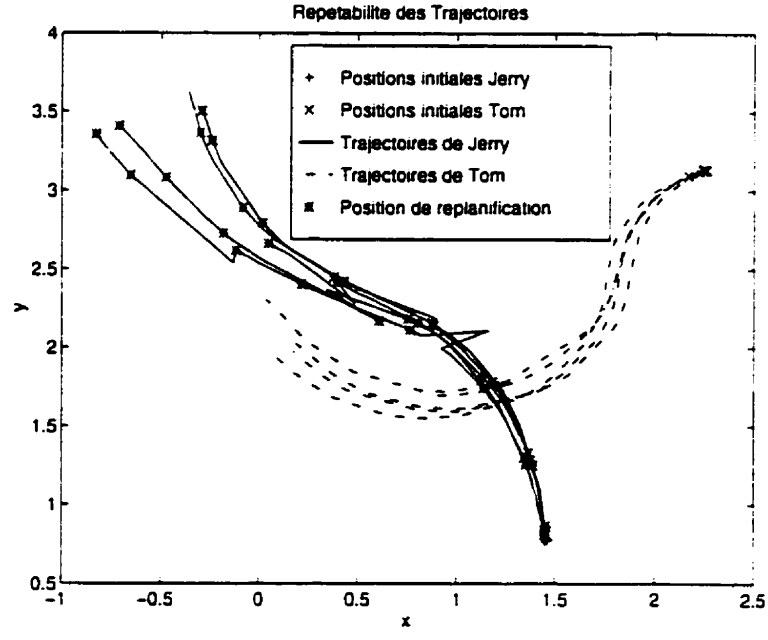
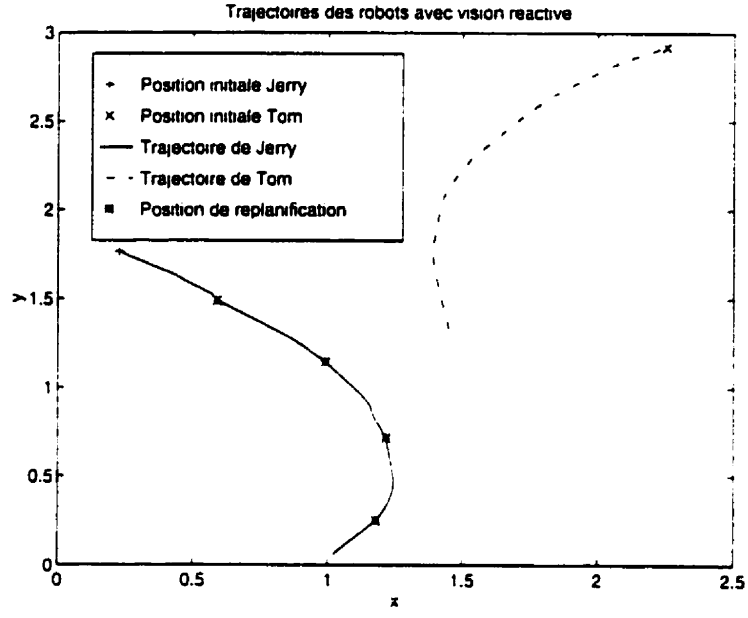


Figure 6.22 : Répétabilité de l'exécution des trajectoires

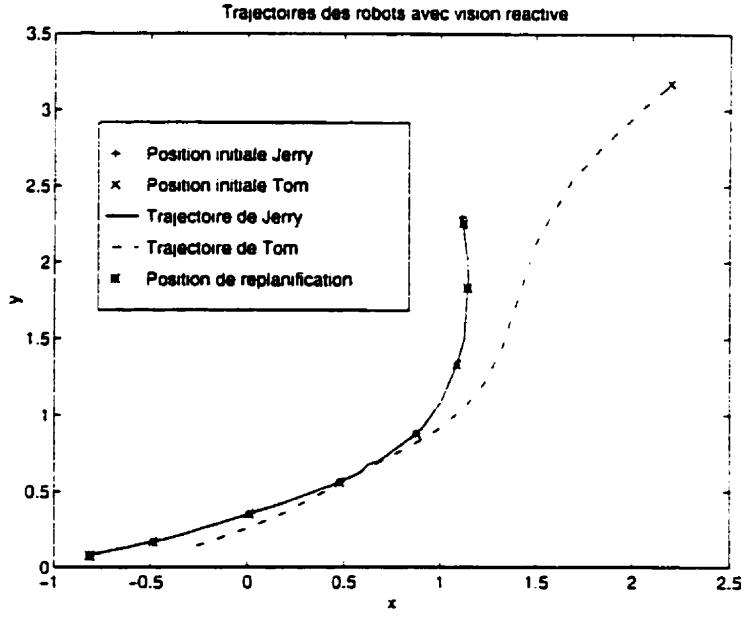
6.6.5 Conclusion sur les expériences

La figure 6.23 présente deux exemples de trajectoires qui ont été réalisées par les deux robots dans le cadre du deuxième scénario. A nouveau, il s'agit ici de montrer qu'il est possible de franchir une étape supplémentaire en direction d'un système complétement autonome.

La conclusion majeure de ces expériences est que le système se comporte bien en fonction du comportement de Tom (le poursuivant). Il n'a pas été possible pour l'instant, par faute de temps, de tester de façon extensive l'ensemble des configurations des robots qui amènent à la capture ou à l'évasion de Jerry. D'après l'ensemble des expériences réalisées, nous pouvons vraiment constater la viabilité du concept de Carte Dynamique comme une alternative à la planification réactive pour l'évasion. Il serait intéressant par la suite de tester un système se basant sur les Cartes Dynamiques pour planifier des trajectoires d'interception afin de pouvoir constater, par exemple, si le système (les deux robots avec leurs systèmes de planification) se dirigerait vers un équilibre stable. Cet équilibre permettrait par exemple de savoir si le poursuivi et le poursuivant ont des chances équivalentes de survie en fonction de leur structure (rayon de courbure maximum et système dynamique sous-jascent).



(a)



(b)

Figure 6.23 : Deux exemples de trajectoires réalisées par les robots avec le poursuivant utilisant la vision pour détecter les mouvements du poursuivi

Chapitre 7

Conclusion

Le but principal de cette thèse était d'introduire une méthodologie de représentation des interactions qui se produisent entre plusieurs robots dans un environnement. Nous avons ainsi proposé le concept de Carte Dynamique pour modéliser les capacités motrices d'un robot, en nous basant sur la théorie des régions atteignables d'un système dynamique. La Carte Dynamique permet également d'unifier les informations acquises par les senseurs du robot sur son environnement, ainsi que celles sur les autres robots. L'indépendance structurelle par rapport au robot que propose la Carte Dynamique est aussi un aspect intéressant de la modélisation. Nous avons présenté une représentation géométrique simple de la Carte Dynamique afin de simplifier l'apprentissage de la forme des Cartes.

Par le biais du concept de Carte Dynamique, nous avons tenté de franchir la barrière existant entre le senseur du robot et ses capacités dynamiques de mouvement. Essentiellement, nous avons montré comment il était possible d'utiliser la Carte Dynamique pour produire des régions d'intérêt dans une image sur lesquelles l'utilisation d'algorithmes usuels d'analyse d'image est possible. Nous avons toutefois considéré seulement cette relation avec une seul robot. L'extension à la combinaison des Cartes Dynamiques directement dans l'image, ou la replanification dynamique de trajectoires à l'aide des Cartes Dynamiques constitue de futurs défis intéressants à l'utilisation

des Cartes Dynamiques.

La combinaison de Cartes Dynamique permet également de représenter les interactions du point de vue d'un robot par rapport aux autres dans l'environnement. En effet, la Carte Dynamique (apprise) du robot considéré est combinée avec les Cartes Dynamiques (estimées) des autres robots en fonction des poses relatives des robots et des informations contenues sur les Cartes Dynamiques. La combinaison est effectuée afin de proposer une représentation des interactions dans le cadre d'une tâche précise à accomplir. La Carte Dynamique obtenue après la combinaison contient à la fois les informations relatives aux régions atteignables du robot mais également de celles des autres robots, indiquant ainsi par exemple des zones de danger ou de sécurité. Des méthodes d'optimisation se basant sur les Cartes combinées afin de produire des trajectoires non-holonomes pour le robot ont aussi été proposées. Une étude des interactions dans le cadre d'une tâche de capture/évasion a été présentée par la suite. La représentation d'interactions coopératives a également été envisagée.

Deux méthodes d'apprentissage ont été également développées : Le première système d'apprentissage de stratégie de capture par renforcement a été proposé comme une alternative à notre concept de planification réactive des Cartes Dynamiques. Nous avons cependant constaté que ces dernières nous ont permis de mieux comprendre les résultats et d'expliquer à nouveau le phénomène des régions atteignables. La deuxième méthode consiste à apprendre la forme du volume de représentation des Cartes Dynamiques, soit du robot lui-même (auto-apprentissage) soit d'un autre robot à partir des informations visuelles disponibles (apprentissage visuel) :

Nous avons introduit dans cette thèse, un banc d'essai expérimental qui a été développé dans le Groupe de Recherche en Perception et Robotique de l'Ecole Polytechnique de Montréal. Ce système présente des caractéristiques originales intéressantes, telles que l'utilisation d'une odométrie simulée centralisée et le partage des ressources de calcul sur un réseau de stations de travail. Plusieurs exemples de travaux réalisés sur le banc d'essai expérimental ont été présentés. Nous avons également

montré que les Cartes Dynamiques pouvaient être utilisées sur notre banc d'essai expérimental. Divers exemples de trajectoires réalisées par nos deux robots dans un cas réel de poursuite/évasion ont été ainsi présentés.

Avant de décrire le futur des travaux reliés aux Cartes Dynamiques, résumons rapidement les différentes contributions originales de notre travail au domaine des systèmes multi-robots :

1. Par le biais des Cartes Dynamiques, nous avons introduit une méthodologie de représentation des capacités de déplacement d'un robot dans son environnement.

2. Cette méthodologie permet de représenter les interactions entre robots pour des tâches de type coopératif et/ou compétitif. En nous basant sur cette représentation, nous avons montré qu'un robot était capable de planifier sa trajectoire en tenant compte des capacités de déplacement des autres robots ou bien d'analyser le type d'interactions obtenues.

3. Nous avons également proposé une représentation "bitmap" des Cartes Dynamiques qui permet d'accélérer les méthodes de construction des interactions entre robots à partir de systèmes d'accélérations graphiques existants.

4. Une des grandes forces de notre méthode de construction des interactions entre robots repose sur le fait qu'elle évite l'explosion combinatoire communément rencontrée dans des systèmes multi-robots.

5. Le banc d'essai expérimental possède plusieurs innovations : la capacité de surveiller les mouvements des robots dans l'espace de travail, l'odométrie simulée, l'utilisation de véhicule radio-guidé simple et finalement le partage du travail sur un réseau de stations de travail.

6. Lors de nos expériences, nous avons démontré l'efficacité des Cartes Dynamiques pour faire de la planification en temps réel. Les Cartes Dynamiques permet-

tent ainsi une planification dynamique des trajectoires d'évasion d'un robot en s'adaptant à la situation courante.

7. Finalement, nous avons également proposé des méthodes originales d'apprentissage des capacités de déplacement (par l'intermédiaire des Cartes Dynamiques) d'un robot par lui-même ou celle d'un autre à partir de la vision.

Comme nous l'avons mentionné dans la thèse, il serait intéressant d'étudier plus en détail les Cartes Dynamiques de chaque type de robot mobile tel que défini par la nomenclature de Campion *et al.* (Campion, Bastin et D'Andréa-Novel 1996). Afin de montrer que le concept de Carte Dynamique constitue réellement un outil générique de génération et d'analyse des interactions entre robots, il est essentiel d'envisager d'autres tâches et diverses façons de combiner les Cartes Dynamiques.

Dans le futur, nous allons étudier le problème de la capture à partir des Cartes Dynamiques. Dans la figure 7.1, un exemple préliminaire de Carte Dynamique combinée pour la capture de trois robots est présenté. Les zones noires correspondent aux positions pour lesquelles le temps des des régions atteignables du robot poursuivant et d'un des robots poursuivis est identique. La fonction de combinaison devra ainsi mettre en relief ce genre de position. Nous avons également décrit comment la coopération entre robots pouvait être faite au travers des Cartes Dynamiques. La figure 7.2 reprend l'exemple simple de ce processus de coopération avec trois robots.

L'intégration des incertitudes à l'intérieur de la Carte Dynamique constitue également un important travail à envisager dans le futur. Malgré que l'incertitude dans la position du robot lui-même ne soit pas importante à cause de l'aspect égocentrique des Cartes, il est nécessaire de pouvoir intégrer l'incertitude sur les positions relatives des autres robots. Par exemple, si une modélisation des incertitudes est disponible quant aux positions des autres robots, il sera possible de générer un nouvel ensemble de courbes T-atteignables, pour chaque robot, avant de procéder à la combinaison qui vont prendre en compte ces incertitudes. Prenons l'exemple d'une modélisation en cercle de l'incertitude, les régions T-atteignables sont alors augmentées en prenant

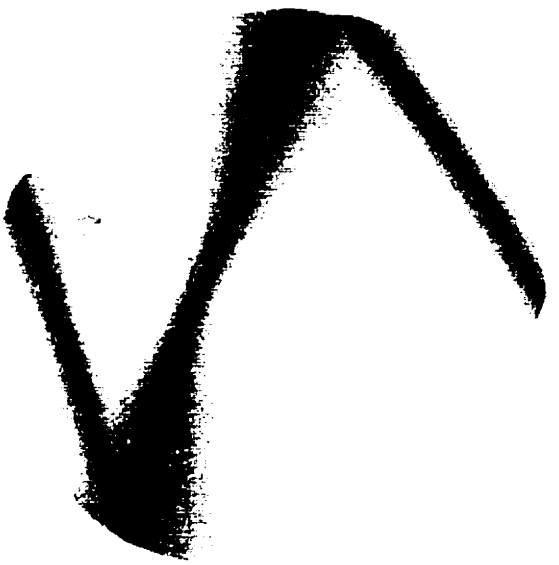


Figure 7.1 : La Carte Dynamique combinée dans le cadre d'une tâche de capture

la périphérie extérieur du cercle se déplaçant sur la courbe T-atteignable. Il s'agit là d'une méthode analogue à l'augmentation des obstacles pour obtenir l'espace C_{libre} présentées dans le chapitre 2.

Une application possible du système dynamique d'évasion se basant sur les Cartes Dynamiques est par exemple la planification d'une trajectoire d'évitement de collision pour une voiture à grande vitesse avec une autre voiture (dont le comportement sera considéré comme agressif pour envisager le pire cas) ou avec un piéton (son déplacement est restreint mais possible) en rajoutant des contraintes sur le maintien du contrôle de la voiture par le chauffeur. En effet, lors de la construction du réseau de courbes il est possible d'ajouter des informations sur la perte de contrôle éventuelle du véhicule lors de changements trop brusques de la trajectoire de la voiture.

