RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

École Nationale Polytechnique

Département d'Automatique

Mémoire en vue de l'obtention
du diplôme d'Ingénieur et du
diplôme de Master en Automatique

Thème :

COMMANDE RÉFÉRENCÉE VISION DU ROBOT MOBILE ROBUCAR

Présenté par :

AMMOUR Rabah
BOUSSIF Abderraouf

Encadré par :

Dr. N. OUADAH
Pr. M. TADJINE

Juin 2012

قُلْ إِنَّ صَلَاتِي وَنُسُكِي وَمَحْيَايَ وَمَمَاتِي لِلَّهِ رَبِّ الْعَالَمِينَ لَا شَرِيكَ لَهُ وَبِذَلِكَ
أُمِرْت وَأَنَا أَوَّل الْمُسْلِمِينَ

REMERCIEMENTS

Nous tenons à exprimer notre gratitude, notre reconnaissance et nos profonds remerciements à notre encadreur Monsieur Mohamed TADJINE pour ses conseils, sa disponibilité, ainsi que le précieux savoir qu'il nous a transmis.

Nous adressons nos sincères remerciements à notre encadreur du (CDTA) Monsieur Noureddine OUADAH qui sans lui, ce travail ne serait pas débuté. Pour sa disponibilité, son aide ainsi que sa perpétuelle bonne humeur.

Nous remercions chaleureusement les membres du jury, Monsieur BOUCHRIT et Monsieur HMICI, pour l'honneur qu'ils nous ont fait en acceptant d'évaluer notre travail.

Nous ne pouvons pas oublier de remercier tous les enseignants de l'école nationale polytechnique (ENP), et plus particulièrement ceux de notre département d'Automatique, ainsi que ceux de l'Ecole Nationale Préparatoire aux Etudes d'ingéniorat (ENPEI).

A tous les « chercheurs » du Centre de Développement des Technologies Avancées. En particulier les membres de la division Robotique et Productique que nous avons côtoyés et appréciés durant toute la durée de notre projet.

Enfin nous remercions tous nos camarades des deux écoles qui nous ont vus devenir ce que nous sommes aujourd'hui… !

R & R.

REMERCIEMENT :

Tout d'abord, je remercie Dieu le tout puissant de m'avoir donné le courage, la volonté, la patience et la bonne santé durant toutes ces années.

Je voudrais également exprimer mon immense gratitude envers mes parents, mes frères et sœurs, ma grand-mère, ainsi que ma grande famille qui, sans nécessairement partager mes rêves, m'ont depuis toujours fait confiance et ont répondu présents dans les bons moments comme dans les moins.

Je n'oublierai non plus de remercier mes amis de ***Rouiba***, spécialement « Section 313 », ainsi que mes amis de ***polytech***, particulièrement les automaticiens, et bien sur tous mes amis d'enfance à ***Babor*** .

Si je devrais citer des prénoms ils seront, mon binôme l'artiste ***Rabah*** (pour tout ce qu'on a partagé entre D117 & C116), ***Mahdus, Med Cherif, Tahar, Raouia & Zed*** .

Sans oublier les huit nains :

Aminette, Badro, Mayssa, Toufik,Hamza, Basset et zakooo.

Et puisqu'il est coutume de garder la meilleure pour la fin, je tiens à remercier tout particulièrement le P'tit Ange

DALIA KAMELIA ABAD !

(La vie a ses raisons que notre raison ignore!)

! Raouf Boussif

Dédicace ...

Avant tout à mes très chers parents qui m'ont tout offert...

A toute ma famille qu'elle soit proche ou lointaine

A toutes mes sœurs : Samia, Nadia, Dalila et leurs maris.

A mes frères : Farid, Djamel ainsi que leurs femmes.

A tous mes petits neveux et nièces.

A celle qui a toujours occupé mon imagination ...

Au traceur de chemins qui me fera sans doute une remarque en lisant cette page...

A tous mes ami(e)s et camarades

Rabah AMMOUR

العنوان: تطبيق تقنيــات التحكم المرئي على آلي متحرك ***(Robucar)***

الملخص:

تتناول هذه المذكرة إشكالية التحكم المرئـي في آلي متحرك ذو عجلات, (قاعدة تجارب بمركز تنمية التكنولوجيـات المتقدمة) المهمة المقترحة تتمثل في تتبع مرئي ببعدين اعتمادا على مستخرجات مرئية ,مستقيمات أو قطع مستقيمة. كحل أولي قمنــا باقتراح تقنية تحكم مبنية على مبدأ " دالـــة المهمة". سعيا إلى تحسين أداء الآلي المتحرك أدرجنــا تقنية تحكم متينــة باعتماد المنطق الغــامض.
من أجل التحقق من فعــالية التــقنيــتين قمنــا بتجريبهمــا على برنــامج محاكاة إضافة إلى تجارب ميدانية على الآلي المتحرك استطعنا من خلال هذا العمل الحصـول على النتــائج المرجوة و بهذا فقد تم تزويد الآلي المتحرك بقدرة تتبع مرئيــة

المفاتيح: آلي متحرك, تحكم مرئي, معالجة الصورة, المنطق الغامض

Le titre : Commande référencée vision du robot mobile ***Robucar***

Résumé :

Le travail présenté dans ce mémoire de fin d'étude traite la problématique de la commande référencée vision du robot mobile de type voiture Robucar (Plateforme expérimentale du Centre de développement des technologies avancées -CDTA-). La démarche proposée est basée sur un asservissement visuel 2D du robot mobile à partir de primitives visuelles de type droites ou segments de droite. Ceci consiste à synthétiser les lois de commande dans l'espace capteur.
La première solution retenue pour résoudre le problème de commande est basée sur le « formalisme de fonction de tâche ». La seconde solution que nous avons adoptée est un régulateur de type flou. Ceci dans le but de bénéficier des caractéristiques de robustesse et de performance de ce dernier.
Tout au long de ce travail, les deux préoccupations ont été d'une part, de valider nos deux lois de commande en simulation, puis dans le cadre d'une implémentation réelle sur le robot mobile Robucar.
A travers ce travail, nous avons abouti à des résultats satisfaisants vis-à-vis de l'application visée. Ceci nous a permis de doter le Robucar de nouvelles capacités de suivi de trajectoire.

Mots clés : robot mobile, asservissement visuel, traitement d'image, logique floue.

Title : Visuel- based control applied to a robot mobile ***Robucar***

Abstract :

This work mainly adresses the problem of visual- based control applied to a non holonomic wheeled mobile robot Robucar (an experimental platform of CDTA) by using the projection of straight lines in the image plane of camera.
The first solution retained in order to solve the problem of control consists in a classical visual servoing, based on the "task function" theory. The second solution we adopted is a more general approach; it consists in a fuzzy controller type 1 which is a robust technique.
Both control techniques are designed and validated through simulation results and real experiments which gave satisfactory results.

Key words: mobile robot, visual servoing, image processing, fuzzy logic

Table des matières

Abréviations et symboles

Abréviations

RMA	Robot Mobile Autonome
PID	Proportionnel intégral dérivé

Symboles

R_0	Repère lié à la scène
R_M	Repère lié à la base du robot
R_P	Repère lié à la platine
R_C	Repère lié à la caméra
C	Matrice de combinaison
e	Fonction de tâche
$r(t)$	Trajectoire du robot dans le repère cartésien
v	La vitesse linéaire du robot (m/s)
z	Distance caméra- droite
f	Distance focale de la caméra (mm)
J_r	Jacobien du robot
λ	Gain de l'asservissement visuel
ω, Ω	Vitesse de rotation (rd/s)
Kp	Gain proportionnel du régulateur PID
Ki	Gain intégral du régulateur PID
Kd	Gain dérivé du régulateur PID
l	Abscisse curviligne du robot
L_s	Matrice d'interaction
L_s^+	Pseudo inverse de la matrice d'interaction
q	Vecteur de configuration du robot
θ	Orientation du robot
τ, T	Torseur cinématique du robot
$s = (\theta, \rho)$	Indices visuelles (angle, rayon)
$U_0, V_0, \alpha_u, \alpha_v$	Caractéristiques intrinsèques de la caméra
Te	Période d'échantillonnage

Introduction générale

Depuis toujours, l'Homme a su se doter d'outils performants pour prolonger sa main et réaliser des actions qui n'auraient pas été possible sans. Ainsi depuis les premiers automates jusqu'aux systèmes disponibles en ce début de XXIe siècle, on mesure tout à la fois le chemin parcouru et celui restant à parcourir avant de réaliser les rêves qui animaient les pionniers. La robotique est un domaine de recherche qui se situe au carrefour de l'intelligence artificielle, de l'automatique, de l'informatique et de la perception par ordinateur ; cette interdisciplinarité est à l'origine d'une certaine complexité. Des applications dans des domaines aussi variés que l'industrie manufacturière, le spatial, l'automobile ou plus récemment les loisirs et le secteur médical, démontrent aujourd'hui l'intérêt économique et social de ces recherches.

Historiquement, l'étude des robots mobiles est venue assez tôt, suivant celle des robots manipulateurs, au milieu des années 70. Leur faible complexité en a fait de bons premiers sujets d'étude pour les roboticiens intéressés par les systèmes autonomes. Cependant, malgré leur simplicité apparente (mécanismes plans, à actionneurs linéaires), ces systèmes ont soulevé un grand nombre de problèmes difficiles. Nombre de ceux-ci ne sont d'ailleurs toujours pas résolus. Ainsi, alors que les robots manipulateurs se sont aujourd'hui généralisés dans l'industrie, rares sont les applications industrielles qui utilisent des robots mobiles. Si l'on a vu depuis peu apparaitre quelques produits manufacturiers (chariots guides) ou grand public (aspirateur), l'industrialisation de ces systèmes bute sur divers problèmes délicats. Ceux-ci viennent essentiellement du fait que, contrairement aux robots manipulateurs Prévus pour travailler exclusivement dans des espaces connus et de manière répétitive, les robots mobiles sont destinés à évoluer de manière autonome dans des environnements peu ou pas structurés.

La volonté des chercheurs de reproduire les capacités humaines de perception et d'action dans les systèmes robotisés a conduit à l'intégration de données issues de capteurs extéroceptifs, et plus particulièrement de celle issus d'une caméra. De plus, Les derniers développement dans le domaine de la technologie des capteurs de vision et du traitement d'image ont permis l'intégration des informations visuelles dans la boucle de réglage. Ainsi l'utilisation de la vision artificielle en robotique mobile trouve un vaste champ d'application avec comme capteur principal, la caméra qui est essentielle pour la résolution des problèmes de perception.

Problématique

Notre travail aborde la problématique de la commande référencée vision du robot mobile *Robucar* par la mise en œuvre d'asservissements visuels. Il s'agit donc d'asservir le véhicule par rapport à une référence construite à partir d'observations de la scène, fournies par une caméra. Comme capteur, une caméra unique permet de percevoir l'environnement. Dans ce contexte, la commande référencée vision revêt une importance capitale puisque c'est l'un des maillons de base de la chaîne de perception-décision-action.

Ce mémoire constitue donc une mise en œuvre de lois de commande référencée vision dans le cas spécifique de l'asservissement du robot mobile *Robucar* de la Division *Productique et Robotique* (DPR) du *Centre de Développement des Technologies Avancées* (CDTA)

L'approche adoptée consiste à synthétiser un asservissement visuel à partir de primitives de type droite qui permet la régulation d'une fonction d'erreur représentative des écarts entre le motif courant et le motif désiré dans l'image. Le but étant de doter le *Robucar* de la capacité de suivi de trajectoire assimilée à des segments de droite.

Organisation du mémoire

Le premier chapitre est une introduction générale à la robotique mobile autonome. Il présente les principales notions fondamentales associées à ce domaine. Les différentes caractéristiques de notre plateforme expérimentale *Robucar* seront exposées.

Le second chapitre propose un état de l'art sur l'asservissement visuel en robotique mobile. En effet ce domaine a connu un développement considérable et plusieurs techniques en sont découlées. Dans un premier temps, des généralités sur la vision en robotique mobile seront présentées. Les différentes techniques d'asservissement visuel seront détaillées. L'accent sera toutefois mis sur l'asservissement visuel 2D. La philosophie globale de cette technique appliquée aux robots consiste à synthétiser les lois de commande dans l'espace capteur. On se basera notamment les travaux réalisés dans ce domaine par François Chaumette.

Le chapitre 3 sera consacré à la synthèse de commandes référencées visions. Pour ce faire, une première approche consistera à exploiter directement les informations visuelles pour la commande de l'angle de braquage du robot. Ainsi, des essais sur le robot nous permettrons de cibler les différents problèmes à traiter. Dans un second temps, une étude détaillée de notre système, qui passera par la modélisation de ce dernier ainsi que de son interaction avec son environnement sera nécessaire pour aborder la seconde approche, celle-ci est basée sur le formalisme de fonction de tâche qui a été détaillée au chapitre 2. Une autre solution adoptée est la synthèse d'un régulateur robuste basé sur le principe de la logique floue.

Au travers de plusieurs exemples de simulation, l'efficacité des approches adoptées dans le chapitre précédent sera validée dans le chapitre 4 de ce mémoire. Ainsi, ce dernier englobera les résultats obtenus en simulation, pour ensuite présenter les résultats expérimentaux qui concernent l'implémentation temps réel sur le *Robucar*.

Enfin, une conclusion et des perspectives sur les travaux présentés clôturent ce présent mémoire.

Chapitre 1

Généralités sur la robotique mobile et présentation du robot « *Robucar* »

1.1 Introduction

De manière générale, on désigne par robots mobiles l'ensemble des robots à base mobile. Ils peuvent donc se déplacer contrairement aux robots manipulateurs possédant une base fixe. L'usage veut néanmoins que l'on désigne le plus souvent par ce terme les robots mobiles à roues. Ce sont en effet les plus utilisés vu leurs simplicité de réalisation comparé aux autres robots mobiles. Ces derniers sont le plus souvent désignés par leur type de locomotion, qu'ils soient marcheurs, sous-marins ou aériens.

Le premier chapitre de ce mémoire propose des notions générales sur la robotique mobile, notamment la notion d'*autonomie*. Ainsi, il présente les aspects matériels et fonctionnels d'un robot mobile autonome. Ce dernier est caractérisé par sa capacité à prendre ses propres décisions et réagir aux différentes situations auxquelles il sera confronté.

Un des points clés en robotique mobile est la perception de l'environnement. Dans cette optique, nous allons étudier les différents outils permettant aux robots de percevoir son environnement et de s'y repérer. En effet, ces outils sont regroupés en deux catégories selon qu'ils fournissent des informations sur l'état même du robot, ou bien sur son environnement.

Suivant son degré d'autonomie, un robot mobile est amené à naviguer dans des environnements plus au moins connus et structurés. Les différentes étapes de la navigation d'un robot mobile autonome seront donc abordées, ainsi que quelques techniques de planification locales et globales les plus utilisées et les différentes architectures de contrôle permettant de les réaliser.

Par la suite, nous allons nous intéresser, après introduction des notions de non-holonomie et de roulement sans glissement, à la modélisation des robots mobiles. Ceci afin d'aboutir au modèle cinématique régissant notre robot *Robucar* qui est de type voiture. Pour terminer, les différentes caractéristiques de ce derniers seront détaillées.

1.2 Les robots mobiles autonomes

Les Robots Mobiles Autonomes (RMA) sont nés avec la robotique dite de troisième génération et sont donc des machines capables avant tout de réaliser le lien entre perception et action. Par définition le comportement d'un RMA n'est pas explicitement programmé à l'avance : il doit pouvoir se reconfigurer au gré des tâches à effectuer et s'adapter en temps réel à toute modification imprévue du monde qui l'entoure. Ce qui paraît caractériser le plus un RMA c'est son pouvoir d'interraction avec l'environnement très complexe dans lequel il évolue, ainsi que son haut degré d'autonomie opérationnelle et décisionnelle [Pichon 91].

Un RMA est donc une machine physique qui agit sur son environnement pour atteindre un objectif. Mais cette définition est incomplète car il existe des machines automatiques antérieures à l'existence des robots qui entrent dans cette définition, par exemple les machines outils. Pour une définition complète, il faut introduire la notion d'***autonomie***.

On dit qu'un robot est *autonome* s'il vérifie ces deux conditions [Coiffet 86] :

- *Versatilité* : Le robot devrait être capable d'effectuer des tâches diverses de plusieurs manières.

- *Auto-adaptativité* : Le robot devrait être capable d'accomplir correctement sa tâche, sans intervention humaine, même s'il rencontre de nouvelles situations inattendues.

1.2.1 Aspects matériels d'un robot mobile autonome

D'un point de vue technologique, la configuration de base d'un robot mobile autonome comprend trois types d'organes [Large 03] :

- Les organes de perception ;
- Les unités de traitement ;
- Les organes de locomotion ;

a - Les organes de perception

Ils comprennent les capteurs proprioceptifs et les capteurs extéroceptifs. Certains capteurs modernes sont capables de prétraiter leurs données pour fournir des informations de plus haut niveau (Segmentation d'une image vidéo, suivi automatique de cible, filtrage du bruit).

b - Les unités de traitement

Elles regroupent tout ce que l'électronique moderne compte de calculateurs, ordinateurs, et autres systèmes capables de manipuler et stocker des données numériques.

c - Les organes de locomotion

Aussi appelés effecteurs, ils ont pour but de permettre au robot d'évoluer dans un monde prévu à l'origine pour l'homme. Les plus courants sont les systèmes à roue. Le type de locomotion définit deux types de contraintes :

- **Les contraintes cinématiques**, qui portent sur la géométrie des déplacements possibles du robot (ex. Bornes sur la courbure de la trajectoire).
- **Les contraintes dynamiques**, liées aux effets du mouvement (accélérations bornées, vitesses bornées, présence de forces d'inertie ou de friction).

Selon sa cinématique, un robot sera dit :

- **Holonome**, s'il peut se déplacer instantanément dans toutes les directions.
- **Non-holonome**, si ses seuls déplacements autorisés sont des courbes dont la courbure est bornée (tel qu'une voiture).

1.2.2 Aspects fonctionnels d'un robot mobile autonome

Les traitements de base réalisés par un RMA, c'est-à-dire par ses unités de traitements, suivent un découpage fonctionnel similaire à celui des équipements. Ainsi nous trouvons [Large 03] :

- La perception
- Le raisonnement
- L'action

a - La perception

Regroupe l'ensemble des traitements visant à minimiser le volume des informations reçues par le robot afin de n'en conserver que les plus pertinentes pour sa mission. Il s'agit essentiellement de filtrage, de mise en forme et de fusion des données issues des capteurs et de la connaissance a priori du robot.

b - Le raisonnement

Il constitue l'intelligence du robot, à l'instar de l'homme le raisonnement du robot lui permet de décider d'une action appropriée à une situation donnée, compte-tenu d'une mission à réaliser. Plusieurs tâches élémentaires s'exécutent en parallèle pour synthétiser un comportement. La façon dont elles interagissent est définie par l'architecture décisionnelle du robot. Celle-ci comprend notamment un contrôleur d'exécution, dont le but est de lancer l'exécution d'un groupe de tâches donné à intervalles de temps réguliers. Chaque cycle du contrôleur correspond à une décision du robot, et génère un ordre (consigne) envoyé à la partie action. Nous parlons de ***traitement itératif.***

c - L'action

Cette fonction est également appelée « contrôle » du robot. L'action essentielle du RMA est de se déplacer. Par conséquent, la consigne est un déplacement désiré et le contrôle regroupe l'ensemble des opérations permettant au robot d'effectuer ce déplacement. Il se fait en deux étapes :

- **La génération de la commande**, ensemble de paramètres de contrôle des effecteurs, supposée impliquer le déplacement escompté.
- **L'application de cette commande**, c'est-à-dire sa conversion en signaux directement utilisables par les effecteurs, on parle alors de contrôle bas niveau.

1.3 La perception en robotique mobile

La notion de perception en robotique mobile est relative à la capacité du système à recueillir, traiter et mettre en forme des informations utiles au robot pour agir et réagir dans le monde qui l'entoure. Alors que pour des tâches de manipulation on peut considérer que l'environnement du robot est relativement structuré, ce n'est plus le cas lorsqu'il s'agit de naviguer de manière autonome dans des lieux très partiellement connus. Aussi, pour extraire les informations utiles à l'accomplissement de sa tâche, il est nécessaire que le robot dispose de nombreux capteurs, leurs choix dépend bien évidemment de l'application envisagée [Bayle 09].

En robotique mobile, on classe traditionnellement les capteurs en deux catégories selon qu'ils mesurent l'état du robot lui-même ou l'état de son environnement. Dans le premier cas, à l'image de la perception chez les êtres vivants, on parle de proprioception et donc de capteurs proprioceptifs.

On trouve par exemple dans cette catégorie les capteurs de position ou de vitesse des roues et les capteurs de charge de la batterie. Les capteurs renseignant sur l'état de l'environnement, donc de ce qui est extérieur au robot lui-même, sont eux appelés capteurs extéroceptifs. Il s'agit

de capteurs donnant la distance du robot à l'environnement, la température, signalant la mise en contact du robot avec l'environnement, etc [Bayle 11].

1.3.1 Capteurs proprioceptifs

Ils fournissent des informations propres au comportement interne du robot, c'est à-dire sur son état à un instant donné. L'intégration de leurs mesures permet d'estimer la situation courante du robot relativement à sa situation initiale. Ces capteurs donnent des résultats qui se dégradent avec le temps, il faut donc leur adjoindre un système permettant de recaler périodiquement la situation absolue du robot. Il existe une panoplie de ce type de capteurs, parmi eux :

- **Odomètres :** L'odométrie permet d'estimer le déplacement à partir de la mesure de rotation des roues. La mesure de rotation est en général effectuée par un codeur optique disposé sur l'axe de la roue, ou sur le système de transmission (par exemple sur la sortie de la boite de vitesse) [Filliat 11]. L'information de déplacement nécessitera la connaissance du diamètre des roues, de l'entraxe des roues et de la structure mécanique et cinématique du véhicule. Ce type de capteur est fortement utilisé en robotique mobile puisqu'il présente l'avantage d'être simple à mettre en œuvre et surtout d'être peu coûteux [Aknin 02].

- **Accéléromètres :** L'accéléromètre est un capteur qui mesure l'accélération linéaire en un point donné. En pratique, le principe de ce capteur est de mesurer l'effort massique non gravitationnel qu'on doit appliquer à une masse M pour la maintenir en place dans un boîtier lorsqu'une accélération est appliquée à ce dernier. Le calcul du déplacement élémentaire du robot est obtenu par double intégration de ces informations. Cette double intégration conduit généralement à des accumulations d'erreurs.

- **Gyroscopes :** Il permet de mesurer une variation angulaire. Il est intéressant en robotique mobile parce qu'il peut compenser les défauts des odomètres. Une erreur d'orientation odométrique peut entraîner une erreur en position cumulative qui peut être compensée par l'utilisation conjointe de gyroscope. Les gyroscopes très précis sont trop onéreux pour être utilisés en robotique mobile. Cependant, les gyroscopes à fibre optique, connus pour leur grande précision, ont vu leur prix chuter et sont donc devenus une solution attractive pour la navigation en robotique mobile [Parrain 00].

- **GPS :** Le GPS (*Global Positioning System*), initialement développé pour les applications militaires américaines, est actuellement à la disposition du grand public. On peut cependant considérer que son utilisation dans ce cadre n'est pas garantie.
  Le GPS fonctionne avec un ensemble de satellites, qui effectuent des émissions synchronisées dans le temps. Par recoupement des instants d'arrivée des signaux et de la position des satellites émetteurs, les récepteurs peuvent calculer leur position. Le principe de calcul de la position est basé sur une triangulation, à l'aide de quatre signaux reçus simultanément (le

quatrième signal assure la robustesse de la mesure).
En termes de précision, la localisation ainsi obtenue est entachée d'une erreur de l'ordre de la quinzaine de mètres, ce qui n'est bien évidemment pas suffisant pour permettre à un robot de naviguer de manière robuste. Ainsi, on a systématiquement recours à une méthode différentielle pour obtenir des résultats plus satisfaisants. La localisation se fait à l'aide de deux récepteurs, dont l'un est statique et positionné avec précision dans l'environnement. On peut alors obtenir une précision de l'ordre du centimètre.
Pour capter les signaux émis par des satellites, ce type de dispositif est donc plutôt destiné au positionnement extérieur. Par ailleurs sa faible précision et son prix élevé le réservent plutôt à des systèmes multi robots, associant par exemple des robots mobiles à des engins volants et sous-marins, comme cela est à l'étude dans les applications militaires les plus récentes. Enfin, ce système de mesure, autorisant des rafraîchissements à des fréquences de l'ordre de seulement 5 Hz, n'est pas utilisable pour une commande en temps-réel. Il est donc principalement utile pour des recalages ponctuels [Bayle 11].

1.3.2 Capteurs extéroceptifs

Ces capteurs informent le robot sur son environnement. Ils peuvent être utilisés tout au long du parcours soit pour mesurer en permanence la situation absolue du robot mobile, soit pour recaler périodiquement la navigation à l'estime. Ils peuvent intervenir également pour assurer la sécurité du robot (perception de l'environnement proche, contrôle de l'attitude de la plate-forme) et pour construire en ligne un modèle de l'environnement exploré.

- **Capteurs infrarouges :** Les capteurs infrarouges sont constitués d'un ensemble *émetteur/ récepteur* fonctionnant avec des radiations non visibles, dont la longueur d'onde est juste inférieure à celle du rouge visible. La mesure des radiations infrarouges étant limitée et, en tout état de cause, la qualité très dégradée au-delà d'un mètre, ces dispositifs ne servent que rarement de télémètres. On les rencontrera le plus souvent comme détecteurs de proximité, ou dans un mode encore plus dégradé de présence. Il faut noter que ce type de détection est sensible aux conditions extérieures, notamment à la lumière ambiante, à la sécularité des surfaces sur lesquelles se réfléchissent les infrarouges, à la température et même à la pression ambiante. Ces capteurs ne sont pas complètement directionnels et leur caractéristique (à l'image des capteurs ultrasons présentés par la suite) présente une zone de détection conique à l'origine d'incertitudes. Enfin, l'alternance de phases d'émission et de réception impose une distance de détection minimale.

- **Télémètres à ultrason :** Les télémètres à ultrason (Fig : 1.1) sont historiquement les premiers à avoir été utilisés. Ils utilisent la mesure du temps de retour d'une onde sonore réfléchie par les obstacles pour estimer la distance. Ces télémètres sont très simples et peu chers, et sont donc très répandus, mais possèdent de nombreux inconvénients [Filliat 11].
En premier lieu, deux télémètres voisins ne peuvent être utilisés simultanément, car il est

impossible de savoir par lequel des deux télémètres une onde réfléchie a été émise (phénomène de *"crosstalk"*). Un robot possédant plusieurs télémètres doit donc les activer l'un après l'autre, ce qui entraine un taux de rafraîchissement global des mesures relativement faible.

FIGURE 1.1 – Télémètre ultrasonore MSU08

- **Télémètres laser :** Les télémètres les plus utilisés à l'heure actuelle pour des applications de cartographie et de localisation sont les télémètres laser à balayage (Fig : 1.2). Ils utilisent un faisceau laser mis en rotation afin de balayer un plan, en général horizontal, et qui permet de mesurer la distance des objets qui coupent ce plan. Cette mesure peut être réalisée selon différentes techniques (mesure du temps de retour, interférométrie,...).
  Les télémètres les plus courants ont une bonne résolution angulaire car ils permettent d'obtenir une mesure de distance tous les demi degrés, sur une zone de 180 ou 360 degrés selon les modèles. La mesure est de plus relativement précise (avec un bruit de l'ordre de quelques centimètres) à une distance relativement grande (plusieurs dizaines de mètres). La fréquence d'acquisition est en général de l'ordre de la dizaine de Hertz, voire proche de la centaine pour certains modèles [Filliat 11].
  Ces télémètres sont très utilisés en environnement intérieur car ils fournissent des données abondantes et précises sur la position des objets caractéristiques de l'environnement tels que les murs. Ils possèdent toutefois un certain nombre d'inconvénients. En premier lieu, leur zone de perception est restreinte à un plan et ne permet donc pas de détecter les obstacles situés hors de ce plan (un petit objet posé au sol par exemple). Ils ne peuvent pas non plus détecter les objets ne réfléchissant pas correctement la lumière du laser (en premier lieu les vitres, mais aussi certains objets très réfléchissants, tels que les objets chromés). Pour limiter ces inconvénients, il est possible de les utiliser en conjonction avec des capteurs à ultrason qui ont un cône de détection plus large et qui peuvent détecter les vitres.

FIGURE 1.2 – La famille des télémètres lasers Sick 04

- **Les capteurs vision :** L'utilisation d'une caméra pour percevoir l'environnement est une méthode attractive car elle semble proche des méthodes utilisées par les êtres humains. Le traitement des données volumineuses et complexes fournies par ces capteurs reste cependant difficile à l'heure actuelle, même si cela reste une voie de recherche très explorée. L'inconvénient majeur est qu'elle peut être un capteur trop riche pour être utilisée dans des applications en temps réel [Jeanpierre 04]. Une caméra peut être utilisée de différentes manières pour la navigation d'un robot mobile. Elle peut être utilisée pour détecter des amers visuels (des points particuliers qui servent de repère, tels que des portes ou des affiches) à partir desquels il sera possible de calculer la position du robot. Si ces amers sont simplement ponctuels, ou de petite taille, il sera en général simplement possible d'estimer leur direction. Dans le cas où les amers sont des objets connus en 2 ou 3 dimensions, il sera en général possible d'estimer complètement la position du robot par rapport à la leur. Elle peut également être utilisée pour détecter des "guides" de navigation pour le robot, tels que des routes ou des couloirs.

FIGURE 1.3 – Caméras CCD (Charge Coupled Device)

1.3.3 Autres capteurs

Il existe d'autres capteurs qui ne sont pas classés dans les catégories cités précédemment, ils visent entre autre à garantir la sécurité du robot, parmis ces capteurs on peut citer [Filliat 11] :

- **Les capteurs tactiles :** Les robots peuvent être équipés de capteurs tactiles, qui sont le plus souvent utilisés pour des arrêts d'urgence lorsqu'il rencontre un obstacle qui n'avait pas été détecté par le reste du système de perception. Ces capteurs peuvent être de simples contacteurs répartis sur le pourtour du robot.
  Il ne détectent alors le contact qu'au dernier moment. Il est également possible d'utiliser des petites tiges arquées autour du robot pour servir d'intermédiaire à ces contacteurs, ce qui

permet une détection un peu plus précoce et donne ainsi plus de marge pour arrêter le robot.

- **Les balises :** Dans certaines applications, il est également possible d'utiliser des balises dont on connaît la position, et qui pourront être facilement détectées par le robot, afin de faciliter sa localisation. Des techniques très diverses peuvent être utilisées pour ces balises. On peut par exemple utiliser un signal radio, émis de manière omnidirectionnel par la balise. Le robot sera alors équipé d'une antenne directionnelle qui lui permettra de détecter la direction des différentes balises, afin de déduire sa position par triangulation. On peut également utiliser des codes couleurs ou des codes-barres qui pourront être détectés par une caméra.

1.4 La navigation en robotique mobile

La navigation autonome des robots, c'est-à-dire la capacité à évoluer sans aide dans leur environnement de travail, est une des fonctions essentielles pour un robot mobile autonome.La complexité de la méthode de navigation mise en œuvre sur un robot mobile dépend donc de l'environnement dans lequel il doit évoluer (milieu intérieur ou environnement naturel , sol plan ou irrégulier,...). Elle dépend également de la connaissance de cet environnement qui peut être figé ou évolutif et du mode de définition de la trajectoire (apprentissage préalable, planification en ligne). Les performances du système de navigation sont étroitement liées à la précision, à la fiabilité et au temps de réponse des capteurs et des méthodes mises en œuvre pour localiser le véhicule.

La navigation est définie comme le procédé permettant de répondre aux trois questions suivantes :

- « où suis-je ? »
- « où sont les autres lieux par rapport à moi ? »
- « comment puis-je atteindre ces autres lieux depuis l'endroit où je me trouve ? ».

La robotique mobile autonome vise plus spécifiquement à concevoir des systèmes capables de se déplacer de façon autonome. Les applications directes se situent notamment dans les domaines de l'automobile, de l'exploration planétaire ou de la robotique de service par exemple. De nombreuses applications restent à découvrir, qui ne découlent pas directement des avancées de la robotique mais qui utilisent ses méthodes et ses développements [Lefebvre 06].

1.4.1 Les étapes de navigation

Il existe trois étapes de navigation :

a - Perception et modélisation

La perception permet de détecter l'environnement proche et éloigné du robot. Elle est nécessaire pour sa sécurité, pour la modélisation de l'environnement et pour sa localisation dans celui-ci [Pruski 96]. L'étape de modélisation a pour but de maintenir un modèle d'environnement dans lequel le robot se déplace, pour ce faire il doit, soit construire son propre modèle du monde, soit

l'acquérir. En plus, il existe plusieurs possibilités pour les représentations utilisées dans ce modèle du monde, parmi lesquelles on cite [Dufourd 05] :

- Une représentation purement géométrique, (par exemple un ensemble de segments laser).
- Une représentation topologique qui décrit les connexions entre couloirs et pièces.

b - Décision et planification

Les informations en provenance des différents capteurs doivent être interprétées comme autant d'éléments utiles à la prise de décision sur l'action à faire, le but étant de délivrer les ordres corrects aux actionneurs.

Dans un robot mobile, il est nécessaire de donner des priorités en fonction des informations reçues. Par exemple, si un capteur de contact informe d'un choc sur l'avant, cette information est prioritaire sur un déplacement du robot vers l'avant et doit entraîner un arrêt ou un déplacement dans une autre direction. On comprend ainsi toute la difficulté de la décision, car c'est elle qui donnera vie à notre robot. C'est lors de cette phase de la conception d'un robot qu'il est nécessaire de lui donner une forme d'intelligence en lui laissant le choix sur l'action à entreprendre [Duval 03].

c - Exécution

Le robot dispose d'actionneurs « moteurs » permettant son déplacement. Le rôle de ces actionneurs est d'exécuter les commandes correspondant aux décisions [Pruski 96].

1.4.2 Les stratégies de navigation

Les stratégies de navigation permettant à un robot mobile de se déplacer pour rejoindre un but sont extrêmement diverses, de même que les classifications qui peuvent en être faites. Nous présentons ici une classification qui a été établie en prenant en compte à la fois les stratégies des robots et des animaux. Elle présente l'avantage de distinguer les stratégies sans modèles internes et les stratégies avec modèle interne . Cette classification comporte cinq catégories, de la plus simple à la plus complexe [Filliat 11] :

a - Approche d'un objet :

L'approche d'un object constitue une tache de positionnement, celle-ci représente une capacité basique d'un robot lui permettant de se diriger vers un objet visible depuis sa position courante. Cette stratégie utilise des actions réflexes, dans lesquelles chaque perception est directement associée à une action. Notons que cette stratégie est fonctionnelle uniquement dans la zone de l'environnement pour laquelle le but est visible, elle est donc dite locale.

b - Guidage :

Un point de l'espace est caractérisé par la configuration spatiale de l'ensemble d'objets remarquables, ou amers, qui l'entourent ou qui en sont voisins. Cette stratégie de navigation consiste alors à se diriger dans la direction qui permet de reproduire cette configuration, et ne vise donc pas à atteindre un objet matériel directement visible. Cette capacité semble être utilisée par certains insectes, comme les abeilles, et a été utilisée sur divers robots. Elle est basée sur des actions réflexes et réalise une navigation locale qui requiert que les amers caractérisant le but soient visibles.

c - Action associée à un lieu :

Contrairement aux approches précédentes, celle-ci permet au robot d'atteindre un but depuis des positions où ce dernier (ou les amers le caractérisants) ne sont pas visibles préalablement, elle constitue donc la première approche permettant de réaliser une navigation globale. Toute fois une étape de représentation interne de l'environnement qui consiste à définir des lieux comme des zones de l'espace dans lesquelles les perceptions restent similaires est nécessaire, ainsi qu'une association d'une action à chacun de ces lieux (Fig : 1.4). L'enchaînement des ces actions définit une route qui permet de rejoindre le but. Malgré le degrés d'autonomie plus important qu'ils offrent aux robots, ces modèles sont limités à un but fixé. Changer de but entraînera donc l'apprentissage d'une nouvelle route, indépendante des routes permettant de rejoindre les autres buts.

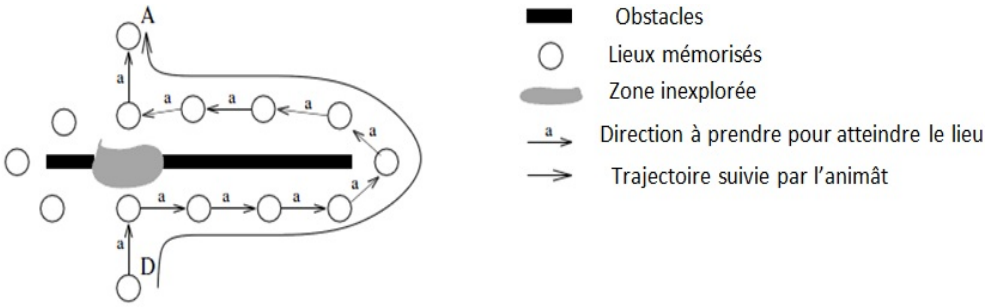

FIGURE 1.4 – Action associée à un lieu

d - Navigation topologique :

Pour faire face aux lacunes de l'approche précédente, la navigation topologique permet de mémoriser dans le modèle interne les relations spatiales entre les différents lieux. Ceci permet d'offrir au robot la possibilité de déplacement d'un lieu à un autre sans que ces derniers soient associées à un but particulier, ainsi différents chemins entre deux lieux arbitraires peuvent être envisagés.(Fig : 1.5).

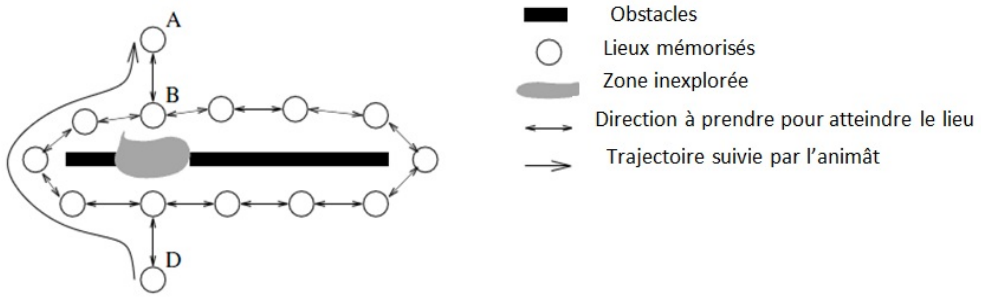

FIGURE 1.5 – Navigation topologique

e - Navigation métrique :

Pour permettre au robot de planifier des chemins au sein de zones inexplorées, la navigation métrique mémorise les positions métriques relatives des différents lieux, en plus de la possibilité de passer de l'un à l'autre. En utilisant une simple composition de vecteurs, ces positions relatives permettent de calculer une trajectoire allant d'un lieu à un autre, même si la possibilité de ce déplacement n'a pas été mémorisée sous forme d'un lien (Fig : 1.6).

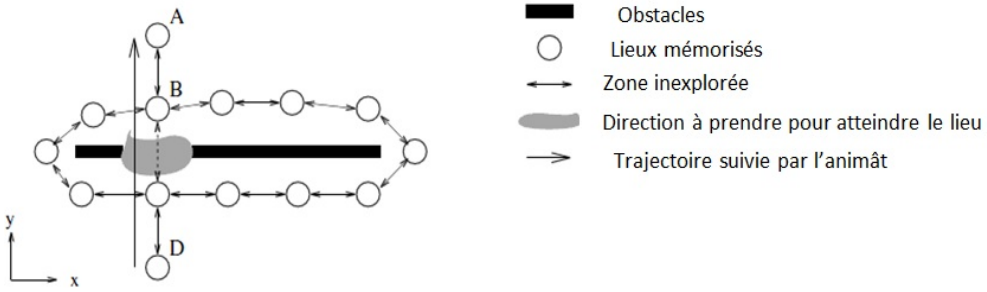

FIGURE 1.6 – Navigation métrique

Remarque : Les modèles des trois premières catégories utilisent des actions réflexes et se regroupent sous le terme générique de *navigation réactive*. Ils se caractérisent par leur simplicité, et de ce fait leur rapidité d'exécution, de plus ils ne nécessitent pas de modèle global de l'environnement mais ont un domaine d'application souvent restreint permettant de réaliser des taches de bas-niveau comme l'évitement d'obstacle imprévu, essentiel à la sécurité du robot.

Les modèles des deux dernières catégories autorisent pour leur part une navigation globale pour atteindre un but arbitraire au sein de l'environnement. S'appuyant pour cela sur un modèle interne du monde, une carte, qui supporte une planification. Ce sont ces deux stratégies qui sont regroupées sous le terme de *navigation par carte*.

1.5 Les architectures de contrôle

Un robot mobile autonome a besoin de capacités de raisonnement et de décision pour pouvoir évoluer dans son environnement et atteindre son but. Il doit aussi satisfaire à des exigences variées et parfois contradictoires. Un exemple typique pour un robot mobile est l'arbitrage qui doit être fait entre l'exécution la plus précise possible d'un plan préétabli pour atteindre un but et la prise

en compte d'éléments imprévus, tels que les obstacles mobiles. Ces arbitrages, que ce soit au niveau du choix de stratégie, ou au niveau de l'utilisation des capteurs, des effecteurs ou des ressources de calcul, sont réglés par un ensemble logiciel appelé architecture de contrôle du robot. Cette architecture sera considérée comme le cerveau du robot, qui permettra à ce dernier, à l'aide de l'architecture matérielle, d'agir d'une manière autonome pour atteindre ses objectifs et évoluer dans son environnement en assurant sa sécurité.

Ainsi, les architecture de contôles sont des systèmes sophistiqués permettant aux robot de faire un travail physique utile dans un environnement réel [Abdou 97]. Elle peuvent être classées en trois grandes catégories : les contrôleurs hiérarchiques, les contrôleurs réactifs et les contrôleurs hybrides (Fig : 1.7). Que nous détaillerons ci-aprés [Filliat 11] :

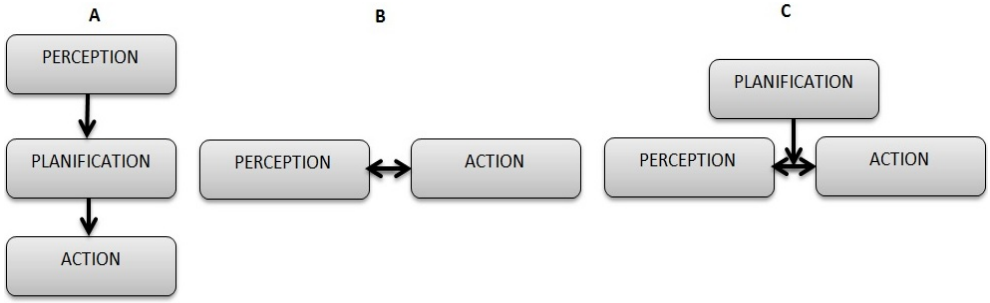

FIGURE 1.7 – Les architectures classiques des contrôleurs pour les robots mobiles : (A) Hiérarchique, (B) Réactive, (C) Hybride.

1.5.1 Contrôleurs hiérarchiques

Les premiers robots mobiles dérivés des recherches en intelligence artificielle utilisaient des contrôleurs hiérarchiques (Fig : 1.7 A). Ils consistent à décomposer les tâches en sous-tâches d'un niveau de l'hirarchie. Ainsi ils fonctionnent selon un cycle rigide de modélisation de l'environnement, planification des actions au sein de cette représentation, puis exécution du plan. Un robot doit donc commencer par percevoir l'environnement en recueillant les données par des capteurs. Il lui faudra ensuite analyser les données et modéliser son environnement pour pouvoir planifier les tâches. Ces architectures ont rapidement montré leurs limites et leur incapacité à fonctionner dans un environnement qui ne soit pas statique et simplifié à l'extrême. L'essentiel des problèmes de ces architectures provient de l'utilisation d'un modèle interne central qui est le seul pris en compte pour guider le robot.

En effet, ces architectures permettent peu de contrôle sur l'exécution des actions. En effet, une fois l'action choisie, elle est exécutée en supposant le modèle du monde correct et il n'y a pas de retour direct de la perception sur l'exécution de l'action. Les écarts *modèles/environnement* ne peuvent être pris en compte que via un nouveau cycle *perception/modélisation/planification* , ce qui, par définition, est très peu réactif et conduit rapidement à de graves problèmes.

De plus, leur principal inconvénient provient de leur faible vitesse de réaction. Elles sont en effet totalement incapables de prendre en compte des obstacles dynamiques ou non détectés lors de la modélisation. Ceci est dû à la lenteur des phases de modélisation et de planification qui ne

peuvent généralement pas être exécutées en temps réel. La construction du modèle sur lequel se base la planification n'est pas une tâche aisée puisqu'il est pratiquement souvent difficile de relier les données sensorielles aux objets du monde réel, et ceci est dû en grande partie à l'imprécision de l'information délivrée par les capteurs utilisés en robotique. Le chercheur R. Brooks résume ce problème en affirmant qu'il existe une grande différence entre capter et percevoir.

1.5.2 Contrôleurs réactifs

Pour remédier aux inconvénients des contrôleurs hiérarchiques, une nouvelle forme d'architecture a été mise au point. il s'agit de l'architecture réactive (Fig : 1.7 B). Dans cette architecture, un ensemble de comportements réactifs, fonctionnant en parallèle, contrôle le robot sans utiliser de modèle du monde.

Cette architecture supprime évidemment les problèmes dûs aux différences entre la réalité, d'une part, et le modèle de l'environnement du robot, d'autre part, mais limite clairement les tâches que peut effectuer le robot En effet, sans représentation interne de l'état de l'environnement, il est très difficile de planifier une suite d'actions en fonction d'un but à atteindre. Les robots utilisant cette architecture sont donc en général efficaces pour la tâche précise pour laquelle ils ont été conçus, dans l'environnement pour lequel ils ont été prévus, mais sont souvent difficiles à adapter à une tâche différente.

Les réussites de ces architectures sont liées au couplage direct entre la perception et l'action qui permet une prise en compte très rapide des phénomènes dynamiques de l'environnement. En donc une bonne robustesse dans des environnements complexes. Comme nous l'avons mentionné, ces architectures sont en général basées sur plusieurs comportements : évitement d'obstacles, déplacement aléatoire, déplacement vers un but, fuite d'un point...etc.

Les systèmes de contrôle réactifs réagissent donc directement aux stimuli provenant du monde extérieur en évitant la construction d'un modèle forcement entaché d'erreurs en environnement réel, ce qui autorise des temps d'exécution relativement courts

1.5.3 Contrôleurs hybrides

Des chercheurs ont essayé de combiner les caractéristiques des deux types de contrôle précédemment cités. Ceci en mettant au point des architectures hybrides (Fig : 1.7 C) permettant notamment d'allier des capacités de raisonnement et de décision de haut niveau, s'appuyant sur des représentations abstraites des connaissances, avec des comportements réactifs garantissant robustesse et flexibilité par rapport à l'environnement.

La plupart des contrôleurs actuellement utilisés choisissent cette solution intermédiaire entre ces deux approches. Cette architecture se compose de deux niveaux. Le premier est chargé des tâches de navigation de haut niveau, telles que la localisation, la cartographie et la planification. Pour cela, il s'appuie sur un second niveau réactif. Celui-ci peut répondre rapidement par réflexe aux événements externes (comme par exemple à l'approche des obstacles) [Abdou 97]. L'action conjointe de ces deux niveaux permet de réagir rapidement face aux variations imprévues de l'environnement, tout en permettant la réalisation d'actions planifiées à plus long terme.

Le bas niveau de ces architectures peut être réalisé sous forme de comportements, tels que ceux utilisés dans les architectures réactives. Ces comportements sont des boucles sensorimotrices qui relient les action aux perceptions avec un phase de décision très courte, qui assure la réactivité. Dans le même temps, les informations sensorielles sont utilisées par le haut niveau dans une boucle sensorimotrice à une échelle de temps beaucoup plus longue. C'est la mise en parallèles de ces deux échelles de temps qui fait la force de ces architectures.

1.6 Modélisation des robots mobiles non holonomes

1.6.1 Rappels sur la notion de non holonomie

Définition 1 : *Etant donné un système mécanique dont l'espace de configuration est une variété différentielle Q de dimension n, on appelle* ***contrainte cinématique*** *une contrainte sur les vitesses du type :*

$$< a(q), \dot{q} >= 0 \ \forall q \in \cup(q_0) \tag{1.1}$$

Avec $\cup(q_0)$un voisinage du point $q_0 \in Q$, et $a(.)$ une forme différentielle (ou champ de covecteurs) de Q dans R^n .

Parmi les contraintes cinématiques du système, il faut distinguer celles qui sont en fait intégrables et qui peuvent, de ce fait, se ramener à des contraintes sur l'état seulement.

Définition 2 : *Une contrainte cinématique $< a(q), \dot{q} >= 0$ est dite contrainte intégrable s'il existe une fonction régulière $h : Q \Rightarrow R$ telle que $dh = a$. Dans ce cas la contrainte est équivalente à la relation statique $h(q) = Cste$.*

Remarquons que la propriété d'intégrabilité est une propriété locale. Dans la pratique, un critère qui permet de repérer l'existence de contraintes intégrables est donné par le résultat suivant.

Théorème d'intégrabilité de Frobenius : *soient un système mécanique sur une variété Q de dimension n, soumis à $k \leq n$ contraintes cinématiques indépendantes.*

$$< a_i(q), \dot{q} >= 0, \forall q \in \cup(q_0), \forall i = 1, \ldots, k \tag{1.2}$$

Et $\{X_1, \ldots, X_{(n-k)}\}$ une famille de champs de vecteurs sur $\cup(q_0)$ orthogonaux aux a_i. Si $Lie(X_1)(q)$ est de dimension $n - p$ pour tout $q \in \cup(q_0)$ alors il existe p contraintes intégrables (au sens où l'on peut trouver des fonctions scalaires $h_1, \ldots, h_p$ indépendantes et des fonctions $\lambda_{1,1}, \ldots, \lambda_{(p,n)}$ telles que $dh_i(q) = \sum_{j=0}^{n} \lambda_{ij}(q) a_j(q)$

Nous sommes à présent prêts à définir un système non holonome.

Définition 3 : *un système non holonome est un système mécanique soumis à un ensemble de contraintes cinématiques :*

$$< a_i(q), \dot{q} >= 0, \forall q \in \cup(q_0), \forall i = 1, \ldots, k \tag{1.3}$$

Indépendantes et non intégrables (au sens où il n'existe pas de contrainte intégrable).
En particulier, on appelle modèle cinématique du système le modèle

$$\dot{q} = X(q)v := dh_i(q) = \sum_{i=1}^{n-k} X_i(q)v_i \tag{1.4}$$

Avec $X_1, \ldots, X_{(n-k)}$ *une famille de champs de vecteur telle que* $< a_i(q), X_j(q) >= 0, \forall i = 1, \ldots, k \forall j = 1, \ldots, n-k$ *La variable* $v = (v_1, \ldots, v_{(n-k)}) = (v_1, \ldots, v_m) \in R^m$ *est assimilable à une variable de commande en vitesse. On appelle m (la dimension de l'espace des vitesses instantanées possibles) le nombre de degrés de liberté du système.*

1.6.2 Roulement sans glissement

La locomotion à l'aide de roues exploite la friction au contact entre roue et sol. Pour cela, la nature du contact (régularité, matériaux en contact) a une forte influence sur les propriétés du mouvement relatif de la roue par rapport au sol. Dans de bonnes conditions, il y a roulement sans glissement (r.s.g.) de la roue sur le sol, c'est-à-dire que la vitesse relative de la roue par rapport au sol au point de contact est nulle.

Théoriquement, pour vérifier cette condition, il faut réunir les hypothèses suivantes [Bayle 11] :

- le contact entre la roue et le sol est ponctuel ;
- les roues sont indéformables, de rayon r .

En pratique le contact se fait sur une surface, ce qui engendre bien évidemment de légers glissements, notamment lorsque les roues ne sont pas réellement rigides (exemple de pneu)[Benseddik 11]. Malgré cela, on supposera toujours qu'il y a r.s.g et, par ailleurs, que le sol est parfaitement plan [Bayle 11].

Mathématiquement, on peut traduire la condition de r.s.g sur une roue :

Soit P le centre de la roue, Q le point de contact de la roue avec le sol, φ l'angle de rotation propre de la roue et θ l'angle entre le plan de la roue et le plan (o, x, y) comme indiqué sur la figure (Fig : 1.8) .

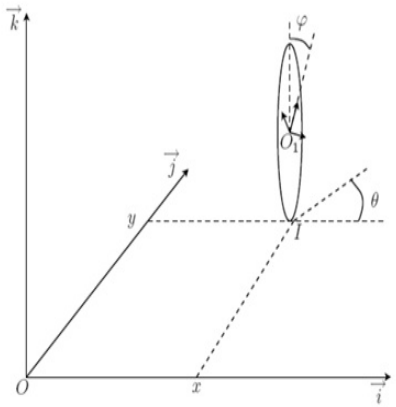

FIGURE 1.8 – Roulement sans glissement d'une roue verticale

L'hypothèse de roulement sans glissement se traduit par une vitesse nulle au point de contact I entre la roue et le sol.

Avec la notation de la figure (Fig : 1.8), on a [Benseddik 11] :

$$\begin{aligned}\overrightarrow{V_{I/R_0}} &= \dot{x}\,\overrightarrow{i} + \dot{y}\,\overrightarrow{j} + (\dot{\theta}\,\overrightarrow{k} + \dot{\varphi}(-sin\theta\,\overrightarrow{i} + cos\theta\,\overrightarrow{j}))\Lambda(-r\,\overrightarrow{k}) \\ &= (\dot{x} - r\dot{\varphi}cos\theta)\,\overrightarrow{i} + (\dot{y} - r\dot{\varphi}sin\theta)\,\overrightarrow{j} = 0\end{aligned} \tag{1.5}$$

On en déduit deux contraintes :

$$\begin{cases} \dot{x} - r\dot{\varphi}cos\theta = 0 \\ \dot{y} - r\dot{\varphi}sin\theta = 0 \end{cases} \tag{1.6}$$

Qu'on peut réécrire :

$$\begin{cases} \dot{x}cos\theta + \dot{y}sin\theta = r\dot{\varphi} \\ -\dot{x}sin\theta + \dot{y}cos\theta = 0 \end{cases} \tag{1.7}$$

Le modèle suivant est formé par l'introduction de $v = r\dot{\varphi}$ qui représente la vitesse de roulement de la roue et $\omega = \dot{\theta}$ sa vitesse de rotation autour de l'axe k :

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} v \\ \omega \end{pmatrix} \tag{1.8}$$

Pour faire apparaître la nature non holonme du roulment sans glissement de la roue sur le sol, dont la configuration est donnée par $q = (x, y, \theta, \varphi)^T \in N = SE(2) * S^1$,les équations (3.5) peuvent être réécrite comme suit :

$$\begin{cases} \dot{x}cos\theta + \dot{y}sin\theta - r\dot{\varphi} = 0 \\ -\dot{x}sin\theta + \dot{y}cos\theta = 0 \end{cases} \tag{1.9}$$

Ses dernières sont de la forme $< a_i(q), \dot{q} >= 0$ avec les covecteurs suivants :

$$\begin{cases} a_1(q) = (cos\theta \quad sin\theta \quad 0 \quad -r) \\ a_2(q) = (-sin\theta \quad cos\theta \quad 0 \quad 0) \end{cases} \tag{1.10}$$

Il apparaît clairement que ces deux covecteurs sont indépendants. Nous obtenons un modèle cinématique de type (S_0) avec $m = 2$

$$X_1(q) = X_1 = \begin{pmatrix} cos\theta \\ sin\theta \\ 0 \\ 1/r \end{pmatrix}, X_2(q) = X_2 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \tag{1.11}$$

Les covecteurs correspondent bien à des contraintes non holonomes.

1.6.3 Modélisation du *Robucar*

a - Hypothèses

Généralement pour la commande de robots mobiles, un modèle de commande en vitesse est utilisé plutôt qu'un modèle de commande en couple. Les principales raisons de ce choix sont les suivantes [Guechi 10] :

- Le calcul de la commande est plus simple pour un modèle cinématique que pour un modèle dynamique.
- Il n y a pas de paramètres géométriques ou inertiels compliqués à identifier pour un modèle cinématique.

Pour ces raisons, nous ne considérons dans la suite que le modèle cinématique en prenant en compte les hypothèses simplificatrices suivantes :

- Le robot mobile est considéré comme un véhicule rigide évoluant dans un plan horizontal ;
- Les roues conventionnelles sont supposées indéformable ;
- Le contact roue/sol est réduit à un point ;
- les roues roulent sans glisser sur le sol.

Nous restreignons notre étude au cas de notre robot mobile *Robucar*, ce dernier est de type voiture et possède plusieurs modèles cinématiques. Il est, par exemple, possible de contrôler indépendamment l'angle de braquage des roues arrières et celui des roues avants. Nous présentons ici le modèle cinématique dans lequel les roues arrières sont bloquées et seules les roues avants peuvent braquer.

b - Le modèle cinématique du *Robucar*

La configuration du *Robucar* peut être représentée par le quadruplet $q = [x, y, \theta, \varphi]$ où le point M de cordonnées (x, y) est le centre de l'essieu arrière du robot, θ est l'orientation du véhicule, φ l'orientation des roues avant et D la distance entre les essieux avant et arrière.

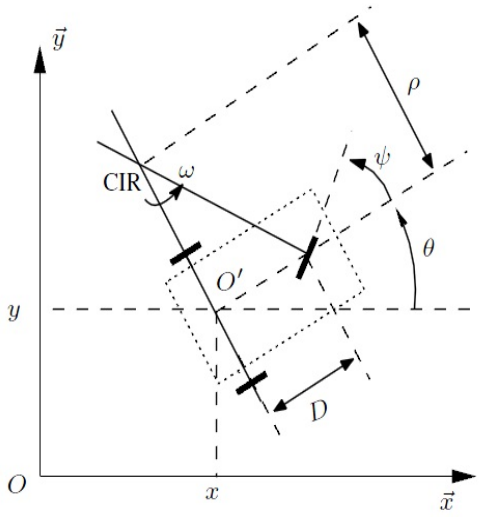

FIGURE 1.9 – Modèle tricycle du *Robucar*

On exploitera dans l'étude de la cinématique de notre robot la notion de *roue directrice centrale*, qui est une roue virtuelle correspondant à la roue directrice d'un tricycle équivalent. Cette astuce permet de simplifier les équations en faisant abstraction du mécanisme de couplage des roues directrices servant à respecter les contraintes de roulement sans glissement en ne considérant ainsi qu'un seul angle de direction.

d'aprés la figure (Fig : 1.9), on peut écrire alors :

$$\begin{cases} \rho = D/tan(\varphi) \\ \omega = Dtan(\varphi)/v \end{cases} \tag{1.12}$$

avec ρ la distance entre l'axe centrale du robot et le CIR (Centre Instantané de Rotation), ω la vitesse angulaire du robot et v la vitesse de translation du robot.

Théoriquement le robot peut se diriger en ligne droite pour $\varphi = 0$ et théoriquement tourner autour du point O (on pourrait dire sur place) pour $\varphi = \pi/2$ [Bayle 11]. Néanmoins, le rayon de braquage de la roue orientable étant généralement limité, ceci impose le plus souvent des valeurs telles $-\pi/2 < \varphi < \pi/2$ ($-20° < \varphi < 20°$ dans le cas du *Robucar*), interdisant cette rotation du robot sur lui-même.

Les conditions de roulement sans glissement s'obtiennent en écrivant que les vitesses latérales des roues avant et arrière sont nulles :

$$\begin{cases} \dot{x}sin\theta - \dot{y}cos\theta = 0 \\ \dot{x}sin(\theta + \varphi) + \dot{y}cos(\theta + \varphi) + D\dot{\theta}cos(\varphi) = 0 \end{cases} \tag{1.13}$$

Ces contraintes sont non intégrables, le *Robucar* est donc non holonome. Ainsi, on peut écrire son modèle cinématique comme suite

$$\begin{cases} \dot{x} = u_1 \cos\theta \\ \dot{y} = u_1 sin\theta \\ \dot{\theta} = u_1 tan(\varphi)/D \\ \dot{\varphi} = u_2 \end{cases} \tag{1.14}$$

Où u_1 correspond à la vitesse longitudinale du robot, alors que u_2 correspond à la vitesse angulaire des roues directrices par rapport au corps du robot [Guechi 10].

On peut écrire aussi le modèle discret correspondant comme suit :

$$\begin{cases} X(k+1) = X(k) + Te.u_1 cos(\theta(k)) \\ Y(k+1) = Y(k) + Te.u_1 sin(\theta(k)) \\ \theta(k+1) = \theta(k) + Te.u_1 tan(\varphi(k))/D \end{cases} \tag{1.15}$$

Avec Te la période d'échantillonnage.

1.7 Présentation du *Robucar*

Le *Robucar* est un prototype de petit véhicule électrique construit sur la base d'un châssis tubulaire des petites voitures utilisées dans les terrains de golf. Dans le cadre des projets de recherche de la division productique et robotique du CDTA, ce robot est utilisé comme plateforme expérimentale.

1.7.1 Caractéristiques techniques

Le robot mobile « *Robucar* » se déplace grâce à ses quatre roues motrices et directrices réparties en deux trains ; avant et arrière. Ces roues sont reliées à quatre moteurs qui assurent la traction du robot et deux vérins qui assurent la rotation des roues avant et arrière.

Les composants du robot mobile *Robucar* sont illustrés sur la figure(Fig : 1.10) et le tableau (Tab : 3.1) suivants :

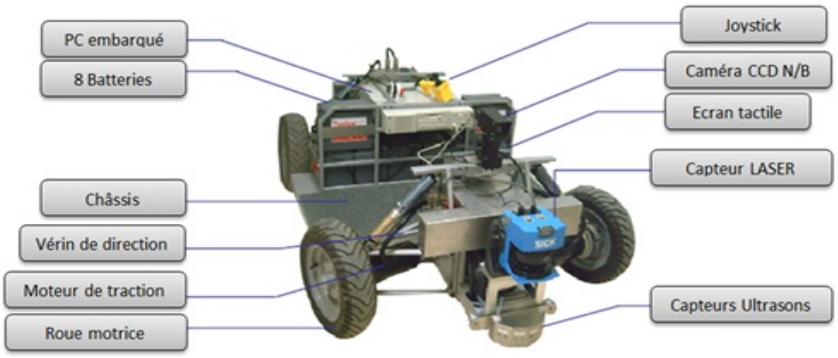

FIGURE 1.10 – Le robot mobile Robucar.

Caractéristiques du *Robucar*	
Longueur /Largeur/ Hauteur	1.836/1.306/0.616 (m)
Poids total	310 (Kg)
Vitesse maximale	18 Km/h (5 m/s)
Autonomie	2 Heures Continues
Capacité d'accueil	2 personnes

TABLE 1.1 – Caractéristiques du *Robucar*

1.7.2 Motorisation

Le *Robucar* fonctionne avec 4 moteurs électriques à courant continu (MP100S B14) de 48 V avec une puissance de 1200 w et une vitesse de rotation de 3500 tr/min.

1.7.3 Les cartes de commande

Le *Robucar* est constitué d'un ordinateur embarqué et de deux cartes ***RSMPC555*** équipées d'un microcontrôleur ***MOTOROLA MPC555***. Ces cartes contrôlent respectivement les moteurs de traction avant et arrière ainsi que ceux de direction. L'ensemble PC embarqué et cartes ***MPC555*** communique via un bus de terrain ***controller area network (CAN)***

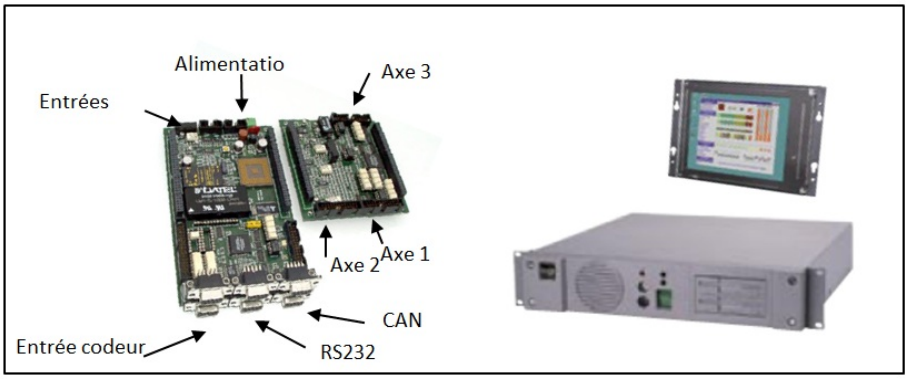

FIGURE 1.11 – Les cartes de commande,le PC embarqué et l'écran tactile

1.7.4 La perception du *Robucar*

Le robot est muni d'un capteur laser et d'une caméra CCD N/B placés sur la face avant du robot, de huit capteurs ultrasons ; quatre placés à l'avant et quatre à l'arrière. Il comporte aussi des encodeurs (liés aux quatre moteurs de traction et aux deux vérins) qui servent à mesurer le déplacement et l'orientation effectués par le robot. Les données provenant de ces capteurs sont présentées de façon à donner la vitesse linéaire et l'angle de braquage du robot à partir desquelles la position du robot peut être calculée.

Le système de mesure laser

Le laser (LMS) émet des impulsions laser qui sont réfléchies par un objet. Le temps mis entre l'émission et la réception de l'impulsion est directement proportionnel à la distance entre le laser et l'objet. Il peut être connecté sur une interface série standard (RS232) ou différentielle (RS422).

Pour permettre son fonctionnement à basse température, un système de chauffage est intégré et déclenché par un thermostat à une température $\leq 10°C$

Les encodeurs (absolu et incrémental)

- Résolution jusqu'à 2048 impulsions
- Sorties A, B, Z en Totem pôle NPN et PNP
- Sorties en Emetteur de ligne pour RS422
- Faible encombrement
- Serrage concentrique par bague
- Consommation 60 mA

- Fréquence de commutation maximale 100 kHz

La caméra

Le *Robucar* est muni d'un capteur caméra CCD SONY EVI 400/401 placé sur l'avant du robot. Les images acquises ont une extension de type « .PPM » avec une dimension de 320 X 240 en niveau de gris.

La figure (Fig : 1.12) regroupe les déférents capteurs du *Robucar* :

FIGURE 1.12 – Les différents capteurs du *Robucar*

1.7.5 Environnement software

Le système actuel de contrôle du robot *Robucar* est composé d'un PC embarqué comportant une version assez ancienne du système d'exploitation ***Red-Hat***, un compilateur ***C/C++*** et deux couches RTAI ainsi qu'un logiciel ***SynDEX*** représentant l'interface entre le programme et le robot.

a - Le logiciel SynDEX

SynDEX (Synchronized Distributed EXecutive) est un logiciel supportant la méthodologie AAA (Adéquation Algorithme Architecture), il est destiné à la conception et à la réalisation de systèmes temps-réel complexes embarqués pour des algorithmes de contrôle et de commande comprenant du traitement de signal et d'images, s'exécutant sur des machines multi composants (réseau de processeur et circuits intégrés spécialisés).

Le logiciel Syndex permet de communiquer avec le robot grâce à la mémoire partagée comportant des informations concernant l'état et le contrôle des composants physiques du *Robucar* permettant leurs contrôles. De plus, pour connaitre l'état des capteurs (LMS, odomètre,..), il suffit d'effectuer une lecture à partir de la structure de données de la mémoire partagée, ou bien une écriture dans cette dernière si on veut envoyer une commande de vitesse aux moteurs. Par ailleurs, Syndex gère tous ces échanges et interactions en temps réel.

b - Modèle global de l'architecture de contrôle

L'architecture actuel du *Robucar* est définie par un ensemble d'entités appelées modules qui s'exécutent en parallèle et qui regroupent des traitements devant être indépendants l'un de l'autre pour qu'ils puissent évoluer en parallèle.

Une tâche sera donc assurée, par un ensemble de modules encapsulant et exécutant les sous-tâches nécessaires. La figure (Fig : 1.13) représente l'architecture de contrôle adaptée au *Robucar*.

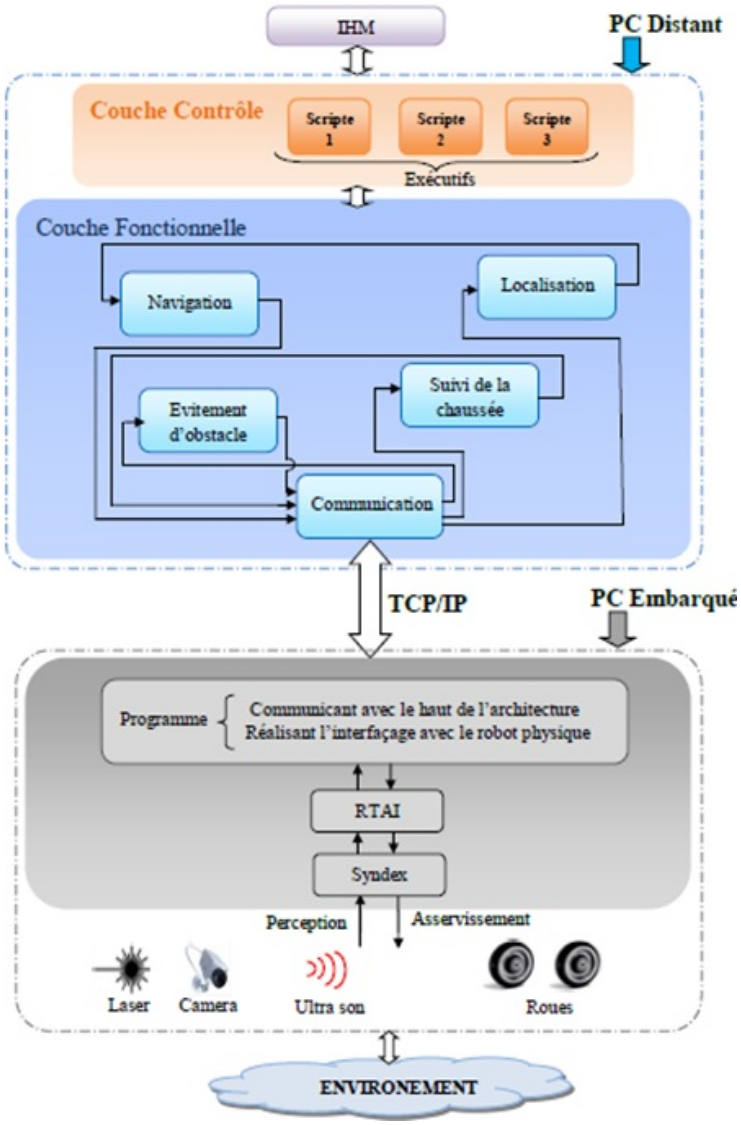

FIGURE 1.13 – Architecture de contrôle du *Robucar*

c - La couche de contrôle

C'est la couche responsable du contrôle des modules de la couche fonctionnelle. Elle regroupe un ensemble de scriptes, comportant des scénarios différents d'exécution (exécutifs), qui seront sélectionnés selon la tâche demandée par l'utilisateur à travers l'interface graphique.

d - La couche fonctionnelle

C'est la couche responsable de tous les traitements réalisés, elle comporte un ensemble de modules : localisation, navigation, communication, évitement d'obstacles et asservissement. Comme il a été déja précisé, ces modules doivent être choisis de manière à ce qu'ils soient indépendants dans les traitements qu'ils exécutent, pour que le parallélisme entre eux soit possible.