Nous pouvons noter également qu'il est possible d'étendre le concept de Carte Dynamique à d'autres types de robots holonomes tels qu'un robot manipulateur. En effet, ce genre de robot est modélisable par un système dynamique auquel le

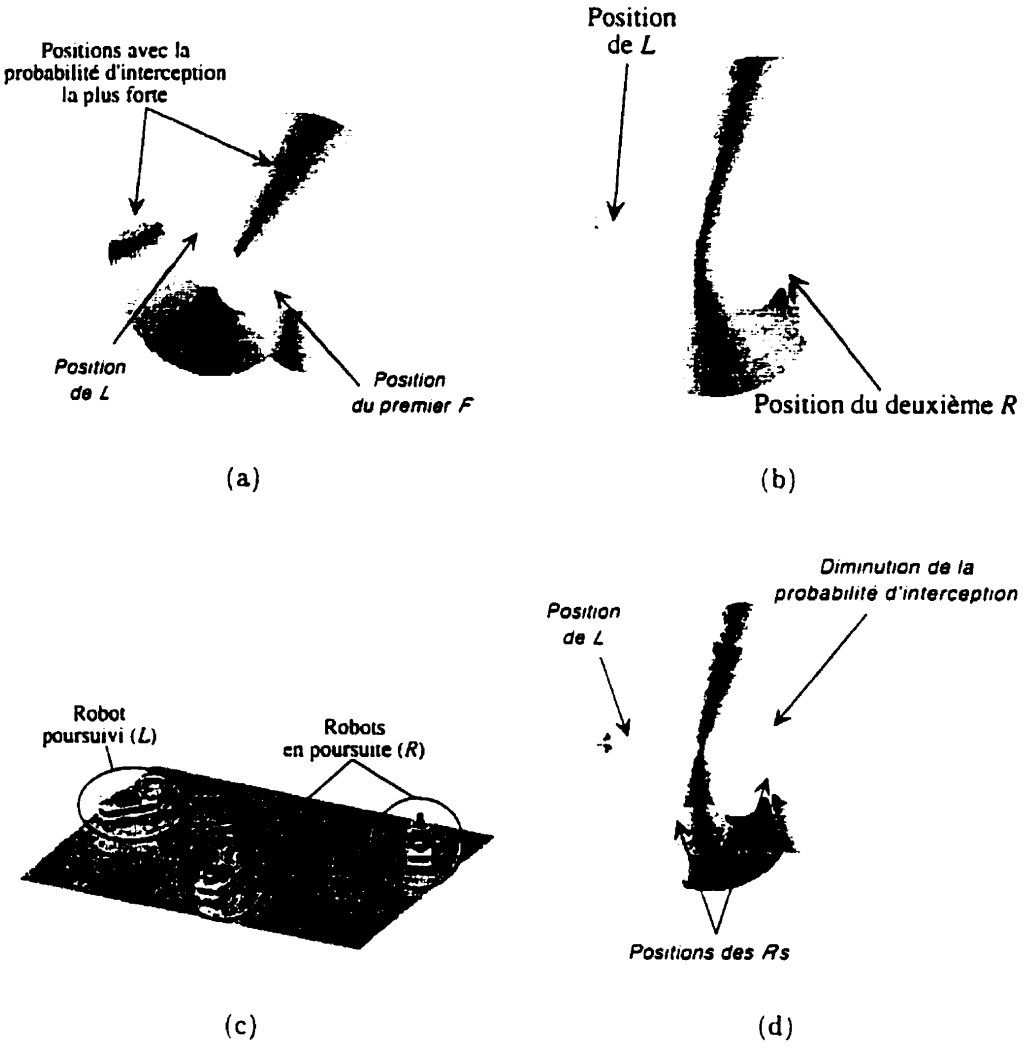
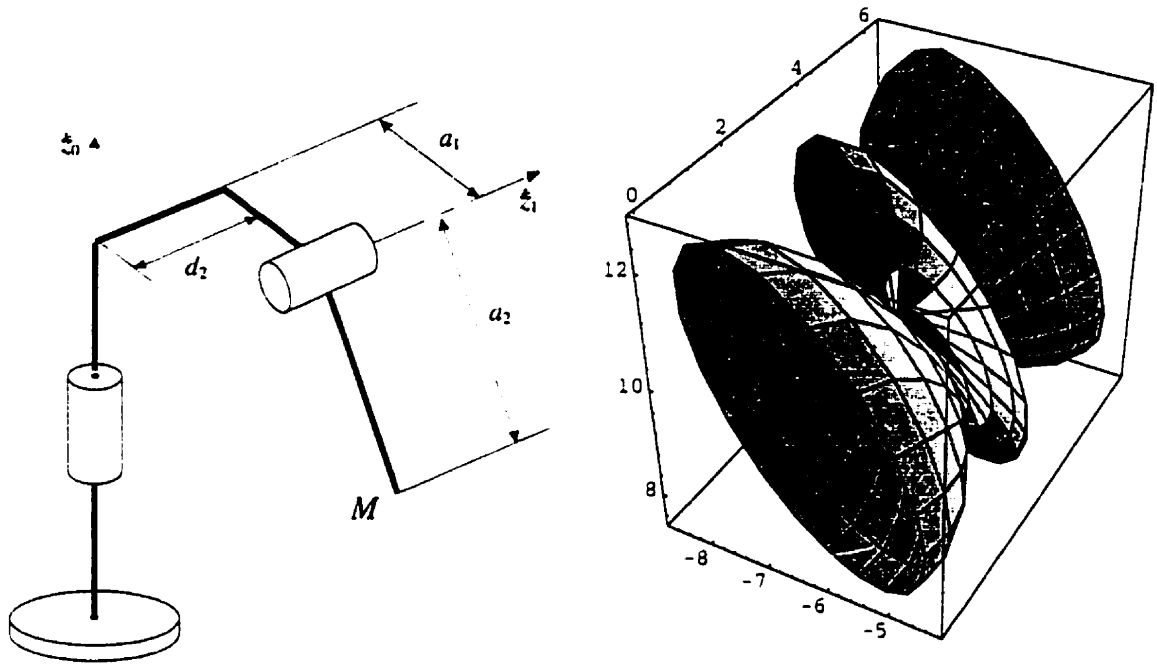


Figure 7.2 : (a) Carte Dynamique combinée entre le premier F et R (b) Carte Dynamique combinée entre le second F et R (c) Pose relative du robot R avec les 2 robots Fs (d) Carte Dynamique combinée du deuxième F avec la coopération du premier F

concept de Carte Dynamique est applicable. Dans cette optique, Hervé (Hervé 1993) a proposé une méthode de représentation des régions atteignables d'un robot avec deux ou trois degrés de liberté, utilisé par la suite pour lier la perception active du robot à ces régions par la Carte Perceptuelle Cinématique (PKM: "Perceptual Kinetic Map"). La Carte Dynamique serait une extension de ce genre de méthode en permettant d'ajouter des informations supplémentaires pour optimiser la planification



(a) (b)

Figure 7.3 : (a) un modèle de robot manipulateur avec deux degrés de liberté (b) la zone atteignable du robot construit en fixant une valeur d'un joint et en procédant à la rotation complète du deuxième joint

des mouvements du robot.

Un aspect intéressant de cette approche est de permettre d'analyser des phénomènes complexes de couplage visuel entre le système visuel et le bras manipulateur (ou coordination main/œil). Il a en effet été proposé par exemple que les saccades visuelles chez le singe sont concentrées autour des positions dans l'image les plus probables des membres du primate (Burt 1988).

Bibliographie

ALAMI, R., ROBERT, F., INGRAND, F. et SUZUKI, S. (1995), Multi-robot cooperation through incremental plan-merging, dans *Proceedings of the IEEE International Conference on Robotics And Automation*, Nagoya, Japan, pp. 2573–2579.

ALOIMONOS, J. (1990), Purposive and qualitative active vision, dans *Proceedings of the IEEE International Conference on Pattern Recognition*, Vol. 1, Atlantic City, New Jersey, USA, pp. 346–360.

ALOIMONOS, Y. et DURIC, Z. (1994), Estimating the heading direction using normal flow, *International Journal on Computer Vision* **13**(1), 33–56.

ANDERSON, R. (1989), Dynamic sensing in a ping-pong playing robot, *IEEE Transactions on Robotics and Automation* **5**(6), 728–739.

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, *Journal of Robotic Systems* **9**(3), 351–364.

ARONOV, B., FORTUNE, S. et WILFONG, G. (1991), Minimum-speed motions, *The International Journal of Robotics Research* **10**, 228–239.

ASADA, M., UCHIBE, E., NODA, S., TAWARATSUMIDA, S. et HOSODA, K. (1994), Coordination of multiple behaviors acquired by a vision-based reinforce-

ment learning, dans *Proceedings of the IEEE/RSJ International Workshop on Intelligent Robots and Systems*, Vol. 2, pp. 917-924.

BAJCSY, R. (1988), Active perception, *Proceedings of the IEEE* **76**(8), 996-1005.

BAJCSY, R. et CAMPOS, M. (1992), Active and exploratory perception, *CVGIP: Image Understanding* **56**(1), 31-40.

BAKER, H. et BOLLES, R. (1988), Generalizing epipolar-plane image analysis on the spatiotemporal surface, dans *Proceedings of DARPA Image Understanding Workshop*, pp. 1022-1031.

BAKER, H. et GARVEY, T. (1991), Motion tracking on the spatiotemporal surface, dans *Proceedings of the IEEE Workshop on Visual Motion*, pp. 340-345.

BALLARD, D. (1991), Animate vision, *Artificial Intelligence* **48**(1), 57-86.

BANDOPADHAY, A. et BALLARD, D. (1991), Egomotion perception using visual tracking, *Computational Intelligence* **7**(1), 39-47.

BARRAQUAND, J. et FERBACH, P. (1995), Motion planning with uncertainty: The information space approach, dans *Proceedings of the IEEE International Conference on Robotics And Automation*, Nagoya, Japan, pp. 1341-1348.

BARRAQUAND, J. et LATOMBE, J.-C. (1993), Robot motion planning: A distributed representation approach, *Algorithmica* **10**, 121-155.

BARTO, A. et SUTTON, R. (1981), Landmark learning: An illustration of associative search, *Biological Cybernetics* **42**(1), 1-8.

BARTO, A., SUTTON, R. et ANDERSON, C. (1983), Neuronlike adaptive elements that can solve difficult learning control problems, *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics* **13**(5), 834-846.

BARTO, A., SUTTON, R. et BROUWER, P. (1981), Associative search network: A reinforcement learning associative memory, *Biological Cybernetics* **40**(3), 201–211.

BELLMAN, R. (1957), *Dynamic Programming*, Princeton University Press.

BICCHI, A., CASALINO, G. et SANTILLI, C. (1995), Planning shortest bounded-curvature paths for a class of nonholonomic vehicles among obstacles, dans *Proceedings of the IEEE International Conference on Robotics And Automation*, Nagoya, Japan, pp. 1349–1354.

BIEN, Z. et LEE, J. (1992), A minimum-time trajectory planning method for two robots, *IEEE Transactions on Robotics and Automation* **8**(3), 414–418.

BOISSONNAT, J.-D. et BUI, X.-N. (1994), Accessibility region for a car that only moves forwards along optimal paths, rapport technique 2181, LAAS, INRIA, Toulouse.

BOUILLY, B., SIMÉON, T. et ALAMI, R. (1995), A numerical technique for planning motion strategies of a mobile robot in presence of uncertainty, dans *Proceedings of the IEEE International Conference on Robotics And Automation*, Nagoya, Japan, pp. 1327–1332.

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

BROOKS, R. (1991*b*), Intelligence without representation, *Artificial Intelligence* **47**, 139–160.

BROWN, R. et JENNINGS, J. (1995), A pusher/steerer model for strongly cooperative mobile robot manipulation, dans *Proceedings of the IEEE/RSJ International Workshop on Intelligent Robots and Systems*, Vol. 3, pp. 562–568.

BUI, X.-N., BOISSONNAT, J.-D., SOUÈRES, P. et LAUMOND, J.-P. (1993), The shortest path synthesis for non-holonomic robots moving forwards, rapport technique 2153, LAAS, INRIA, Toulouse.

BURGER, W. et BHANU, B. (1992), *Qualitative Motion Understanding*, Kluwer Academic Publishers, Boston, MA.

BURT, P. (1988), Attention mechanisms for vision in a dynamic world, dans *Proceedings of the IEEE International Conference on Pattern Recognition*, pp. 977–987.

CAMPION, G., BASTIN, G. et D'ANDRÉA-NOVEL, B. (1996), Structural properties and classification of kinetic and dynamic models of wheeled mobile robots, *IEEE Transactions on Robotics and Automation* **12**(1), 47–62.

CAO, Y., FUKUNAGA, A., KAHNG, A. et MANG, F. (1995), Cooperative mobile robotics: Antecedents and directions, dans *Proceedings of the IEEE RSJ International Workshop on Intelligent Robots and Systems*, Vol. 1, pp. 226–234.

CHATILA, R., ALAMI, R., DEGALLAIX, B. et LARUELLE, H. (1992), Integrated planning and execution control of autonomous robot actions, dans *Proceedings of the IEEE International Conference on Robotics And Automation*, Nice, France, pp. 2689–2695.

CHEN, Q. et LUH, J. (1994), Distributed motion coordination of multiple robots, dans *Proceedings of the IEEE/RSJ International Workshop on Intelligent Robots and Systems*, Vol. 3, pp. 1493–1500.

CHERIF, M. et LAUGIER, C. (1995), Motion planning of autonomous off-road vehicles under physical interaction constraints, dans *Proceedings of the IEEE International Conference on Robotics And Automation*, Nagoya, Japan, pp. 1687–1693.

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

CIPOLLA, R. et BLAKE, A. (1992), Surface orientation an time to contact from image divergence and deformation, dans *Proceedings of the Second European Conference on Computer Vision*, Santa Margherita Ligure, Italy, pp. 187–202.

CONNELL, J. (1990), *Minimalist Mobile Robots: A Colony-Style Architecture for an Artificial Creature*, Academic Press, San Diego, CA.

CONNOLLY, C. et GRUPEN, R. (1994), Nonholonomic path planning using harmonic functions, rapport technique, Laboratory for Perceptual Robotics.

COOMBS, D. et ROBERTS, K. (1993), Centering behavior using peripheral vision, dans *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 440–445.

CRAIG, J. (1989), *Introduction to robotics*, Addison-Wesley Publishing Company.

CRISMAN, J. et THORPE, C. (1993), Scarf: A color vision system that tracks roads and intersections, *IEEE Transactions on Robotics and Automation* **9**(1), 49–58.