1.7.6 Les modules intégrés

a - Module de localisation

Le module de localisation calcule la position actuelle du robot à l'état i, qui sera exprimée dans un environnement $3D(X_i, Y_i, \theta_i)$ où θ_i représente l'angle d'orientation du robot. Cette position sera calculée grâce aux :

- Valeurs odométriques (valeur vitesse V et angle de braquage ϕ) fournies par le module de *communication*.
- La position de l'état $(i-1)$ du robot (l'état précédant $(X_{i-1}, Y_{i-1}, \theta_{i-1})$).

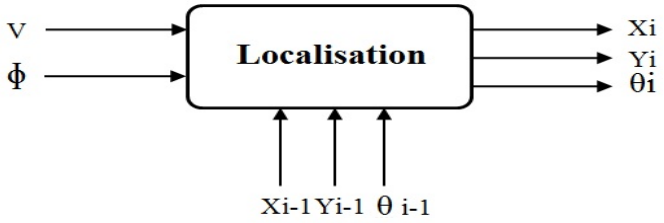

FIGURE 1.14 – Module de localisation

b - Module de navigation

Le module de navigation décide de l'asservissement à entreprendre (commande vitesse V , commande angle roues avants $\phi 1$, angle roues arrières $\phi 2$), pour modifier la trajectoire du robot afin d'aboutir à la position voulue (position but). Pour cela il doit connaitre la position actuelle du robot, qui sera bien sûre fournie grâce au module de *localisation*.

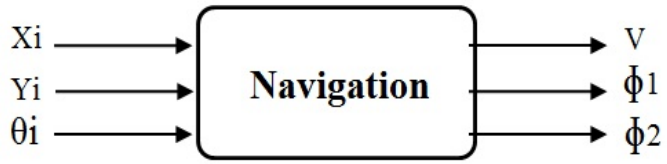

FIGURE 1.15 – Module de navigation

c - Module de communication

Ce module est responsable de la communication entre la couche fonctionnelle et le robot. D'une part, il reçoit les valeurs récoltées par les capteurs et envoyées par le robot et les stocke dans sa

base de données pour que les autres modules puissent les utiliser, d'une autre part, il envoie les commandes d'asservissements $(V\prime, \phi 1, \phi 2)$ au robot pour qu'il les injecte aux effecteurs (moteurs). Ce module peut comporter une phase de traitement, qui vérifie la compatibilité des commandes envoyées au robot.

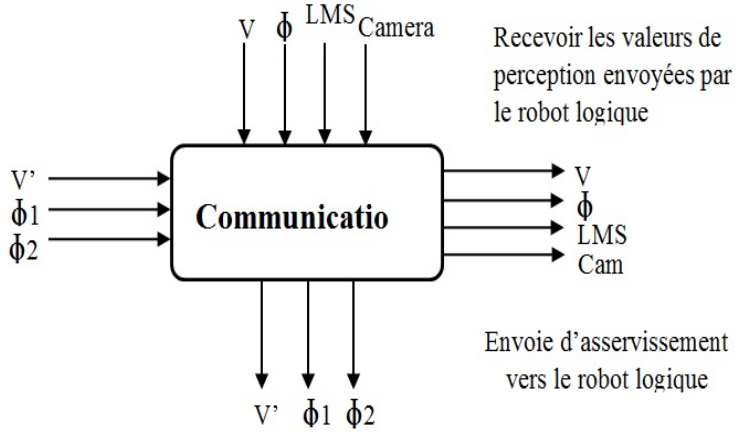

FIGURE 1.16 – Module de communication

d - Module d'évitement d'obstacles

Le module d'évitement d'obstacles permet d'éviter un obstacle détecté grâce aux valeurs de LMS ou de l'image Camera prises à partir du module de communication.

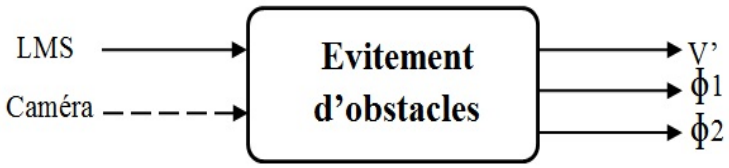

FIGURE 1.17 – Module de d'évitement d'obstacles

e - Module d'asservissement

Ce module calcule les valeurs d'asservissement $(V\prime, \phi 1, \phi 2)$ dans le but d'effectuer une tache bien spécifique telle que le suivi de trajectoire et cela en utilisant les images caméra fournies par le module de communication, ainsi les différentes commandes que nous allons développer dans le cadre de notre travail seront destinées à enrichir ce module afin de permettre au robot d'effectuer différentes taches d'asservissement visuel.

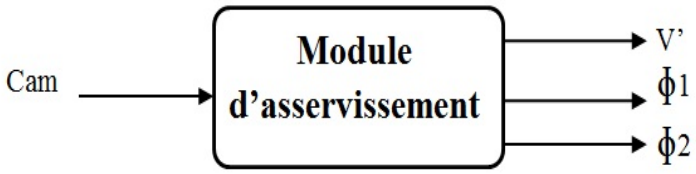

FIGURE 1.18 – Module d'asservissement

1.8 Conclusion

Dans ce premier chapitre, nous avons définit les principales notions fondamentales de la robotique mobile. Ainsi, après une définition des robots mobiles autonomes et l'introduction des aspects matériels et fonctionnels qui les caractérisent, les différents moyens de perception utilisés ont été abordés. En effet, ces derniers sont indispensables pour la réalisation des différentes tâches de navigation. Ils permettent au robot de percevoir l'environnement dans lequel il évolue et faire face aux différentes situations imprévues afin de mener à bien sa mission.

Par la suite, nous avons abordé les différentes étapes de la navigation d'un robot mobile autonome, allant de la perception et la modélisation de l'environnement à l'exécution des actions, en passant par l'étape de décision et de planification. Toutes ces étapes sont menées selon différentes stratégies de navigation et moyennant différentes architectures de contrôle.

Pour la modélisation des robots mobiles non-holonomes, nous avons commencé par des rappels sur la notion de non-holonomie, ainsi que celle du roulement sans glissement, pour nous intéresser ensuite au modèle cinématique des robots mobiles de type voiture. En effet, ce dernier sera utilisé dans le cas de notre robot *Robucar*.

A la fin de ce chapitre, les différentes caractéristiques du robot mobile *Robucar*, sujet de notre étude, ont été présentées. Elles incluent ses différents composants et moyens de perception, ainsi que les modules intégrés visant sa navigation de manière autonome.

Dans le prochain chapitre, nous présenterons un état de l'art sur l'asservissement visuel en robotique mobile. Ce dernier englobera toutes les notions théoriques nécessaires à l'étude des problématiques liées à ce domaine. Aussi, une vue générale sur les avancées ainsi que les travaux réalisés en asservissement visuel sera présentée.

Chapitre 2

Etat de l'art sur l'asservissement visuel

2.1 Introduction

L'utilisation de l'asservissement visuel en robotique mobile connait un essor important notamment pour les applications de conduite automatique de véhicules.

Dans ce chapitre nous commençons par introduire le principe général d'une commande référencée vision et présenter ensuite les différentes techniques d'asservissement visuel qu'il est possible d'employer afin de positionner un robot mobile par rapport à son environnement. Ces différentes techniques seront classées en fonction des mesures utilisées en entrée du module de commande. Ainsi, nous commençons par l'asservissement 3D , pour aborder ensuite des techniques plus récentes à savoir l'asservissement visuel $2D\frac{1}{2}$ et $\frac{d2D}{dt}$, et détailler ensuite l'asservissement 2D sur lequel se base notre travail.

2.2 Commande référencée vision

La commande référencée vision représente la variété la plus répandue en termes de commande référencée capteurs, dans le domaine de la robotique. Elle consiste à contrôler les mouvements d'un système robotique en utilisant des informations visuelles, notées s, issues d'un ou plusieurs capteurs de vision, embarqués ou non sur le système. Dans la littérature, de nombreux travaux sont basés sur l'exploitation de ces données pour réaliser différentes tâches robotiques (positionnement face à un objet, suivi, saisie, etc.). La première utilisation de la vision en boucle fermée date des années soixante-dix on parlait alors de retour visuel (Visual Feedback) et ce n'est qu'à partir de 1979 que ce type de loi de commande est appelé Asservissement Visuel (Visual Servoing) [Ouadah 11].

Malheureusement, l'évolution de ce type de commande a été freinée, par les limitations des moyens de calcul, jusqu'à la fin des années 80. Au début des années 90, plusieurs études se sont alors intéressées à cette discipline émergente, essentiellement pour des applications sur des bras manipulateurs [Chaumette 90].

L'approche de l'asservissement visuel répond à une idée intuitive du mouvement associé à la vision : le robot génère ses mouvements de façon à ce que sa cible visuelle atteigne une certaine configuration dans l'image qu'il perçoit du monde à l'aide d'une ou plusieurs caméras. Il existe plusieurs types d'asservissement visuel. Par exemple, on peut utiliser dans la boucle de commande des informations tridimensionnelles, tirées du traitement d'images. On peut aussi utiliser directement les primitives visuelles, ou faire employer un mélange des deux (données 3D et données 2D en même temps) [Malis 98]. Une classification détaillée des différentes structures de commande en asservissement visuel est présentée par la suite, après introduction des généralités concernant la vision en robotique mobile.

2.3 Généralités sur la vision en robotique mobile

Chez l'être humaine, la vision est le sens qui transmet le plus d'informations au cerveau. Elle permet de construire une représentation très riche de notre environnement et ainsi d'entreprendre une interaction intelligente avec cet environnement qu'il soit statique ou dynamique. De ce point

de vue, il est intéressant de constater que les chercheurs en robotique ont rapidement essayé de faire bénéficier les robots de capacités de vision pour accomplir des tâches essentielles telles que la navigation dans un environnement inconnu [Benseddik 11].

Le capteur de vision est assurément l'un des capteurs extéroceptifs les plus riches en informations : il fournit, en effet, non seulement une seule information, comme c'est par exemple le cas des capteurs proximétrique [Chaumette 90], mais un signal vidéo extrêmement riche, prenant la forme d'un tableau numérique de pixels définis par leur position et leur couleur. L'exploitation de ce signal est donc lié à l'informatique, et peut se situer à différents niveaux. Le plus bas niveau est de considérer directement les informations pixelliques, qui sont les plus riches mais n'ont pas de sens géométrique à proprement dit. Au niveau du dessus, l'extraction de primitives géométriques (points, lignes ,contours et surfaces)est une approche populaire par sa facilité d'implantation et la richesse des informations délivrées quant à l'interaction avec l'environnement.

Enfin, il est possible de suivre un objet dans l'image tout en estimant sa pose par rapport à la caméra, à partir d'un modèle 3D de l'objet ou de sa texture [Lefebvre 06].

• **capteurs vision** Trois grandes classes de capteurs apparaissent chez les utilisateurs :

- Les caméras passives (une , deux ou trois caméras).
- les capteurs semi-actifs à lumière structurée.
- la vision basée sur les télémètres imageurs [Khadraoui 96].

En général, les applications de vision pour la robotique passent par deux principales phases, qui consistent à l'extraction de certaines informations pertinentes de l'image, puis à l'utilisation de ces informations pour une tâche donnée (contrôle, cartographie, etc.). la première phase d'extraction de l'information est plutôt un problème de traitement d'image, tandis que la seconde n'est pas forcément spécifique aux applications de vision, par exemple, la planifications d'une trajectoire à exécuter [Benseddik 11]. Ainsi les informations que l'on peut extraire d'une image peuvent donc être extrêment variées de plus ou moins haut niveau selon les différents traitements que l'on effectue sur cette image : de l'intensité lumineuse d'un pixel à la reconnaissance et à la localisation dans l'image d'un objet. Bien entendu, plus les informations sont de haut niveau, plus les algorithmes de traitement d'image nécessaires à leur extractions sont complexes [Chaumette 98].

2.3.1 Rappel sur la notion d'image

a - Image numérique Une image numérique $a = [m, n]$ décrite dans un espace discret 2D, est obtenue à paritr d'une image analogique $a(x, y)$ par un processus d'échantillonnage, appelé numérisation [Benseddik 11].

La numérisation est le processus qui permet de passer de l'état d'image physique (image optique par exemple) qui est caractérisée par l'aspect continu du signal qu'elle représente (une infinité de valeurs dans l'intensité lumineuse par exemple), à l'état d'image numérique qui est caractérisée par l'aspect discret (l'intensité lumineuse ne peut prendre que des valeurs quantifiées en un nombre fini de points distincts). C'est cette forme numérique qui permet une exploitation ultérieure par des outils logiciels sur ordinateur[Alendre et al 06].

Une image numérique est donc un signal fini bidimensionnel échantillonné a valeurs quantifiées

dans un certain espace de couleurs, elle est représentée par une matrice d'entiers pour l'image monochrome ou par 3 matrices d'entiers pour une image couleur (chacune représentant un sous-espace de l'espace des couleurs, ex : la couleur rouge dans le système RGB)[Map-toul].

b - Caractéristiques d'une image

1. **Le pixel**

   Un pixel, c'est littéralement "un élément d'une image" : le mot pixel vient de l'anglais "*picture element*".

   Un pixel correspond au plus petit élément d'une surface d'affichage auquel on puisse associer individuellement une couleur (ou un niveau de gris) et une intensité. Dans le cas d'un écran monochrome, le pixel s'identifie à un point et, dans le cas d'un écran couleur, il est constitué de trois points de couleurs différentes (rouge, vert, bleu). En variant l'intensité de chacun des points, on peut faire apparaître des milliers de couleurs différentes. La quantité de pixels composant l'écran détermine la résolution [Laumond 01].

   Plus le nombre de pixels est élevé par unité de longueur de la structure à numériser, plus la quantité d'informations qui décrit cette structure est importante.

2. **Définition**

   On appelle définition le nombre de points (pixel) constituant l'image, c'est-àdire le nombre de colonnes de l'image que multiplie son nombre de lignes. Exemple : Une image possédant 3000 pixels en largeur et 2000 en hauteur aura une définition de 3000 pixels par 2000, notée $3000 * 2000$.

3. **Résolution**

   La résolution est le nombre de points par unité de surface, exprimé en points par pouce PPP, en anglais DPI pour(Dots Per Inch) ; un pouce représentant 2.54cm [Lerond 06].

4. **Niveau de gris**

   Le niveau de gris est la valeur de l'intensité lumineuse en un point. Une image en niveau de gris est une image composée de points gris plus ou moins foncés (Fig : 2.1 [Rouilly 00]).

   Pour chaque point, l'ordinateur enregistre une valeur de gris entre le noir et le blanc. Par convention la valeur Zéro représente le noir (intensité lumineuse nulle) et la valeur 255 le blanc (l'intensité lumineuse maximale), chaque pixel étant représenté par un octet, (en général on sauvegarde les images à 256 teintes de gris).

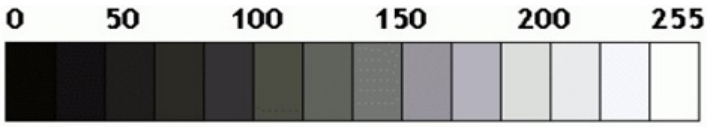

FIGURE 2.1 – Les niveaux de gris

5. **Histogramme** Un histogramme (Fig : 2.2 [Rouilly 00]) est un graphe statique, appelé aussi « diagramme en bâtons ». Il représente la fonction H définie sur l'ensemble des entiers naturels par :

$$H(x) = CardP : I(P) = x$$

Où H(x) traduit le nombre d'apparitions du niveau de gris x dans l'image I.

Par convention un histogramme représente le niveau d'intensité en abscisse en allant du plus foncé (à gauche) au plus clair (à droite). L'histogramme est un outil privilégié en analyse d'images car il représente un résumé simple[Boucher]. Ainsi l'histogramme d'une image en 256 niveaux de gris sera représenté par un graphique possédant 256 valeurs en abscisses,et le nombre de pixels de l'image en ordonnées.

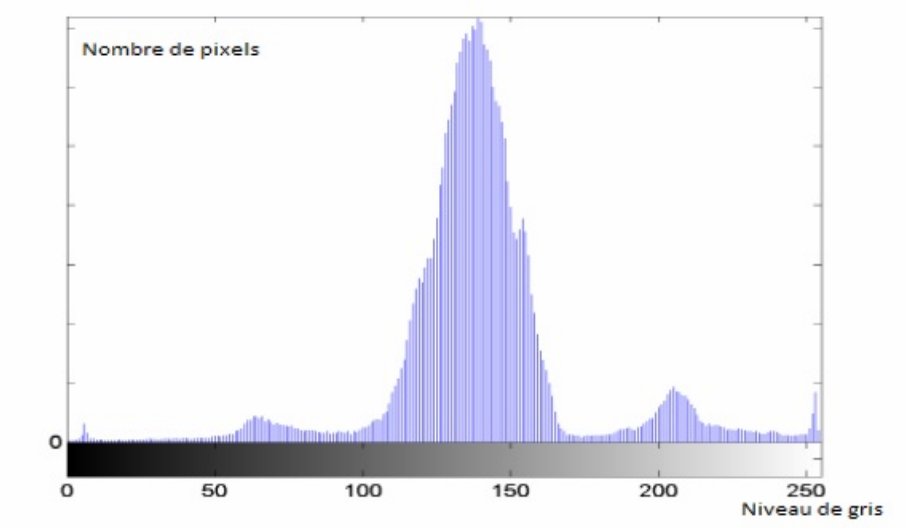

FIGURE 2.2 – Histogramme d'une image

2.3.2 Les techniques de traitement d'images en robotique mobile

Les techniques de traitement d'images utilisées en robotique mobile peuvent être classées en trois catégories, selon le niveau d'abstraction :

- Les processus de bas niveau, qui nécessitent très peu d'informations sur le contenu des images. Il s'agit de processus qu'on peut regrouper sous l'appellation prétraitement d'images et extractions d'indices ;
- Les processus de niveau intermédiaire, qui englobent les techniques de segmentation d'images en régions ;
- Les processus de haut-niveau, qui peuvent nécessiter des informations sur le contenu des images, et dans lesquels on trouve, la reconstruction tridimensionnelle (modélisation de l'environnement), la reconnaissance de formes, l'analyse de mouvements, etc.

a - Les techniques de bas-niveau (les prétraitements) Les prétraitements permettent la préparation des données reçues des capteurs à la phase suivante d'analyse consacrée à l'extraction

des paramètres. Cette phase n'est possible et surtout fiable que si les données des capteurs sont dénuées de bruit, corrigées de leurs erreurs éventuelles, homogénéisées et réduites à l'essentiel [Belaid 91]. Les opérateurs de prétraitement sont divisés en trois catégories :

- Opérateurs de filtrage (réduction de bruit),
- Opérateurs de rehaussement du contraste,
- Opérateurs de modification d'histogramme[Cocquerez et Philip 95].

1. **Le filtrage :**

   L'image possède dans sa nature une redondance spatiale en information ; les pixels voisins en 4-voisins ou en 8-voisins possèdent les mêmes caractéristiques du point de vue niveau de gris. Une présence de bruit dans une image provoque une variation brutale de niveau de gris d'un pixel par rapport aux autres voisins. Sur un profil d'une image, cette variation apparait de la façon suivante (Fig : 2.3 [Rouilly 00])

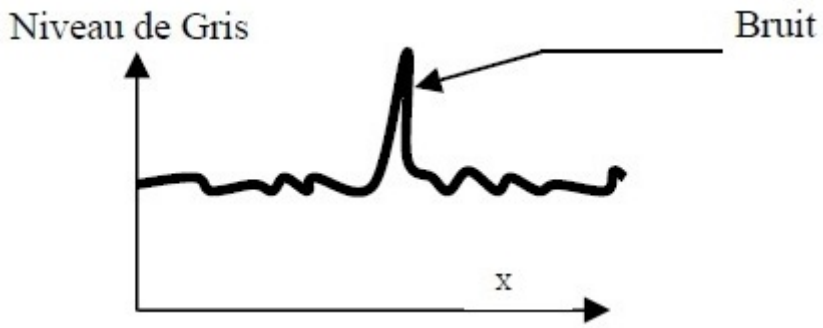

FIGURE 2.3 – Variation du bruit sur un histogramme

   Les opérations de réduction (filtrage) du bruit sont divisées en deux catégories ; les filtres linéaires et les filtres non linéaires. Le type de filtrage le plus simple et facile à implémenter est basé sur le filtrage linéaire[Boucher].

   Le principe de filtrage est de modifier la valeur des pixels d'une image, généralement dans un but d'améliorer son aspect. En pratique, il s'agit de créer une nouvelle image en se servant des pixels de l'image d'origine. Sa qualité sera améliorée.

   Le filtrage manipule uniquement les données de l'image numérisée. On ne fait aucune supposition sur ce que représente l'image. Le résultat obtenu suite à un filtrage dépend donc énormément de la qualité du signal de l'image d'origine [Alendre et al 06]. Il existe deux type de filtrage ; le filtrage global et le filtrage local. Dans le filtrage global chaque pixel de la nouvelle image est calculé en prenant compte de la totalité des pixels de l'image de départ. Quant au filtrage local, chaque pixel de la nouvelle image est calculé en prenant en compte seulement un voisinage du pixel correspondant dans l'image d'origine.

FIGURE 2.4 – Exemple de filtrage

2. **Opérateurs du rehaussement du contraste :**

   Pour un certain type d'images, il arrive que la transition entre les régions soit floue ; l'objectif cherché par le rehaussement du contraste est de diminuer l'étendue de la zone de transition sans affecter l'intensité moyenne des régions[Cocquerez et Philip 95].

   Modification d'histogramme :

   La modification d'histogramme est une fonction qui fait correspondre à chaque niveau de gris d'un pixel de l'image originale un autre niveau de gris. Chaque niveau de gris est modifié dans le but d'accroître le contraste. L'approche la plus simple se caractérise par l'équation :

$$G(x, y) = t(g(x, y))$$

   Où t est une fonction prédéfinie, et $g(x, y)$ correspond à la valeur du niveau de gris du pixel (x, y)[Boucher]. Les deux opérations principales de la modification de l'histogramme sont l'égalisation et l'élargissement.

-]- **Egalisation :**

  L'égalisation d'histogramme a pour but d'harmoniser la répartition des niveaux de luminosité de l'image, de telle manière à tendre vers un même nombre de pixels pour chacun des niveaux de l'histogramme. Cette opération vise à augmenter les nuances dans l'image[Harchaoui 08].

-]- **Elargissement (étirement) :**

  L'élargissement d'histogramme (aussi appelé « linéarisation d'histogramme » ou « expansion de la dynamique ») consiste à répartir les fréquences d'apparition des pixels sur la largeur de l'histogramme. Ainsi, il s'agit d'une opération consistant à modifier l'histogramme de telle manière à répartir au mieux les intensités sur l'échelle des valeurs disponibles. Ceci revient à étendre l'histogramme afin que la valeur d'intensité la plus faible soit à zéro et que la plus haute soit à la valeur maximale. De cette façon, si les valeurs de l'histogramme sont très proches les unes des autres, l'étirement va permettre de fournir une meilleure répartition afin de rendre les pixels clairs encore plus clairs et les pixels foncés proches du noir [Harchaoui 08].

b - Les techniques de segmentation La segmentation des images est l'outil de base du traitement d'images. Quelle que soit son origine, une image constitue une représentation d'un

univers composé d'entités : objets dans une scène intérieure, cellules, organes du corps humain, routes, etc.

Le but de toute méthode de segmentation est l'extraction d'attributs caractérisant ces entités. Les attributs étudiés correspondent à des points d'intérêt ou à des zones caractéristiques de l'image : primitives. Les images acquises par les caméras ne sont pas exploitables à l'état brut pour la compréhension de la scène, car elles contiennent trop d'informations. Il faut donc, procéder à l'extraction d'indices pertinents et ne pas considérer l'image dans sa totalité, ceci réduit l'information contenue dans l'image sans trop la dégrader. Les indices visuels, souvent appelés primitives, doivent être tels que leur extraction soit avec une extrême sécurité, sans perte d'informations utiles. D'où la nécessité de certains critères de choix auxquels ces indices doivent répondre[Djekoune 98].

Parmi les primitives existant nous citons : primitive arcs de cercle, primitive points d'intérêts, primitive région et primitive segment de contours.

b.1 Détection de contours : La détection de contours dans les images a débuté de façon extrêmement expérimentale par des opérateurs locaux qui, soit estimaient un gradient, soit appliquaient la convolution sur l'image par des masques caractéristiques des contours[Haralick et Paindavoinee 00].

Les contours sont caractérisés par des discontinuités de la fonction d'intensité dans les images. Le principe de la détection de contours repose donc sur l'étude des dérivées de la fonction d'intensité dans l'imag, les extrema locaux du gradient de la fonction d'intensité et les passages par zéro du Laplacien. La difficulté réside dans la présence de bruit dans les images[Ufrima 07].

Le principe de la détection de contours repose donc sur l'étude des dérivées de la fonction d'intensité dans l'image.

Trois techniques dérivatives générales sont définies :

- Approche dérivée première (gradient).
- Approche dérivée seconde (Laplacien).
- Approche du filtrage optimal[Cocquerez et Philip 95].

Parmi les filtres les plus utilisés, on peut citer :

- Filtre de CANNY.
- Filtre de DERICHE
- Filtre de SHEN *et* CASTAN.

b.2 La transformation de Hough : La Transformée de Hough (TH) a été développée par Paul Hough en 1962, et a été brevetée par IBM. Dans les dernières décennies, la transformée de Hough est devenue un outil standard dans le domaine de la vision artificielle. Elle permet la détection de droites, de cercles ou d'ellipses en exploitant la primitive de type segments de contours. Elle peut aussi être étendue à des cas de description d'objets plus complexes[Rouilly 00].

-] **Principe de la transformée de Hough :**

Cette transformée part du principe que pour un point donné, il existe une infinité de droite passant par celui-ci, la seul chose qui change entre chaque droite est son angle. Le but de la transformée est de trouver les lignes potentiels se rapprochant le plus de ce que l'on attend. Pour déterminer si

plusieurs points sont approximativement sur une même droite, il faut faire une représentation de la droite et des points dans un repère permettant cette comparaison.

Dans le repère complexe, chaque ligne est un vecteur de coordonnées paramétriques, ρ étant la norme du vecteur et θ son angle. Si l'on trace toutes les lignes possible caractérisant ce point, en calculant donc le ρ et le θ de chacune des possibilités, on obtient une sinusoïde unique appelée espace de Hough.

Comme les possibilités sont infinies, on stocke ces valeurs dans un tableau que l'on appelle accumulateur : ce tableau à deux entrées (ρ et θ), est un ensemble d'intervalles de ρ et θ.

Une fois le parcours de tous les points effectué, on prend la case contenant la plus grande valeur, et ainsi on obtient la droite qui passe approximativement par le plus de points de l'image[Roger 07].

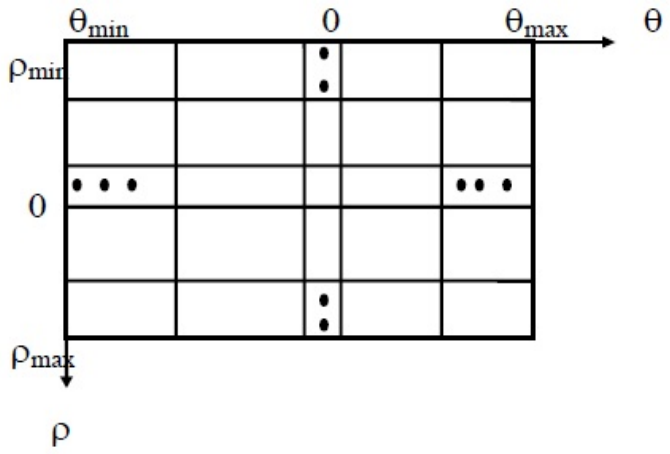

FIGURE 2.5 – Quantification du plan des paramètres ρ, θ

c - Les techniques de haut-niveau : Une fois les informations extraites sous forme d'un ensemble d'entités, la dernière étape consiste à prendre une décision en fonction de ces informations. Selon l'application visée, cette décision peut être une simple valeur booléenne (l'image fait partie de la catégorie visée ou non). Ou bien une commande moteur (le cas de l'asservissement et de l'évitement d'obstacles).ou un ensemble plus complexe de traitements (mise à jour d'une carte par exemple).

2.4 Techniques d'asservissement visuel

2.4.1 Asservissement visuel 3D

L'approche 3D est la plus intuitive et c'est sans doute pour cela qu'elle a été implémentée en premier. A cette époque, la période de l'asservissement visuel était de l'ordre de la dizaine de secondes. Ce n'était d'ailleurs pas vraiment un asservissement mais plutôt une commande séquentielle de type ≪ look then move ≫ [Gangloff 04].

Comme son nom l'indique, l'asservissement visuel 3D (position-based visual servoing) utilise en entrée de la boucle de commande des informations tridimensionnelles,[Chaumette 98], à savoir la situation r de la caméra, par rapport à l'objet d'intérêt. La tâche à réaliser s'exprime alors

sous la forme d'une situation de référence $s^* = r^*$ à atteindre. La commande repose ainsi sur la détermination de la situation $s = r$ de la caméra, à partir des informations visuelles extraites de l'image [Folio 07].

La figure (Fig : 2.6) d'écrit le schéma de principe d'un asservissement visuel basé sur une mesure 3D. La cosigne r^* est un vecteur donnant les coordonnées d'attitude désirées de l'objet par rapport à la caméra. Cette consigne est comparée à la mesure r issue d'un algorithme de reconstruction 3D. Le correcteur fournit au robot les commandes adéquates pour que r converge vers r^* [Gangloff 04].

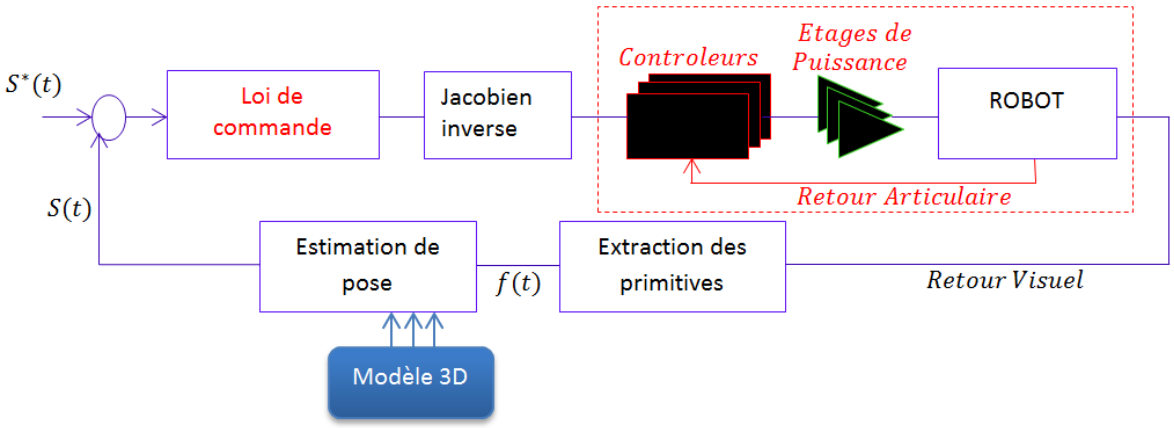

FIGURE 2.6 – Asservissement visuel 3D

De nombreuses méthodes permettent d'estimer la situation d'une caméra par rapport à un objet à partir de l'image perçue de cet objet. Elles reposent très généralement sur la connaissance a priori d'un modèle (3D ou 2D) de l'objet et des paramètres intrinsèques de la caméra. Ces méthodes utilisent des informations visuelles de différente nature, telle que des points, des droites, des ellipses résultant de la projection de cercles ou de sphères ou encore des objets cylindriques. Par contre peu de méthodes combinent des types différents d'informations visuelles. Toutefois des travaux, issus des recherches en reconstruction 3D par vision dynamique, permettent d'estimer le modèle de l'objet d'intérêt, ou de localiser la caméra à partir de mesures de mouvement 2D ou 3D, élargissant ainsi les tâches considérées [Folio 07].

En théorie, une seule itération est suffisante et une commande du robot en position permet d'atteindre la position désirée. En pratique évidement, les diverses erreurs de modèle et de calibration nécessitent un rebouclage et un asservissement en vitesse pour assurer la stabilité du système [Chaumette 98] Parmi les avantages liés à l'utilisation d'une telle approche, on peut citer celui de l'espace de contrôle qui est équivalent à l'espace opérationnel du robot, et qui permet de simplifier la définition de la tâche robotique [Martinet 99].

Mais, il faut noter que dans cette méthode, il n'y a strictement aucun contrôle dans l'image, donc les informations visuelles nécessaires peuvent sortir de l'image pendant l'asservissement, ce qui empêche la réalisation de la tâche. De plus, cette approche requiert une interprétation du motif visuel pour caractériser la situation du robot, et nécessite donc une reconstruction de l'état du système. On y retrouve alors les mêmes problèmes d'incertitude et de précision des schémas de commande des robots mobiles classiques. De très récentes études ont montré l'efficacité de ce type

de loi de commande, pour des applications d'assemblages en micromanipulation [Ouadah 11].

2.4.2 Asservissement visuel $2D\frac{1}{2}$

Cette technique, proposée par *Chaumette*, est une variante de la loi de commande habituellement utilisée pour un asservissement 2D. Elle est particulièrement intéressante puisqu'elle ne présente pas la plupart des inconvénients respectifs des deux approches précédemment présentées. Il s'agit de l'asservissement visuel $2D\frac{1}{2}$, car les informations utilisées comme mesures et consignes sont, pour certaines d'entre elles, exprimées directement dans l'image et, pour les autres, exprimées dans le repère de la caméra [Ouadah 11].

Cette technique est basée sur l'estimation de l'homographie, notée H, qui relie l'image d'au moins trois points entre différents plans projectifs. L'homographie est une application projective bijective, correspondant à une transformation linéaire entre deux plans projectifs. Plus précisément, elle permet d'établir (à un facteur d'échelle près) une bijection entre un objet de l'espace 3D et une image 2D, ou bien entre deux images 2D d'un même objet [Folio 07]. un minimum de quatre points appariés sur les images courante et désirée permet l'estimation de l'homographie par la résolution d'un système linéaire.

Dans le cas d'un objet quelconque, un appariement de huit points permet d'aboutir, là-encore par la résolution de systèmes linéaires. A partir de l'homographie obtenue, il est possible de calculer le déplacement en rotation que la caméra doit effectuer pour atteindre sa position spécifiée ainsi que la direction de son déplacement en translation. L'homographie fournit également le rapport $\frac{d}{d^*}$ entre les distances courante et désirée de la caméra à l'objet. Comme la translation à réaliser n'est connue qu'à facteur d'échelle prés, un asservissement visuel purement 3D est impossible. Cependant, la combinaison des informations 2D et 3D disponibles permet d'aboutir à une solution aux propriétés particulièrement intéressantes [Chaumette 98].

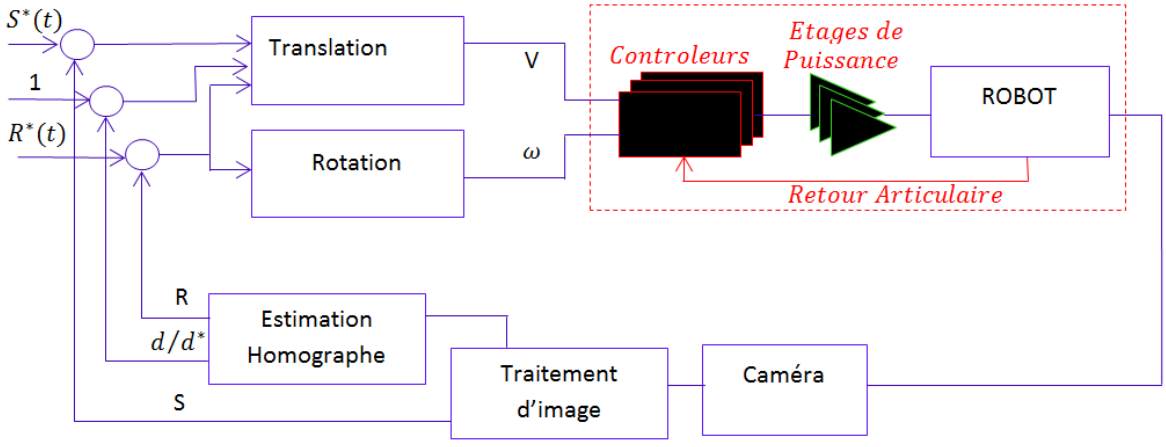

FIGURE 2.7 – Asservissement visuel $2D\frac{1}{2}$

Le vecteur des informations visuelles s est donc défini sur la base de donnée 2D et 3D. Ainsi, l'asservissement visuel $2D\frac{1}{2}$ est une approche intermédiaire entre l'asservissement 3D et 2D. La boucle d'asservissement ainsi synthétisée permet de séparer la rotation et la translation de la

caméra [Folio 07] ce qui permet :

- Un fort découplage de la loi de commande ;
- Un contrôle partiel dans l'image permettant de conserver la cible en permanence dans le champ de vision de la caméra ;
- Une étude de la stabilité et du domaine de convergence de la loi de commande ;

Il a été ainsi obtenu des conditions suffisantes de stabilité prenant en compte les erreurs de calibrage et les erreurs d'estimation de l'homographie qui en découlent. De ce fait, pour des positions initiales de la caméra très éloignées de la position désirée, la convergence est obtenue là où l'asservissement visuel 2D échoue [Ouadah 11].

L'avantage majeur de cette approche est que la connaissance d'un modèle géométrique 3D de l'objet n'est plus nécessaire, ce qui lui donne un domaine d'application très important à partir du moment où le motif à atteindre est préalablement déterminé, par exemple lors d'une phase d'apprentissage. La seule information 3D utilisée dans la commande est la profondeur désirée approximative d'un point de l'objet, soit un besoin d'informations a priori bien moindre qu'en asservissement visuel 3D et 2D. L'utilisation de droites comme informations visuelles pour estimer l'homographie est également possible dans le cas d'un objet planaire. Par contre, l'utilisation de primitives plus complexes (telles des ellipses) semble plus délicate [Chaumette 98].

2.4.3 Asservissement visuel $\frac{d2D}{dt}$

Les méthodes que nous avons décrites jusqu'à présent reposaient sur l'utilisation d'informations visuelles géométriques (coordonnées de points, paramètres représentant l'image d'une droite, etc.). Les contraintes imposées sont donc que ces primitives géométriques existent dans la scène, mais surtout qu'il est possible de les extraire et de les suivre par traitement d'image à une cadence assez élevée afin de conserver la robustesse et la stabilité des lois de commande. Actuellement, les systèmes les plus fiables sont limités soit au suivi de segments, soit au suivi de points caractéristiques ou de formes simples.

En pratique, on utilise le plus souvent des marqueurs disposés sur la scène observée afin d'extraire les positions de leur projection perspective dans l'image [Chaumette 98]. Ce type d'asservissement visuel utilise une vitesse relative entre la caméra et la cible comme grandeur asservie. La référence est définie dans le plan image par un champ de vitesse des points. Le principe de la commande consiste alors à contrôler les mouvements de la caméra de telle sorte que le mouvement 2D mesuré atteigne un champ de vitesse désirée d'où l'appellation d'asservissement visuel $\frac{d2D}{dt}$ [Ouadah 11].

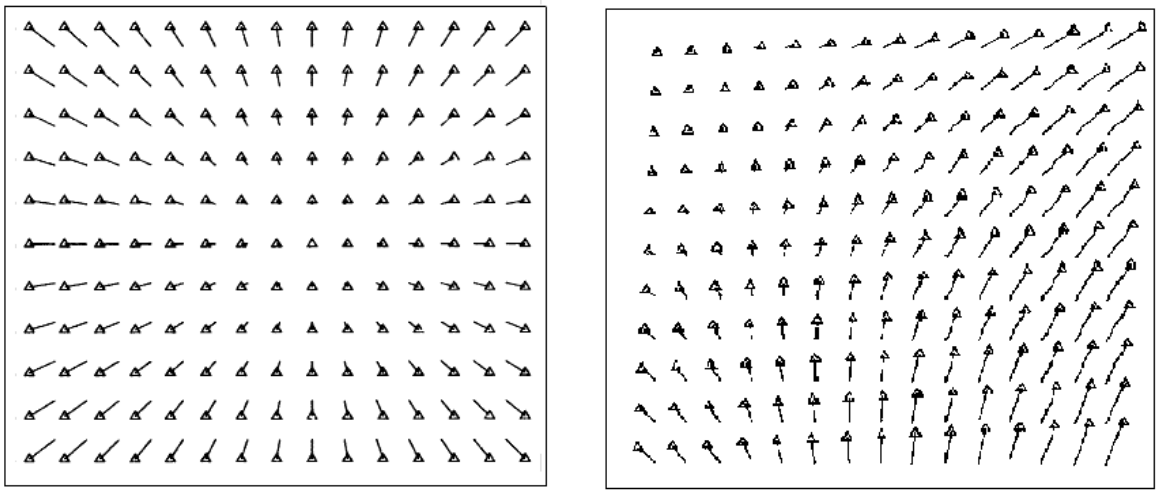

FIGURE 2.8 – Exemples de champs de vitesse

Contrairement au cas de mesure géométrique, la mesure s ainsi que sa valeur désirée s^* correspondent désormais à des paramètres P_i du modèle de mouvement 2D dans l'image. Dans ce cas alors, la loi de commande ne contraint plus seulement le torseur cinématique de la caméra τ mais aussi son accélération Γ. La mesure s peut être établie par des techniques classiques de flot optique. Une fonction quadratique est souvent choisie pour modéliser le champ de vitesse. La loi de commande est définie par rapport aux paramètres de cette fonction. Par ailleurs, l'emploi de ces nouvelles informations visuelles permet d'appréhender de manière plus simple des tâches extrêmement délicates à modéliser en utilisant des informations visuelles géométriques [Ouadah 11].

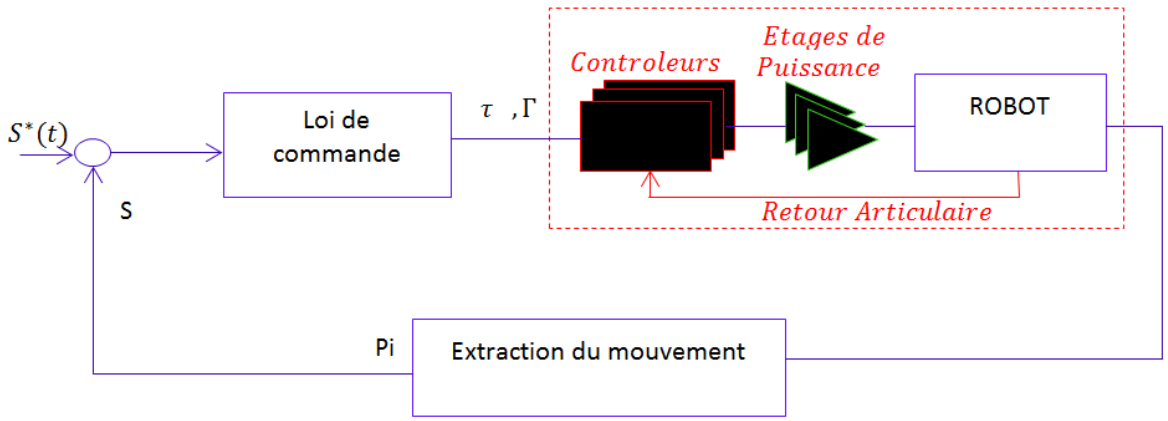

FIGURE 2.9 – Asservissement visuel $\frac{d2D}{dt}$

L'asservissement visuel $\frac{d2D}{dt}$ n'a pas connu une très grande utilisation, et reste aujourd'hui encore à un stade émergeant, car les traitements d'image qu'il nécessite sont encore très longs et fournissent des informations très bruitées. Des applications pratiques ont néanmoins été réalisées. Par exemple, un asservissement de l'alignement de l'axe optique d'une caméra animée d'un mouvement de translation suivant la direction de cette translation, une tâche de positionnement telle que le plan image de la caméra soit parallèle au plan d'une cible observée .Dans un cadre plus général, la relation linéaire entre les variations de l'information visuelle et la vitesse de la caméra, représentée par la matrice d'interaction pour des informations visuelles géométriques, n'est généralement plus valide, en particulier en raison de l'apparition de termes liés à l'accélération de la caméra et de termes quadratiques. Actuellement, les travaux sont plus orientés vers l'élaboration de lois de commande en accélération ou, si besoin, combinant accélération et vitesse.

Il faut finalement signaler que, même si des progrès énormes ont été récemment faits, les algorithmes de traitements d'image utilisés sont encore relativement coûteux en temps de calcul si l'on considère un modèle affine du mouvement (et encore davantage pour un modèle quadratique). Ils fournissent en outre des résultats beaucoup plus bruités que ceux employés pour extraire des informations visuelles géométriques. Á des cadences de l'ordre de 1 Hertz, la validation expérimentale des techniques d'asservissement visuel $\frac{d2D}{dt}$ est donc particulièrement délicate. Par ailleurs, il est possible de remonter à des informations visuelles géométriques par simple intégration des paramètres de mouvement estimés (si une localisation initiale est disponible) .On peut alors utiliser directement les lois de commande qui ont été développées en asservissement visuel 2D. Cette approche a été validée sur une tâche de poursuite d'une cible quelconque (tel un piéton par exemple) en contrôlant l'orientation de la caméra [Ouadah 11].

2.4.4 Asservissement visuel 2D

Les techniques d'asservissement visuel 2D utilisent elles directement les informations visuelles, notées s, extraites de l'image. La tâche à réaliser est alors spécifiée directement dans l'image en termes d'indices visuels de référence s^* à atteindre [Folio 07]. Les lois de commande consistent alors à contrôler le mouvement de la caméra afin que les mesures dans l'image s(t) atteignent une valeur désirée s^* , voir suivent une trajectoire spécifiée $s^*(t)$ [Chaumette 98]. Cette approche permet donc de s'affranchir de l'étape de reconstruction 3D de la cible et des problèmes qui y sont liés [Folio 07].

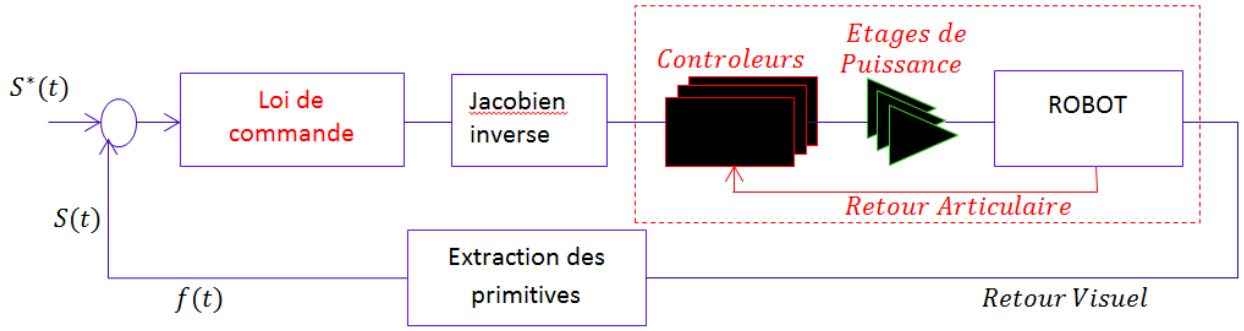

FIGURE 2.10 – Asservissement visuel 2D

Avantages des asservissements visuels 2D

Alors que l'asservissement visuel 3D implique la connaissance d'un modèle tridimensionnel de la cible et de pouvoir la situer dans un repère commun, et de définir une consigne d'asservissement en termes de position et orientation, l'asservissement visuel 2D travaille au niveau de l'image, avec des primitives visuelles (indices) définies au préalable [Ouadah 11].

Ainsi, les asservissements visuels 2D sont, d'une manière générale, des lois de commandes relativement rapides à calculer, puisqu'il n'y a pas de phase de reconstruction 3D. Ce gain de temps n'est pas négligeable puisque le délai d'application de la commande est réduit, ce qui contribue à la stabilité du système [Folio 07].

De plus, les asservissements basés sur l'image permettent la réalisation de tâches de manière très efficace et précise. C'est ainsi que ce type de commande se rencontre de plus en plus dans différents domaines d'application. Citons par exemple la conduite d'engins sous-marins, de véhicules autonomes personnels ou agricoles [Khadraoui 96]. La robotique chirurgicale apparaît également comme un champ d'application privilégié des dernières avancées de l'asservissement visuel 2D. [Folio 07].

2.5 Asservissement visuel 2D en robotique mobile

Les techniques de commande référencées capteurs, présentées ci-dessus, permettent à un robot de réaliser des tâches de diverses natures.

L'implémentation d'une loi de commande sur un système robotique ne peut se faire correctement sans passer par une phase de modélisation (ou d'identification du modèle). Cette dernière doit être menée avec rigueur, en vue d'obtenir le modèle le plus proche de la réalité, et garantir de meilleures performances en termes de stabilité et de précision.

De plus, l'implémentation d'une loi de commande référencée vision sur un robot mobile requiert la détermination du lien entre l'évolution des indices visuels dans l'image et les mouvements de la caméra.

Dans ce but, nous décrivons dans le paragraphe ci-dessous le formalisme de la fonction de tache, ainsi que les méthodes de calcul de la matrice d'interaction introduite dans la boucle d'asservissement visuel.

2.5.1 Formalisme de fonction de tache :

Ce formalisme permet de modéliser la tâche considérée sous la forme d'une fonction de tâche, puis à synthétiser un asservissement permettant de la réguler à zéro. Cette technique offre un cadre rigoureux pour la synthèse de lois de commande référencées capteur [Folio 07].

Initialement conçue pour des bras manipulateurs [Chaumette 90] [ESPIAU et al], ce formalisme ne peut être directement étendu au cas des robots mobiles non-holonomes du fait de la contrainte de roulement sans glissement. En effet, cette contrainte entrave les mouvements de la caméra, et les tâches robotiques nécessitent alors la réalisation de manoeuvres qui peuvent conduire à la perte du motif visuel. Une première solution a consisté à introduire (au moins) un degré de liberté

supplémentaire permettant à la caméra de se déplacer indépendamment du robot mobile. Ainsi, bien que la base mobile reste non-holonome, les mouvements de la caméra deviennent holonomes, et il est alors possible d'appliquer le formalisme des fonctions de tâches [Folio 07].

Ainsi, dans le cadre de ce formalisme, une tâche robotique peut être exprimée comme la régulation sur un certain horizon temporel $t \in \{0, T\}$ d'une fonction e(q,t) de classe C2, appelée fonction de tâche (où q est le vecteur de configuration du robot) [Ouadah 11].

Les fonctions de tâche peuvent s'exprimer en termes de données proprioceptives ou extéroceptives selon l'application considérée et les capteurs disponibles [Folio 07].Nous présentons ci-après quelques exemples de fonctions classiques :

- e(q,t) = q(t) - q*(t) où q*(t) est une trajectoire désirée dans l'espace des configurations.
- e(q,t) = r(q) - r*(t) où r*(t) est une trajectoire désirée dans l'espace cartésien.
- e(q,t) = s(q,t) - s*(t) où s*(t) est une trajectoire désirée dans l'espace du capteur.

Cependant, [Samson et al] exigent dans leur travaux des conditions pour que le problème de régulation de e(q, t) soit bien posé :

- La fonction de tâche doit posséder la propriété de l'unicité de la trajectoire q_r(t) solution de l'équation : $e(q,t) = 0, \forall t \in \{0, T\}$. Or, la fonction de tâche étant non linéaire en q, la résolution de cette équation peut conduire à la définition de plusieurs trajectoires solutions distinctes. Pour éviter cela, une condition initiale q_0 est introduite, telle que : $q_r(0) = q_0$ vérifie (q(0) , 0) = 0. Notons qu'une trajectoire vérifiant cette propriété est dite trajectoire idéale. Cependant, il se peut qu'aucune solution n'existe, excepté la condition initiale, ou au contraire, qu'il y en ait une infinité de solutions [Folio 07]. Dans ce dernier cas on parle alors de tâche redondante.
- D'une part, il est nécessaire que la régulation de la fonction de tâche e(q, t) à zéro impose la convergence de la trajectoire du robot vers la trajectoire solution désirée, et d'autre part qu'une petite variation de e(q, t) n'induise pas une grande variation de la configuration q du robot.

Lorsque toutes les conditions requises sont satisfaites, la fonction de tâche est dite *admissible*, ce qui permet alors la synthèse de lois de commande efficaces. La notion de *ρ-admissibilité* offre alors un cadre rigoureux, réunissant les conditions suffisantes que doivent vérifier les fonctions de tâche pour que le problème de commande soit bien posé [Samson et al.].

Nous rappelons brièvement ci-après cette notion.

Notion de ρ-admissibilité : La *ρ-admissibilité* permet de prouver l'existence d'une application bijective de classe C 1 dont la réciproque est aussi de classe C 1, c'est-à-dire l'existence d'un difféomorphisme entre l'ensemble des couples (q, t) et (e, t). Ce difféomorphisme permet alors d'établir un lien régulier entre l'espace d'état et l'espace capteur dans lequel s'exprime la loi de commande. Ce lien est important dans la définition d'une fonction de tâche *ρ-admissible*, comme le souligne le théorème énoncé ci-dessous [samson et al] :

Théorème : *ρ-admissibilité*

Soit e(q, t), q_0 une fonction de tâche admettant une trajectoire idéale $q_r(t)$ définie sur l'intervalle de temps $\{0, T\}$. Alors :

- *Si e(q, t) est de classe C^2*
- *Si la jacobienne de la tâche $J_e = \frac{\delta e}{\delta q}$ est inversible au voisinage de la trajectoire $q_r(t)$*
- *Si l'horizon de temps reste fini : $T < \infty$*

Il existe un rayon $\rho > 0$ tel que la tâche considérée soit « *ρ-admissible* ».

Ainsi, la notion de *ρ-admissibilité* permet de montrer d'une part qu'il y a équivalence entre la synthèse de la loi de commande dans l'espace d'état et la synthèse dans l'espace capteur, et d'autre part que la trajectoire de référence $q_r(t)$ est unique. Finalement, la condition essentielle permettant d'établir la *ρ-admissibilité* est de définir e(q, t) de telle sorte que sa jacobienne soit inversible au voisinage de la trajectoire solution. Il nous suffit donc pour cela de contraindre autant de degrés de liberté que d'actionneurs disponibles sur le robot. L'obtention de cette propriété se réduit alors à un problème de modélisation de la tâche à effectuer. Néanmoins, certaines applications ne nécessitent pas l'utilisation de tous les degrés de liberté disponibles, et donc toute tâche robotique ne satisfait pas la propriété de *ρ-admissibilité* [Folio 07]. La fonction de tâche est alors dans ce cas redondante.

2.5.2 Intéraction caméra \environnement :

Comme nous l'avons préalablement énoncé, dans le cadre de l'asservissement visuel 2D, le choix des informations visuelles et l'obtention de la relation caractérisant leur variation sont deux points fondamentaux de cette approche. Les informations visuelles s(q,t) caractérisent les mesures effectuées au moyen de la caméra, en fonction de la configuration q du robot, et du temps t pris en tant que paramètre indépendant. On supposera aussi que seuls les mouvements de la caméra, et éventuellement celui de la cible, sont susceptibles de faire varier les valeurs du signal sensoriel. En différentiant s(q,t) par rapport au temps, il est possible d'exprimer la variation des informations visuelles en fonction des mouvements de la caméra [Chaumette 90].

$$\dot{s}(q,t) = \frac{\delta s}{\delta q}\frac{dq}{dt} + \frac{\delta s}{\delta t} = \frac{\delta s}{\delta r}\frac{\delta r}{\delta q}\dot{q} + \frac{\delta s}{\delta t} \tag{2.1}$$

Ou l'on reconnait :

$\frac{\delta s}{\delta r}$: La *matrice d'intéraction* ou *jacobienne de l'image*, notée L(s,z), que nous détaillerons ci-aprés.

$J(q) = \frac{\delta r}{\delta q}$: Le jacobien du robot, qui ne dépend que de la géométrie du robot et de sa configuration.

$\frac{dq}{dt} = \dot{q}$: La commande en vitesse du système robotique

$\frac{\delta s}{\delta t}$: Terme qui est dû au mouvement propre de l'objet par rapport à la caméra. Si l'amer visuel reste fixe, ceci impliquera que ce terme sera nul $\frac{\delta s}{\delta t} = 0$

Ainsi par le biais du torseur cinématique de la caméra T_C, ($T_C = J(q).\dot{q}$), nous pouvons relier la variation des informations sensorielles au mouvement réalisé par la caméra, soit :

$$\dot{s}(q,t) = L(s,z).T_C$$

On retrouve ici la notion d'action réflexe ou les stimuli du capteur, $\dot{s}$, sont aussitôt retranscrits par une action, T_C.

a - La matrice d'interaction :

Le choix des informations visuelles et l'obtention de la relation les liant au mouvement de la caméra sont deux aspects fondamentaux de l'asservissement visuel 2D. Cette relation, obtenue par dérivation des informations sensorielles s par rapport à la situation r de la caméra, est définie par une matrice appelée *jacobienne de l'image* ou *matrice d'intéraction*.

Dans le cas général, il s'agit de l'interaction entre le mouvement relatif de la caméra par rapport à la scène représentée par un torseur cinématique T et la variation des mesures représentée par un vecteur des vitesses des mesures $\dot{s}$ [Gangloff 04].

La matrice d'interaction L(s,z) permet donc de relier le mouvement de la caméra aux informations visuelles. Cette matrice dépend non seulement de la nature des informations visuelles choisies, mais aussi de la situation de la caméra par rapport à l'objet observé, ou, plus particulièrement, de la profondeur z, c'est-à-dire de la distance entre la caméra et les informations sensorielles s(q,t). Différentes méthodes existent pour caractériser la matrice d'interaction. Ainsi, certains auteurs utilisent les méthodes d'apprentissage permettant une estimation en ligne pour obtenir L. Et c'est directement le produit L.J qui est estimé. Toutefois, avec cette approche il n'est pas possible de démontrer la stabilité des lois de commande associées [Chaumette 98].

Une autre méthode consiste à formuler une expression analytique, à partir du modèle géométrique de l'objet considéré. Ainsi F. Chaumette propose une méthode générale de calcul analytique de la matrice d'interaction pour différentes primitives géométriques simples, telles que des points, des ségments, des droites, des cercles, des sphères, des ellipses, etc [Chaumette 90] ou des moments de l'image.

Remarque :

Le calcul de la matrice d'interaction L(s,z) nécessite généralement une mesure ou un modèle de la profondeur z. Toutefois, cette information n'étant pas toujours disponible, il est nécessaire, dans ce cas, d'utiliser un modèle ou une approximation de la matrice d'interaction, notée $\hat{L}$.

Ainsi la matrice d'interaction associée joue un rôle essentiel dans l'élaboration des diverses lois de commande possibles [Chaumette 98]. Notons que pour qu'il y ait unicité de la position de la caméra par rapport à la cible pour une image donnée, il est nécessaire d'utiliser au moins 4 points [Gangloff 04].

a.1 - Matrice d'interaction pour une primitive du type point : Dans le cas d'un point p de coordonnes $(x,y,z)^T$, exprimé dans le repère caméra R_c, est projeté sur le plan image en un point P, de coordonnées métriques $(X,Y)^T$, la matrice d'interaction $L(P,z)$ se déduit des équations

du flot optique, et s'écrit comme suit [Chaumette 90] :

$$L(P,z) = \begin{pmatrix} L(X,z) \\ L(Y,z) \end{pmatrix} = \begin{pmatrix} -f/z & 0 & X/z & XY/f & -(f+X^2/f) & Y \\ 0 & -f/z & Y/z & (f+Y^2/f) & -XY/f & -X \end{pmatrix} \tag{2.2}$$

ou f désigne la focale de la camera. Cette expression de la matrice d'interaction suppose que le torseur cinématique est de dimension (6 x 1). Mais on peut ne considerer que les degrés de liberté réellement commandables selon le système considéré.

Nous pouvons généraliser ce résultat dans le cas d'un amer visuel constitué de $k > 1$ points. Le vecteur des informations sensorielles s(q,t) est alors défini par :

$s = (X_1, Y_1, ..., X_i, Y_j, \ldots, X_k, Y_k)$, ou les couples (X_i, Y_j) représentent les coordonnées métriques du j^{eme} point P_j de l'amer projeté dans le plan image de la caméra. La matrice d'interaction résultante est alors données par :

$$L_s = (L_{P_1} \ldots L_{P_k})^T = (L_{X_1}^T L_{Y_1}^T \ldots L_{X_k}^T L_{Y_k}^T)^T \tag{2.3}$$

Ou chaque ligne L_{P_j} correspond à la matrice d'interaction d'un point P_j .

Remarque : différents types de primitives peuvent être construits sur la base de point : centre de gravité, segments, polygones, cercles, etc. [Chaumette 90].

a.2 - Matrice d'interaction pour une primitive du type droite Nous allons maintenant exploiter les résultats précédents pour le calcul de la matrice d'interaction pour une primitive de type droite.

Dans un univers 3D, une droite est représentée par l'intersection de 2 plans non parallèles, cette représentation est donnée dans le référentiel lié à la caméra sous la forme [Chaumette 90] [Debain 96] :

$$h(X,Y,Z,Q) = \begin{cases} a_1X + b_1Y + c_1Z + d_1 = 0 \\ a_2X + b_2Y + c_2Z + d_2 = 0 \end{cases} \tag{2.4}$$

Projetons ces deux équations dans le plan image grâce aux formules de projections pour une distance focale $f = 1$:

$$\begin{cases} x = \dfrac{X}{Z} \\ y = \dfrac{Y}{Z} \end{cases} \tag{2.5}$$

Ainsi l'équation (2.4) s'écrit :

$$\begin{cases} 1/Z = -(a_1X + b_1Y + c_1)/d_1 \\ 1/Z = -(a_2X + b_2Y + c_2)/d_2 \end{cases} \tag{2.6}$$

On peut donc par égalité écrire :

$$\frac{-(a_2X + b_2Y + c_2)}{d_2} + \frac{(a_1X + b_1Y + c_1)}{d_1} = 0 \tag{2.7}$$

D'ou

$$(a_1d_2 + a_2d_1)X + (b_1d_2 + b_2d_1)Y + (c_1d_2 + c_2d_1) = 0 \tag{2.8}$$

L'équation cherchée est de la forme : $\mathbf{Ax + By + C = 0}$ Avec

$$\begin{cases} A = a_1d_2 + a_2d_1 \\ B = b_1d_2 + b_2d_1 \\ C = c_1d_2 + c_2d_1 \end{cases} \tag{2.9}$$

Cette représentation n'est pas minimale ; c'est pourquoi nous choisirons une autre représentation n'utilisant que 4 paramètres.

- Représentation polaire

L'équation de la droite est donnée en utilisant cette représentation par

$$g(X, q) = x\cos\theta + y\sin\theta - \rho = 0 \tag{2.10}$$

avec $q = (\rho, \theta)$ et $X = (x, y)$.

Si nous considérons la représentation de la droite : $\mathbf{Ax+By+C=0}$

On obtient :

$$\begin{cases} A = \cos\theta \\ B = \sin\theta \\ \theta = \arctan(B/A) \\ \rho = -C/\sqrt{A^2 + B^2} \end{cases} \tag{2.11}$$

En dérivant $g(X, q) = 0$:

$$\frac{\partial g(X,q)}{\partial q}\dot{q} = \frac{\partial g(X,q)}{\partial x}\dot{x} \tag{2.12}$$

$$\dot{\rho} + \dot{\theta}(x\sin\theta + y\cos\theta) = \dot{x}\cos\theta + \dot{y}\sin\theta \tag{2.13}$$

En utilisant (2.10) remplaçons x en fonction de y :

$$x = \frac{\rho - y\sin\theta}{\cos\theta} \tag{2.14}$$

Cette équation devient :

$$\dot{\rho} + \dot{\theta}(\frac{\rho - ysin\theta}{\cos\theta}sin\theta - ycos\theta) = \dot{x}cos\theta + \dot{y}sin\theta \tag{2.15}$$

D'où on obtient :

$$-\frac{\dot{\theta^2}}{\cos\theta}.y + \dot{\rho} + \rho\tan\theta\dot{\theta} = \dot{x}\cos\theta + \dot{y}\sin\theta \tag{2.16}$$

A partir de l'équation (2.10), écrivons x en fonction de y si $cos\theta \neq 0$ (ou y en fonction de x sinon). L'équation (2.16) peut alors s'écrire, en utilisant l'équation du flot optique :

$$-\frac{\dot{\theta^2}}{\cos\theta}.y + \dot{\rho} + \rho\tan\theta\dot{\theta} = Y.K_1\tau + K_2\tau \tag{2.17}$$

On en déduit aussitôt :

$$\begin{cases} \dot{\theta} = -K_1\cos\theta.\tau \\ \dot{\rho} = K_2 + K_1\sin\theta.\tau \end{cases} \tag{2.18}$$

D'ou en posant :

$$s = (\theta, \rho)^T \tag{2.19}$$

$$\dot{s} = \begin{pmatrix} \lambda_\theta\cos\theta & \lambda_\theta\sin\theta & -\lambda_\theta\rho & -\rho\cos\theta & -\rho\sin\theta & -1 \\ \lambda_\rho\cos\theta & \lambda_\rho\sin\theta & \lambda_\rho\rho & (1+\rho^2)\sin\theta & -(1+\rho^2)\cos\theta & 0 \end{pmatrix} \tag{2.20}$$

Avec :

$$\begin{cases} \lambda_\theta = \dfrac{1}{d_i}.(a_i\sin\theta - b_i\cos\theta) \\ \lambda_\rho = \dfrac{1}{d_i}.(a_i.\rho\sin\theta + b_i.\rho\cos\theta + c_i) \end{cases} \tag{2.21}$$

Nous venons ainsi de déterminer un lien entre les variations des coordonnées (θ, ρ) d'une droite de l'image et les mouvements de la caméra.

2.5.3 Méthodes de calcul des lois de commande pour l'asservissement visuel 2D :

Différentes techniques de l'automatique classique permettent de synthétiser des lois de commande exploitant des données visuelles. Les commandes ainsi synthétisées garantissent a priori certaines propriétés telles que la convergence, la non-saturation des actionneurs, la visibilité de l'objet, la satisfaction de contraintes 3D pendant le déplacement, etc. Une autre approche consiste à exploiter le formalisme des fonctions de tâches proposé par SAMSON C. dans [SAMSON et al., 1991] [Folio 07].

On peut ainsi intégrer parfaitement le domaine de la commande sous l'approche fonction de tâche qui a été évoquée précédemment. En effet, les tâches qui peuvent être réalisées par asservissement visuel peuvent s'exprimer comme la régulation de la fonction de tâche suivante [Chaumette 98] :

$$e(q,t) = C(s(q,t) - s^*) \tag{2.22}$$

Ou C est une matrice, dite de combinaison, qui permet notamment de prendre en compte dans la commande un nombre d'informations visuelles supérieur au nombre de degrés de liberté contraints par e.

La matrice C peut être choisit comme étant la pseudo inverse à gauche de la matrice d'interaction dont la méthode de calcul sera détaillé ci-dessous.

a - Calcul de la matrice d'interaction :

La synthèse de la commande repose sur l'élaboration d'une méthode de calcul explicite de cette matrice souvent associée à des primitives géométrique simples, telles que des points, des droites, des cercles, des ellipses [Chaumette 90], ou encore des moments de l'image. Dans ce contexte, les lois de commande synthétisées dépendent de la nature des informations visuelles choisies. Par conséquent la tâche robotique à réaliser est elle aussi dépendante de l'objet observé par la présence ou non d'informations visuelles dont on est capable de calculer la matrice d'interaction associée [Folio 07].

La question est donc de savoir comment mettre à jour cette matrice dans la boucle d'asservissement. Deux cas de figure sont donc possibles, selon que L(s,z) est calculée hors ligne ou en ligne [Ouadah 11] :

- L(s,z) calculée hors ligne : Dans ce cas, elle n'est calculée qu'une seule fois avant l'asservissement, en prenant les indices visuels au point d'équilibre souhaité s^* comme suit : $(L(s,z) = L(s^*,z^*))$. Cette approche allège les calculs dans la boucle d'asservissement tout en préservant sa convergence. Toutefois, des difficultés sont enregistrées quand la position initiale est loin de celle désirée.
- - L(s,z) calculée en ligne : Nous considérons dans ce cas qu'elle est calculée à chaque itération en fonction des primitives visuelles courantes s comme suit : $(L(s,z) = L(s(q,t),\hat{z}))$. Cette matrice dépend alors d'une information métrique (la profondeur z). Une estimation de z (notée $\hat{z}$) peut se faire si on dispose d'un modèle 3D de la cible [Chaumette 98].