DICKMANNS, E. et GRAEFE, V. (1988*a*), Applications of dynamic monocular machine vision, *Machine Vision and Applications* **1**(4), 241–261.

DICKMANNS, E. et GRAEFE, V. (1988*b*), Dynamic monocular machine vision, *Machine Vision and Applications* **1**(4), 223–240.

DUBINS, L. (1957), On curves of minimal length with a contraint on average curvature and with prescribeb initial and terminal position and tangent, *American Journal of Mathematics* pp. 497–516.

DURIC, Z., ROSENFELD, A. et DUNCAN, J. (1994), The applicability of Green's theorem to computation of rate of approach, rapport technique CAR-TR-710. Computer Vision Laboratory, Center for Automation Research, University of Maryland, College Park, MD.

FEDDEMA, J. et LEE, C. (1990), Adaptive image feature prediction and control for visual tracking with a hand-eye coordinated camera, *IEEE Transactions on Systems, Man, and Cybernetics* **20**(5), 1172–1183.

FEDDEMA, J. et MITCHELL, O. (1989), Vision-guided servoing with feature-based trajectory generation, *IEEE Transactions on Robotics and Automation* **5**(5), 691–700.

FERMÜLLER, C. (1993*a*), Basic visual capabilities, rapport technique CS-TR-3064, Computer Vision Laboratory, Center for Automation Research, University of Maryland, College Park, MD.

FERMÜLLER, C. (1993*b*), Motion constraint patterns, dans *IEEE Workshop on Qualitative Vision*, New York, New York, pp. 61–70.

FERRARI, C., PAGELLO, E. et ARAI, T. (1995), Planning multiple autonomous robots motion in space and time, dans *Proceedings of the IEEE/RSJ International Workshop on Intelligent Robots and Systems*, Vol. 2, pp. 253–259.

FIORINI, P. et SHILLER, Z. (1996), Time optimal trajectory planning in dynamic environments, dans *Proceedings of the IEEE International Conference on Robotics And Automation*, Minneapolis, MN, pp. 1553–1558.

FLEURY, S., SOUÈRES, P., LAUMOND, J.-P. et CHATILA, R. (1995), Primitives for smoothing mobile robot trajectories, *IEEE Transactions on Robotics and Automation* **11**(3), 441–448.

FRANçOIS, E. (1991), Interprétation qualitative du mouvement à partir d'une séquence d'images, Thèse de doctorat, Université de Rennes I, Rennes.

FUJIMURA, K. (1989), Motion planning in dynamic domains, Thèse de doctorat, Science Department, University of Maryland, College Park, MD.

FUJIMURA, K. et SAMET, H. (1989), A hierarchical strategy for path planning among moving obstacles, *IEEE Transactions on Robotics and Automation* **5**(1), 61–69.

FUKUDA, T. et IRITANI, G. (1995), Construction mechanism of group behavior with cooperation, dans *Proceedings of the IEEE/RSJ International Workshop on Intelligent Robots and Systems*, Vol. 3, pp. 535–542.

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

GRISWOLD, N. et EEM, J. (1990), Control for mobile robots in the presence of moving obstacles, *IEEE Transactions on Robotics and Automation* **6**(2), 263–268.

GROSSO, E. et BALLARD, D. (1993), Head-centered orientation strategies in animate vision, dans *Proceedings of the International Conference on Computer Vision*, Berlin, Germany, pp. 395–402.

GUPTA, N. (1981*a*), An overview of differential games, *Control and Dynamic Systems* **17**, 1–25.

GUPTA, N. (1981*b*), Reachable set methods, *Control and Dynamic Systems* **17**, 323–344.

GUTSCHE, R., LALONI, C. et WAHL, F. (1994), Path planning for mobile vehicles within dynamic worlds using statistical data, dans *Proceedings of the IEEE/RSJ International Workshop on Intelligent Robots and Systems*, Vol. 1, pp. 454–461.

HAYKIN, S. (1991), *Neural Networks: A Comprehensive Foundation*, Macmillan, New York.

HERVÉ, J.-Y. (1993), Navigational Vision, Thèse de doctorat, Computer Science Departement, University of Maryland, College Park, MD.

HINTON, G. (1989), Connectionist learning procedures, *Artificial Intelligence* **40**(1), 185-234.

HORN, B. (1986), *Robot Vision*, MIT press.

HORSWILL, I. et BROOKS, R. (1988), Situated vision in a dynamic world: Chasing objects, dans *Proceedings of the AAAI-88*, pp. 796-800.

HWANG, Y. et AHUJA, N. (1992), Gross motion planning—a survey, *ACM Computing Surveys* **24**(3).

INABA, M., KAGAMI, S., ISHIKAWA, T., KANEHIRO, F., TAKEDA, K. et INOUE, H. (1994), Vision-based adaptive and interactive behaviors in mechanical animals using the remote-brained approach, dans *Proceedings of the IEEE RSJ International Workshop on Intelligent Robots and Systems*, Vol. 2, pp. 933-940.

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

ISHIGURO, A., WATANABE, Y. et UCHIKAWA, Y. (1995), An immunological approach to dynamic behavior control for autonomous mobile robots, dans *Proceedings of the IEEE/RSJ International Workshop on Intelligent Robots and Systems*, Vol. 1, pp. 495–500.

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*, Nagoya, Japan, pp. 1379–1384.

JOCHEM, T. et POMERLEAU, D. (1996), Life in the fast lane, the evolution of an adaptive vehicle control system, *AI Magazine* **17**(2), 11–50.

JULIER, S. et DURRANT-WHYTE, H. (1995), Process models for the high-speed navigation of road vehicles, dans *Proceedings of the IEEE International Conference on Robotics And Automation*, Nagoya, Japan, pp. 101–105.

KANT, K. et ZUCKER, S. (1988), Planning collision-free trajectories in time-varying environments: A two-level hierarchy, dans *Proceedings of the IEEE International Conference on Robotics And Automation*, pp. 1644–1649.

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

KOENDERINK, J. (1986), Optic flow, *Vision Research* **26**(1), 161–180.

KOENDERINK, J. et DOORN, A. V. (1975), Invariant properties of the motion field parallax field due to the movement of rigid bodies relative to an observer, *Optica Acta* **22**(9), 773–791.

KOSECKA, J. (1996), A Framework for Modeling and Verifying Visually Guided Agents: Design, Analysis and Experiments, Thèse de doctorat, University of Pennsylvania.

KOSECKA, J., CHRISTENSEN, H. et BAJCSY, R. (1993), Discrete event modeling of navigation and gaze control, rapport technique, GRASP Laboratory, University of Pennsylvania, Philadelphia, PA.

KUNDUR, S. et RAVIV, D. (1994), An image texture-independent visual motion cue for autonomous navigation, rapport technique 5567, National Institute of Standards and Technology (NIST), Gaithersburg, MD.

LABONTÉ, F. (1997), Codage stéréoscopique par région d'intérêt, Thèse de doctorat, GRPR, École Polytechnique de Montréal.

LACROIX, P., POLOTSKI, V., COHEN, P. et HERVÉ, J.-Y. (1997), Decentralized control of two car-like robots performing common transportation tasks, rapport technique, GRPR, Ecole Polytechnique, Montréal, QC, Canada.

LATHUILIÈRE, F. (1997), Coordination visuelle entre robots mobiles autonomes, rapport technique GRPR-RT-9710, GRPR, École Polytechnique de Montréal.

LATOMBE, J.-C. (1991), *Robot Motion Planning*, Kluwer Academic Publishers, Boston, MA.

LAUMOND, J.-P., JACOBS, P., TAÏX, M. et MURRAY, R. (1994), A motion planner for nonholonomic mobile robots, *IEEE Transactions on Robotics and Automation* **10**(5), 577–593.

LIN, L.-J. (1991), Programming robots using reinforcement learning and teaching, dans *Proceedings of the AAAI-91*, Vol. 2, pp. 781–786.

LIN, L.-J. (1993), Reinforcement Learning for Robots Using Neural Networks, Thèse de doctorat, School of Computer Science, Carnegie Mellon University.

LITTMAN, M. (1996), A generalized reinforcement-learning model: Convergence and application, rapport technique CS-96-10, Department of Computer Science, Brown University.

LONGUET-HIGGINS, H. (1981), A computer algorithm for reconstructing a scene from two projections, pp. 61–62.

LUMELSKY, V., MUKHOPADHYAY, S. et SUN, K. (1990), Dynamic path planning in sensor-based terrain, *IEEE Transactions on Robotics and Automation* **6**(4), 462–472.

LUMELSKY, V. et SHKEL, A. (1995), Incorporating body dynamics into the sensor-based motion planning paradigm, the maximum turn strategy, dans *Proceedings*

of the IEEE International Conference on Robotics And Automation, Nagoya, Japan, pp. 1637–1642.

LYUSTERNIK, L. (1964), *Shortest Paths Variational Problems*, Pergamon Press.

MAHADEVAN, S. et CONNELL, J. (1991), Automatic programming of behavior-based robots using reinforcement learning, dans *Proceedings of the AAAI-91*, Vol. 2, AAAI-91, The MIT Press, pp. 768–773.

MARR, D. (1982), *Vision*, W.H. Freeman, San Francisco.

MATARIĆ, M. (1994), Interaction and Intelligent Behavior, Thèse de doctorat, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology.

MATARIĆ, M., NILSSON, M. et SIMSARIAN, K. (1995), Cooperative multi-robot box-pushing, dans *Proceedings of the IEEE/RSJ International Workshop on Intelligent Robots and Systems*, Vol. 3, pp. 556–561.

MATTHIES, L., KANADE, T. et SZELISKI, R. (1989), Kalman filter-based algorithms for estimating depth from image sequences, *International Journal on Computer Vision* **3**, 209–236.

MITSUMOTO, N., FUKUDA, T., SHIMOJIMA, K. et OGAWA, A. (1995), Micro autonomous robotic system and biologically inspired immune swarm strategy as a multi agent robotic system, dans *Proceedings of the IEEE International Conference on Robotics And Automation*, Nagoya, Japan, pp. 2187–2192.

NAKAMURA, T. et ASADA, M. (1995), Motion sketch: Acquisition of visual motion guided behaviors, dans *Proceedings of the International Joint Conference on Artificial Intelligence*, Vol. 1, Montréal, Canada, pp. 126–132.

NAM, Y., LEE, B. et KO, N. (1995), An analytic approach to moving obstacle avoidance using an artificial potential field, dans *Proceedings of the IEEE/RSJ International Workshop on Intelligent Robots and Systems*, Vol. 2, pp. 482–487.

NELSON, R. (1991), Qualitative motion detection by a moving observer, *International Journal on Computer Vision* **7**(1), 33–46.

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

NOREILS, F. (1993), Toward a robot architecture integrating cooperation between mobile robots: Application to indoor environment, *The International Journal of Robotics Research* **12**(1), 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/RSJ International Workshop on Intelligent Robots and Systems*, Vol. 3, pp. 543–548.

OVERMARS, M. et SVESTKA, P. (1996), A paradigm for probabilistic path planning, rapport technique, Department of Computer Science, Utrecht University.

PAGE, L. et SANDERSON, A. (1995), Robot motion planning for sensor-based control with uncertainties, dans *Proceedings of the IEEE International Conference on Robotics And Automation*, Nagoya, Japan, pp. 1333–1340.

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.

POGGIO, T. et KOCH, C. (1985), Ill-posed problems in early vision: from computational theory to analogue networks, dans *Proceedings of the Research Society in London*, Vol. B 226, London, Great Britain, pp. 303–323.

POLOTSKI, V. (1995), Guidance of an articulated vehicle, dans *Proceedings of the IFAC conference on Motion Control.*

PRESS, W., FLANNERY, B., TEUKOLSKY, S. et VETTERLING, W. (1965), *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, Cambridge.

RAVIV, D. et HERMANN, M. (1991), A new approach to vision control for road following, dans *Proceedings of the IEEE Workshop on Visual Motion*, The Institute of Electrical and Electronics Engineers, Inc., IEEE Computer Society Press, pp. 217–225.

REEDS, J. et SHEPP, L. (1990), Optimal paths for a car that goes both forward and backward, *Pacific Journal of Mathematics* **145**, 367–393.

REISTER, D. et PIN, F. (1994), Time-optimal trajectories for mobile robots with two independently driven wheels, *The International Journal of Robotics Research* **13**(1), 38–54.

RIVLIN, E. et ROSENFELD, A. (1994), Navigational functionalities, rapport technique CAR-TR-733, CS-TR-3343, Computer Vision Laboratory, University of Maryland, College Park, MD.

RUDE, M. (1994), Cooperation of mobile robots by event transforms int local space-time, dans *Proceedings of the IEEE/RSJ International Workshop on Intelligent Robots and Systems*, Vol. 3, pp. 1501–1507.

SAHOTA, M. (1994), Reactive deliberation: An architecture for real-time intelligent control in dynamic environments, dans *Proceedings of the AAAI-94*, pp. 1303–1308.

SAITO, F. et FUKUDA, T. (1995), Two-link-robot brachiation with connectionist Q-learning, dans *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*.

SANDERSON, A. et WEISS, L. (1983), Adaptive visual servo control of robots, dans e. A. Pugh, éditeur, *Robot Vision*, Springer-Verlag, New York.

SANTOS-VICTOR, J., SANDINI, G., CUROTTO, F. et GARIBALDI, S. (1993), Divergent stereo for robot navigation: Learning from bees, dans *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 434–439.

SCHÖNER, G. et DOSE, M. (1992), A dynamical systems approach to task level system integration used to plan and control autonomous vehicle motion, *Robotics and Autonomous Systems* **10**, 253–267.

SEKHAVAT, S., SVESTKA, P., LAUMOND, J.-P. et OVERMARS, M. (1996), Multi-level path planning for nonholonomic robots using semi-holonomic subsystems, rapport technique, Department of Computer Science, Utrecht University.

SHARMA, R. (1992), Locally efficient path planning in an uncertain, dynamic environment using a probabilistic model, *IEEE Transactions on Robotics and Automation* **8**(1), 105–111.

SHILLER, Z. et GWO, Y.-R. (1991), Dynamic motion planning of autonomous, *IEEE Transactions on Robotics and Automation* **7**(2), 241–249.

SIMMONS, R. (1996), The curvature-velocity method for local obstacle avoidance, dans *Proceedings of the IEEE International Conference on Robotics And Automation*, Minneapolis, MN, pp. 3375–3382.

SINCLAIR, D., BLAKE, A. et MURRAY, D. (1994), Robust estimation of egomotion using normal flow, *International Journal on Computer Vision* **13**(1), 57–70.

SINGH, S. et KELLY, A. (1996), Robot planning in the space of feasible actions: Two examples, dans *Proceedings of the IEEE International Conference on Robotics And Automation*, Minneapolis, MN, pp. 3309–3316.

SNOW, D. (1967), Determining reachable regions and optimal controls, dans C. Leondes, éditeur, *Advances in Control Systems*, Vol. 5, pp. 133–196.

SPITERI, R., ASCHER, U. et PAI, D. (1995), Numerical solution of differential systems with algebraic inequalities arising in robot programming, dans *Proceedings of the IEEE International Conference on Robotics And Automation*, Nagoya, Japan, pp. 2373–2380.

SUBBARAO, M. (1990), Bounds on time-to-collision and rotational component from first-order derivatives of image flow, *Computer Vision, Graphics and Image Processing* **50**, 329–341.

SUNDAR, S. et SHILLER, Z. (1995), Time-optimal obstacle avoidance, dans *Proceedings of the IEEE International Conference on Robotics And Automation*, Nagoya, Japan, pp. 3075–3080.

SUTTON, R. (1988), Learning to predict by the methods of temporal differences, *Machine Learning* **3**, 9–44.

SUTTON, R. (1990), Integrated architectures for learning, planning, and reacting based on approximating dynamic programming, dans *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, pp. 216–224.

SVESTKA, P. et OVERMARS, M. (1995), Coordinated motion planning or multiple car-like robots using probabilistic roadmaps, dans *Proceedings of the IEEE International Conference on Robotics And Automation*, Nagoya, Japan, pp. 1631 - 1636.

TAKEDA, H., FACCHINETTI, C. et LATOMBE, J.-C. (1994), Planning the motion of a mobile robot in a sensory uncertain field, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **16**(10), 1002–1017.

TAN, M. (1991), Cost-sensitive reinforcement learning for adaptive classification and control, dans *Proceedings of the AAAI-91*, Vol. 2, The MIT Press, pp. 774–780.

TESAURO, G. (1992), Practical issues in temporal difference learning, *Machine Learning* **8**, 257–277.

TESAURO, G. et SEJNOWSKI, T. (1989), A parallel network that learns to play backgammon, *Artificial Intelligence* **39**, 357–390.

THOMPSON, W. et PONG, T.-G. (1990), Detecting moving objects, *International Journal on Computer Vision* **4**, 39–57.

TOMASI, C. et KANADE, T. (1991), Factoring image sequences into shape and motion, dans *Proceedings of the IEEE Workshop on Visual Motion*, The Institute of Electrical and Electronics Engineers, Inc., IEEE Computer Society Press, pp. 21–28.

TSAI, R. et HUANG, T. (1984), Uniqueness and estimation of three-dimensional motion parameters of rigid objects with curved surfaces, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **6**(1), 13–27.

VALLE, S. L. et HUTCHINSON, S. (1996), Optimal motion planning for multiple robots having independent goals, dans *Proceedings of the IEEE International Conference on Robotics And Automation*, Minneapolis, MN, pp. 2847–2852.

VIDAL, T., GHALLAB, M. et ALAMI, R. (1996), Incremental mission allocation to a large team of robots, dans *Proceedings of the IEEE International Conference on Robotics And Automation*, Minneapolis, MN, pp. 1620–1625.

WANG, Z.-H., NAKANO, E. et MATSUKAWA, T. (1994), Cooperating multiple behavior-based robots for object manipulation, dans *Proceedings of the IEEE/RSJ International Workshop on Intelligent Robots and Systems*, Vol. 3, pp. 1524–1531.

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

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

YAMAGUCHI, H. et ARAI, T. (1994), Distributed and autonomous control method for generating shape of multiple mobile robot group, dans *Proceedings of the IEEE/RSJ International Workshop on Intelligent Robots and Systems*, Vol. 2, pp. 800–807.

YOSHIDA, E., ARAI, T., OTA, J. et MIKI, T. (1994), Effect of grouping in local communication system of multiple mobile robots, dans *Proceedings of the IEEE/RSJ International Workshop on Intelligent Robots and Systems*, Vol. 2, pp. 808–815.

ZANARDI, C. (1995), Apprentissage d'une stratégie de poursuite par renforcement, dans *Polyneurones95, Ecole Polytechnique de Montréal*.

ZANARDI, C. (1996), Learning pursuit strategies by reinforcement for a mobile robot, rapport technique GRPR-RT-9601, GRPR, École Polytechnique de Montréal.

ZANARDI, C., HERVÉ, J. et COHEN, P. (1996*a*), Dynamic map: Representation of interactions between robots, dans *Proceedings of the American Association on Artificial Intelligence Conference*.

ZANARDI, C., HERVÉ, J. et COHEN, P. (1996*b*), Mutual learning of unsupervised interactions between mobile robots, dans *Proceedings of the IEEE International Conference on Pattern Recognition*.

ZANARDI, C., HERVÉ, J.-Y. et COHEN, P. (1995), Escape strategy for a mobile robot under pursuit, dans *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, Vancouver, BC. 1430-1436.

ZANARDI, C., HERVÉ, J.-Y. et COHEN, P. (1997*a*), A low cost testbed for the study of interactions between mobile robots, rapport technique GRPR-RT-9702, GRPR, École Polytechnique de Montréal.

ZANARDI, C., HERVÉ, J.-Y. et COHEN, P. (1997*b*), A reactive planner for multiple robots involved in competitive tasks, dans *IEEE Conference on System, Man, and Cybernetics*.

ZHANG, Z., WEISS, R. et HANSON, A. (1994), Qualitative obstacle detection, rapport technique, Computer and Information Science Department, University of Massachuset.

ZHENG, Y. et MOORE, P. (1995), The design of time-optimal control for two-wheel driven carts tracking a moving target, dans *Proceedings of the* 34^{th} *Conference on Decision and Control*, pp. 3831–3836.

ZHU, Q. (1991), Hidden markov model for dynamic obstacle avoidance of mobile robot navigation, *IEEE Transactions on Robotics and Automation* **7**(3), 390–397.

Annexe A

Le filtre de Kalman (base et étendu)

A.1 Principe de base du filtre de Kalman

Le filtre de Kalman est un processus permettant d'obtenir les paramètres d'un système linéaire à partir d'un ensemble de mesures bruitées. Il existe beaucoup d'exemples d'applications basées sur le filtre de Kalman, en contrôle, en commande optimal, en robotique et même en vision (comme le montre par exemple (Matthies, Kanade et Szeliski 1989) et les travaux de Dickmanns (Dickmanns et Graefe 1988*b*)).

Le filtre est dérivé à partir du critère d'optimalité d'un estimateur non biasé au sens des moindres carrés. L'algorithme est en deux phases : une première phase de prédiction de l'état suivant du système et une seconde phase de correction des valeurs en fonction de la variance du bruit et des valeurs mesurées.

Considérons le système linéaire ayant la description dans l'espace d'état suivant :

$$\begin{cases} \mathbf{y}_{k+1} = A_k \mathbf{y}_k + B_k \mathbf{u}_k + \Gamma_k \underline{\xi}_k \\ \mathbf{w}_k = C_k \mathbf{y}_k + D_k \mathbf{u}_k + \underline{\eta}_k \end{cases} \tag{A.1}$$

avec A_k, B_k, Γ_k, C_k, D_k matrices connues de taille respective $n \times n$, $n \times m$, $n \times q$, $q \times n$,

$q \times m$ ($1 \leq m, p, q \leq n$), et avec $\{\mathbf{u}_k\}$, une séquence connue de vecteurs de taille m (appelée séquence d'entrée deterministe) et finalement avec $\{\underline{\xi}_k\}$ et $\{\underline{\eta}_k\}$ des séquences inconnues de bruits correspondant respectivement au système et à l'observation dont les moyennes, variances et covariances sont connues.

Ce genre de système est appelé linéaire deterministe/stochastique à cause de la présence de $\{\mathbf{u}_k\}$, $\{\underline{\xi}_k\}$ et $\{\underline{\eta}_k\}$ dans les équations.

Ce système est décomposable en deux sous systèmes, un linéaire deterministe :

$$\begin{cases} \mathbf{z}_{k+1} = A_k \mathbf{z}_k + B_k \mathbf{u}_k \\ \mathbf{s}_k \ = C_k \mathbf{z}_k + D_k \mathbf{u}_k \end{cases} \tag{A.2}$$

et un purement stochastique :

$$\begin{cases} \mathbf{x}_{k+1} = A_k \mathbf{x}_k + \Gamma_k \underline{\xi}_k \\ \mathbf{v}_k \ = \ C_k \mathbf{x}_k + \underline{\eta}_k \end{cases} \tag{A.3}$$

avec $\mathbf{w}_k = \mathbf{s}_k + \mathbf{v}_k$ et $\mathbf{y}_k = \mathbf{z}_k + \mathbf{x}_k$.

r_k corresponds aux observations à l'instant k de l'état du système (observation partielle).

La solution du premier est donnée par l'équation de transition :

$$\mathbf{z}_k = \Pi_{n=0}^{k-1} A_n z_0 + \Sigma_{i=1}^{k} \Pi_{n=i-1}^{k-1} A_n B_{i-1} \mathbf{u}_{i-1} \tag{A.4}$$

Il faut donc résoudre la solution du deuxième système par un estimateur optimal $\hat{\mathbf{x}}_k$ et obtenir $\hat{\mathbf{y}}_k = \mathbf{z}_k + \hat{\mathbf{x}}_k$.

Les séquences de bruits de mesure et sur le système $\{\underline{\xi}_k\}$ et $\{\underline{\eta}_k\}$ sont blancs. Gaussiens, et de moyenne nulle, avec $Var(\underline{\xi}_k) = Q_k$[1] et $Var(\underline{\eta}_k) = R_k$ matrices positives définies, et avec finalement $E(\underline{\xi}_k \underline{\eta}_l^T)$[2] $\forall k, l$.

---

[1] $Var(x)$ désigne la variance de la variable aléatoire x

[2] $E(x)$ désigne l'espérance mathématique de x

A.2 Prediction/Correction

La formule de calcul en temps réel du filtre de Kalman que l'on dérive à partir d'une méthode d'estimateur optimal au sens des moindres carrés est :

$$\begin{cases} \hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + G_k(\mathbf{v}_k - C_k\hat{\mathbf{x}}_{k|k-1}) \\ \hat{\mathbf{x}}_{k|k-1} = A_{k-1}\hat{\mathbf{x}}_{k-1|k-1} \end{cases} \tag{A.5}$$

La matrice G_k est la matrice de gain de Kalman. Le point de départ de la formule récursive est $\hat{\mathbf{x}}_0 = \hat{\mathbf{x}}_{0|0}$. G_k doit aussi être calculé de façon récursive à l'aide des matrices de variance de l'erreur de mesure et du système.

Ainsi, la formule récursive complète du filtre de Kalman est :

$$\begin{cases} P_{0,0} = Var(\mathbf{x}_0) \\ P_{k,k-1} = A_{k-1}P_{k-1,k-1}A_{k-1}^T + \Gamma_{k-1}Q_{k-1}\Gamma_{k-1}^T \\ G_k = P_{k,k-1}C_k^T(C_kP_{k,k-1}C_k^T + R_k)^{-1} \\ P_{k,k} = (I - G_kC_k)P_{k,k-1} \\ \hat{\mathbf{x}}_{0|0} = E(\mathbf{x}_0) \\ \hat{\mathbf{x}}_{k|k-1} = A_{k-1}\hat{\mathbf{x}}_{k-1|k-1} + B_{k-1}\mathbf{u}_{k-1} \\ \hat{\mathbf{x}}_{k,k} = \hat{\mathbf{x}}_{k|k-1} + G_k(\mathbf{v}_k - D_k\mathbf{u}_k - C_k\hat{\mathbf{x}}_{k|k-1} \\ k = 1, 2, \cdots, \end{cases} \tag{A.6}$$

A.3 Filtre de Kalman étendu

Le filtre de Kalman estime le vecteur d'état d'un modèle linéaire d'un système. Il est cependant possible d'étendre le filtre à des systèmes non linéaires en procédant à une approximation linéaire du modèle (comme par exemple une approximation de Taylor au premier ordre).

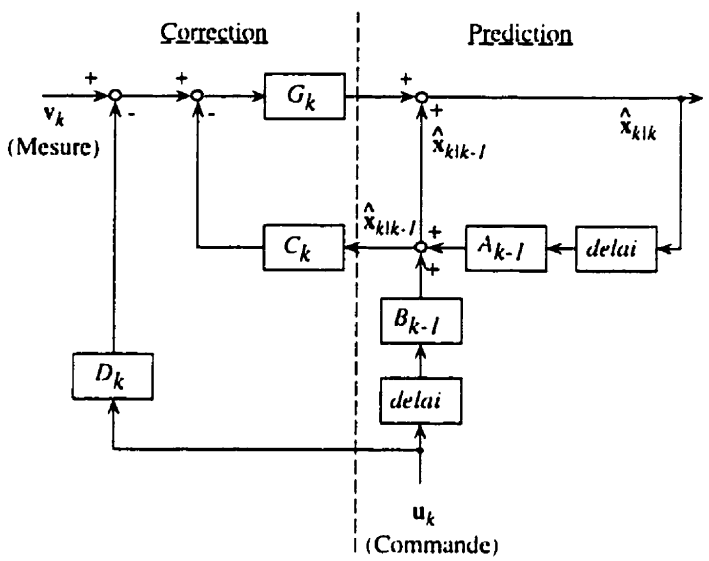


Figure A.1 : Algorithme de prediction/correction du filtre de Kalman

Soit alors le système non linéaire suivant :

$$\begin{cases} \mathbf{x}_{k+1} = \mathbf{f}_k(\mathbf{x}_k) + H_k(\mathbf{x}_k)\underline{\xi_k} \\ \mathbf{v}_k \;=\; \mathbf{g}_k(\mathbf{x}_k) + \underline{\eta_k} \end{cases} \tag{A.7}$$

avec $\mathbf{f}_k$ et $\mathbf{g}_k$ des fonctions vecteurs dont l'espace de définition est respectivement $\mathbb{R}^n$ et $\mathbb{R}^q$ ($1 \leq q \leq n$), et avec H_k une fonction matrice définit dans $\mathbb{R}^n \times \mathbb{R}^q$.