Ces deux possibilités présentent chacune un certain nombre d'avantages, mais aussi des inconvénients [Chaumette 98] :

- D'une part, le calcul de L(s,z) en ligne contraint fortement la trajectoire que doivent suivre les informations visuelles dans l'image pour atteindre la position désirée. Par exemple, si l'on choisit des points, ce choix implique que chaque point doit aller en ligne droite vers sa position désirée (Fig : 2.11 a). Cependant, ce comportement souhaitable dans l'image peut parfois entraîner des mouvements inadéquats de la caméra (singularité), voire même impossible à réaliser. La caméra peut ainsi atteindre un minimum local de l'erreur dans l'image à partir duquel il est impossible de rejoindre le minimum global correspondant à une erreur nulle.
- D'autre part, calculer L(s,z) hors ligne ne contraint quasiment pas la trajectoire dans l'image, comme l'illustre la figure (Fig : 2.11 b). Du coup, et même si nous ne pouvons pas l'expliquer, il s'avère en pratique que la commande est alors beaucoup moins sujette aux problèmes des mouvements irréalisables et des minima locaux. Par contre, il est tout à fait possible que la cible sorte du champ de vue de la caméra au cours du positionnement.

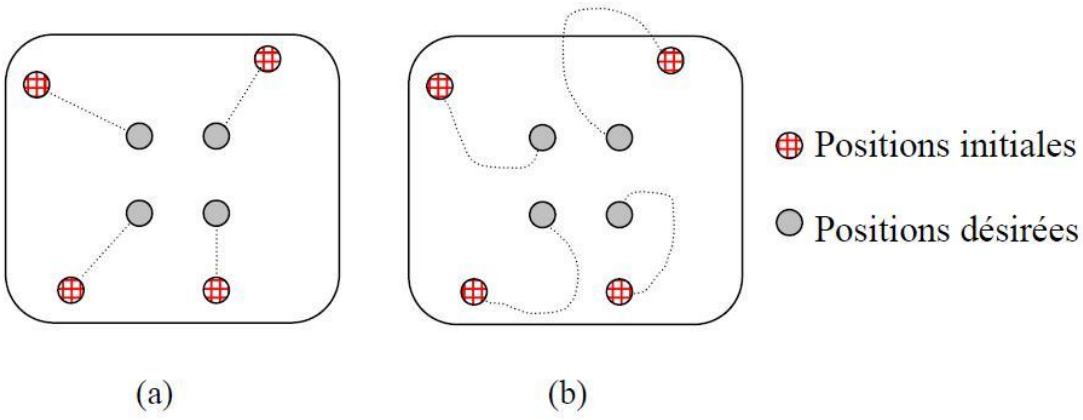

FIGURE 2.11 – Trajectoires des primitives visuelles dans l'image

b - Synthèse de la loi de commande :

Comme il a été précisé précédemment, le principe de l'asservissement visuel 2D consiste à contrôler le mouvement de la caméra de manière à annuler l'erreur entre les informations visuelles courantes s(q, t) et la configuration désirée s^*. De ce fait, la synthèse du contrôleur est basée sur le formalisme des fonctions de tâche. Classiquement, la convergence de la fonction de tâche vers zéro est effectuée en lui imposant de décroitre de manière exponentielle [Folio 07].

Dans la mesure où le système robotique est commandable en vitesse, par l'intermédiaire de $\dot{q}$, on considée uniquement des schémas de commande cinématiques ainsi le comportement désiré du système en boucle fermée peut être décrit par :

$$\dot{e}(q) = J_e \dot{q} = -\lambda e(q) \tag{2.23}$$

où λ (le gain) est un scalaire positif ou une matrice définie positive, permettant de fixer la vitesse de décroissance exponentielle de e(q). De plus, J_e qui est égale à $\frac{\delta e}{\delta q}$ est une matrice représentant la jacobienne de la fonction de tâche. On rappelle que si e(q) est *ρ-admissible* (i.e. J_e est inversible), cela suffit pour s'assurer que le problème de commande est bien posé, et assure la détermination de la commande $\dot{q}$ de manière unique. Toutefois, lorsque la fonction de tâche e(q) est définie par l'expression (2.1), le calcul de Je s'avère en général difficile, et la synthèse de la commande requiert une estimation de cette matrice, notée $\hat{J}_e$ [Chaumette 90] [Folio 07]. La loi de commande (3.24) s'écrit alors sous la forme :

$$\dot{q} = -\hat{J}_e^{-1} \lambda e(q) \tag{2.24}$$

La dynamique de la fonction de tâche e(q) devient alors en réalité dépendante du produit de Matrices $(\hat{J}_e \hat{J}_e^{-1})$. En effet, la stabilité de la loi de commande nécessite que ce produit soit égal à une matrice définie positive. Or, si les conditions mentionnées de *ρ-admissibilité* sont satisfaites, la jacobienne J_e est définie positive autour de la trajectoire solution q_r(t). Dans ce cas, une solution satisfaisant la condition $(\hat{J}_e \hat{J}_e^{-1}) > 0$ consiste à choisir $\hat{J}_e$ égale à la matrice identité I, et la loi de commande ci-dessus se simplifie alors comme suit [Espiau] :

$$\dot{q} = -\lambda e(q) \tag{2.25}$$

c - Synthèse d'un controleur classique :

La synthèse d'un contrôleur permettant à un robot de réaliser une navigation référencée vision, en se basant sur le formalisme des fonctions de tâches passe nécessairement par la spécification de l'objectif à réaliser sous la forme d'une régulation à zéro d'une fonction particulière e(q), dite *fonction de tâche* . Dans cette étude, nous avons opté pour un choix classique de la fonction de tâche pour un asservissement visuel 2D permettant d'amener les indices visuels courants s vers la configuration désirée s^*, comme suit :

$$e(q) = C(s(q)–s^*) \tag{2.26}$$

Où C représente la matrice de combinaison déjà définie. Toutefois, l'introduction de cette matrice peut perturber la convergence des indices visuels, en introduisant des minimas locaux. En effet, la régulation de e(q) vers zéro n'assure que la convergence de Cs vers Cs^*, ce qui ne garantit pas forcement la convergence de s vers s^* [Folio 07].

Comme il a été cité plus haut, la convergence de la fonction de tâche vers zéro est réalisée en lui imposant de décroître de manière exponentielle. Le comportement désiré en boucle fermée peut donc être décrit par :

$$\dot{e}(q) = -\lambda C(s(q)–s^*) = -\lambda \hat{L}^+(s - s^*) \tag{2.27}$$

Ou $\hat{L}^+$ est la pseudo-inverse d'un modèle ou d'une approximation de L. Pour assurer la stabilité et la convergence de e, on dispose de la condition suffisante suivante :

$\hat{L}^+ L > 0$

C'est-à-dire que $\hat{L}^+$ doit être suffisamment correct et proche de L pour ne pas trop perturber le système (puisque $\hat{L}^+ = L$, le produit de matrice est égal à l'identité).

Concernant l'influence de la calibration, les expérimentations présentées dans [Episau 93] indiquent une forte robustesse des lois de commande par rapport à des variations des paramètres intrinsèques. Par contre, et excepté pour des cas très simples (comme le contrôle de l'orientation de la caméra pour qu'un objet soit situé au centre de l'image), la complexité des calculs fait qu'il est quasiment impossible de démontrer la positivité du produit de matrices ci-dessus dans les cas les plus généraux [Chaumette 98]

2.5.4 Travaux récents dans la synthèse des lois de commande

Les premiers travaux en asservissement visuel datent de la fin des années 1980. Depuis lors, ils n'ont cessé de prendre de l'essor, via des contributions méthodologiques propres ou via l'élargissement de leur domaine d'application en se confrontant aux autres domaines de la robotique [Andreff 07].

Dans ce qui suit, nous allons exposer les avancées récentes qui concernent la synthèse des lois de commande en asservissement visuel, et l'analyse de leur stabilité.

Comme nous l'avons cité précédemment, les commandes en asservissements visuel basées sur le formalisme de fonctions de tâche dépendent directement de la manière dont est calculée la matrice $\hat{L}^+$. Nous avons ainsi cité deux méthodes, selon que la matrice soit calculée en ligne ou bien hors ligne.

Des études plus récentes ont testé d'autres possibilités de calcul de la matrice d'interaction, que celles citées plus haut. Il a été constaté qu'une combinaison linéaire des deux formes précédentes donnait de meilleurs résultats. Il en résulte une matrice d'interaction de la forme.

$$L(s,z) = \frac{L(s^*,z^*) + L(s(q,t),\hat{z})}{2} \tag{2.28}$$

D'une manière encore plus générale, une forme originale a été proposée dans [Marey 08], en introduisant un paramètre de comportement β, celle-ci est donnée par :

$$\hat{L_s} = L_\beta = \beta L_{s^*} + (1-\beta)L_{s(t)} \tag{2.29}$$

Ce nouveau schéma de commande nous permet donc d'adapter le comportement du système à partir de la valeur choisie de β. Ainsi, nous pouvons retrouver les trois cas de figure précédents en faisant varier β entre 0 et 1.

Analyse de la stabilité

Pour les schémas de commande précédents, seule la stabilité asymptotique locale peut être démontrée et cela pour des informations visuelles redondantes telles que les coordonnées des points dans l'image. Nous présentons à présent une loi de commande construite pour tenter d'obtenir la stabilité asymptotique globale développée dans [Marey 08] :

Soit la fonction de tâche :

$$e = L_{s^*}^+(s - s^*) \tag{2.30}$$

La dérivée temporelle de e est donnée par :

$$\dot{e} = L_e T_c = L_{s^*}^+ L_s T_c \tag{2.31}$$

Pour obtenir une décroissance exponentielle, la loi de commande sera de la forme :

$$T_c = -\lambda L_e^{-1} e = -\lambda (L_{s^*}^+ L_s)^{-1} L_{s^*}^+ (s - s^*) \tag{2.32}$$

Pour étudier la stabilité de cette loi de commande, on considère une fonction de Lyapunov condidate :

$$V = \frac{1}{2}\|e(t)\|^2 \tag{2.33}$$

En dérivant on a :

$$\dot{V} = e^T \dot{e} = e^T L_e T_c \tag{2.34}$$

En utilisant l'équation (2.32), on obtient :

$$\dot{V} = -\lambda e^T L_e {L_e}^{-1} e = -\lambda e^T e, \forall e \neq 0 \tag{2.35}$$

La loi de commande (2.32) est donc globalement asymptotiquement stable (GAS) dans l'espace de la fonction de tâche e. De plus, elle assure une convergence exponentielle découplée de chaque composante de e.

La figure ci dessous présente les résultats de simulation obtenus pour les différentes méthodes de calcul des lois de commande [Marey 08] :

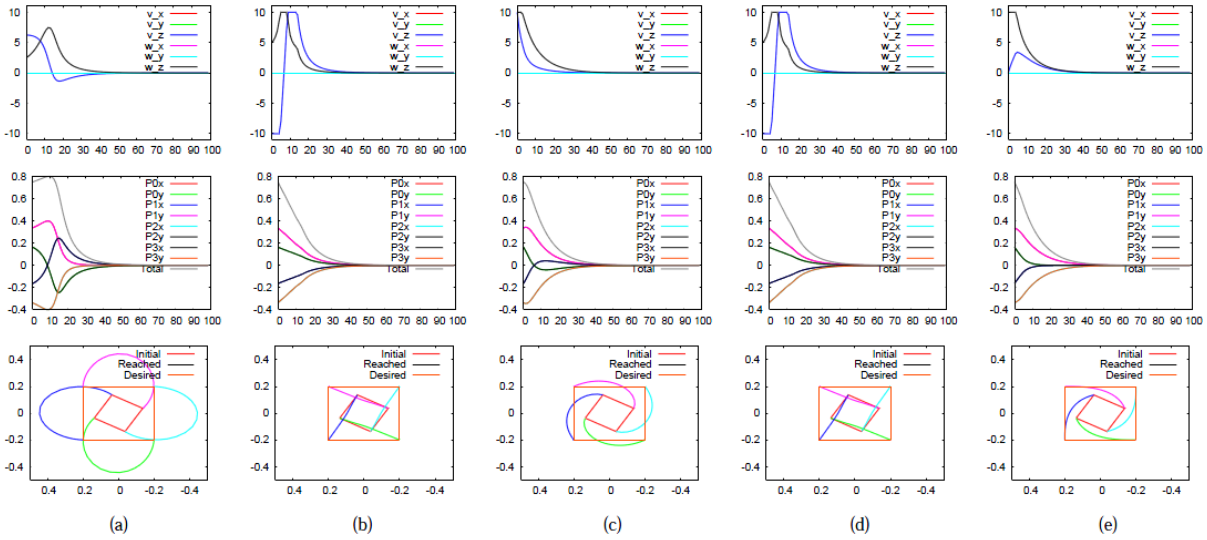

FIGURE 2.12 – Résultats de simulation

Tel que :

(a) : Résultats pour $(L(s,z) = L(s^*,z^*))$

(b) : Résultats pour $(L(s,z) = L(s(q,t),\hat{z}))$

(c) : Résultats utilisant l'equation (2.28)

(d) : Résultats utilisant la loi Pseudo GAS

(e) : Résultats utilisant l'équation (2.29) pour $\beta = 0.285$.

2.6 Conclusion

Dans ce chapitre, nous avons exposé un état de l'art sur l'asservissement visuel. Les généralités sur la vision en robotique mobile nous ont permis d'aborder les principes généraux de l'utilisation de cette faculté dans ce domaine. Après quoi, les différentes techniques utilisées en asservissement visuel ont été abordées. Ceci nous a permis de nous focaliser sur celle qui présente le plus d'intérêt pour notre application. Dans cette optique, nous avons donc détaillé l'asservissement visuel 2D appliqué en robotique mobile. Ainsi, nous avons exposé les dernières avancées permettant la synthèse de lois de commande dans l'espace capteur.

Chapitre 3

Synthèse de la commade référencée vision pour le robot « Robucar »

3.1 Introduction

Nous allons, dans ce chapitre, nous intéresser à la synthèse de lois de commande du robot mobile « *Robucar* ». Ceci dans le but de réaliser un suivi de trajectoire, définit par des droites ou approchée par morceaux à des segments de droite.

Dans un premier temps, nous allons exploiter directement les données visuelles issues de la caméra pour commander l'angle de braquage du robot mobile « *Robucar* », et cela pour effectuer le suivi de la trajectoire.

Ensuite, nous allons effectué une étude théorique de notre système, et synthétiser une loi d'asservissement visuel basée sur la notion de formalisme de fonction de tâche que nous avons détaillé dans le chapitre précédent. Nous allons donc commencer par une modélisation de notre système robotique, ainsi que de l'interaction de ce dernier avec l'environnement.

Nous allons donc commencer par la synthèse d'un régulateur classique de type PID, en détaillant la méthode de calcul de chaque action de ce type de régulateur. Pour ensuite étudier les différentes manières qui nous permettrons le passage de la forme analogique à la forme numérique de ce régulateur tout en évitant les problèmes liés à cette transition.

Après quoi, un régulateur de type flou sera synthétisé. La détermination de ses fonctions d'appartenances, ainsi que de la base des règles sera basée sur les essais déjà effectués sur le robot.

A la fin de ce chapitre, nous aurons synthétisé nos lois de commandes qui seront d'abord validées en simulation avant d'être implémentées en temps réels sur le robot mobile « *Robucar* »

3.2 Exploitation directe des informations visuelles pour la commande

3.2.1 Le choix des informations visuelles

L'utilisation d'un capteur de vision pour commander le « *Robucar* » requiert un choix sur les primitives visuelles à utiliser (points, droite, ellipses, cercles, etc.). En effet, ce choix dépend de la tâche à réaliser (suivi de trajectoire, poursuite de cibles,...), et devra correspondre à une seule configuration du robot dans l'espace 3D, par exemple un choix de primitives visuelles de type points, nous contraint à choisir un nombre de points supérieur à trois pour satisfaire cette condition.

Pour notre application qui vise à asservir le robot pour le suivi d'une trajectoire assimilée par morceaux à des segments de droites, le choix évident sera une primitive de type droite, l'unicité de la configuration du robot correspondant à une position de la droite dans l'image est assurée par une liaison de type prismatique entre le robot et le sol (cette liaison interdit des mouvements de roulis et de tangage du robot). La droite dans l'image sera caractérisée par son équation en coordonnée polaires dans un repère lié au centre de l'image , tel que ρ représente la distance entre le centre de l'image et sa projection sur la droite, θ représente l'angle entre la droite et l'axe verticale du repère, voir figure (Fig : 3.1).

Un algorithme de traitement d'image qui sera détaillé dans le chapitre suivant nous permet d'extraire ces informations visuelles qui seront utilisées pour la commande de l'angle de braquage

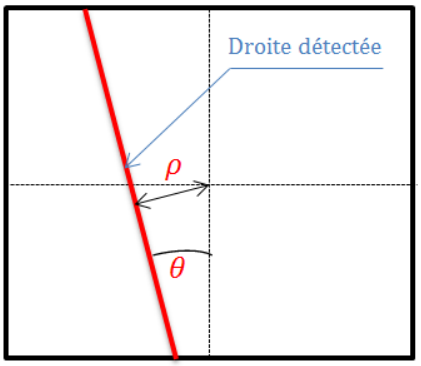

FIGURE 3.1 – Représentation de la droite dans l'image.

du robot, le choix de la convention pour les signes de ρ et θ selon la position de la droite détéctée dans l'image est illustré sur la figure suivante :

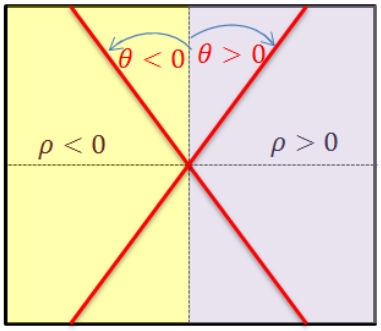

FIGURE 3.2 – Signes de ρ et θ selon la position de la droite dans l'image

3.2.2 Droite de référence et droite détectée

La droite de référence sera prise comme étant la première droite détectée par le robot dans sa position initiale, ainsi l'asservissement du robot correspondra à une régulation à zero de l'erreur entre les informations visuelles liées à la droite détectée pour une position du robot à un instant donné notées $s = (\theta, \rho)$, et celles de la droite de référence notées $s = (\theta^*, \rho^*)$, voir Figure (Fig : 3.3). Le but sera donc de garder une même configuration du robot par rapport à la trajectoire parcourue. L'erreur de régulation est donnée par :

$$e = S - S^* = \begin{pmatrix} \theta - \theta^* \\ \rho - \rho^* \end{pmatrix} \tag{3.1}$$

3.2.3 Commande directe de l'angle de braquage du Robot mobile « *Robucar* »

Dans cette approche, nous avons exploité directement l'erreur précédemment définie pour la commande de l'angle de braquage du robot, ainsi le schéma de régulation sera comme suit :

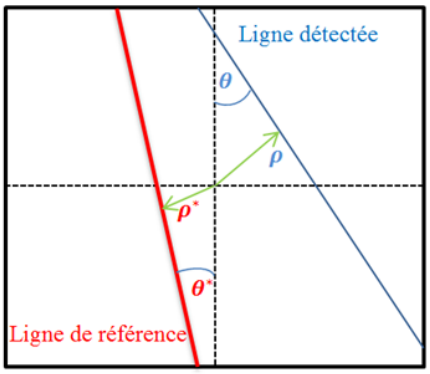

FIGURE 3.3 – Droite de référence et droite détectée

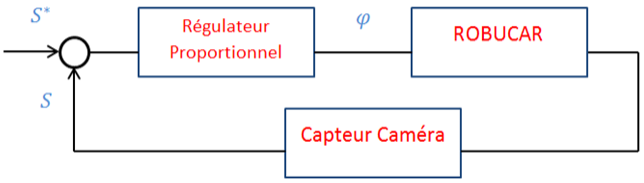

FIGURE 3.4 – Schéma du régulateur exploitant directement les informations visuelles

Le premier problème rencontré est dû au fait que deux erreurs sont régulées par une seule commande (l'angle de braquage), de plus l'évolution de ces deux erreurs est liée, ainsi une régulation du rayon implique une erreur sur l'angle de la droite du fait que la droite ne peut se déplacer à $\theta = 0$ dans l'image, et une régulation de θ engendre une erreur non nulle sur le rayon, ce phénomène est d'autant plus accentué pour des positions du robot éloignées par rapport à sa position d'équilibre, ce problème implique notamment des oscillations de la commande.

Les figures (Fig : 3.5 , 3.6) suivantes illustrent ce phénomène, elles représentent une régulation par rapport à une droite de référence de coordonnées $(\theta^*, \rho^*) = (0, 0)$, pour une position initiale décalée du robot par rapport à la droite à suivre.

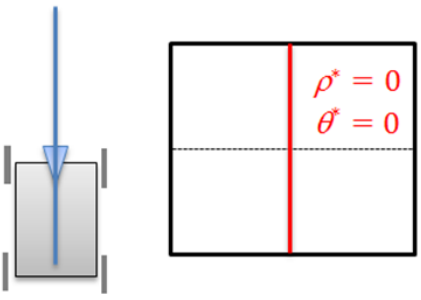

FIGURE 3.5 – Position désirée

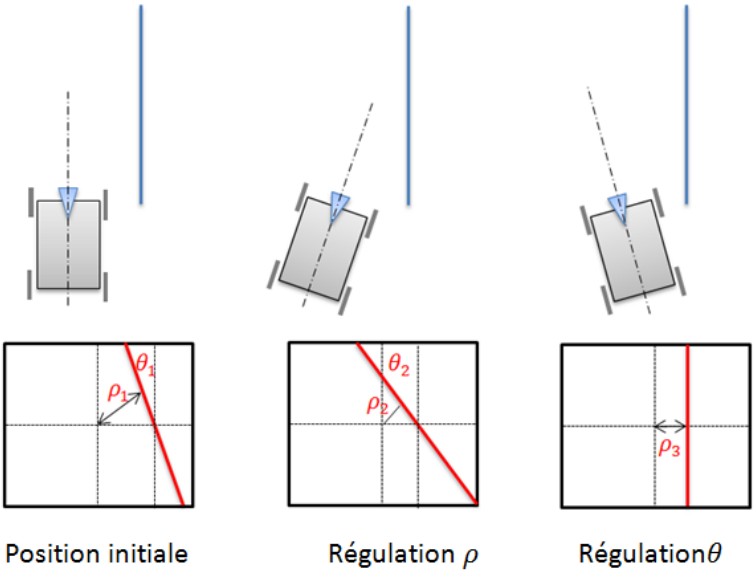

FIGURE 3.6 – Fonctionnement du régulateur

Pour remédier à ce problème, une méthode consiste à définir des priorités sur les erreurs à réguler, ainsi pour une position éloignée du robot une régulation sur le rayon sera prioritaire, à l'inverse lorsque le robot est proche de sa position de référence la priorité sera donnée à une régulation de l'angle.

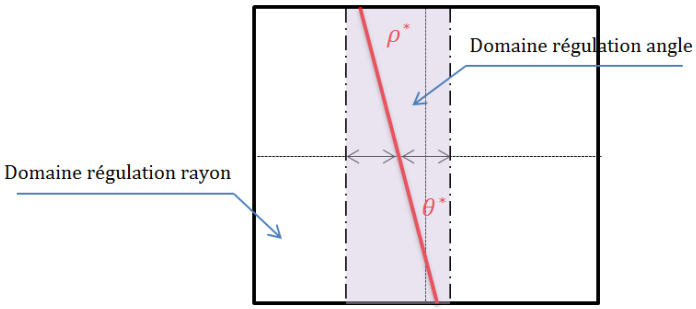

FIGURE 3.7 – Intervalles de fonctionnement

Le régulateur est de type proportionnel, sa forme sera donnée comme suit :

$$\varphi = -K_1((\rho - \rho^*) + K_2(\theta - \theta^*)). \tag{3.2}$$

Le choix de K_1, K_2 est fait de telle sorte à respecter les priorités de régulation du rayon et de l'angle.

3.2.4 Analyse critique du résultat

Plusieurs tests effectués sur le robot en utilisant ce régulateur (résultats exposés au chapitre quatre) nous ont permis de s'apercevoir que malgré le suivi de la trajectoire, cette méthode engendre plusieurs problèmes :

- Nous remarquons que ce régulateur n'assure pas un bon découplage entre θ, ρ .
- Pour certaines valeurs de θ et ρ de signes différents, on peut avoir une commande nulle (exemple de $(\theta - \theta^*) = -(\rho - \rho^*)/K_2$.
- Le régulateur engendre des variations brusques de l'angle de braquage dû au passage entre les intervalles de régulation de θ et ρ.

Le problème majeur est dû au fait qu'aucune modélisation de l'interaction entre l'évolution des indices visuels dans l'image et les mouvements du robot par rapport à sa trajectoire n'a été faite,

Les problèmes cités ci-dessus montrent la nécessité de passer par une modélisation complète du système, cette dernière nous permettra de synthétiser une loi de commande basée sur l'asservissement visuel 2D.

3.3 Modélisation du système robotique

3.3.1 Modélisation du *Robucar*

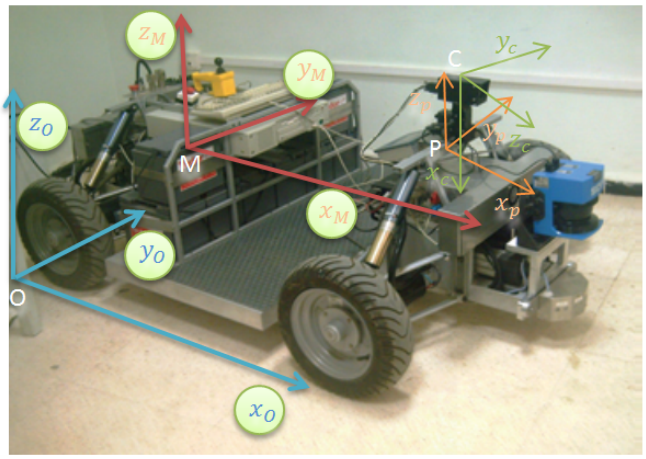

FIGURE 3.8 – Robucar

Notre objectif est d'établir la modélisation de notre système robotique représenté sur la figure (Fig : 3.8). Nous considérons le robot mobile « *Robucar* »muni d'une caméra montée sur une platine P commandable en lacet, le point P désigne le centre de la rotation de la platine, le choix des repères pour notre étude est défini comme suit :

- $R_0(O, \overrightarrow{X_0}, \overrightarrow{Y_0}, \overrightarrow{Z_0})$ le repère lié à la scène ;
- $R_M(M, \overrightarrow{X_M}, \overrightarrow{Y_M}, \overrightarrow{Z_M})$ le repère lié à la base mobile ;
- $R_P(P, \overrightarrow{X_P}, \overrightarrow{Y_P}, \overrightarrow{Z_P})$ le repère lié à la platine ;
- $R_C(C, \overrightarrow{X_C}, \overrightarrow{Y_C}, \overrightarrow{Z_C})$ le repère lié à la caméra ;

Aussi, nous définissons les paramètres suivants qui caractérisent le système étudié :

- D la distance entre le centre de l'essieu arrière (M) du robot et le centre de la platine P selon l'axe $\overrightarrow{X_M}$;
- a, b, c représentent les distances entre le centre de la platine P et le centre du repère caméra C respectivement selon les axes $\overrightarrow{X_P}, \overrightarrow{Y_P}, \overrightarrow{Z_P}$;
- h la distance entre le centre de l'essieu M et le centre de la platine P selon l'axe $\overrightarrow{Z_M}$

Les angles caractérisant la caméra sont les suivants :

- α l'orientation horizontale de la platine qui est donnée par l'angle formé entre $\overrightarrow{X_P}$ et $\overrightarrow{X_M}$;
- β inclinaison horizontale de la caméra qui est donnée par l'angle formé entre $\overrightarrow{X_P}$ et $\overrightarrow{Z_C}$;

les différents repères du système robotique sont illustrés sur la figure suivante :

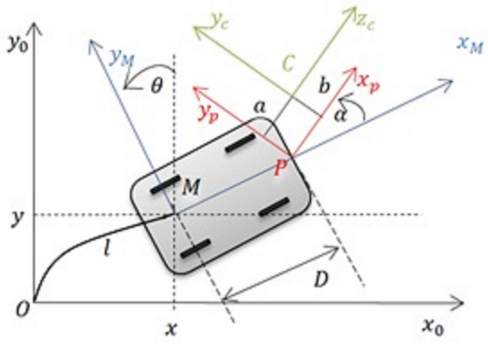

FIGURE 3.9 – Repères du système Robot-Caméra

Nous définissons le vecteur de configuration du système robotique, noté q, comme suit $q = (x, y, \theta, \alpha)^T$.

- (x, y) désignant les coordonnées du centre de l'essieu M de la base mobile dans le repère lié à la scène R_0
- θ est l'orientation de la base mobile par rapport à l'axe $\overrightarrow{X_0}$.

La dérivée de ce vecteur par rapport au temps n'est autre que le vecteur de commande, noté $\dot{q}$ et exprimé par : $\dot{q} = \frac{dq}{dt} = (v, \dot{\theta}, \dot{\alpha})^T$.

Où v et $\dot{\theta}$ désignent respectivement les vitesses linéaire et angulaire de la base mobile et $\dot{\alpha}$ la vitesse angulaire de la platine.

L'ensemble de notre système robotique étant commandable en vitesse, nous utilisons alors tout au long de ce travail le modèle cinématique décrivant son comportement. Il s'agit donc de déterminer le torseur cinématique de la caméra noté $\tau_{\mathbf{C/R_0}}$, par raport au repère de la scène R_0, en fonction des composantes du vecteur de commande $\dot{q}$ de notre robot. Pour cela, nous exploitons l'approche qui s'appuie sur des relations bien connues de la mécanique classique .

3.3.2 Le torseur cinématique

Notre objectif est d'établir l'expression du torseur cinématique de la caméra par rapport au repère de la scène donné par l'équation mécanique suivante [Folio 07] :

$$\tau_{C/R_0} = \begin{bmatrix} V_{c/R_M} + V_{M/R_0} + \Omega_{R_M/R_0} \wedge \overrightarrow{MC} \\ \Omega_{R_C/R_M} + \Omega_{R_M/R_0} \end{bmatrix} \tag{3.3}$$

Dans un premier temps, nous allons commencer par déterminer le torseur cinématique de la

caméra relativement au repère mobile, ce dernier s'exprime par la relation mécanique suivante :

$$\tau_{C/R_M} = \begin{bmatrix} V_{c/R_M} \\ \Omega_{R_c/R_M} \end{bmatrix} = \begin{bmatrix} V_{c/R_P} + V_{P/R_M} + \Omega_{R_P/R_M} \wedge \overrightarrow{PC} \\ \Omega_{R_C/R_P} + \Omega_{R_P/R_M} \end{bmatrix} \tag{3.4}$$

Or les vitesses V_{C/R_P} etV_{P/R_M} sont nulles puisque les points P et C ne se déplacent pas par rapport à ces reperes. De plus R_C restant fixe par rapport à R_P , $\Omega_{R_C/R_P} = 0$ [Cadenat 99].

L'équation (3.4) s'écrit donc comme suit :

$$\tau_{C/R_M} = \begin{bmatrix} \Omega_{R_P/R_M} \wedge \overrightarrow{PC} \\ \Omega_{R_P/R_M} \end{bmatrix} \tag{3.5}$$

Où :

$$\Omega_{R_P/R_M} = \begin{bmatrix} 0 \\ 0 \\ \dot{\alpha} \end{bmatrix} \tag{3.6}$$

Le vecteur $\overrightarrow{PC}$ est défini par les coordonnées de la position de la caméra dans le repère R_P, ainsi ces valeurs seront nulles pour une position centrée de la caméra sur la platine. pour d'autres positions de la caméra sur le *Robucar* il suffira de modifier les valeurs de ce vecteur.

$$\overrightarrow{PC}^{R_P} = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \tag{3.7}$$

En utilisant la matrice de passage entre$R_M et R_P$ définie comme suit :

$$P_{R_M,R_P} = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.8}$$

Le vecteur $\overrightarrow{PC}$ exprimé dans R_M est donné par :

$$\overrightarrow{PC}^{R_M} = P_{R_M,R_P}.\overrightarrow{PC}^{R_P} = \begin{bmatrix} acos(\alpha) - bsin(\alpha) \\ asin(\alpha) + bcos(\alpha) \\ c \end{bmatrix} \tag{3.9}$$

D'où :

$$\Omega_{R_P/R_M} \wedge \overrightarrow{PC}^{R_P} = \begin{bmatrix} -asin(\alpha) - bcos(\alpha) \\ acos(\alpha) - bsin(\alpha) \\ c \end{bmatrix} \dot{q} \tag{3.10}$$

Finalement, en remplaçant l'équation (3.10) dans (3.5) nous déduisons l'expression du torseur cinématique de la caméra relativement à R_M :

$$\tau_{C/R_M}^{R_M} = \begin{bmatrix} V_{c/R_M}^{R_M} \\ \Omega_{C/R_M}^{R_M} \end{bmatrix}_{R_M} = \begin{bmatrix} -asin(\alpha) - bcos(\alpha) \\ acos(\alpha) - bsin(\alpha) \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \dot{q} \tag{3.11}$$

Afin de définir complètement l'équation (3.3), il reste à déterminer le vecteur MC. Ce dernier est calculé d'après la relation de Chasles, $\overrightarrow{MC} = \overrightarrow{MP} + \overrightarrow{PC}$, en projetant dans le repère R_M nous obtenons :

$$\overrightarrow{MC}^{R_M} = \begin{bmatrix} D + acos(\alpha) - bsin(\alpha) \\ asin(\alpha) + bcos(\alpha) \\ h + c \end{bmatrix} \tag{3.12}$$

Où h désigne la hauteur de la platine par rapport à la plateforme mobile. Finalement, à partir des équations (3.12) et (3.11) on obtient le torseur cinématique de la caméra par rapport au repère de la scène exprimé dans le repère R_M :

$$\tau_{C/R_M}^{R_M} = \begin{bmatrix} 1 & -(asin(\alpha) + bcos(\alpha)) & -(asin(\alpha) + bcos(\alpha)) \\ 0 & D + acos(\alpha) - bsin(\alpha) & acos(\alpha) - bsin(\alpha) \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} v \\ \dot{\theta} \\ \dot{\alpha} \end{bmatrix} \tag{3.13}$$

Il reste maintenant à projeter ce vecteur dans le repère de la caméra. La matrice de passage entre R_M et R_C est définie par la relation suivante :

$$P_{R_M/R_C} = P_{R_M/R_P}.P_{R_P/R_C}$$

P_{R_M/R_P} Etant déjà calculée, on calcule la matrice de passage entre R_P et R_C

$$P_{R_M/R_P} = \begin{bmatrix} -sin(\beta) & 0 & cos(\beta) \\ 0 & cos(\beta) & 0 \\ -cos(\beta) & 0 & -sin(\beta) \end{bmatrix} \tag{3.14}$$

D'où

$$P_{R_M/R_C} = P_{R_M/R_P}.P_{R_P/R_C} = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -sin(\beta) & 0 & cos(\beta) \\ 0 & cos(\beta) & 0 \\ -cos(\beta) & 0 & -sin(\beta) \end{bmatrix} \tag{3.15}$$

D'ou

$$P_{R_M/R_C} = \begin{bmatrix} -cos(\alpha)sin(\beta) & -sin(\alpha) & cos(\alpha)cos(\beta) \\ -sin(\alpha)sin(\beta) & cos(\alpha) & sin(\alpha)cos(\beta) \\ -cos(\beta) & 0 & -sin(\beta) \end{bmatrix} \tag{3.16}$$

on a $P_{R_C/R_M} = (P_{R_M/R_C})^T$ Donc :

$$P_{R_M/R_C} = \begin{bmatrix} -cos(\alpha)sin(\beta) & -sin(\alpha)sin(\beta) & -cos(\beta) \\ -sin(\alpha) & cos(\alpha) & 0 \\ cos(\alpha)cos(\beta) & sin(\alpha)cos(\beta) & -sin(\beta) \end{bmatrix} \tag{3.17}$$

Enfin en utilisant l'équation de projection ci-après :

$$\tau^{R_C}_{C/R_0} = P_{R_C,R_M}.\tau^{R_M}_{C/R_0} \tag{3.18}$$

On obtient le torseur cinématique écrit dans le repère de la caméra :

$$\tau^{R_C}_{C/R_0} = \begin{bmatrix} V^{R_C}_{C/R_0} \\ \Omega^{R_C}_{R_C/R_0} \end{bmatrix} \begin{bmatrix} v_x c \\ v_y c \\ v_z c \\ \Omega_x c \\ \Omega_y c \\ \Omega_z c \end{bmatrix} \begin{bmatrix} -cos(\alpha)sin(\beta) & sin(\beta)(b - Dsin(\alpha) & bsin(\beta) \\ -sin(\alpha) & a + cos(\alpha) & a \\ cos(\alpha)cos(\beta) & cos(\beta)(Dsin(\alpha) - b) & -bcos(\beta) \\ 0 & -cos(\beta) & -cos(\beta) \\ 0 & 0 & 0 \\ 0 & -sin(\beta) & -sin(\beta) \end{bmatrix} \begin{bmatrix} v \\ \dot{\theta} \\ \dot{\alpha} \end{bmatrix} \tag{3.19}$$

Ce torseur s'écrit alors comme le produit d'une matrice dite ***Jacobienne du robot*** noté **J(q)**, qu'est fonction de la configuration de la caméra et de la caractéristique du système robotique, avec $\dot{q}$ le vecteur de commande du robot :

$$\tau^{R_C}_{C/R_0} = J(q).\dot{q} \tag{3.20}$$

La matrice Jacobienne précédente permet de décrire le mouvement de la caméra par rapport au repère de la scène. La ligne de zéro représente un mouvement impossible pour la caméra. Ainsi, celle-ci ne peut avoir une rotation autour de l'axe y_c.