On suppose que $\forall k, l$:

$$E(\underline{\xi_k}\underline{\xi_l}^T) = Q_k \delta_{kl} \tag{A.8}$$

$$E(\underline{\eta_k}\underline{\eta_l}^T) = R_k \delta_{kl} \tag{A.9}$$

$$E(\underline{\xi_k}\underline{\eta_l}^T) = 0 \tag{A.10}$$

$$E(\underline{\xi_l}\mathbf{x}_0^T) = 0 \tag{A.11}$$

$$E(\underline{\eta_l}\mathbf{x}_0^T) = 0 \tag{A.12}$$

Dans le filtre de Kalman étendu, la prédiction est obtenue par :

$$\hat{\mathbf{x}}_{k+1|k}=\mathbf{f}_k(\mathbf{x}_k) \tag{A.13}$$

et la description linéaire de l'espace d'état du système est donnée par :

$$\begin{cases} \mathbf{x}_{k+1}=A_k\mathbf{x}_k + \mathbf{u}_k + \Gamma_k\underline{\xi}_k \\ \mathbf{w}_k = \quad C_k\mathbf{x}_k + \underline{\eta}_k \end{cases} \tag{A.14}$$

où les matrices $A_k, \mathbf{u}_k, \Gamma_k, \mathbf{w}_k$ et C_k doivent être prises de la façon suivante. Considérons les approximations de Taylor à l'ordre deux de $\mathbf{f}_k(\mathbf{x}_k)$ en $\hat{\mathbf{x}}_k$ et celle de $\mathbf{g}_k(\mathbf{x}_k)$ en $\hat{\mathbf{x}}_{k|k-1}$:

$$\begin{cases} \mathbf{f}_k(\mathbf{x}_k)\simeq \quad \mathbf{f}_k(\hat{\mathbf{x}}_k) + A_k(\mathbf{x}_k - \hat{\mathbf{x}}_k) \\ \mathbf{g}_k(\mathbf{x}_k)\simeq\mathbf{g}_k(\hat{\mathbf{x}}_{k|k-1}) + C_k(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1}) \end{cases} \tag{A.15}$$

où

$$A_k=\left[\frac{\partial \mathbf{f}_k}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_k)\right] \tag{A.16}$$

$$C_k=\left[\frac{\partial \mathbf{g}_k}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_{k|k-1})\right] \tag{A.17}$$

De là, en ayant

$$\begin{cases} \mathbf{u}_k = \quad \mathbf{f}_k(\hat{\mathbf{x}}_k) - A_k\hat{\mathbf{x}}_k \\ \Gamma_k = \quad H_k(\hat{\mathbf{x}}_k) \\ \mathbf{w}_k=\mathbf{v}_k - \mathbf{g}_k(\hat{\mathbf{x}}_{k|k-1}) + C_k\hat{\mathbf{x}}_{k|k-1} \end{cases} \tag{A.18}$$

le modèle non linéaire est approximé par le modèle linéaire proposé. La formule de correction est alors :

$$\hat{\mathbf{x}}_k=\hat{\mathbf{x}}_{k|k-1} + G_k(\mathbf{w}_k - C_k\hat{\mathbf{x}}_{k|k-1}) \tag{A.19}$$

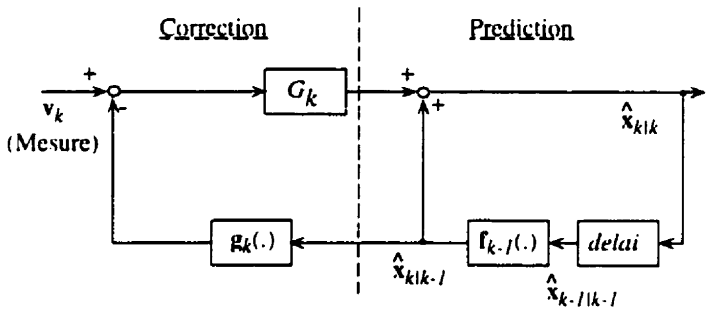


Figure A.2 : Diagramme de l'algorithme de Kalman étendu

$$= \hat{\mathbf{x}}_{k|k-1} + G_k(\mathbf{v}_k - \mathbf{g}_k(\hat{\mathbf{x}}_{k|k-1})) \tag{A.20}$$

L'algorithme complet est donné par :

$$\begin{cases} P_{0,0} = & Var(\mathbf{x}_0) \\ \hat{\mathbf{x}}_0 = & E(\mathbf{x}_0) \\ P_{k,k-1} = & \left[\frac{\partial \mathbf{f}_{k-1}}{\partial \mathbf{x}_{k-1}}(\hat{\mathbf{x}}_{k-1})\right] P_{k-1,k-1} \left[\frac{\partial \mathbf{f}_{k-1}}{\partial \mathbf{x}_{k-1}}(\hat{\mathbf{x}}_{k-1})\right]^T \\ & + H_{k-1}(\hat{\mathbf{x}}_{k-1}) Q_{k-1} H_{k-1}(\hat{\mathbf{x}}_{k-1})^T \\ \hat{\mathbf{x}}_{k|k-1} = & \mathbf{f}_{k-1}(\hat{\mathbf{x}}_{k-1}) \\ G_k = & P_{k,k-1} \left[\frac{\partial \mathbf{g}_k}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_{k|k-1})\right]^T \\ & \cdot \left[\left[\frac{\partial \mathbf{g}_k}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_{k|k-1})\right] P_{k,k-1} \left[\frac{\partial \mathbf{g}_k}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_{k|k-1})\right]^T + R_k\right]^{-1} \\ P_{k,k} = & \left[I - G_k \left[\frac{\partial \mathbf{g}_k}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_{k|k-1})\right]\right] P_{k,k-1} \\ \hat{\mathbf{x}}_{k|k} = & \hat{\mathbf{x}}_{k|k-1} + G_k(\mathbf{v}_k - bfg_k(\hat{\mathbf{x}}_{k|k-1})) \end{cases} \tag{A.21}$$

$$\forall k = 1, 2, \cdots, \tag{A.22}$$

Annexe B

Apprentissage par renforcement

Le principe de base de l'apprentissage par renforcement repose sur un processus d'essais et de récompenses, dans le but d'apprendre une association entre l'état du système et les actions à y appliquer pour maximiser une valeur de performance appelée signal de renforcement (Haykin 1991).

Il existe plusieurs méthodes se basant sur ce paradigme, mais elle s'inspirent principalement de la fameuse méthode de programmation dynamique de Bellman (Bellman 1957) : soit un système dynamique X, avec des états discrets $x_i, i = 0, ..., m$, et un ensemble A également discret d'actions $a_j, j = 0, ..., n$, qui permettent de modifier l'état courant du système. Pour une action $a(t) \in A$ appliquée à l'état courant $x(t)$, on reçoit à l'instant $t+1$ un signal de renforcement $r(t+1)$ indiquant une récompense ou une punition dépendant de l'état $x(t+1)$ atteint et de l'action accomplie.

Pour maximiser la récompense envers les actions à prendre à chaque état, il faut maximiser la valeur de la fonction suivante[1] :

$$J(x) = E\left[\sum_{k=0}^{+\infty} \gamma^k r(k+1) | x(0) = x\right], \tag{B.1}$$

avec γ un paramètre d'influence rétroactive d'une récompense (si $\gamma = 1$ alors la valeur

[1] $E[x]$ correspond à l'espérance mathématique de x

de r lorsque t est à l'infini influence $r(1)$, si $\gamma = 0$ il n'y a pas d'influence entre deux renforcements). A cause de la complexité de $J(x)$, les méthodes d'apprentissage par renforcement tentent seulement de maximiser une estimation de cette fonction de performance.

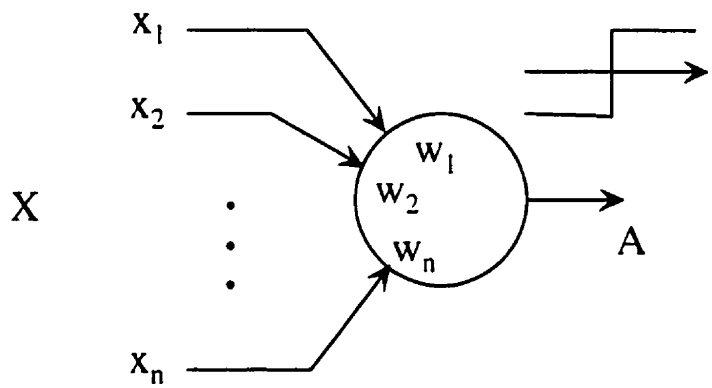


Figure B.1 : Le système de Recherche Associative

Une première méthode, présentée par Barto, Sutton *et al.* (Barto et Sutton 1981, Barto, Sutton et Brouwer 1981, Barto, Sutton et Anderson 1983, Sutton 1990), se base sur un réseau à une seule couche de neurones ayant en entrée l'état X du système et en sortie l'action A. Le changement des poids dans le cas d'un seul neurone à sortie bipolaire est le suivant :

$$w_i(t+1) = w_i(t) + \alpha r(t) e_i(t) \tag{B.2}$$

$$e_i(t+1) = \delta e_i(t) + (1-\delta) a(t) x_i(t-1). \tag{B.3}$$

avec α le taux d'apprentissage, $r(t)$ le renforcement au temps t et $e_i(t)$ une variable indiquant si le poids w_i est éligible à un changement de sa valeur. L'équation B.3 donne la loi d'éligibilité suivant un coefficient d'atténuation δ. Il est possible de rajouter un "critique", sous la forme d'un autre réseau de neurones prédisant la valeur du renforcement $\hat{r}$ suivant les valeurs de l'état courant. Malheureusement, ce type d'apprentissage n'est valide que pour une seule couche de neurones et il est nécessaire

d'avoir des entrées orthogonales. Le nouveau système est donné à la figure B.2. Les lois de prédiction vont être données de la façon suivante : on prend p comme la sortie du deuxième neurone.

$$p(t)=\sum_{i=1}^{n} v_i(t)x_i(t).$$

La loi de changement des poids v_i est :

$$v_i(t+1)=v_i(t)+\beta[r(t)+\gamma p(t)-p(t-1)]\bar{x}_i(t) \tag{B.4}$$

$$\bar{x}_i(t)=\lambda\bar{x}_i(t)+(1-\lambda)x_i(t), \tag{B.5}$$

avec β le taux de changement, γ le facteur d'oubli, et $\bar{x}_i(t)$ une autre fonction d'éligibilité (c'est à dire, une fonction indiquant si, en correspondance avec l'état courant du système, la valeur du neurone doit être modifiée). Une extension de ce

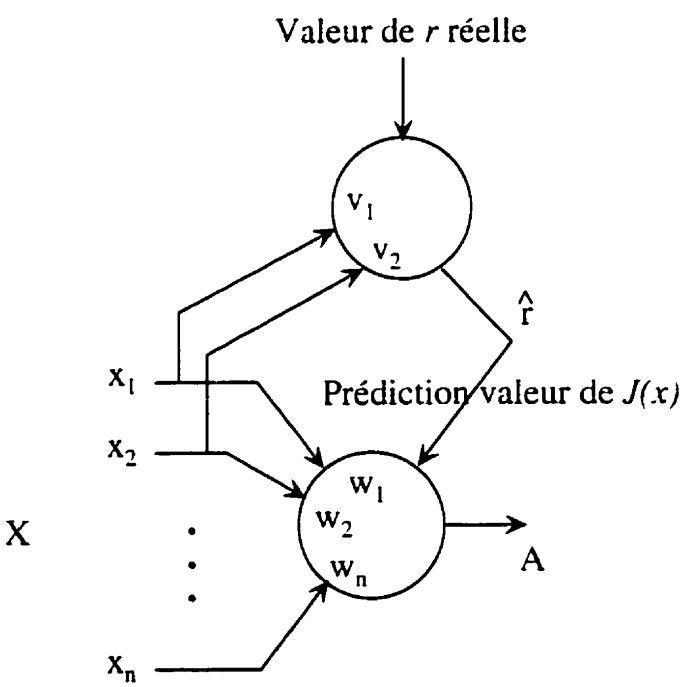


Figure B.2 : Le système "Heuristic Adaptive Critic"

genre d'approche est donnée par Tesauro et Sutton (Sutton 1988, Tesauro et Sejnowski 1989, Tesauro 1992) avec l'apprentissage par "différence temporelle". Le principe de cette méthode est d'analyser les différences entre les prédictions successives et non pas entre les états prévus et observés comme cela est fait usuellement. Un système proposé dans (Tesauro 1992) permet à un ordinateur d'apprendre le backgammon sans connaissance initiale et juste grâce aux résultats de parties jouées en solitaire. La loi de changement des poids s'exprime par :

$$W = W + \sum_{t=1}^{m} \Delta W_t, \quad \Delta W_t = \alpha(P_{t+1} - P_t) \sum_{k=1}^{t} \nabla_W P_k, \tag{B.6}$$

avec α le taux d'apprentissage et P_t la prédiction au temps t.

Par la suite, le fameux algorithme du "Q-learning" fut proposé dans (Watkins 1989). Un grand nombre d'articles furent et sont encore écrits sur l'utilisation ou l'extension de ce principe (Mahadevan et Connell 1991, Lin 1991, Tan 1991, Nakamura et Asada 1995, Littman 1996, Zanardi 1996). Le "Q-learning" permet en effet de rendre explicite la relation entre les entrées et les sorties, par le biais d'une fonction Q à deux variables :

$$Q : (X, A) \mapsto Q(X, A) \in \mathbb{R} \tag{B.7}$$

avec X l'ensemble des états et A l'ensemble des actions. A chaque état $x \in X$, la valeur de $Q(x, a), a \in A$ va indiquer la corrélation relative entre l'état et l'action. $V(x)$ la valeur de Q maximum pour x indique la meilleure action à prendre pour maximiser la valeur du renforcement. L'algorithme procède en prenant une action $a \in A$ suivant une distribution de Boltzman (à température variable T), permettant le passage d'un état x vers un autre état y avec un renforcement associé r, et en changeant $Q(x, a)$ suivant la formule :

$$Q(x, a) = Q(x, a) + \beta\,(r + \gamma V(y) - Q(x, a)) \tag{B.8}$$

$$V(x) = \max_{b \in actions} Q(x, a) \tag{B.9}$$

Un des points forts de l'algorithme est que, pour un processus markovien, la convergence de Q vers la politique optimale est assurée (Watkins 1989), le point faible étant que la discrétisation induit une faible propension à la généralisation, et un coût mémoire énorme. Par exemple, dans (Tan 1991), les états du système représentent les cases d'une grille dans laquelle le robot peut se déplacer suivant quatre directions indiquant chacune une action possible. La volonté du robot est de parvenir au but sans entrer en contact des obstacles et en ayant aucune connaissance *a priori*.

Il est clair, d'après les résultats expérimentaux et certaines expériences personnelles, que le temps de calcul et la place mémoire pour stocker la valeur de $Q(x,y)$ deviennent complètement prohibitifs avec une grille de taille supérieure à 16×16. À cause de ce problème, les espaces de représentation des actions et des états sont généralement simples. Cela ne diminue cependant pas les capacités d'apprentissage de cet algorithme, comme le prouve (Asada, Uchibe, Noda, Tawaratsumida et Hosoda 1994), où un robot apprend à envoyer une balle dans un filet de football tout en évitant un autre robot. Toutefois, il est important de constater que l'espace de représentation devient alors le produit cartésien des espaces des deux tâches (évitement d'obstacle et tir au but) concurrentes et que la procédure d'apprentissage est nettement plus complexe.

Afin de diminuer la taille de cet espace de représentation, plusieurs travaux ont utilisé un réseau de neurones pour faire une approximation de la fonction $Q(x, y)$. Dans (Saito et Fukuda 1995), par exemple, un réseau de type CMAC est utilisé. Ce type de réseau a en entrée les états du systèmes et une liste de paramètres. Le comportement du réseau est contrôlé par les actions sur le système. En sortie, la valeur de Q est donnée, ainsi qu'une mesure de la fiabilité de cette valeur. Ce réseau permet à un robot pendulaire d'apprendre un impressionnant mouvement de brachiation (balancement d'un singe de branche en branche). Plusieurs autres types de réseau de

neurone sont utilisés dans (Lin 1993) pour faire l'approximation de la relation entre les entrées et les sorties d'un système. La réduction de l'espace de représentation ainsi que les possibilités inhérentes de généralisation rendent les réseaux de neurones attrayants pour l'apprentissage par renforcement.

Annexe C

Étude du problème de poursuite/évasion dans le cadre des jeux différentiels

Nous avons introduit succintement les jeux différentiels dans l'étude biblographique du chapitre 2. Dans cette section, nous allons tenter de dériver un ensemble de résultats à partir de la théorie des jeux différentiels. Nous allons dans un premier temps nous intéresser au problème du chauffeur homicide. Par la suite, nous allons dériver l'étude du problème des deux voitures. Dans ces deux exemples, nous allons montrer que l'introduction des capacités de déplacement de chaque robot rend les problème difficile à résoudre. C'est ce genre d'argument qui nous ont poussé vers la construction des Cartes Dynamiques qui permettent l'analyse et la représentation heuristique du problème de poursuite et d'évasion avec plus de 2 opposants. Dans la méthode d'analyse des jeux différentiels, le problème principal est de caractériser les frontières entre des zones de capture ou d'évasion. Ces frontières forment des surfaces semi-perméables, qui correspondent aux terminaisons neutres du jeu différentiel. Sur ces surfaces, pour chaque adversaire il existe un déplacement (matérialisé par la variable de contrôle de l'angle des roues par exemple) qui fait que quel que soit le déplacement

de l'autre participant, le jeu sera neutre.

C.1 Problème du chauffeur homicide

Dans ce problème, on suppose qu'un chauffeur probablement Montréalais tente d'écraser un pauvre piéton. A l'encontre de la voiture, le piéton n'a pas de restriction dans son déplacement. Il peut donc à chaque instant se déplacer omnidirectionnellement. La voiture a par contre une contrainte de courbure maximale. La figure C.1 présente une vue d'ensemble de la situation.

Le mouvement de la voiture est donnée par la formule familière :

$$\begin{cases} \dot{x}_1 = V_1 \cos(\theta_1) \\ \dot{y}_1 = V_1 \sin(\theta_1) \\ \dot{\theta}_1 = \frac{V_1}{L_1} \tan(\phi) \\ |\phi| \leq \quad \phi_{max} \end{cases} \tag{C.1}$$

Le mouvement du piéton est simplement :

$$\begin{cases} \dot{x}_2 = V_2 \cos(\psi) \\ \dot{y}_2 = V_2 \sin(\psi) \end{cases} \tag{C.2}$$

La dimension de l'espace d'état est ici 5 : $(x_1, y_1, x_2, y_2, \theta_1)$. On peut cependant utiliser un espace de dimension réduite avec (x, y) (voir figure C.1).

On obtient alors les équations :

$$f_1(\mathbf{x}, \phi, \psi) = \dot{x} = -\frac{V_1}{L_1} \tan(\phi) * y + V_2 \sin(\psi) \tag{C.3}$$

$$f_2(\mathbf{x}, \phi, \psi) = \dot{y} = \frac{V_1}{L_1} \tan(\phi) * x - V_1 + V_2 \cos(\psi) \tag{C.4}$$

La solution optimale du jeu différentiel est obtenue par l'intermédiaire de l'équa-

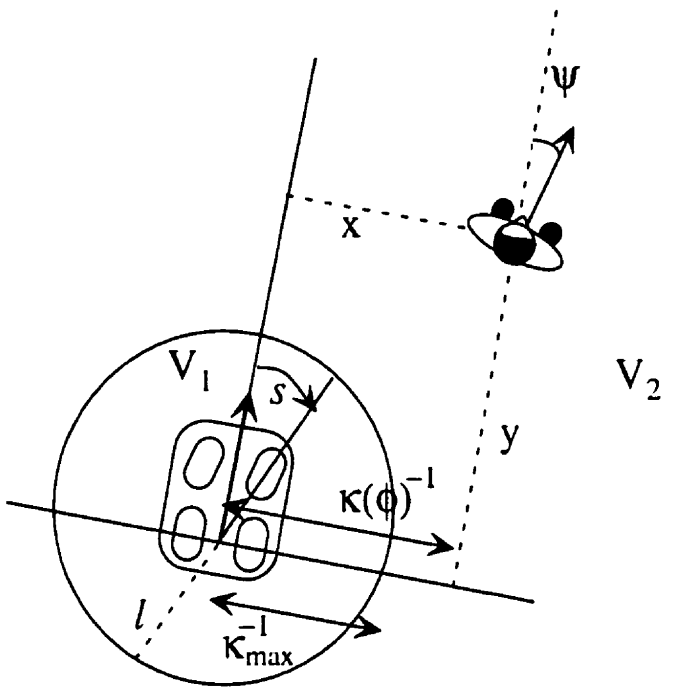


Figure C.1 : Problème du chauffeur homicide : la voiture tente de poursuivre le pauvre piéton

tion principale suivante :

$$\min_{\phi} \max_{\phi} \sum_{i=1}^{n} \nu_i f_i(\mathbf{x}, \phi, \psi) = 0 \tag{C.5}$$

avec n la dimension de l'espace (ici 2), ν_i la normale à la barrière entre les zones de capture et d'évasion (surface semi-perméable), f_i les fonctions du systèmes d'équations différentielles, $\mathbf{x}$ l'état du système. Cette équation suppose bien entendu que le poursuivant (la voiture) tente de minimiser le résultat du jeu (comme par exemple le temps de capture), tandis que le poursuivi tente de maximiser la terminaison du jeu (augmenter le temps avant capture).

Avec les équations C.3 et C.4 à l'interieur de l'équation on obtient :

$$\min_{\phi} \max_{\phi} \left\{ -\frac{V_1}{L_1} [y * \nu_1 - x * \nu_2] \tan(\phi) - V_1 \nu_2 + V_2(\nu_1 \sin(\psi)) + \nu_2 \cos(\psi)) \right\} = 0 \tag{C.6}$$

Posons $A = y * \nu_1 - x * \nu_2$. Alors, la solution optimale du poursuivant s'exprime

par $\bar{\phi} = signe(A)\phi_{max} = \sigma$ et avec $\rho = \sqrt{\nu_1^2 + \nu_2^2}$ alors on obtient :

$$\cos(\bar{\psi}) = \frac{\nu_2}{\rho} \tag{C.7}$$

$$\sin(\bar{\psi}) = \frac{\nu_1}{\rho} \tag{C.8}$$

Afin de permettre la résolution du problème du chauffeur homicide, la technique des équations retrogrades est utilisée. L'idée est relativement simple, les équations différentielles sont inversées de façon à partir d'une position terminale vers la position initiale du système. Les équations s'exprime de façon générale par :

$$\dot{x}_j = -f_j(\mathbf{x}, \bar{\phi}, \bar{\psi}) \tag{C.9}$$

$$\dot{\nu}_j = \sum_i \nu_i f_{ij}(\mathbf{x}, \bar{\phi}, \bar{\psi}) \tag{C.10}$$

Dans le cas du chauffeur homicide, les équations renversées (RPE) sont :

$$\dot{x} = cy - V_2 \frac{\nu_1}{\rho} \tag{C.11}$$

$$\dot{y} = -cx + V_1 - V_2 \frac{\nu_2}{\rho} \tag{C.12}$$

$$\dot{\nu}_1 = c\nu_2 \tag{C.13}$$

$$\dot{\nu}_2 = -c\nu_1 \tag{C.14}$$

$$c = \frac{V_1}{L_1} tan(\sigma) \tag{C.15}$$

Nous allons maintenant déterminer les conditions initiales pour ce système d'équations renversées.

Nous avons défini la zone de capture par le cercle centrée sur la voiture et de rayon l. Une paramétrisation de cette zone est donnée par :

$$x = l \sin s \tag{C.16}$$

$$y = l \cos s \tag{C.17}$$

On recherche les positions initiales utilisables séparant les points sur la zone de capture où une capture est immédiate et inévitable (comme par exemple devant la voiture) et des positions où l'évasion est encore possible. Cette limite (BUP) est caractérisée par :

$$\min_{\phi} \max_{\psi} \sum_{i} \gamma_i f_i(\mathbf{x}, \phi, \psi) = 0 \tag{C.18}$$

avec γ_i le vecteur normal à la surface de terminaison.

Afin de trouver la partie utilisable (où la capture est possible) on considère l'équation suivante :

$$\min_{\phi} \max_{\psi} [\gamma_1 \dot{x} + \gamma_2 \dot{y}] \leq 0 \tag{C.19}$$

puisqu'une valeur négative indique une capture.

On obtient alors :

$$V_2 - V_1 \cos(s) \leq 0 \tag{C.20}$$

En définissant S par $0 \leq S \leq \pi/2$ et $\cos(S) = V_2/V_1$, la partie utile est spécifiée par : $|s| < S$.

On notera que sur la zone de capture, on a $\dot{A} = 0$. De ce fait, il n'est pas possible de déterminer le signe de $\dot{A}$ et ainsi d'obtenir $\bar{\phi}$. Cependant, on peut s'intéresser à calculer la valeur renversée de $\dot{A}$. En différenciant la valeur de A et en utilisant les équations renversées du système, on obtient :

$$\dot{A} = y\dot{\nu}_1 - x\dot{\nu}_2 + \dot{y}\nu_1 - \dot{x}\nu_2 \tag{C.21}$$

$$\dot{A} = y(c\nu_2) - x(-c\nu_1) + \nu_1 \left(-cx + V_1 - V_2 \frac{\nu_2}{\rho} \right) - \nu_2 \left(cy - V_2 \frac{\nu_1}{\rho} \right) \tag{C.22}$$

$$\dot{A} = V_1 \nu_1 \tag{C.23}$$

On pourra donc prendre sur la zone de capture $\sigma = signe(\nu_1)\phi_{max} = signe(s)\phi_{max}$.

Les conditions initiales pour l'intégration des équations renversées sont donc :

$$x = l \sin S \tag{C.24}$$

$$y = l \cos S = l\frac{V_2}{V_1} \tag{C.25}$$

$$\nu_1 = \sin S \tag{C.26}$$

$$\nu_2 = \cos S \tag{C.27}$$

L'intégration des équations C.13 et C.14 donne :

$$\nu_1 = \sin(S + c\tau) \tag{C.28}$$

$$\nu_2 = \cos(S + c\tau) \tag{C.29}$$

De ce fait, les deux premières équations renversées sont alors :

$$\dot{x} = cy - V_2 \sin(S + c\tau) \tag{C.30}$$

$$\dot{y} = -cx + V_1 - V_2 \cos(S + c\tau) \tag{C.31}$$

Les solutions de ces équations, avec les conditions initiales précisées plus haut sont :

$$x = (l - V_2\tau)\sin(S + c\tau) + \frac{L_1}{\tan\sigma}(1 - cos(c\tau)) \tag{C.32}$$

$$y = (l - V_2\tau)\cos(S + c\tau) + \frac{L_1}{\tan\sigma}\sin c\tau \tag{C.33}$$

En intégrant l'équation différentielle correspondant à A, on obtient :

$$A = \frac{L_1}{tan(\sigma)}[\cos S - \cos(S + c\tau)] \tag{C.34}$$

Dans le cas où $\sigma > 0$ (ce qui correspond à la barrière droite), on peut remarquer

que d'après l'équation C.34, A change de signe lorsque $S + c\tau = 2\pi - S$. De ce fait, la barrière droite se termine lorsque $\tau = 2(\pi - S)/c$.

Dans la figure C.2, nous présentons quelques exemples de frontières. Dans le cas de l'exemple de la figure C.2(a), le piéton ne peut jamais s'échapper. Les frontières ne délimite donc pas dans ce cas les positions initiales de capture ou d'évasion, mais plutot change la topologie d'accès de la voiture vers le piéton. Dans la figure C.2(a), la suite des positions relatives du piéton et de la voiture dans le cours de l'éxecution d'un jeu différentiel est indiqué par le trait en pointillé. Il est nécessaire pour la capture que la voiture fasse que cette suite de position ne traverse pas la frontière car il s'agit alors dans ce cas d'un jeu sans gain pour l'un ou l'autre. Lorsque les frontières gauche et droite se croise, alors l'espace est dans ce cas délimité en position de capture et d'évasion. Dans le cas de la figure C.2(c), la zone hachurée correspond à des positions où la voiture peut intercepter le piéton alors que tous le reste de l'espace correspond à une évasion certaine du piéton.

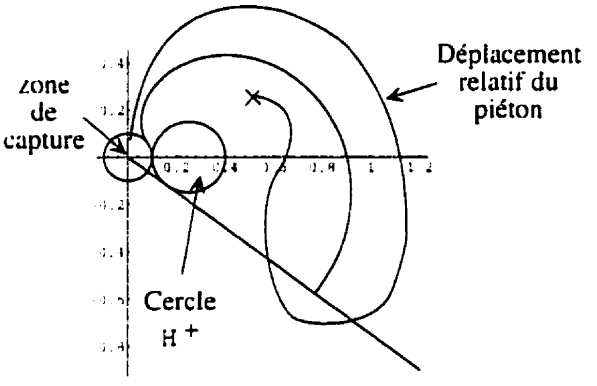
C.2 Problème des deux voitures

Dans le cas du problème des deux voitures, nous avons un poursuivi et un poursuivant ayant des contraintes de courbure maximum sur leurs déplacements. La figure C.3 présente la situation du face-à-face.

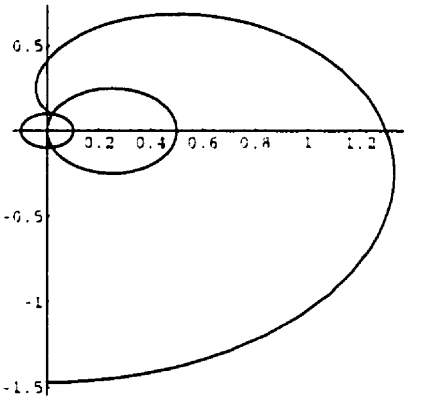
Le déplacement de chaque voiture est modélisée par le système C.1 pour les paramètres respectifs des deux voitures. A nouveau, la voitures possède la capacité de changer l'orientation de ses roues à vitesse infinie.