3.4 Interaction caméra-environnement

Une fois la jacobienne du robot établie, il reste à déterminer la relation entre l'évolution des indices visuels perçus par le robot et le mouvement de la caméra, pour cela une modélisation de la caméra est nécessaire, celle-ci nous permettra par la suite la détermination de la matrice d'interaction.

3.4.1 Modélisation de la caméra

Différentes techniques de modélisation d'une caméra sont utilisées dans le cadre des travaux en vision artificielle, cependant, le modèle sténopé ou « trou d'épingle » (Pinhole camera model), présenté sur la figure (Fig : 3.10) est le plus couramment utilisé en vision par ordinateur, car il permet de modéliser finement la plupart des capteurs projectifs et de simplifier les équations mises en jeu. D'autant plus acceptable que l'on utilise des focales de faible dimension.

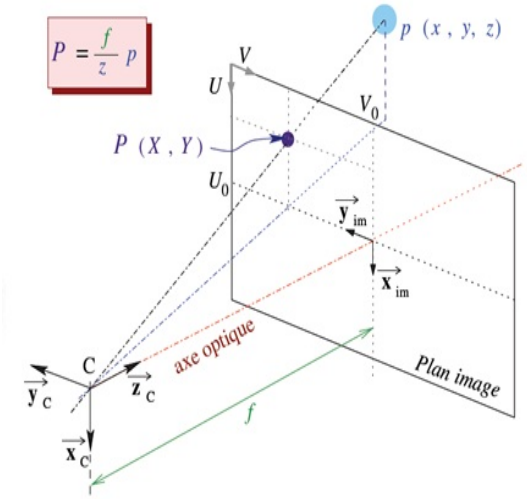

FIGURE 3.10 – Modèle sténopé de la camera.

Le modèle sténopé prend pour hypothèse que tous les rayons passent par un seul point : le centre optique C. de ce fait, les points sont projetés sur le plan image par une projection perspective.

Ainsi, un point p de la scène, de coordonnées $(x, y, z)^T$, exprimé dans le repère caméra R_c, est projeté sur le plan image en un point P, de coordonnées métriques $(x, y)^T$, selon la relation suivante [Folio 07] :

$$\begin{pmatrix} X \\ Y \end{pmatrix} = \begin{pmatrix} \frac{f}{Z} & 0 & 0 \\ 0 & \frac{f}{Z} & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{3.21}$$

Où f représente la distance focale de la caméra. En introduisant la matrice des paramètres intrinsèques, nous pouvons relier les coordonnées métriques $(X, Y)^T$ de p à leurs coordonnées en

pixel (U, V) selon la relation suivante :

$$\begin{pmatrix} U \\ V \end{pmatrix} = \begin{pmatrix} \alpha_u & 0 & U_0 \\ 0 & -\alpha_u & V_0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} \tag{3.22}$$

Où U_0 et V_0 correspondent aux coordonnées de la projection orthogonale du centre optique C dans le plan image, et α_u et α_v représentent respectivement la taille des pixels selon les lignes et les colonnes.

Les paramètres $(U_0, V_0, \alpha_u, \alpha_v)$, appelés paramètres intrinsèques de la caméra, sont obtenus après calibrage de celle-ci, et ne dépendent que du capteur optique proprement dit.

Ainsi grâce à ces relations, il est possible de projeter les points exprimés dans le repère de la caméra R_C sur le plan image. Si les coordonnées de ces points sont données dans le repère de la scène R_0, il faut de plus déterminer la matrice des paramètres extrinsèques. Celle- ci décrit la situation de la caméra par rapport au repère du robot R_M. Ainsi, connaissant la situation du robot dans l'environnement, nous pouvons projeter les points 3D de la scène dans le repère de la caméra R_C , et donc sur le plan image.

A présent, nous avons établi le lien entre la position des points de la scène exprimés dans le repère caméra et les mesures effectuées dans l'image.

Afin de réaliser l'asservissement visuel du *Robucar* pour le suivi d'une trajectoire composée de segments de droite, il s'avère nécessaire de modéliser l'interaction entre la caméra et l'environnement, ceci passe par la détermination de la matrice d'interaction pour une primitive de type droite.

3.4.2 Matrice d'interaction pour une primitive du type droite

Dans le chapitre 2 de ce mémoire, une étude de l'interaction caméra-environnement a été abordée, ainsi nous allons exploiter les résultats de cette étude pour la synthèse d'un asservissement visuel du *Robucar*, notamment la matrice d'interaction pour une primitive de type droite sera adaptée pour notre cas d'étude. Cette matrice sera donc écrite, dans le repère de la caméra R_C, comme suit :

$$L_s^T = \begin{pmatrix} -\lambda_\theta sin\theta & \lambda_\theta cos\theta & -\lambda_\theta \rho & \rho sin\theta & -\rho cos\theta & -1 \\ -\lambda_\rho sin\theta & \lambda_\rho cos\theta & \lambda_\rho \rho & (1+\rho^2)cos\theta & (1+\rho^2)sin\theta & 0 \end{pmatrix} \tag{3.23}$$

Avec :

$$\begin{cases} \lambda_\theta = \dfrac{1}{d_i}.(a_i cos\theta + b_i sin\theta) \\ \lambda_\rho = \dfrac{1}{d_i}.(-a_i.\rho cos\theta + b_i.\rho sin\theta + c_i) \end{cases} \tag{3.24}$$

Donc, pour :

$$s = (\theta, \rho)^T \tag{3.25}$$

$$\dot{s} = L_s^T.\tau \tag{3.26}$$

3.4.3 Calcul de la matrice d'interaction à l'équilibre

L'algorithme de traitement d'images donne les coordonnées polaires de la droite détectée dans l'image, par rapport à une ligne de référence. Cette dernière correspond à la ligne détectée lorsque le robot est correctement asservi. Dans notre cas nous considérons la droite centrée verticale comme étant la droite de référence, elle a donc pour coordonnées dans le repère image :

$$s^* = [\theta^*, \rho^*]^T = [0, 0]^T \tag{3.27}$$

Dans la scène, cette droite de référence appartient au plan du sol et au plan $y = 0$, exprimés tous les deux dans le repère R_C $(O, \overrightarrow{X_C}, \overrightarrow{Y_C}, \overrightarrow{Z_C})$ voir figure (Fig : 3.11)

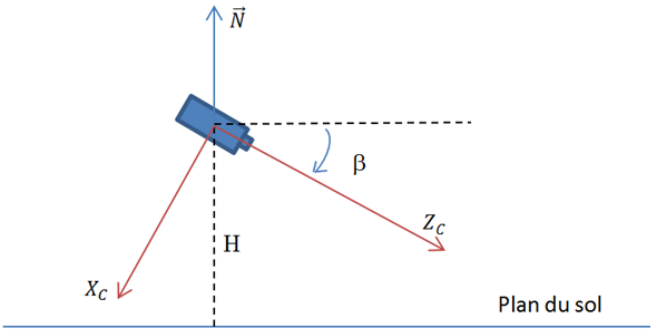

FIGURE 3.11 – position de la caméra

$\overrightarrow{N}$ Etant le vecteur normal au plan du sol.

A présent, nous calculons l'équation du plan auquel appartient la droite détectée dans le repère caméra $R_C(O, \overrightarrow{X_C}, \overrightarrow{Y_C}, \overrightarrow{Z_C})$:

En projetant le vecteur $\overrightarrow{N}$ dans le repère R_C, on obtient les coordonnées de ce vecteur normal au plan du sol :

$$\overrightarrow{N} = \begin{pmatrix} -cos\beta \\ 0 \\ -sin\beta \end{pmatrix} \tag{3.28}$$

L'équation du plan est de la forme :

$$\overrightarrow{N} = \quad -xcos\beta - zsin\beta + Cte = 0 \tag{3.29}$$

On considère un point P appartenant au plan :

$$P = \begin{pmatrix} -H/cos\beta \\ 0 \\ 0 \end{pmatrix} \tag{3.30}$$

H étant la hauteur du point C par rapport au sol, tel que : $H = h + c$ En remplaçant dans l'équation du plan on a : $Cte = H$

D'où l'équation du plan :

$$\overrightarrow{N} = \quad -xcos\beta - zsin\beta + H = 0 \tag{3.31}$$

La représentation de la droite de référence dans le repère 3D $R_C(O, \overrightarrow{X_C}, \overrightarrow{Y_C}, \overrightarrow{Z_C})$ est donnée par les équations suivantes :

$$P = \begin{cases} -xcos\beta - zsin\beta + H = 0 \\ y = 0 \end{cases} \tag{3.32}$$

A partir de la matrice d'interaction pour une primitive de type droite calculée précédemment, et en utilisant les équations de la droite de référence on peut calculer la matrice d'interaction à la positon d'équilibre tel que :

$$s^* = [\theta^*, \rho^*]^T = [0, 0]^T \tag{3.33}$$

D'après l'équation (3.25) on trouve :

$$\begin{cases} \lambda_\theta = -cos\beta/H \\ \lambda_\rho = -sin\beta/H \end{cases} \tag{3.34}$$

La matrice d'interaction est donc égale à :

$$L^T_{s=s^*} = \begin{pmatrix} 0 & -cos\beta/H & 0 & 0 & 0 & -1 \\ 0 & -sin\beta/H & 0 & 1 & 0 & 0 \end{pmatrix} \tag{3.35}$$

Nous venons ainsi de déterminer un lien entre les variations des coordonnées (θ, ρ) d'une droite de l'image et les mouvements de notre caméra. Ce lien est la matrice d'interaction semblable à une liaison mécanique virtuelle entre la scène décrite par le mouvement d'une droite et la caméra.

3.5 Développement de la commande par l'approche fonction de tâche

Dans la modélisation de la scène, nous avons considéré une primitive visuelle (une droite) donnée par le vecteur S. A partir de cette information, et par comparaison avec une référence (S^*), voir figure(Fig :3.12) nous désirons établir une loi de commande pour le *Robucar*. *Chaumette* dans

[Chaumette 90] propose une approche par fonction de tâche qui induit un mouvement de la caméra selon les 6 axes en fonction de l'erreur détectée dans les informations visuelles $(S - S^*)$. En effet, on va chercher à exprimer les mouvements de la caméra en fonction de ceux de la droite détectée dans l'image .

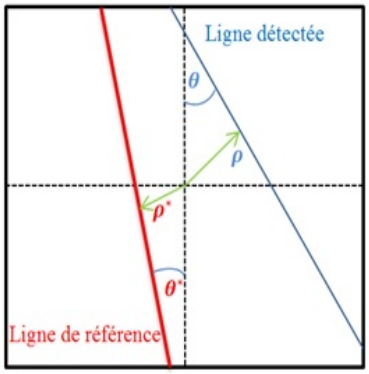

FIGURE 3.12 – Détection des droites dans l'image

Considérons la fonction de tâche (ou d'erreur) donnée par Chaumette [Chaumette 90] pour la commande référencée vision :

$$e = C.(S - S^*) \tag{3.36}$$

- C est une matrice de combinaison permettant d'utiliser un nombre d'informations visuelles supérieur au nombre de degrés de liberté contraints par la fonction de tâche e. Elle traduit l'erreur $(S - S^*)$ exprimée dans l'espace image dans l'espace cartésien du robot. Cela signifie que l'erreur dans l'image devient, par la matrice C, une erreur sur les axes du robot.
- e est un vecteur à 6 composantes correspondant aux 6 degrés de liberté du robot.
- $S - S^*$ est un vecteur à 2 composantes qui sont les informations visuelles.
- C est une matrice à 6 lignes et deux colonnes ($C1$ et $C2$). Ecrivons : $C = (C1, C2)$.

nous avons :

$$e = C_1(\theta - \theta^*) + C_2(\rho - \rho^*) = e_\theta + e_\rho \tag{3.37}$$

Pour notre application qui vise à asservir un robot évoluant à vitesse moyenne constante par rapport à une droite, nous admettons que l'objet considéré (la droite), ne se déplace pas par rapport à la caméra, ce qui est réaliste pour une trajectoire rectiligne ou pour une trajectoire « approximée » par morceaux à une trajectoire rectiligne.

3.5.1 Commande proportionnelle

Classiquement en commande référencée vision nous prenons le torseur cinématique représentant les mouvements de la caméra comme vecteur de commande. Dans le cas où la primitive reste fixe par rapport au repère caméra, une loi de commande possible est :

$$T = -\lambda e \tag{3.38}$$

λ étant un réel positif non nul. Ceci correspond à une décroissance exponentielle de la fonction de tâche :

$$\dot{e} = -\lambda e \tag{3.39}$$

Nous choisissons une loi de commande similaire donnée par :

$$T = -\lambda(\beta e_\theta + e_\rho) \tag{3.40}$$

λ et β étant deux réels positifs non nuls.

Nous avons introduit un gain supplémentaire sur un des deux termes de l'erreur visuelle, ainsi nous n'imposons pas la même décroissance au cap « θ » et à l'erreur latérale « ρ ». ce qui nous permettra de différencier l'importance que l'on donne à l'erreur latérale et au cap.

C'est pourquoi nous imposerons une première décroissance exponentielle à e_θ et une autre décroissance exponentielle à e_ρ en posant :

$$\begin{cases} \dot{e_\theta} = -\lambda\beta.e_\theta \\ \dot{e_\rho} = -\lambda.e_\rho \end{cases} \tag{3.41}$$

Donc :

$$\dot{e} = \dot{e_\theta} + \dot{e_\rho} = -\lambda\beta.e_\theta - \lambda.e_\rho \tag{3.42}$$

En utilisant les méthodes classiques du formalisme de la commande référencée vision nous pouvons calculer notre matrice de commande [Debain 96]. Pour cela nous dérivons l'équation (3.37)

$$\dot{e} = C\dot{S} = C_1\dot{\theta} + C_2\dot{\rho} = \dot{e_\theta} + \dot{e_\rho} \tag{3.43}$$

Avec l'équation (3.42) nous avons :

$$\dot{e} = -\lambda.[\beta C_1(\theta - \theta^*) + C_2(\rho - \rho^*)] \tag{3.44}$$

Nous obtenons :

$$\dot{e} = -\lambda C \begin{pmatrix} \beta & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \theta - \theta_* \\ \rho - \rho_* \end{pmatrix} \tag{3.45}$$

Avec l'équation précédente :

$$\dot{e} = C\dot{S} = -\lambda.C.B(S - S^*) \tag{3.46}$$

Comme nous l'avons précisé au chapitre 2, une valeur possible de C pour satisfaire les conditions de convergence au voisinage de la position désirée est [Chaumette 90] :

$$C = L_{s=s^*}^{T_+} \tag{3.47}$$

$L_{s=s^*}^{T_+}$ est la pseudo inverse de la matrice d'interaction calculée pour la position d'équilibre.

$$L_{s=s^*}^{T_+}.L_{s=s^*}^{T} = I_6 \tag{3.48}$$

La modélisation de la scène nous donne au voisinage de la position d'équilibre :

$$S = L^{T}_{s=s^*}.T_c \tag{3.49}$$

T_c est le torseur cinématique représentant les 6 degrés de liberté du robot.

Donc

$$T_c = -\lambda L^{T_+}_{s=s^*} B.(S - S^*) \tag{3.50}$$

T_c donne une commande en vitesse contrôlant les mouvements de la caméra en fonction des coordonnées de la droite détectée et de la droite de référence exprimées dans l'espace image. Cette commande ne nécessite aucune reconstruction de la position des droites dans la scène.

3.5.2 Commande proportionnelle intégrale

Pour un processus sans intégration propre, la loi de commande proportionnelle laisse, en général, subsister une erreur résiduelle e. En effet, lorsque l'erreur est nulle, la sortie du correcteur proportionnel est également nulle. Pour diminuer au maximum l'erreur résiduelle, la solution est d'augmenter le gain de la commande. Le gain idéal est infini, mais cette commande de type ≪ tout ou rien ≫ sollicite énormément le système tout en étant instable (en particulier si le système commandé comporte des retards purs...).

Au contraire, la commande proportionnelle intégrale est une commande progressive et persévérante. Tant que subsiste une erreur, la commande croît ou décroît, mais l'annulation de l'erreur n'implique pas forcément une commande nulle. En effet, la sortie du correcteur proportionnel intégral dépend de l'erreur mais aussi du point de fonctionnement considéré.

Ainsi l'équation de commande sera de la forme :

$$\dot{e} = -\lambda \quad (\quad \beta(e_\theta + K_1 \quad \int_0^t e_\theta \quad dt) + (\quad e_\rho + K_1 \int_0^t e_\rho \quad dt)) \tag{3.51}$$

λ, β, k_1, k_2 étant des réels positifs.

En procédant de la même manière que pour le cas de la commande proportionnelle, on obtient :

$$T_c = -\lambda \quad L^{T_+}_{s=s^*} B \quad [\quad (S - S^*) + K_i \quad \int_0^t (S - S^*) \quad dt \quad] \tag{3.52}$$

Avec $K_i = \begin{bmatrix} K_{i1} & 0 \\ 0 & K_{i2} \end{bmatrix}$

Nous avons donc calculé une loi de commande utilisant un correcteur proportionnel intégral. Son entrée est dans le cas de la commande référencée vision l'information visuelle $(S - S^*)$ exprimée dans le repère caméra. Cette dernière loi comporte l'avantage d'accroître la robustesse quant aux erreurs de modélisation de notre système ou des perturbations basses fréquences auxquelles il sera soumis [Debain 96].

Remarque :

Pour les deux lois de commande précédemment citées nous utilisons la matrice d'interaction calculée à la position d'équilibre. Ceci représente une approximation forte mais qui ne remet pas

en cause la convergence du système même pour des conditions initiales très éloignées de la position d'équilibre. Les erreurs de modélisation seront compensées par la boucle fermée.

3.5.3 Commande proportionnelle intégrale dérivée (PID)

La loi de commande proportionnelle intégrale permet d'annuler l'erreur résiduelle et fait en sorte que le robot converge vers la position désirée, mais cette dernière ne tient pas compte du temps nécessaire à cette convergence, de plus l'application réel de cette commande nécessite des temps de calcul relativement grands, ceci est particulièrement dû à l'application de traitement d'image qui extrait les informations visuelles or le temps de calcul dans une tâche d'asservissement est déterminant pour la stabilité du système, ceci nous amène à ajouter une action dérivée, celle-ci anticipe en effet les variations de la sortie du système et accélère de ce fait le temps de convergence.

Dans ce cas l'équation de commande que nous retiendrons sera de la forme :

$$\dot{e} = -\lambda(\beta(e_\theta + K_{i1}\int_0^t e_\theta dt + K_{d1}\frac{d(\theta-\theta^*)}{dt}) + (e_\rho + K_{i1}\int_0^t e_\rho dt + K_{d2}\frac{d(\rho-\rho^*)}{dt}))) \tag{3.53}$$

La même démarche que dans le cas de la commande PI nous amène à écrire :

$$T_c = -\lambda \quad L_{s=s^*}^{T+} B \quad [\quad (S-S^*) + K_i \quad \int_0^t (S-S^*) \quad dt \quad + K_d \quad \frac{d(S-S*)}{dt}] \tag{3.54}$$

Avec $K_d = \begin{bmatrix} K_{d1} & 0 \\ 0 & K_{d2} \end{bmatrix}$

Le schéma général de l'asservissement 2D utilisé dans notre travail, avec la mise en place de ce régulateur PID est le suivant :

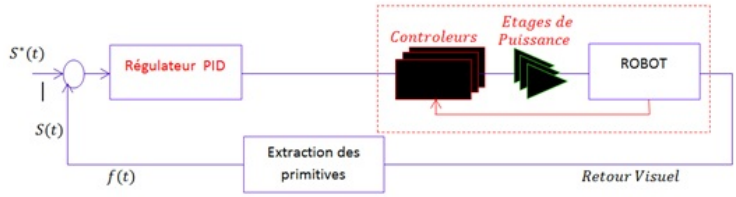

FIGURE 3.13 – Asservissement 2D avec régulateur PID

Cependant la mise en œuvre de l'action dérivée dans le cas pratique engendre des problèmes que nous allons détailler par la suite, ceci est notamment du au cas où les signaux d'erreur sont numériques et soumis à une période d'échantillonnage, ainsi qu'aux bruits de mesure.

Les lois de commande calculées ci-dessus seront validées dans un premier temps en simulation, et ceci dans le but de les implémenter sur le robot et les tester dans le cas réel, nous allons donc voir dans ce qui suit les méthodes qui nous permettrons d'adapter ces lois de commande pour l'implémentation.

3.5.4 Numérisation de la commande

Le régulateur PID calculé précédemment est une forme analogique de la forme suivante :

$$u(t) = K_p \quad [\quad e(t) + K_i \quad \int_0^t e(t) \quad dt \quad + K_d \quad \frac{de(t)}{dt}] \tag{3.55}$$

L'équivalant de ce régulateur en numérique est :

$$u(k) = K_p \quad [\quad e(k) + K_i Te \quad \sum_{i=0}^{k-1} e(i) \quad dt \quad + K_d \quad \frac{e(k-1) - e(k))}{Te}] \tag{3.56}$$

Avec Te la période d'échantillonnage.

Tout fois la mise en œuvre de ce régulateur en pratique a engendré un certain nombre de problèmes que nous détaillerons ci-dessous, tel que l'effet de la période d'échantillonnage Te, la saturation due à l'accumulation des erreurs (Action intégrale) ainsi que l'amplification du bruit par l'action dérivée [Mudry 06].

a- Effet de la période d'échantillonnage

Lorsqu'on a un même signal échantillonnée à plusieurs fréquences, on remarque que par exemple lorsque la période d'échantillonnage $Te = 2/f = 2T$, f étant la fréquence du signal, il devient impossible de reconstituer le signal original [Cabodevila 08].

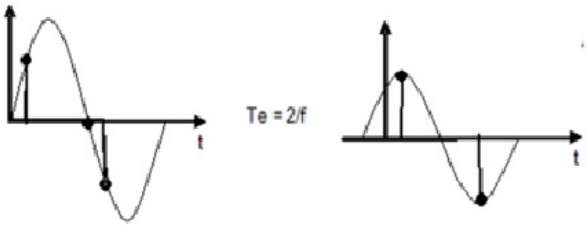

FIGURE 3.14 – signal original et signal échantillonné

En augmentant encore la période d'échantillonnage on tombe sur des aberrations similaires. De plus lorsque la fréquence maximale du signale $Fmax > Fe/2$, il se produit le phénomène de *recouvrement de spectre* (à ne pas confondre avec *le repliement de spectre*). Si ce phénomène se produit, il est alors impossible de reconstituer le signal original :

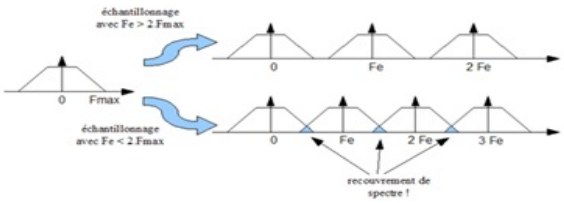

FIGURE 3.15 – phénomène de recouvrement de spectre

Afin d'éviter ces phénomènes, il est impératif de respecter **le théorème de Shannon**.

b- Effets liés à l'action intégrale

Après application du correcteur précédent, la deuxième imperfection qui apparait est due aux non linéarités du système et en particulier les saturations de l'organe de commande (vérins, angle de braquage maximale. . . etc.)

En particulier lorsque l'erreur est importante (au démarrage par exemple), l'intégrateur intègre une erreur grande et donc sa sortie est très grande. Lorsque le système arrive à la valeur de consigne, l'intégrateur est encore « plein » et donc le système dépasse largement cette valeur de consigne.

Pour éviter ce problème, deux solutions sont envisageables :

* Soit par la mise en place d'un ***anti-windup***, qui limite la valeur stockée dans l'intégrateur pur au maximum des commandes admissibles par notre système [Cabodevila 08]. Le schéma du régulateur PID avec ***Anti-Windup*** est donné ci-dessous :

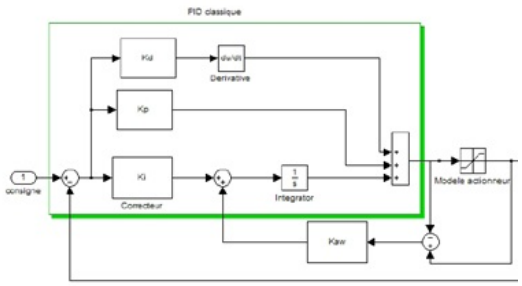

FIGURE 3.16 – Régulateur PID classique avec un Anti-windup

* Soit par la limitation de la somme des erreurs calculées pour l'action intégrale par un nombre N tout en calculant la moyenne, ainsi l'action intégrale sera calculée :

$$u_i(k) = K_i \quad \frac{(N-1)}{N}(\sum_{i=K-N-1}^{k-1} e(i)) \tag{3.57}$$

Ainsi on peut agir non seulement sur la valeur du paramètre de l'action intégrale, mais aussi sur le paramètre N pour avoir la bonne valeur qui nous évitera la saturation tout en annulant l'erreur résiduelle lors de l'asservissement (notamment lorsque le robot suit une ligne droite)

Afin d'éviter que l'intégrateur accumule l'écart au-delà de valeurs pouvant conduire à des saturations, il est judicieux de limiter le terme $u_i[k]$ entre u_{imin} et u_{imax} [Mudry 06].

c- Effets du bruit sur l'action dérivée

Quand la période d'échantillonnage tend vers sa valeur minimale, toute en respectant le théorème de Shannon, l'action dérivée calculée par le taux de variation entre deux périodes d'échantillonnage tend vers la dérivée de la fonction. Ceci n'est malheureusement vrai que dans le cas d'un signal sans bruit. Or dans notre cas les signaux, viennent du traitement d'image, sont entachés de bruit, et dans ce cas la proposition précédente n'est plus vraie. La figure suivante montre que l'erreur sur le calcul de la dérivée croit lorsque Te décroît.

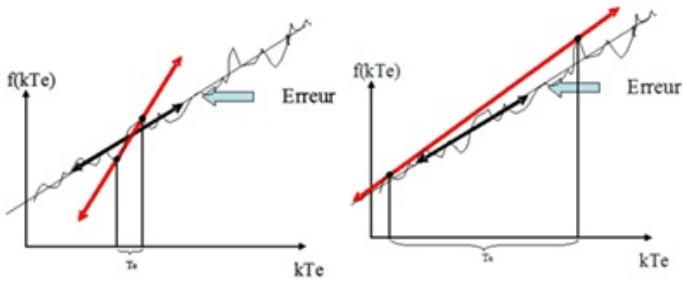

FIGURE 3.17 – Influence de la période d'échantillonnage sur la dérivée dans la présence de bruit.

3.6 Synthèse de la commande par logique floue

A partir de concepts mathématiques relativement simples, la logique floue permet de résoudre des problèmes complexes là où les outils de résolution conventionnels sophistiqués peinent. Son succès réside principalement dans son potentiel à résoudre des problèmes du monde réel. L'industrie et les services peuvent bénéficier des atouts de la logique floue, la panoplie des applications à l'échelle planétaire en témoigne. Tous les problèmes de contrôle, de diagnostic ou d'aide à la décision sont des clients potentiels de la logique floue. Ainsi, La logique floue s'est imposée dans des domaines aussi variés que l'électrotechnique, l'industrie, l'automatisme, la robotique, , le contrôle aérien, afin de résoudre les problèmes d'identification, de régulation de processus, d'optimisation, de classification, de détection de défauts ou de prise de décision [Elougli 09].

Le développement des algorithmes flous se fait à travers les méthodes par lesquelles l'homme essaye de copier la nature et de reproduire des modes de raisonnement et de comportement qui lui sont propres.

Dans ce contexte, nous allons appliquer les principes de la logique floue pour l'asservissement visuel du *Robucar*. Nous allons donc commencer par définir ces principes d'une façon générale, pour ensuite nous interesser à notre sujet d'étude qui concerne l'asservissement du *Robucar* et cela à partir de l'expertise acquise lors des essais.

3.6.1 Le principe de réglage flou

Le principe de réglage par logique floue part du constat suivant : dans les problèmes de régulation auxquels il est confronté, l'homme ne suit pas, à l'image de ses inventions, un modèle mathématique précis fait de valeurs numériques et d'équations. Au contraire, il utilise des termes tels que « un peu, beaucoup, plus, à fond. etc... » Ainsi que ses propres connaissances qu'il a dans le domaine. Ces connaissances sont, le plus souvent, acquises de façon empirique [Ouadah 04].

Le principe du réglage par logique floue s'approche de la démarche humaine dans le sens où les variables traitées ne sont pas des variables logiques (au sens de la logique binaire par exemple) mais des variables linguistiques, proche du langage humain. De plus, ces variables linguistiques sont traitées à l'aide de règles qui font référence à une certaine connaissance du comportement du système à régler.

Les éléments indispensables du principe du réglage par logique floue sont :

- Les variables floues.
- Les règles d'inférences.
- Les fonctions d'appartenance.

3.6.2 Contrôleur flou

Les grandeurs de sortie d'un système à commander et éventuellement d'autres mesures déterminantes pour saisir l'évolution dynamique du système ainsi que les consignes définissent les variables d'entrée du contrôleur flou. Les variables de sortie de ce contrôleur sont les commandes à appliquer au système [Elougli 09].

Le contrôleur flou est constitué de 4 blocs principaux (Fig : 3.18) la base de connaissance, le système d'inférence, l'interface de fuzzification et l'interface de défuzzification.

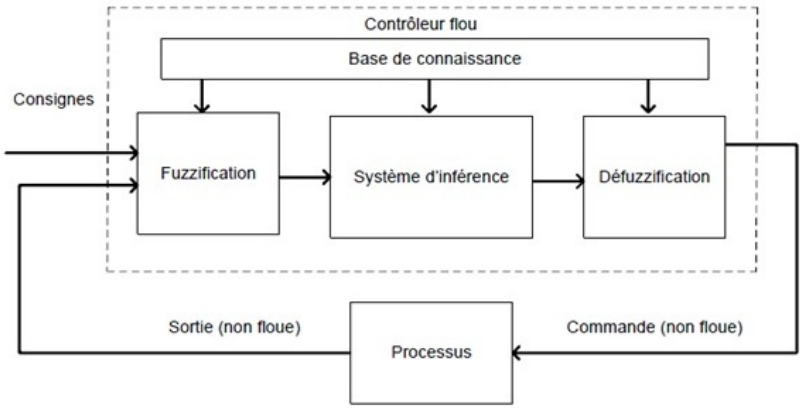

FIGURE 3.18 – Schéma synoptique d'un contrôleur flou.

Un contrôleur flou passe généralement par les étapes suivantes :

- Choix de la stratégie de fuzzification.
- Etablissement de la base de règles.
- Choix de la méthode d'inférences.
- Choix de la stratégie de défuzzification.

a - Fuzzification

Les variables d'entrée et de sortie choisies pour modéliser ou commander un système sont des grandeurs numériques. L'étape de fuzzification consiste à transformer ces grandeurs réelles en variables linguistiques en vue d'un traitement d'inférence. Ainsi, à chaque variable d'entrée et de sortie est associé des ensembles caractérisant les termes linguistiques pris par ces variables. Ces termes seront utilisés pour écrire les règles d'inférence [Elougli 09].

b - La base de règles

Une base de règles floues est une collection de règles qui permet de lier les variables floues d'entrée et de sortie. La description de la commande se fait par l'intermédiaire de ces règles

c - Méthode d'inférence floue

Elle permet de calculer l'ensemble flou associé à la commande et se fait par les opérations d'inférence floue et l'agrégation des règles. L'inférence floue repose sur l'utilisation d'un opérateur d'implication floue pour chaque règle à analyser. Cet opérateur quantifie la force de liaison entre la prémisse et la conclusion de la règle.

d - Défuzzification

Le traitement des règles d'inférences fournit une valeur floue. L'étape de défuzzification consiste à transformer l'ensemble flou résultant de l'agrégation des règles en une grandeur de commande précise à appliquer au processus. Dans la littérature, il existe plusieurs stratégies pour réaliser cette opération telle que la moyenne des maxima, le centre des aires, le centre des maxima. La méthode de défuzzification par le centre de gravité est la méthode la plus utilisée en commande floue du fait qu'elle fournit intuitivement la valeur la plus représentative de l'ensemble flou issu de l'agrégation des règles. Elle consiste à calculer le centre de gravité de la surface formée par la fonction d'appartenance résultante. Mais on trouve aussi la méthode de Takagi-Sugeno où les conclusions sont polynomiales [Elougli 09].

3.6.3 Le régulateur flou pour l'asservissement visuel du *Robucar*

L'application se résume par la synthèse d'un régulateur flou pour l'asservissement du *Robucar*. Les entrées du régulateur sont le rayon ρ et l'angle θ de la droite extraite de l'image, la sortie du régulateur est l'angle de braquage nécessaire pour le suivi de la trajectoire.

En se basant sur les caractéristiques techniques du robot, la conception du contrôleur flou est proposée en définissant les propriétés fonctionnelles et opérationnelles ci-dessous :

a - Les fonctions d'appartenance

Cela consiste à spécifier le domaine de variation des variables : l'univers de discours, q'on divise en intervalles (sous-ensembles flous). Cette répartition, consiste à fixer le nombre de ces valeurs et les distribuer sur le domaine, est faite en se basant sur des connaissances du système et selon la précision désirée. Les fonctions d'appartenance sont explicitées dans les figures qui suivent.

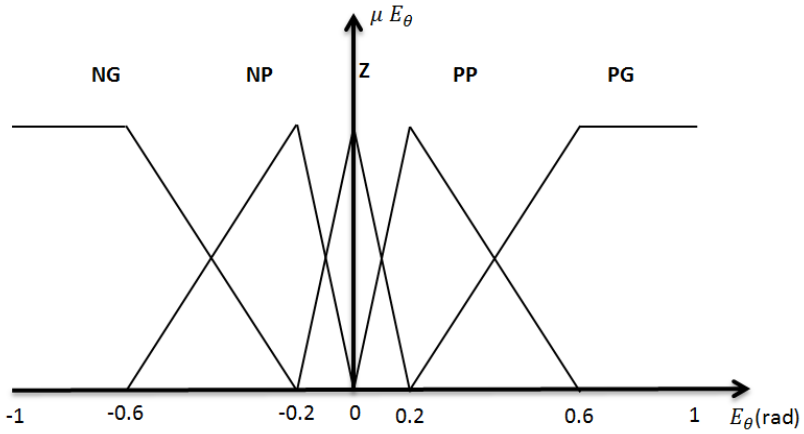

FIGURE 3.19 – Les fonctions d'appartenance de l'angle

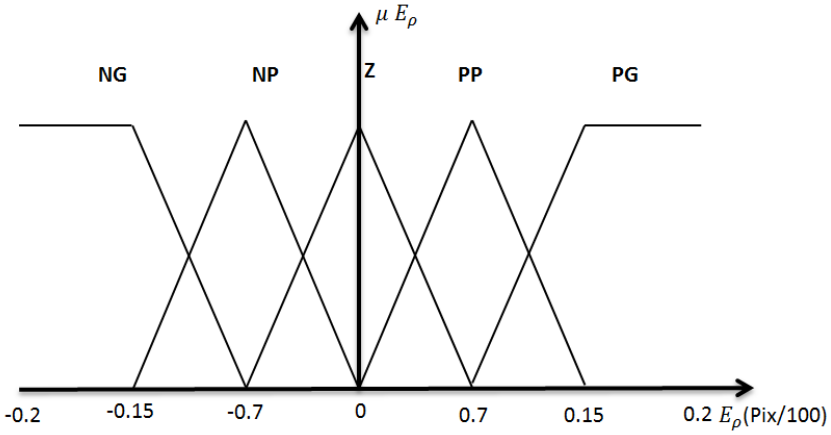

FIGURE 3.20 – Les fonctions d'appartenance du rayon

Tel que les variables linguistiques sont :

PG : Positif grand ;

PM : Positif moyen ;

PP : Positif petit ;

Z : Zéro ;

NP : Négatif petit ;

NM : Négatif moyen ;

NG : Négatif grand ;

b - Les bases des régles

Cette étape concerne l'élaboration des régles, pour définir le comportement attendu du robot selon ses paramètres intrinsèques : angle de braquage et vitesse. Pour chacune des combinaisons des valeurs des variables d'entrées une action sur les variables de sorties lui est associée. La base des règles floues (Situation Action), proposée dans la table est construite manuellement.

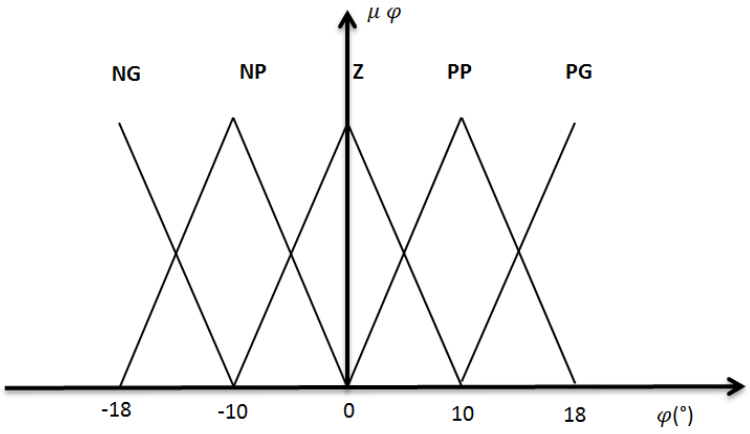

FIGURE 3.21 – Les fonctions d'appartenance de la commande (MAMDANI)

$E_\rho \backslash E_\theta$	**NG**	**NP**	**Z**	**PP**	**PG**
NG	PG	PG	PG	PP	PP
NP	PP	PP	Z	Z	Z
Z	PG	PP	Z	NP	NG
PP	Z	Z	Z	NP	NP
PG	NP	NP	NG	NG	NG

TABLE 3.1 – Le tableau des règles (MAMDANI)

La construction des règles s'est faite sur la base des expériences déja effectuées sur le robot, en effet les conclusions des règles ne sont pas seulement fonction des valeurs de ρ et θ, mais aussi en fonction de leurs signes. La figure suivante montre un exemple sur le signes de ces informations visuelles en fonction de la position du robot, ainsi des valeurs de rayon et de l'angle positives correspondent à une phase d'éloignement du robot de sa position de référence, l'angle de braquage sera donc grand (GN dans le tableau), contrairement au cas ou le rayon est positif et l'angle négative, ce qui correspond à une phase d'approche du robot vers sa position de référence, l'angle de braquage sera dans ce cas moins grand (PN dans le tableau).

c - Méthode de défuzzyfication

Une fois la mise en place des fonctions d'appartenance et l'établissement des règles définissant le comportement du régulateur ont été effectués, on passe à la sélection d'une méthode de défuzzyfication. C'est cette étape qui permet de transformer les valeurs de commande du domaine flou vers le domaine réel (variables physiques).

Pour notre cas, des résultats de simulations significatifs, présentés au chapitre 4, sont obtenus en utilisant la méthode du centre de gravité (MAMDANI), mais étant donné que cette méthode requiert des temps de calculs assez élevés, ce qui est une contrainte pour une implémentation temps réel ou le temps du calcul est un facteur déterminant pour la bonne stabilité de notre système.

Pour pallier à ce problème, nous avons adopté la méthode *Takagi - Sugeno*, cette dernière bien

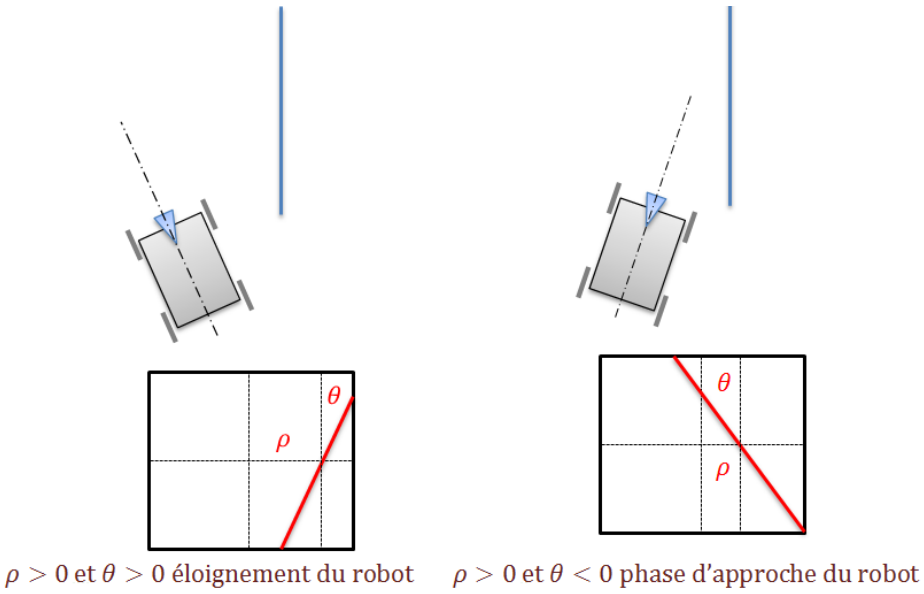

FIGURE 3.22 – Signes de ρ et θ selon la position du robot

qu'elle soit rapide en temps d'éxécution, ne fournit pas une valeur plus représentative comparé à la méthode du centre de gravité.

La figure suivante montre les fonctions d'appartenance des sorties, en utilisant la méthode *Takagi - Sugeno*.

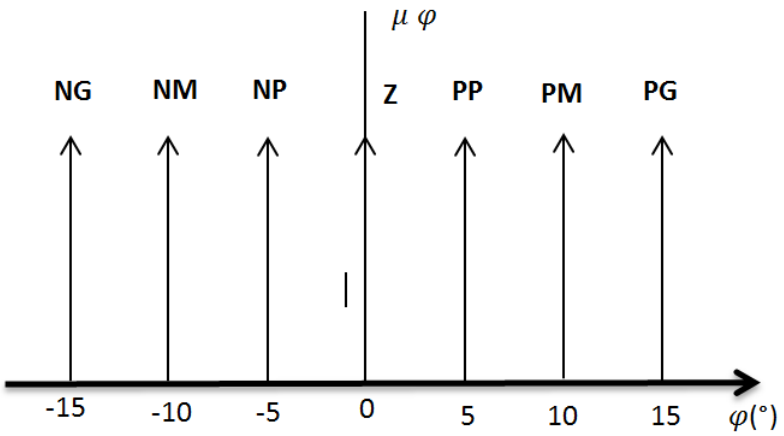

FIGURE 3.23 – Les fonctions d'appartenance de la commande (TSK)

Dans le tableau des règles devient :

$E_\rho \backslash E_\theta$	**NG**	**NP**	**Z**	**PP**	**PG**
NG	PG	PG	PG	PM	PM
NP	PM	PM	PP	PP	PP
Z	PG	PM	Z	NM	NG
PP	NP	NP	NP	NM	NM
PG	NM	NM	NG	NG	NG

TABLE 3.2 – Le tableau des règles (TSK)

3.7 Conclusion

Dans ce chapitre, nous avons réalisé une étude détaillée de notre système dans le but de synthétiser un asservissement visuel de notre robot. La modélisation complète du système robotique nous a permis de lier les mouvements du robot à ceux des indices visuels. En exploitant ce résultat, ainsi que le formalisme de fonction de tache, on a synthétisé différentes lois de commande qui permettent un contrôle dans l'image et qui ne nécessitent donc pas une reconstruction de la scène. Ce qui équivaut à un gain considérable en temps de calcul.

Dans un deuxième temps, un régulateur de type flou a été synthétisé en se basant sur l'expertise acquise durant l'étude de notre système. Ce dernier nous permettra de situer les performances des différentes lois de commande synthétisées à travers une comparaison entre ces dernières.

Les lois de commande précédemment calculées seront testées en simulation. Ce qui nous permettra de définir les gains qui éviteront la saturation des actionneurs du robot tout en assurant un suivi de la trajectoire de référence.

Chapitre 4

Simulation, Implémentation et Essais Temps Réel

4.1 Introduction

Dans le but de valider les lois de commande synthétisées précédemment, nous allons procéder dans ce chapitre à une série de simulation sous l'environnement *Matlab*. Ceci nous permettra de mesurer les performances et limitations de nos approches.

Deux séries de tests seront effectuées : la première concerne une tâche de positionnement du robot par rapport à une droite. L'autre sera une tâche d'asservissement par rapport à des segments de droites. Des interprétations ainsi que des comparaisons sur les différents résultats obtenus seront exposées.

Dans un second temps, nous allons procéder à l'implémentation en temps réel de ces lois de commande sur le *Robucar*, après avoir détaillé la chaine de traitement d'image.

Nous terminerons ce chapitre avec un essai à l'extérieur, ce qui présente une vraie mise à l'épreuve de la robustesse de notre loi de commande synthétisée.

4.2 Partie simulation

Dans un premier temps, nous allons tester nos lois de commandes en simulation sous l'environnement logiciel *Matlab*. Une tâche de positionnement par rapport à une primitive de type droite sera donc considérée. On déterminera ainsi l'influence des valeurs des différents gains (pondération, gain de régulation...), et celles des actions du régulateur PID (K_i, K_d) sur la réalisation de la tâche.

Après fixation des gains, différentes positions initiales du robot seront envisagées. Ceci nous permettra de voir les limitations de nos lois de commande. Ceci dit, les positions initiales doivent être choisies de tel sorte que la droite soit toujours dans le champ de vision de la caméra.

Enfin, des tâches d'asservissement par rapport à différentes trajectoires constituées de segments de droites seront réalisées. La scène considérée est constituée d'une droite tracée au sol, et qui a comme équation $(y = 2)$. Le robot prend différentes configurations initiales (x_0, y_0, θ_0), la caméra du robot sera inclinée d'un angle $\beta = 15°$ vers le sol afin de permettre la visibilité de la droite dans le champ de vision.

Notons que la vitesse moyenne du robot sera fixée à $0.5m/s$, étant donné que c'est la vitesse du robot dans le cas réel.

Le robot sera représenté par un triangle afin de mieux apprécier le sens de déplacement et son angle d'orientation.

4.2.1 Tâche de positionnement par rapport à une droite

a - Régulateur proportionnel

Ce test de simulation est réalisé pour une configuration initiale du robot $(1, 1, 0)$. Avec un gain de régulateur $\lambda = 0.5$ et sans pondération sur l'angle $(\beta = 1)$. La trajectoire du robot est représentée sur la figure (Fig :4.1), les figures (Fig : 4.2) et (Fig :4.3) représentent respectivement l'évolution de la droite perçue par le robot et l'évolution des fonctions de tâche, les figures (Fig :4.5) et (

Fig :4.4) représentent les commandes injectées au robot ainsi que l'évolution des erreurs sur l'angle et le rayon de droite.

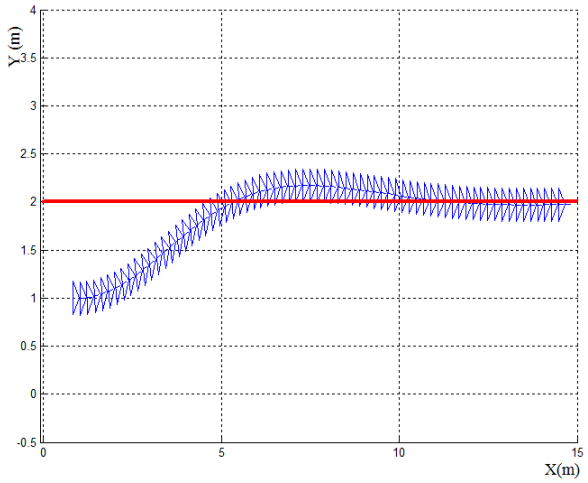

FIGURE 4.1 – La trajectoire du robot

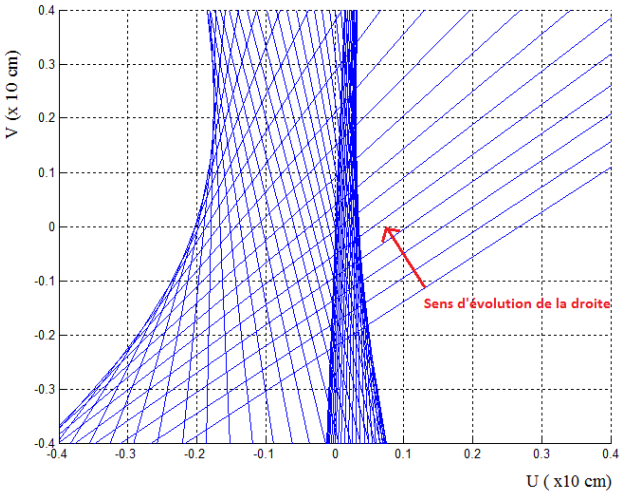

FIGURE 4.2 – L'évolution de la droite

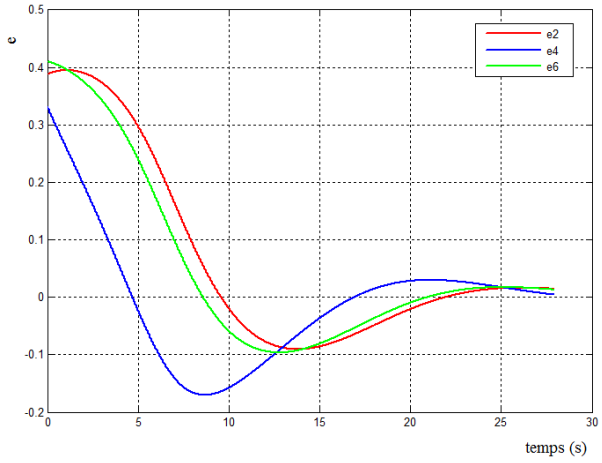

FIGURE 4.3 – Les fonctions de tâche

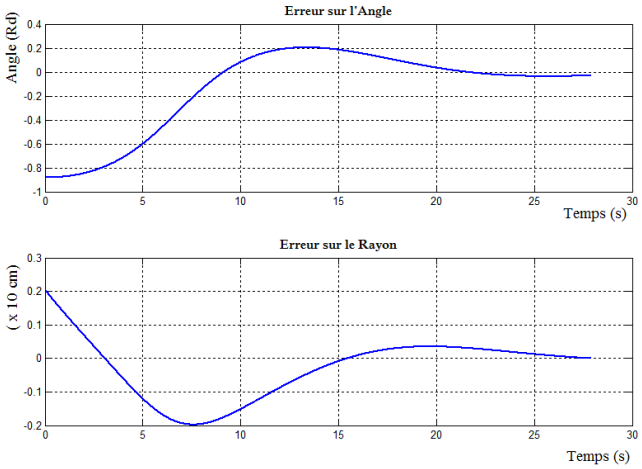

FIGURE 4.4 – Erreurs rayon, angle

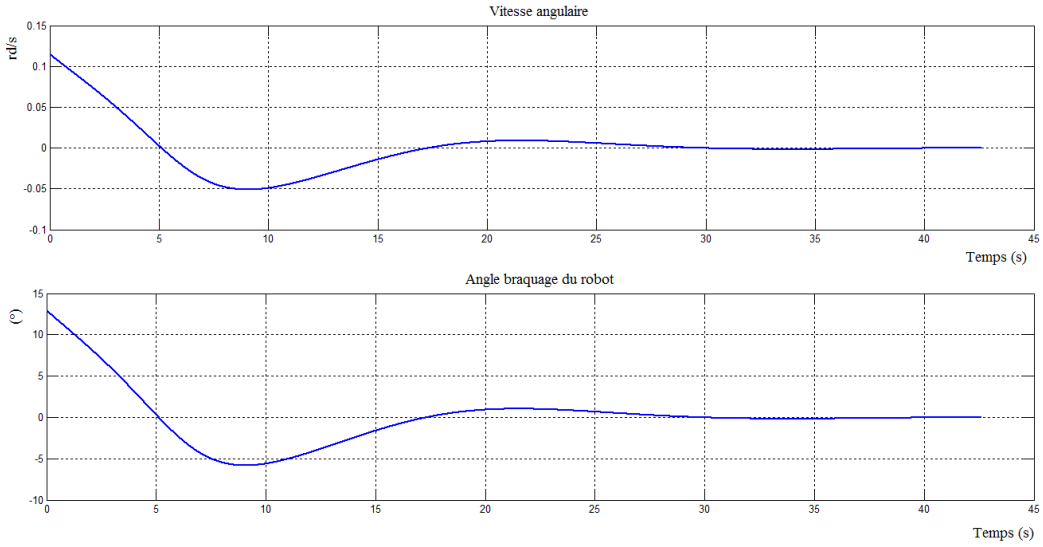

FIGURE 4.5 – Les commandes

Interprétation

Dans cette première simulation, le robot a effectué la tâche de positionnement sur une distance de $14m$ en un temps de $27secondes$. Nous remarquons qu'au démarrage le robot prend une vitesse angulaire importante. Celle-ci correspond à la valeur maximale de l'angle de braquage, cette dernière est contrôlée par les valeurs des gains choisis de telle façon qu'elle respecte les contraintes de notre système réel. La commande proportionnelle assure ainsi l'annulation des erreurs de fonctions de tâche avec une erreur de 2.10^{-2}. Cela se traduit par la convergence de la droite détectée par le robot vers la droite de référence (droite perpendiculaire centrée dans l'image) , comme le montre les figures (Fig : 4.2) et (Fig :4.4). Nous remarquons ainsi l'annulation des erreurs sur le rayon et sur l'angle, sans dépasser les limites de commande admissibles par le système réel, comme le montre la figure (Fig :4.5), avec un angle de braquage inférieur à 20°.

Néanmoins, un dépassement de la trajectoire par le robot est bien visible sur la figure (Fig :4.1), ce qui nécessite une pondération sur l'angle. Ce sera le point envisagé dans la deuxième série de simulations.

b - Proportionnel avec pondération sur l'angle

Pour ce test, nous allons introduire une pondération sur l'angle de ($\beta = 0.3$). La configuration initiale du robot reste la même $(1, 1, 0)$, ainsi que le gain de régulation ($\lambda = 0.5$). Nous obtenons les résultats suivants :

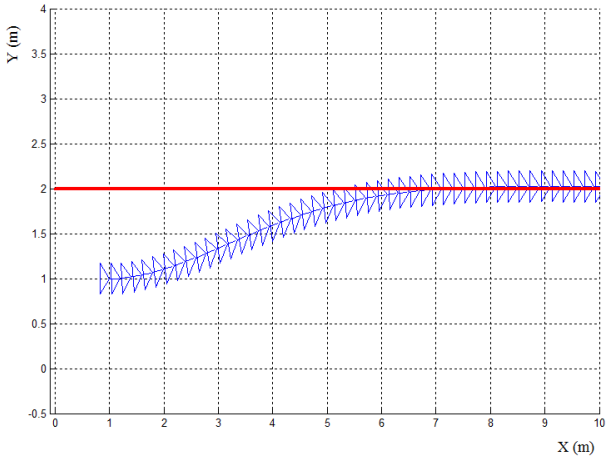

FIGURE 4.6 – La trajectoire du robot

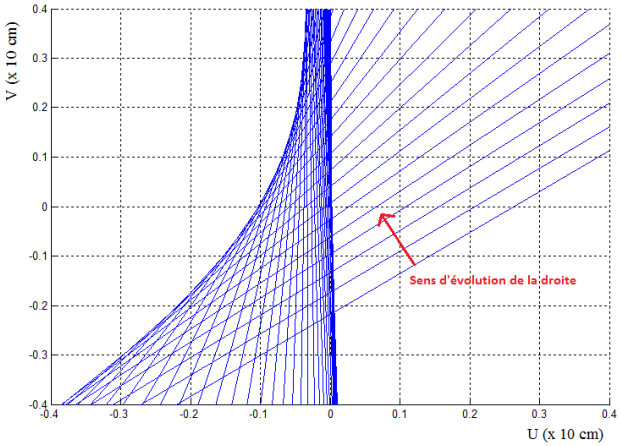

FIGURE 4.7 – L'évolution de la droite

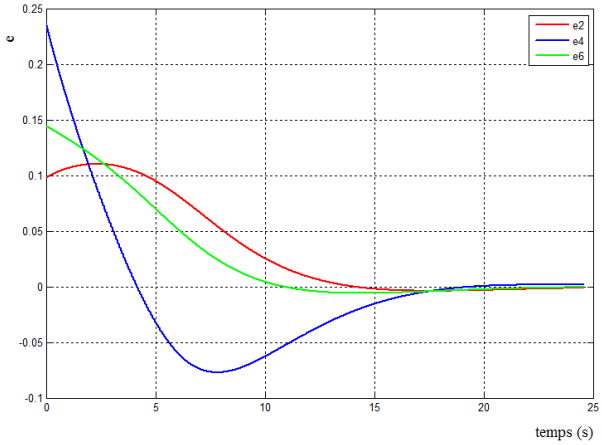

FIGURE 4.8 – les fonctions de tâche

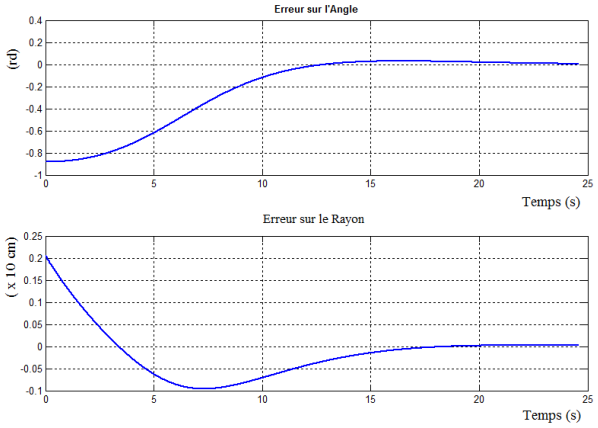

FIGURE 4.9 – Erreurs rayon,angle

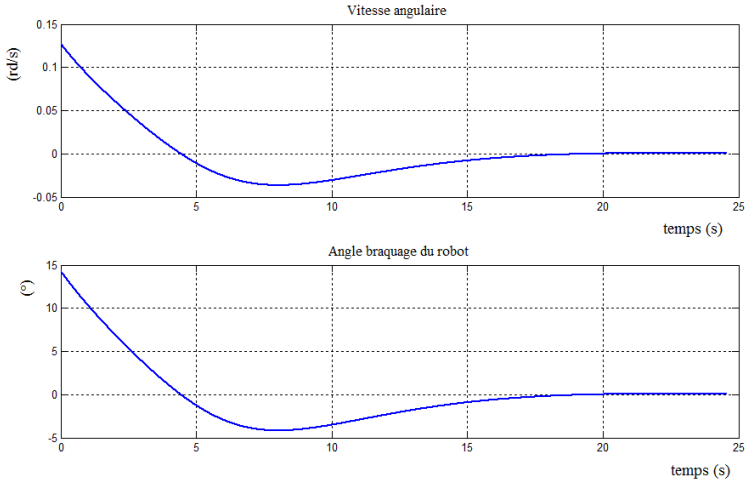

FIGURE 4.10 – Les commandes

Interprétation

Par rapport à la première simulation, on remarque l'absence du dépassement sur la figure représentant la trajectoire du robot. Cela est dû à l'introduction de la pondération sur l'erreur de l'angle, avec un facteur inférieur à 1. Ce qui privilégie dans ce cas la régulation du rayon tant que le robot est loin de sa trajectoire de référence.

Dans un second temps, on essayera d'améliorer le temps de convergence qui est égale à $24,5$ secondes pour atteindre une erreur sur la fonction de tache de l'ordre de 2.10^{-2} pour le régulateur proportionnel. On introduira donc les actions intégrale et dérivée pour ce dernier.

c - Régulateur PID

Pour une même configuration initiale du robot $(1, 1, 0)$, avec un gain du régulateur $\lambda = 0.5$ et une pondération sur l'angle ($\beta = 0.3$) et les gains suivants des actions $K_{ie2} = 0.001 \quad K_{ie4} = 0.03 \quad K_{ie6} = 0.3$; $K_{de2} = 1 \quad K_{de4} = 0.7 \quad K_{de6} = 2$.

Tel que :

K_{iej} représente le gain de l'action intégrale appliqué sur la j^{eme} erreur des fonctions de tâches.

K_{dej} représente le gain de l'action dérivée appliqué sur la j^{eme} erreur des fonctions de tâches.

Nous obtenons les résultats suivants :

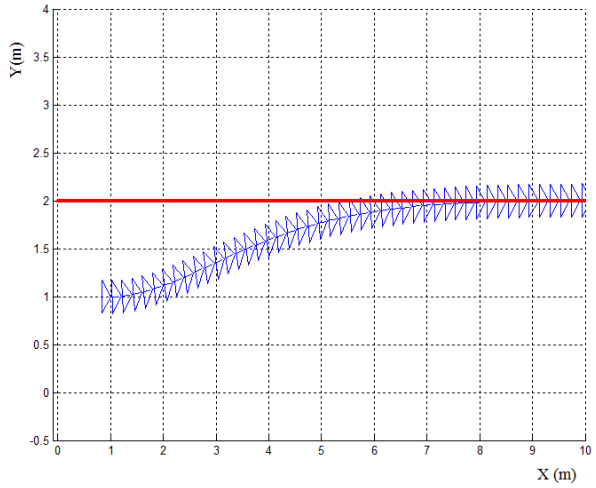

FIGURE 4.11 – La trajectoire du robot

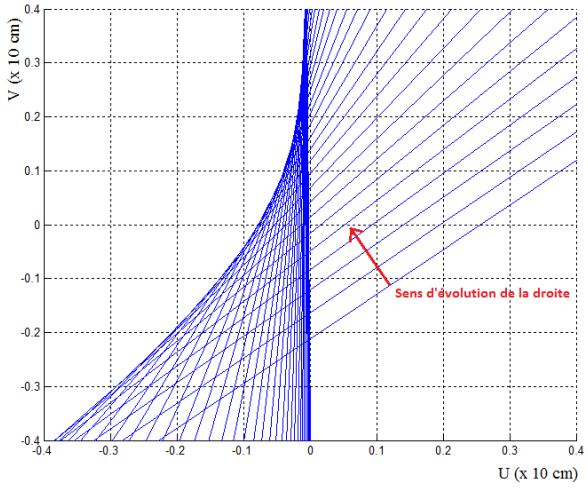

FIGURE 4.12 – L'évolution de la droite

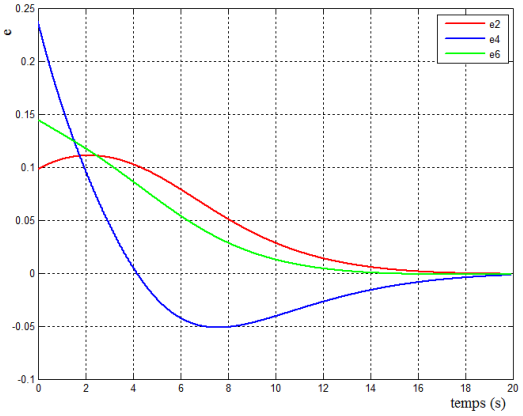

FIGURE 4.13 – les fonctions de tâche

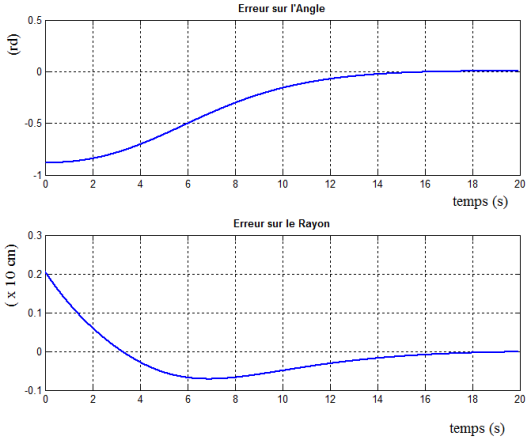

FIGURE 4.14 – Erreurs rayon,angle

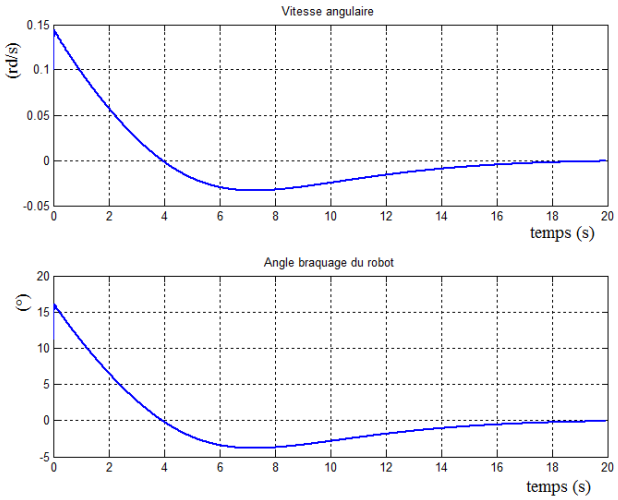

FIGURE 4.15 – Les commandes

Interprétation

Dans cette simulation, on remarque clairement l'apport des actions intégrale et dérivée. En effet, l'erreur de 2.10^{-2} est atteinte en un temps de 13 secondes ce qui correspond à un gain de $12 secondes$ par rapport au régulateur proportionnel utilisé dans la simulation précédente. La distance aussi a été raccourcie car le robot converge sur une distance de $7m$ et cela sans la saturation des actionneurs du robot.

d - Régulateur PID (Autre position initiale)

En utilisant le régulateur PID testé précédemment avec les même valeurs des gains. Nous avons, dans le but de valider ce régulateur, testé d'autres positions initiales pour voir les limitations de notre loi de commande. Ainsi la simulation ci-après donne les résultats de convergence pour une position initiale du robot de $(1, 3, -pi/6)$.