Nous allons appliquer la même technique que dans le cas du chauffeur homicide. En premier lieu, nous devons décrire les équations différentielles du système des deux voitures :

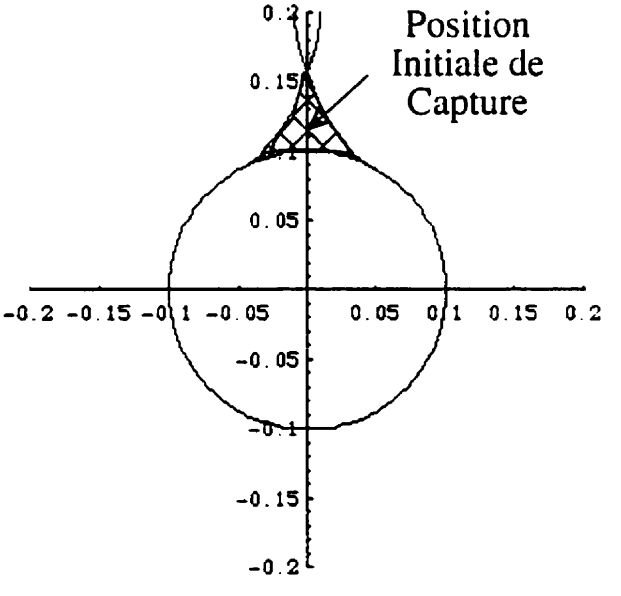
$$f_1(\mathbf{x}, \phi, \psi) = \dot{x} = -\frac{V_1}{L_1} \tan(\phi) * y + V_2 \sin(\theta) \tag{C.35}$$



(a) $V_2 = V_1/2$



(b) $V_2 = V_1$



(c) Intersection des frontières

Figure C.2 : Les frontières de capture pour le problème du chauffeur homicide : (a) $V_2 = V_1/2$ (b) $V_2 = V_1$ (c) Intersection entre frontières droite et gauche.

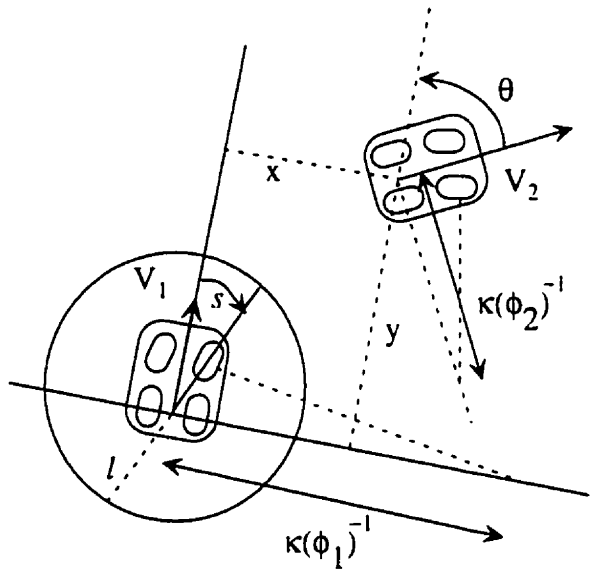


Figure C.3 : Situation du problème des deux voitures.

$$f_2(\mathbf{x}, \phi, \psi) = \dot{y} = \frac{V_1}{L_1} \tan(\phi) * x - V_1 + V_2 \cos(\theta) \tag{C.36}$$

$$f_3(\mathbf{x}, \phi, \psi) = \dot{\theta} = -\frac{V_1}{L_1} \tan(\phi) + \frac{V_2}{L_2} \tan(\psi) \tag{C.37}$$

L'équation principale est alors :

$$\min_{\phi} \max_{\psi} \{ -\frac{V_1}{L_1} \tan\phi + V_2(\nu_1 \sin\theta + \nu_2 \cos\theta) + \frac{V_2}{L_2} \nu_3 \tan\psi - V_1 \nu_1 = 0 \tag{C.38}$$

$$A = \nu_1 y - \nu_2 x \tag{C.39}$$

Ce qui donne alors les conditions :

$$\tilde{\phi} = \sigma_1 = signe(A) * \phi_{max} \tag{C.40}$$

$$\tilde{\psi} = \sigma_2 = signe(\nu_3) * \psi_{max} \tag{C.41}$$

Les équations renversées sont alors :

$$c = \frac{V_1}{L_1} \tan\sigma_1 \tag{C.42}$$

$$\dot{x} = cy - V_2 \cos\theta \tag{C.43}$$

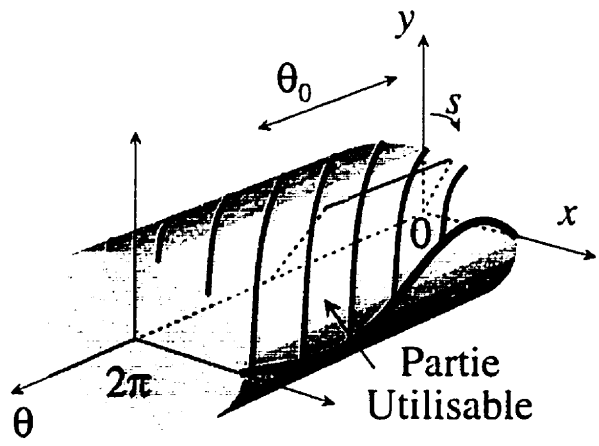


Figure C.4 : Cylindre de capture : la séparation permet de connaître les portions utilisables de ce cylindre pour générer les solutions aux jeux différentiels.

$$\dot{y} = -cx + V_1 - V_2 \cos\theta \tag{C.44}$$

$$\dot{\theta} = c - \frac{V_2}{L_2} \tan\sigma_2 \tag{C.45}$$

$$\dot{\nu}_1 = c\nu_2 \tag{C.46}$$

$$\dot{\nu}_2 = -c\nu_1 \tag{C.47}$$

$$\dot{\nu}_3 = V_2(\nu_1 \cos\theta - \nu_2 \sin\theta) \tag{C.48}$$

et on a aussi $\dot{A} = V_1\nu_1$.

Pour les conditions initiales, on parametrise la zone de capture par s (voir figure C.3) et $\theta_0 \in [0; 2\pi[$:

$$x = l\cos s \tag{C.49}$$

$$y = l\sin s \tag{C.50}$$

$$\theta = \theta_0 \tag{C.51}$$

La figure C.4 présente le volume de capture dans l'espace x, y, θ. En prenant $r^2 = x^2 + y^2$ et en différenciant, on obtient :

$$r\dot{r} = \dot{x}x + \dot{y}y \tag{C.52}$$

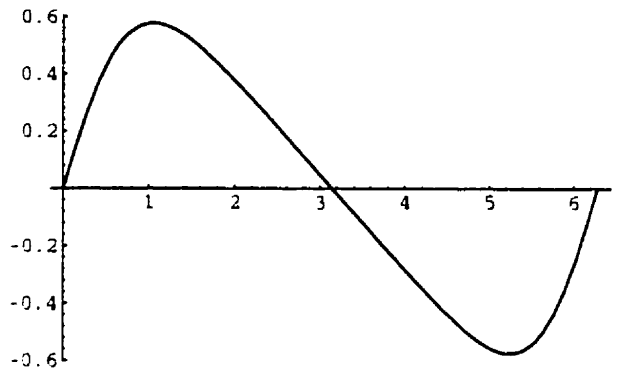


Figure C.5 : Projection sur zone de capture

$$=x(V_2 \sin\theta) + y(-V_1 + V_2 \sin\theta) \tag{C.53}$$

Lorsque que l'on se trouve sur la zone de capture, on a $r = l$ et $\dot{r} = 0$ ce qui définit alors la partie utilisable par :

$$0=l \sin s(V_2 \sin\theta_0) - l \cos s(V_1 - V_2 \cos\theta_0) \tag{C.54}$$

$$\text{atan}s = \frac{V_1 - V_2 \cos\theta_0}{V_2 \sin\theta_0} \tag{C.55}$$

La forme générale de la fonction atan$s$ obtenue est donnée dans la figure C.5 et projeté sur la zone de capture dans la figure C.4.

Il est maintenant nécessaire de déterminer les valeurs initiales du signe de $\dot{A}$ et de ν_3 sur la zone de capture. Sur cette zone, $\nu_1 = \sin s$ et $\nu_2 = \cos s$ (normale au cercle de capture), et $\nu_3 = 0$. Ce qui donne finalement $\dot{A} = 0$.

Les signes de $\dot{A}$ et ν_3 vont donc dépendre des valeurs des dérivées renversées (équation C.42 à C.48) $\ddot{A}$ et $\dot{\nu}_3$. On a donc $signe\sigma_1 = signe\ddot{A} = signe\nu_1 = \sin s$ qui est positif à droite (axe des x positif) et négatif de l'autre coté. D'un autre coté, on a $signe\sigma_2 = signe\dot{\nu}_3 = signe(V_1 \cos\theta_0 - V_2)$. Nous allons supposer par la suite que l'on s'intéresse au cas où $V_1 > V_2$ et σ_1 positif.

En étudiant le signe de σ_2, il est facile de voir que pour S tel que $\cos S = V_2/V_1$ on a trois intervalles pour σ_2 :

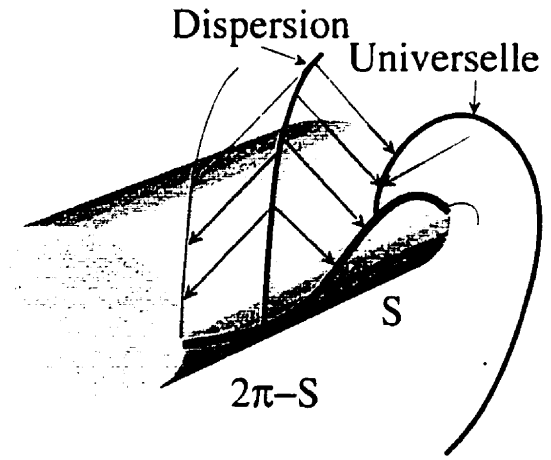


Figure C.6 : Partage entre capture et évasion

1. $\theta_0 \in [0; S[, \sigma_2 = 1$

2. $\theta_0 \in [S; 2\pi - S[, \sigma_2 = -1$

3. $\theta_0 \in [2\pi - S; 2\pi[, \sigma_2 = 1$

Les brusques changements de signe produisent soit des courbes de dispersion ou des courbes universelles. Les premières correspondent à des courbes pour lesquelles les solutions ayant une fin de jeu nulle (équivalent de la courbe trouvée dans l'exemple du chauffeur homicide) divergent à partir de cette courbe. Les courbes universelles correspondent au contraire à une fusion d'un ensemble de courbes. La figure C.6 présente une idée des deux types de courbes obtenues sur la surface à jeu nulle. Cette surface partage de nouveau l'espace entre les zones de capture et d'évasion.

C.3 Conclusion sur les jeux différentiels

Les jeux différentiels nous ont permis de générer explicitement les barrières d'atteignabilité dans le cadre d'une tâche de capture entre deux robots. La représentation des barrières devient cependant de plus en plus fastidieuse à chaque fois qu'une dimension est ajoutée au problème. Par exemple, pour traiter le cas où les deux voitures ont une contrainte de courbure maximale du chemin mais également une contrainte

sur la vitesse de changement de l'orientation induit d'étuder la forme de la surface utile dans un espace de dimension 5.

Par définition, les jeux différentiels ont besoin des modèles dynamiques de chaqu'un des joueurs et il ne sera donc possible de faire qu'un apprentissage paramètrique des valeurs d'un modèle de robots. Il semble également difficile d'intégrer les conditions environnementales (tel que des obstacles) à l'intérieur du jeu différentiel lui-même.

Annexe D

Etalonnage des véhicules

La phase d'étalonnage est une étape primordiale de la construction d'un robot. Il est en effet nécessaire de connaître à l'avance la relation qui existe entre les différents actuateurs du robot (dans notre cas les potentiomètres des télécommandes) et les performances réelles du robot (sa vitesse, la courbure de son déplacement ou son accélération en fonction des tensions envoyées au télécommande). Lorsque cette relation est connue, il est alors possible de s'abstraire des détails techniques (ici la modélisation des tensions applicables) et de commander le robot par l'intermédiaire de techniques usuelles de contrôle en accélération, vitesse ou position. L'etalonnage obtenu permet ainsi de procéder à une simple interpolation entre les valeurs désirées de contrôle et les tensions à appliquer.

Une automatisation de la technique de calibrage décrite par la suite a été réalisée dans le cadre du projet de fin d'études de Fabienne Lathuilière (Lathuilière 1997) sous la supervision conjointe de Philippe Lacroix et Christian Zanardi. Nous allons décrire dans cette partie la technique générale d'étalonnage de nos véhicules. Cette phase d'étalonnage est divisée en trois parties consécutives : la calibration du rayon de courbure, de la vitesse et de l'accélération du véhicule.

D.1 Estimation du rayon de courbure

L'étalonnage de l'orientation des roues du robot consiste à obtenir pour chaque valeur désirée de l'orientation des roues du véhicule (ou la valeur équivalente du rayon de courbure de la trajectoire), la valeur de la tension à appliquer à la télécommande du véhicule. Cette phase se déroule en deux étapes : en premier lieu, les caractéristiques importantes du déplacement du robot, obtenues lors de l'application d'un ensemble de valeurs de tension possible, sont extraites et analysées. Par la suite, lors de la deuxième étape, les données ainsi obtenues sont alors utilisées pour générer la fonction reliant le rayon de courbure désiré à la tension à appliquer à la télécommande.

Calcul du rayon de courbure : Le but est, ici, de trouver la fonction de correspondance entre la tension V et la courbure κ du robot.

$$f : V \longmapsto \kappa = f(V).$$

En utilisant le système d'odométrie comme un observateur central, nous pouvons obtenir les positions absolues et les orientations du robot dans son espace de travail. Afin d'obtenir le rayon de courbure du robot (le rayon de courbure R est inversement proportionnel à la courbure κ), une tension constante est appliquée à la fois pour la vitesse longitudinale du robot et pour l'angle de ses roues. Le robot décrit alors un cercle dont le rayon correspond au rayon de courbure de la trajectoire.

Afin de pallier aux erreurs de localisation de l'odomètre, la trajectoire du robot est approximée par un cercle parfait, dont le centre O et le rayon R sont obtenues par un algorithme de minimisation dite du simplexe. Cette algorithme utilise comme fonction de coût l'erreur quadratique moyenne entre les points formant le cercle et les paramètres $O = (x_0, y_0)$ et R :

$$\mathcal{E}(x_O, y_O, R) = \frac{1}{N} \sum_{i=1}^{N} \left[R^2 - (x_i - x_O)^2 - (y_i - y_O)^2 \right]^2. \qquad \text{(D.1)}$$

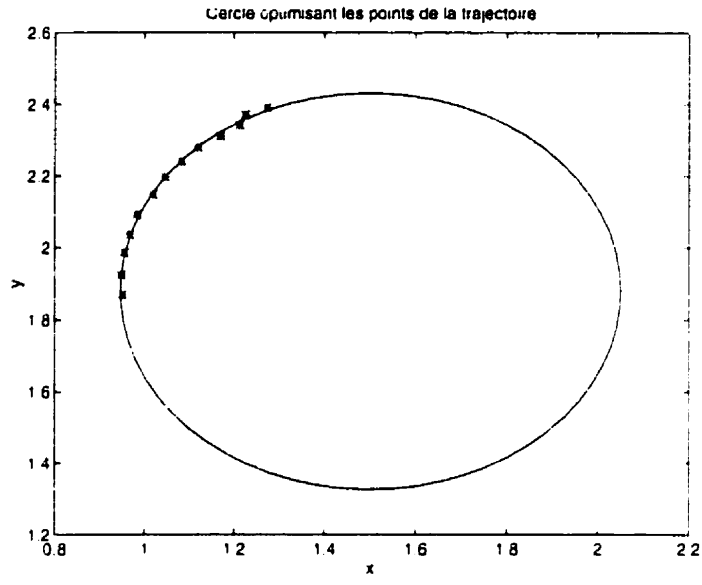


Figure D.1 : Cercle obtenue par la minimisation du simplexe sur les points désignés par les étoiles.

(x_i et y_i) étant l'ensemble des N couples de points extraits par l'odomètre. L'algorithme du simplexe (Press, Flannery, Teukolsky et Vetterling 1965) est ensuite directement appliqué pour obtenir les valeurs de x_0, y_0 et R.

La figure D.1 présente un exemple de cercle obtenue par la procédure de minimisation du simplexe appliquées sur les points extraits par l'odométrie simulée.

Interpolation des tensions : L'étape précédente d'étalonnage a permis d'obtenir l'association d'une tension à un rayon de courbure. Afin de permettre le contrôle du robot par l'intermédiaire du rayon de courbure (ou par la courbure $\kappa = 1/R$ ou l'angle des roues du robot $\tan(\phi)/L = 1/R$), il faut maintenant pouvoir générer pour toute valeur du rayon de courbure la tension correspondante. Pour cela, nous allons interpoler les valeurs discrètes obtenues lors de l'étape précédente. Le but est donc d'obtenir la fonction suivante :

$$f : \kappa \longmapsto V = f(\kappa).$$

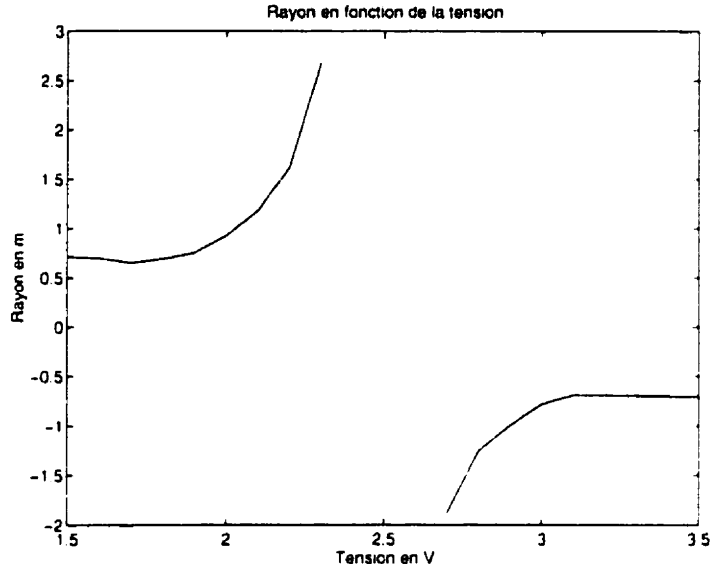


Figure D.2 : Allure des courbes exprimant le rayon de courbure en fonction de la tension de braquage appliquée.

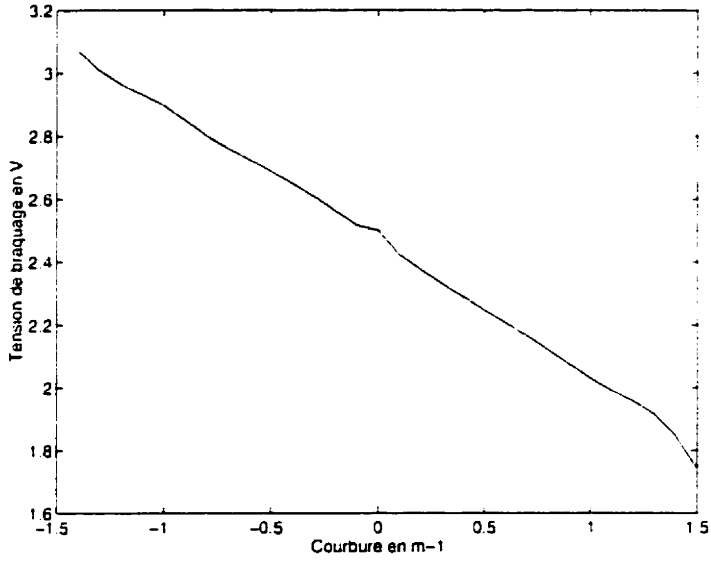


Figure D.3 : Interpolation des valeurs de tension en fonction de la courbure.

L'interpolation directe des rayons de courbure est assez difficile, essentiellement à cause de la forme hyperbolique de la courbe (voir figure D.2).

Nous avons alors choisi d'interpoler la courbure de la trajectoire ($\kappa = 1/R$) qui sera quasiment linéaire, par des splines cubiques. Le résultat de l'interpolation des courbures est présenté dans la figure D.3.

D.2 Etalonnage de la vitesse et de l'accélération

L'objectif est ici de parvenir à modéliser les caractéristiques dynamiques des véhicules en fonction des tensions appliquées. Dans le cas d'un contrôle en accélération par exemple, la valeur de l'accélération est fonction de la vitesse actuelle du robot. Dans ce cas, un tableau à deux dimensions (Vitesse et Accélération) est utilisée donnant pour une vitesse donnée et une accélération désirée, la tension à appliquer correspondante. On ne peut pas, en effet, calibrer la vitesse indépendamment de l'accélération instantanée.

Pour obtenir une loi exprimant la vitesse en fonction de la tension, il serait normalement facile d'observer la réponse du véhicule à une rampe de tension, afin d'approximer la vitesse par une réponse du premier ordre par exemple. Il est cependant délicat d'obtenir une rampe de tension, étant donné que les niveaux de tension sont discrétisés et le fait que l'on dispose de relativement peu de points pour visualiser la trajectoire sur un intervalle de temps suffisamment long. De ce fait, nous utiliserons des courbes qui seront les approximations aux moindres carrés des accélérations et vitesses des véhicules.

Etalonnage de la vitesse seule : la vitesse du véhicule est extraite simplement à partir de l'odomètre central pour une trajectoire linéaire et une tension constante appliquée. Pour les deux véhicules les résultats obtenus sont les suivants :

- Voiture 1 (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 (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

La fonction donnant la tension correspondante à la valeur de vitesse constante désirée est décrite par une simple interpolation linéaire entre les différentes couples tension/vitesse obtenus.

Etalonnage vitesse/accélération : au véhicule initialement au repos, une tension constante est appliquée. L'odomètre permet alors d'acquérir le profil de vitesse correspondant au passage d'une vitesse nulle à la vitesse finale désirée.

Expérimentalement, nous avons observer que la vitesse en fonction du temps était similaire à une exponentielle. La réponse de la vitesse à un échelon de tension en fonction du temps est ainsi décrite par la courbe exponentielle à trois paramètres (V_{max}, Δ, τ) d'équation suivante:

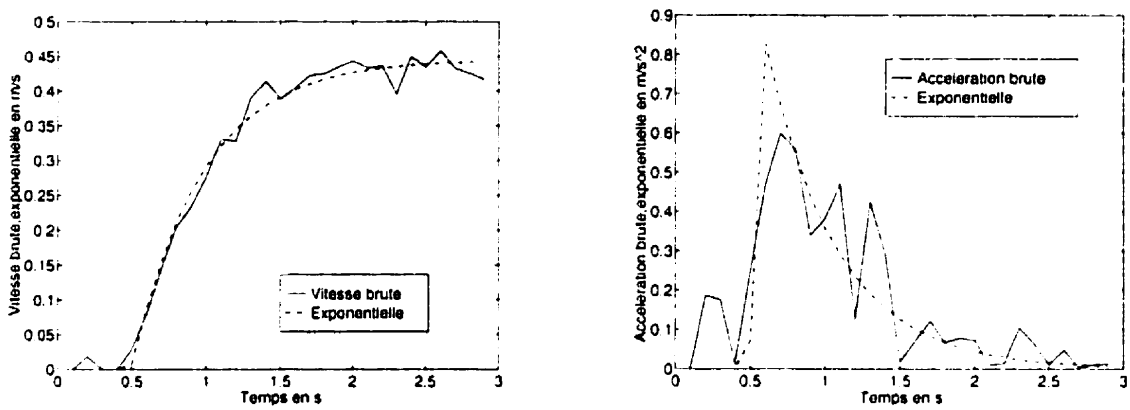
$$V(t) = V_{max}(1 - e^{-\frac{t-\Delta}{\tau}}) \tag{D.2}$$

La technique de minimisation du simplexe est utilisée afin de réduire l'écart entre la courbe obtenue expérimentalement par l'odomètre de la vitesse en fonction du temps et l'exponentielle. L'optimisation est faite dans ce cas par rapport aux trois paramètres V_{max}, Δ, τ. La figure D.4(a) présente un exemple de profil de vitesse approximé par une fonction de ce type (équation D.2).

La dérivée $A(t) = dV(t)/dt$ de la fonction $V(t)$ par rapport au temps permet de calculer simplement l'accélération du véhicule (voir figure D.4(b)). Les courbes de tension observées expérimentalement en fonction de l'accélération et d'une vitesse donnée ont l'allure de paraboles (voir figure D.5). La courbe d'équation suivante :

$$U_{vitesse}(t) = c - a.A(t)^2 \tag{D.3}$$

est ainsi choisi pour interpoler la valeur de tension en fonction de la vitesse et de l'accélération. Cette fonction est à nouveau optimisé par l'algorithme du simplexe pour choisir les valeurs adéquates de a et c.



(a) Vitesse (b) Accélération

Figure D.4 : Vitesse et accélération approximées par une exponentielle

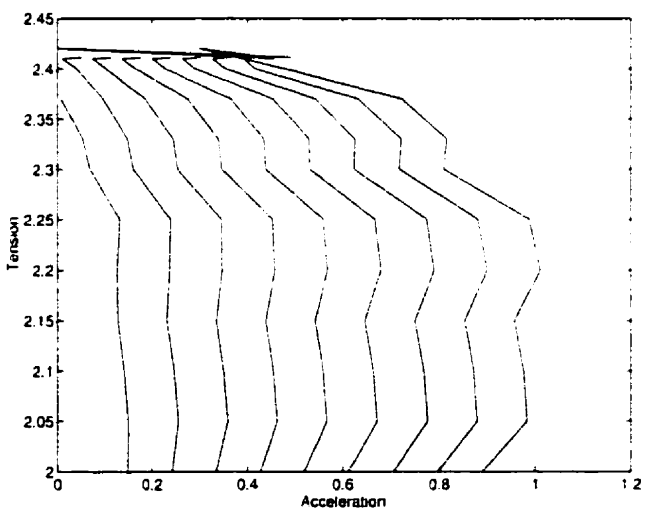


Figure D.5 : Tension en fonction de l'accélération pour une vitesse donnée

Une procédure similaire est utilisée pour l'étalonnage de la voiture en marche arrière et en décélération. La figure D.6 présente finalement l'ensemble des courbes permettant d'obtenir la valeur de la tension pour une accélération positive ou négative par rapport à l'ensemble des valeurs de vitesse.

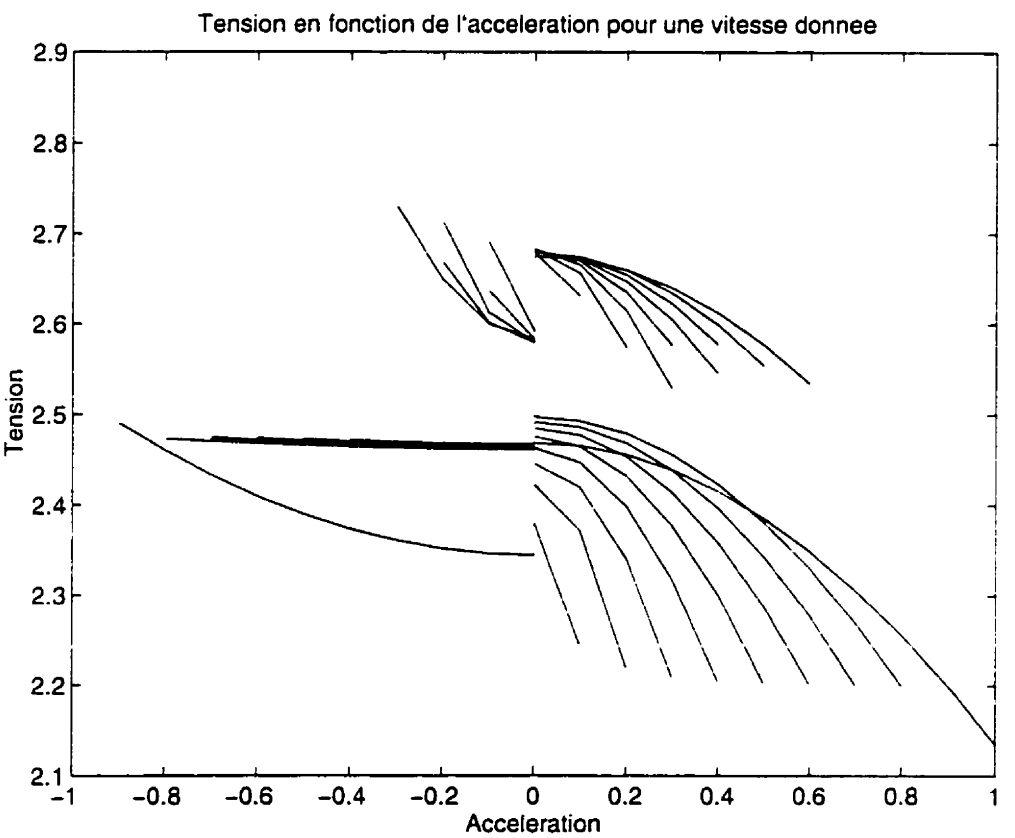


Figure D.6 : Réseau de courbes de calibration pour une vitesse donnée dans les quatre quadrants