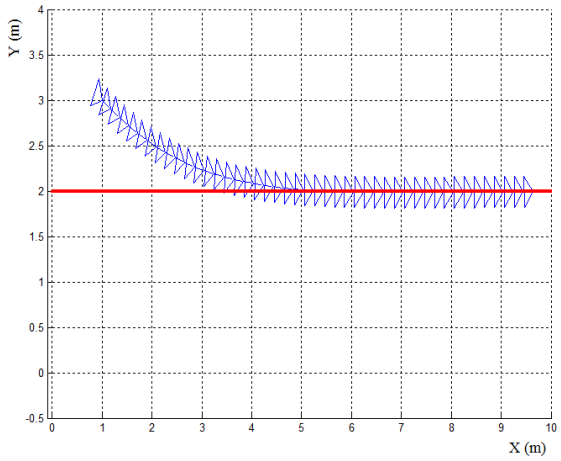

FIGURE 4.16 – La trajectoire du robot

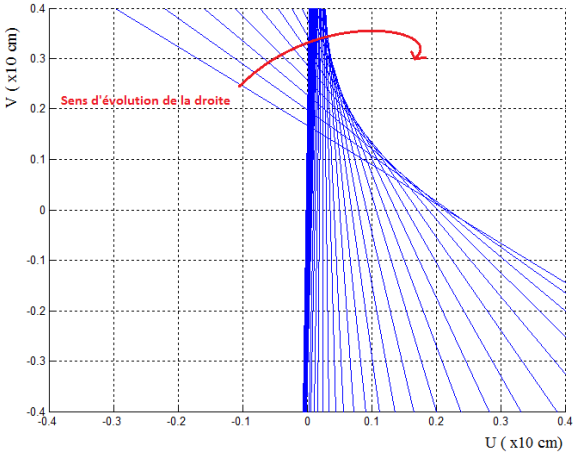

FIGURE 4.17 – L'évolution de la droite

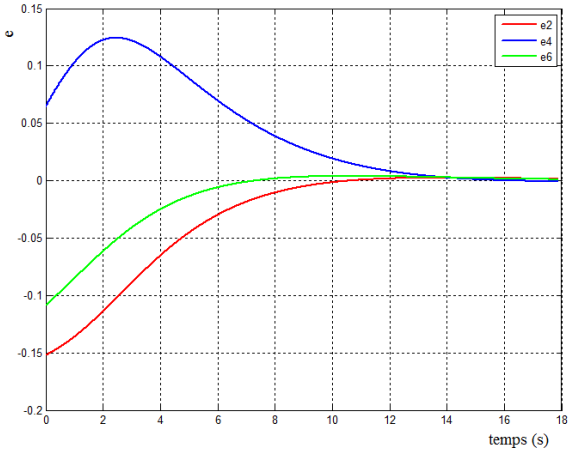

FIGURE 4.18 – les fonctions de tâche

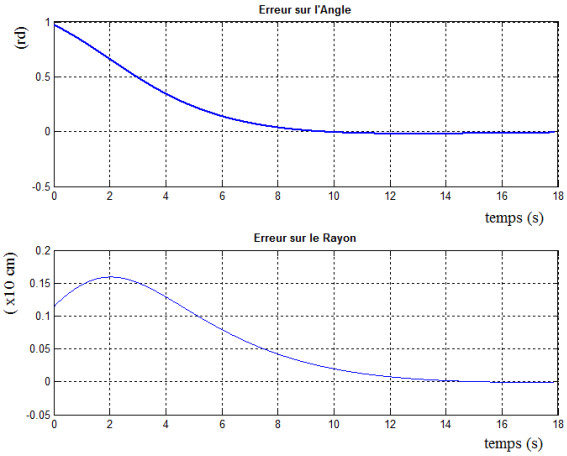

FIGURE 4.19 – Erreurs rayon,angle

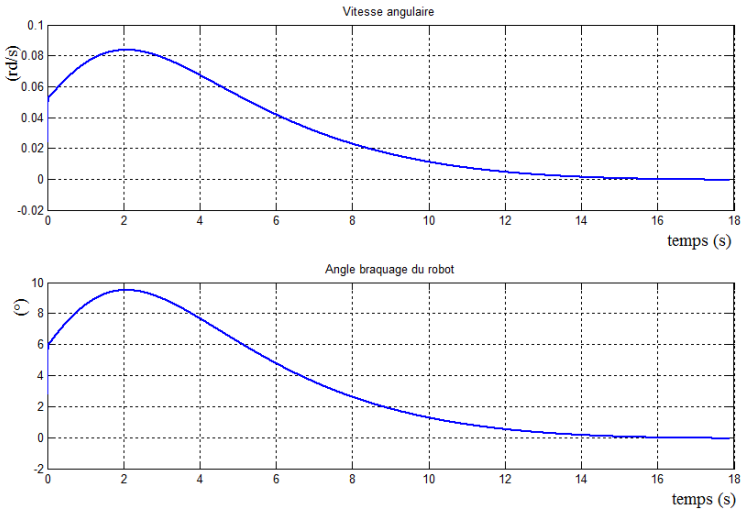

FIGURE 4.20 – Les commandes

Interprétation :

On remarque à partir de ces simulations que le changement de la position initiale n'influe pas sur le suivi du robot de la trajectoire de référence. La seule condition à satisfaire est bien que la droite soit dans le champ de vision de la caméra.

e - Régulateur flou

Les résultats présentés ci–dessous correspondent au régulateur flou synthétisé au chapitre précédent. La position initiale du robot est la même que pour le cas du régulateur PID synthétisé par approche fonction de tâche. Ceci nous permettra de comparer les performances des différentes lois qui ont été utilisées. La figure (Fig : 4.21) montre la trajectoire du robot dans l'espace 2D, la figure (Fig : 4.24) correspond aux valeurs des commandes injectées au robot et enfin la figure (Fig : 4.23) montre l'évolution des erreurs sur le rayon et l'angle de la droite détectée dans l'image.

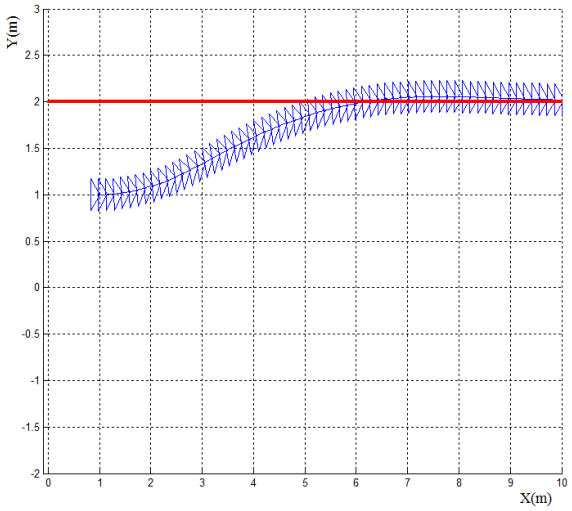

FIGURE 4.21 – La trajectoire du robot

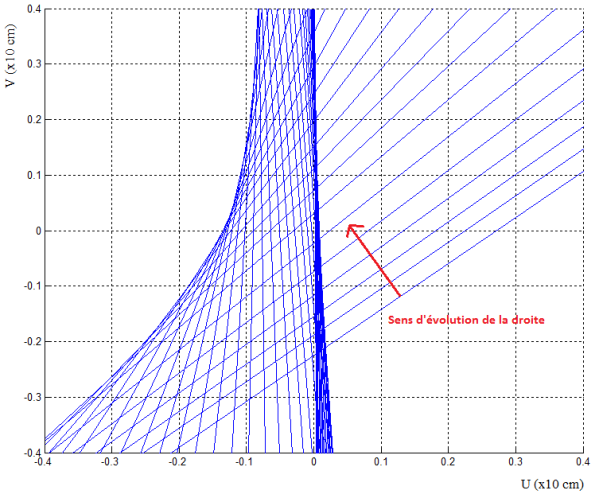

FIGURE 4.22 – L'évolution de la droite

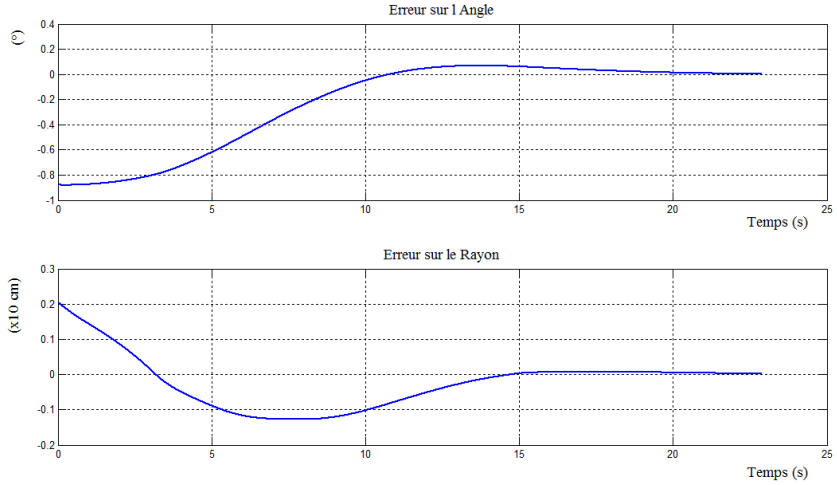

FIGURE 4.23 – Erreurs rayon, angle

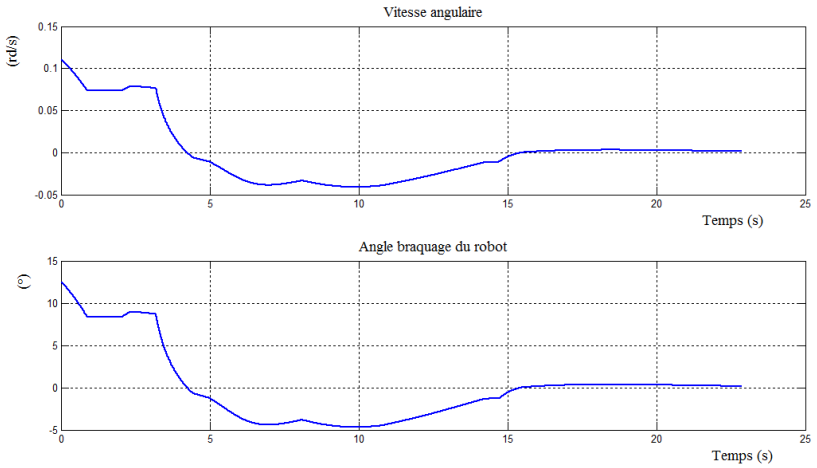

FIGURE 4.24 – Les commandes

Interprétation

La tâche de positionnement du robot en utilisant le régleur flou s'est effectuée en un temps de 18,4 secondes, ce dernier est plus élevé que le temps de convergence obtenu en utilisant le régulateur PID. On remarque aussi que la commande injectée (angle de braquage) présente des allures brusques, ceci est dû au couplage entre les erreurs du rayon et de l'angle de la droite.

4.2.2 Asservissement sur une trajectoire formée par des segments de droites

a - Régulateur PID

Après validation du régulateur PID pour des tâches de positionnement, le but de cette simulation est de tester ce régulateur pour une tâche d'asservissement. La trajectoire sera constituée de segments de droites. Dans ce cas, la pondération privilégie l'erreur sur le cap plutôt que l'erreur sur le rayon, et ceci est dû au fait que la position du robot n'est pas éloignée de sa trajectoire de référence.

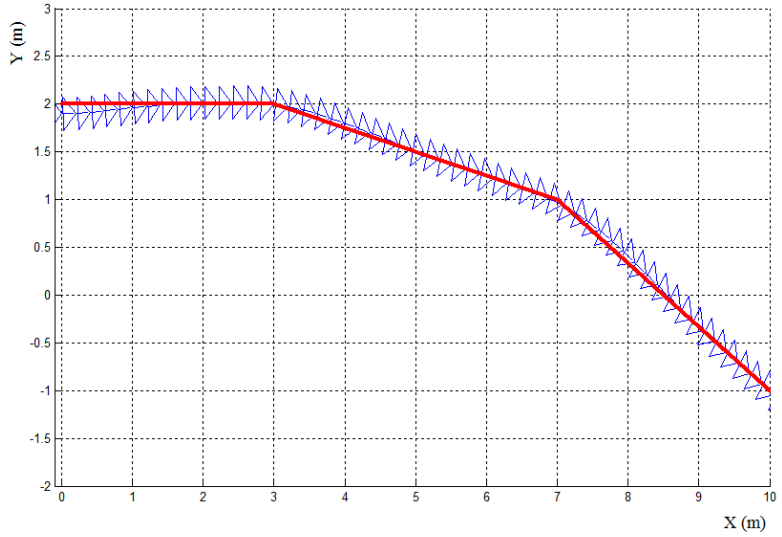

FIGURE 4.25 – La trajectoire du robot

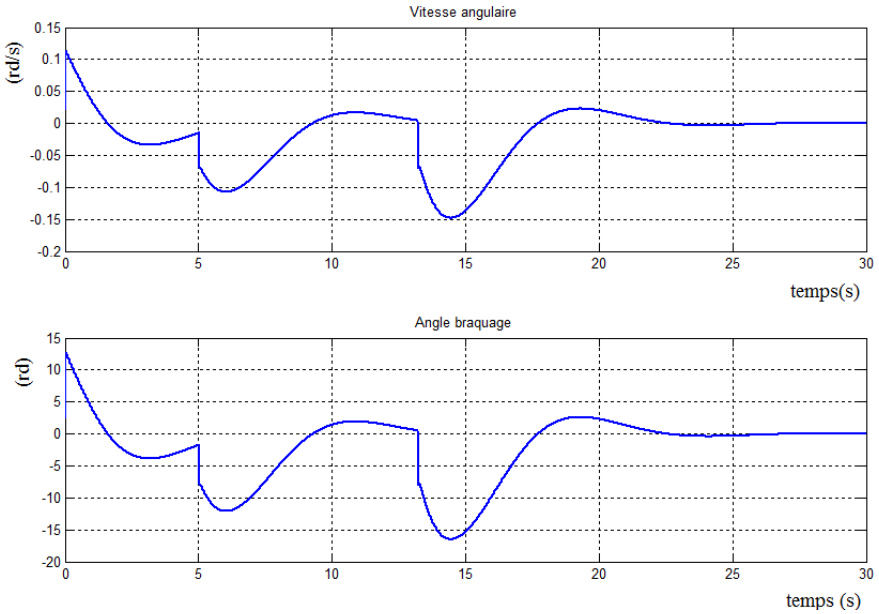

FIGURE 4.26 – Les commandes

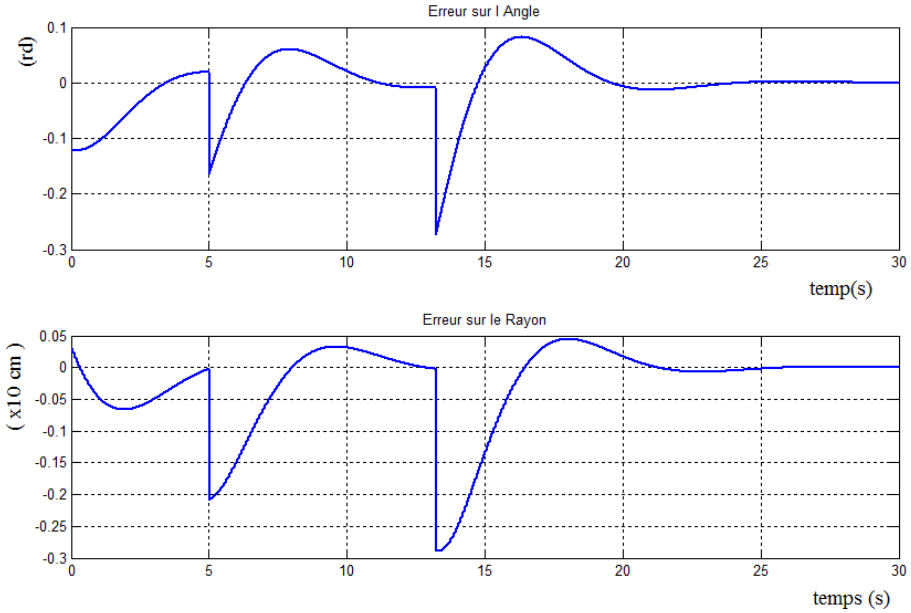

FIGURE 4.27 – Erreurs rayon, angle

Interprétation :

On remarque que le régulateur PID a bien assuré la convergence du robot vers la trajectoire ainsi que le suivi, et cela avec des commandes admissibles (sans saturer les actionneurs).

b - Régulateur flou

Nous allons à présent, pour la même trajectoire composée de segments de droites (Fig : 4.28), tester le régulateur flou et ceci avec une vitesse du robot de 0.5 m/s. La position initiale sera légèrement décalée du premier segment de la droite, et ceci correspondra à une erreur initiale. La figure(Fig : 4.29) montre les valeurs des commandes injectées au robot, la figure (Fig : 4.30) illustre l'évolution des erreurs sur le rayon et l'angle de la droite détectée dans l'image.

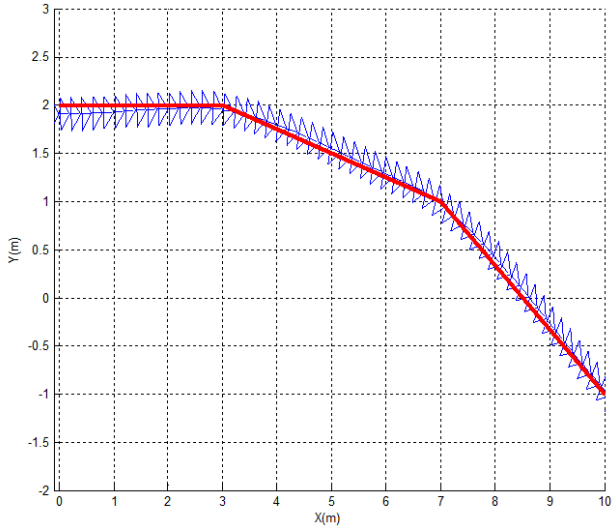

FIGURE 4.28 – La trajectoire du robot

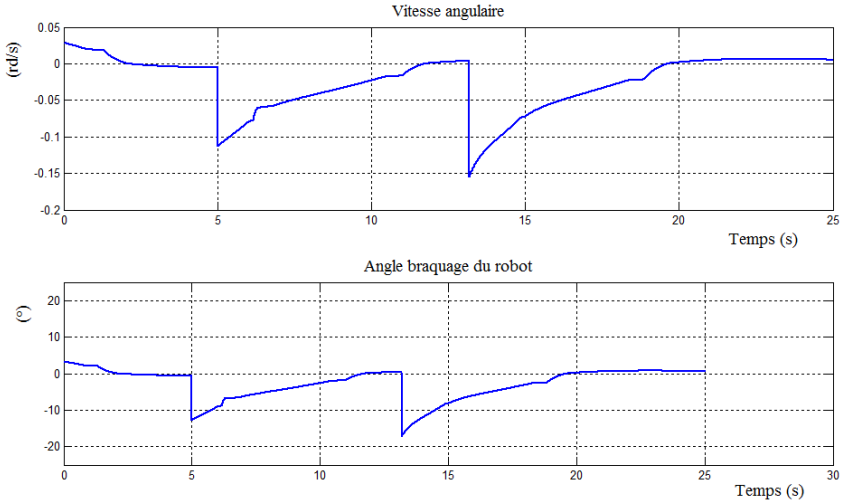

FIGURE 4.29 – Les commandes

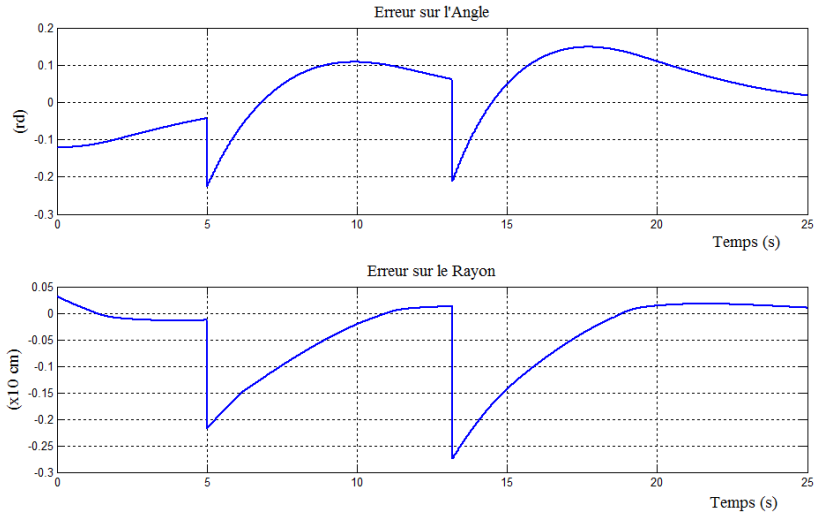

FIGURE 4.30 – Erreurs rayon, angle

Interprétation :

Le régulateur flou a assuré le bon suivi de la trajectoire. On remarque au même titre que le cas du positionnement utilisant ce même type de régulateur, que l'allure de la commande (angle de braquage) n'est pas lisse. L'angle formé entre les segments de droites correspond à un brusque changement des valeurs des erreurs du rayon et de l'angle. La commande réagit donc avec des pics bien visibles sur la figure (Fig : 4.29).

4.2.3 Asservissement sur une trajectoire en forme d'arc de cercle (virage)

a - Régulateur PID

Afin de s'approcher du cas réel des trajectoires que le robot sera amené à suivre, nous allons considérer dans cette simulation un virage approximé par morceaux à des segments de droites. Ce qui nous permettra de valider la loi de commande en vue d'une implémentation réelle.

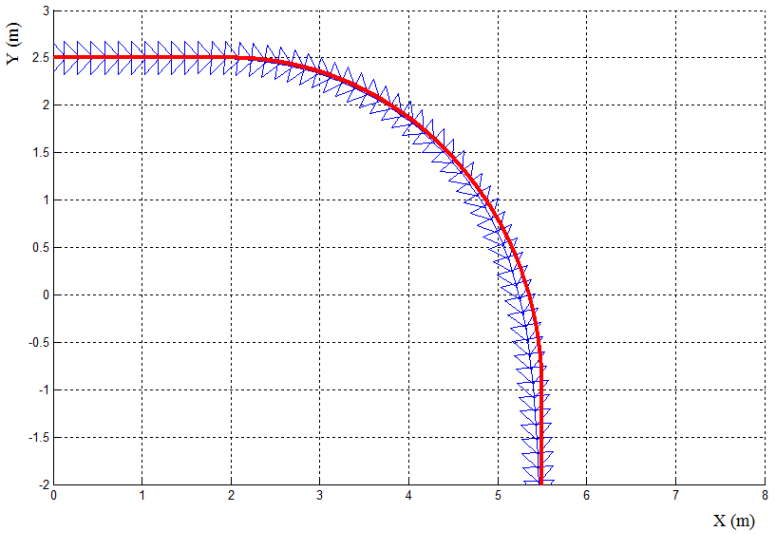

FIGURE 4.31 – La trajectoire du robot

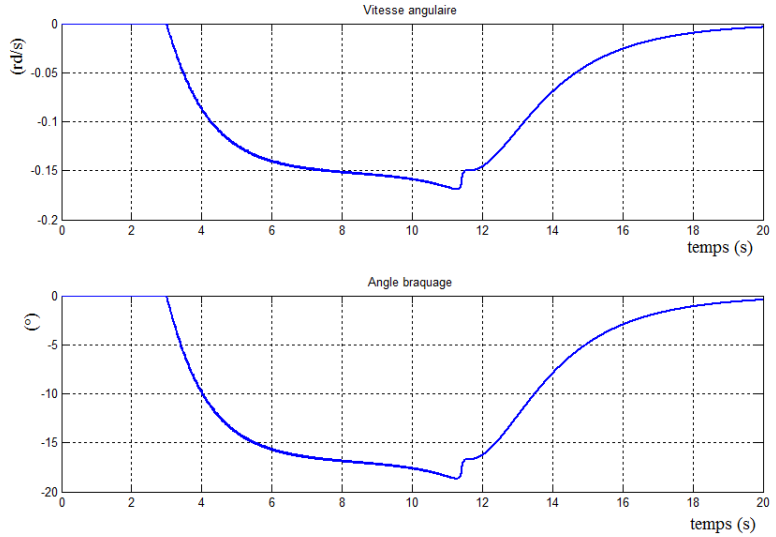

FIGURE 4.32 – Les commandes

Interprétation :

Les résultats de la simulation montre le bon suivi de la trajectoire par le robot. Ceci avec une vitesse fixée à $0.5m/s$, sans saturation des organes du robot (angle de braquage $< 20°$) ce qui est admissible pour le cas du *Robucar*. Ainsi notre loi de commande est complètement validée et pourra donc être testée dans le cas réel.

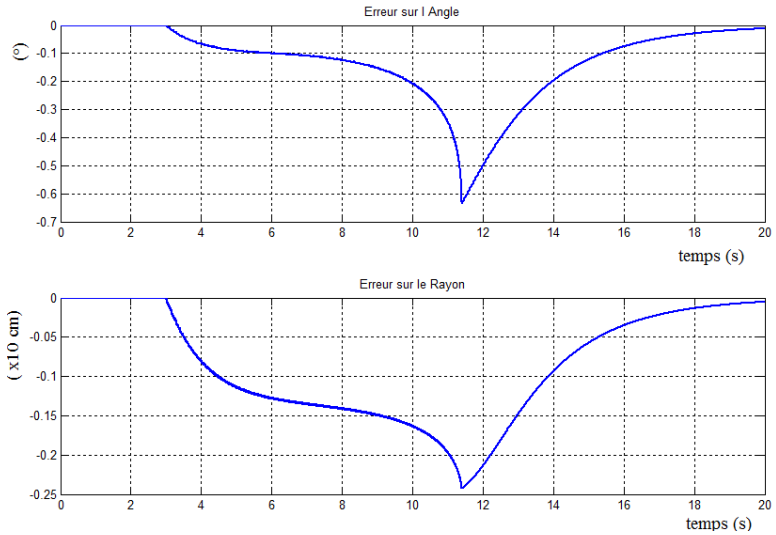

FIGURE 4.33 – Erreurs rayon, angle

b - Régulateur flou

Dans les mêmes conditions du test précédent (trajectoire en forme de virage), le régulateur flou est testé pour le suivi de cette trajectoire.

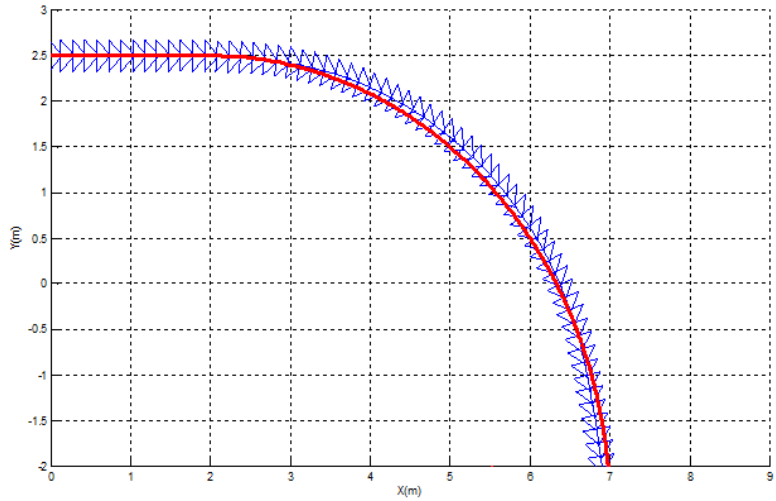

FIGURE 4.34 – La trajectoire du robot

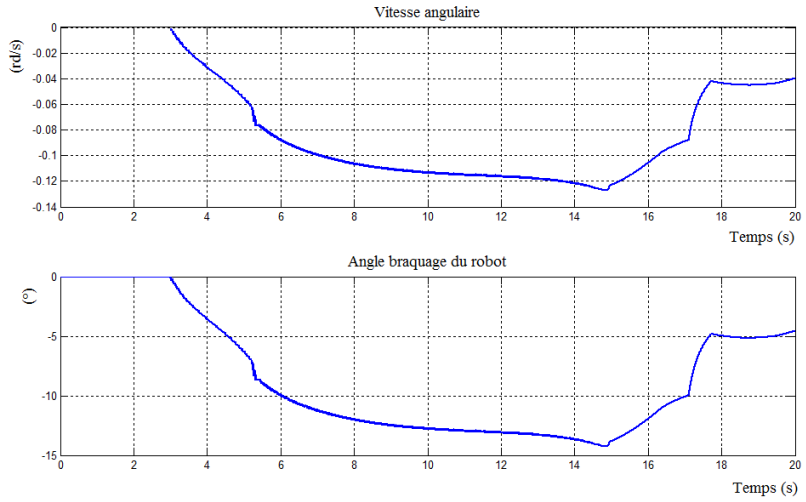

FIGURE 4.35 – Les commandes

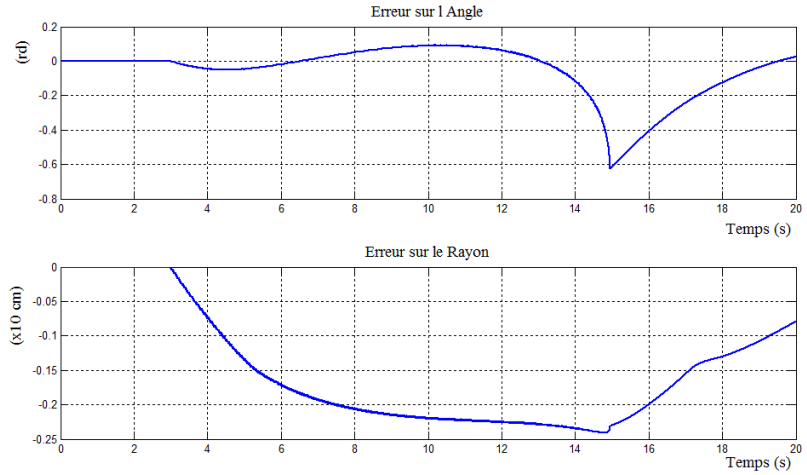

FIGURE 4.36 – Erreurs rayon, angle

Interprétation :

Pour assurer le suivi du virage, le régulateur flou maintient un angle de braquage de $-13°$ (bien visible sur la figure (Fig : 4.35). On remarque que cet angle est inférieur à celui du régulateur PID précédent, ceci se traduit par une erreur de suivi bien visible sur la trajectoire du robot (Fig : 4.34) ainsi que sur le graphe des erreurs sur le rayon et l'angle (Fig : 4.36).

4.2.4 Étude comparative

Après avoir effectué tous ces test sur les déférents régulateurs, ainsi que pour des déférents situations, nous allons maintenant conclure cette partie pour une comparaison entre les régulateurs utilisés , cela nous permettra de dire à la fin quel est le régulateur qui peux satisfaire nos besoins en vue d'une implémentation réelle.

Le tableau ci-dessous résume cette étude.

	Régulateur P	Régulateur PID	Régulateur flou
Temps de convergence	27 secondes	13 secondes	18 secondes
Distance de convergence	14 m	7 m	10 m
Angle de braquage maximal	environ 10 deg	environ 15 deg	environ 15 deg
L'allure de la commande	lisse	lisse	variations brusques

TABLE 4.1 – comparaison du régulateur

A partir de cette étude, nous pouvons déduire que le PID présente plus d'avantages par rapport aux deux autres régulateurs. Ainsi, il sera le régulateur utilisé dans les implémentations réels. Le régulateur flou sera aussi implémenté malgré ses quelques désavantages concernant les variations brusques de la commande et le temps de calcul relativement grand.

4.3 Partie implémentation temps réel

4.3.1 Implémentation sur le « Robucar »

Notre application consiste à asservir le *Robucar* en utilisant des informations visuelles. Le facteur temps est donc un élément très important qu'on cherche à optimiser afin de rendre le robot plus réactif.

Du fait que le PC embarqué est relativement lent (P1 à 133 MHz), un PC portable a été exploité afin de partager les différents traitements de telle sorte à gagner en temps d'exécution. Pour cela une architecture Client/Serveur a été utilisée. Cette architecture va également permettre l'utilisation de bibliothèques de traitement d'images plus performantes. ce qui est impossible sur le PC embarqué à cause de son système d'exploitation qui est une version très ancienne de Redhat. Les deux PC coopèrent pour la réalisation des différentes tâches. Ils sont connectés à un même réseau et leur communication est faite grâce aux sockets en mode connecté en exploitant le protocole TCP/IP. Les tâches qui demandent des temps de traitement élevés seront exécutées sur le serveur, ainsi après avoir acquis l'image par la caméra embarquée, des prétraitements sont nécessaires pour rendre cette image exploitable. Une phase de détection de contours est ensuite réalisée en utilisant le filtre de Deriche, suivie de la sélection de la droite par la transformée de Hough.

Grâce à ces informations, le robot va déterminer l'angle avec lequel il doit tourner pour réaliser le suivi. Le client à son tour, participe à la réalisation des tâches d'asservissement du robot et cela en établissant la connexion au serveur, la réception des données et l'envoi des commandes au niveau bas du robot. A l'exécution des deux programmes (Serveur, Client), ils doivent d'abord se connectés grâce aux sockets (TCP/IP). Chaque image acquise par le client est traitée et les données extraites sont envoyées au client, ce dernier applique les commande reçues sur le robot.

Nous allons exploiter le programme de traitement d'image utilisé pour la détection du bord de la chaussé et développé dans [Boudar 10]. Ainsi nous allons améliorer cet algorithme en l'adaptant à nos besoin.

Les tâches du serveur

Les différentes tâches qui sont réalisées sur le serveur sont :

- Extraction des données visuelles des images (chaine de traitement d'images) :
  - Acquisition de l'image.
  - Prétraitements de l'image.
  - Détections des contours.
  - Détection de la droite
- Calcul de la commande à partir des données extraites ;
- Envoie des commandes au Client ;

Comme les différentes tâches citées sont pratiquement lentes en terme de temps d'exécution à cause de la quantité d'informations à traiter d'une part, et vu que le robot doit naviguer en temps réel d'autre part ; une solution en Multithreads a été utilisée. Les différentes tâches sont donc exécutées en parallèle. La communication entre ces dernières est faite grâce à des zones mémoires partagées, et la synchronisation de l'accès à ces zones est assurée par des sémaphores. Ces threads sont illustrés sur la figure suivante :

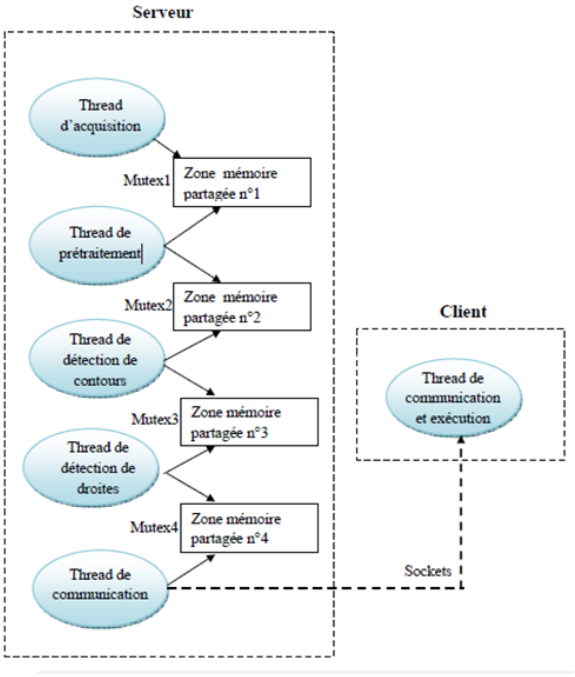

FIGURE 4.37 – Architecture multithreads

L'acquisition des images et leur traitement qui permet l'extraction de la droite sont donc faits sur le serveur. Cinq Threads (qui seront détaillés par la suite) sont exécutés en parallèle et coopèrent pour l'extraction des données représentant les informations visuelles de la droite, ainsi que pour établir la connexion avec le client et l'envoi des données extraites. La communication entre ces threads est réalisée grâce à la mémoire partagée entre ces derniers. La synchronisation de l'accès à la mémoire est assurée grâce aux sémaphores (Mutex1,.., Mutex4).

La chaine de traitement d'image

a - Thread d'acquisition d'images : Notre but consiste à extraire la primitive visuelle de type droite. Pour l'extraction de ces informations, une acquisition d'image est nécessaire. Nous avons donc fixé une caméra (webcam Logitech) sur l'avant du robot.

FIGURE 4.38 – la caméra utilisée

Pour avoir un meilleure vue de la droite, nous avons orienté la caméra avec un angle vers le sol. L'acquisition des images est réalisée grâce à la bibliothèque de traitement d'images OpenCV développée par Intel.

L'image sera donc sauvegardée dans la zone mémoire partagée numéro 1. La synchronisation de l'accès à cette zone est assurée par un sémaphore Mutex1. Le thread d'acquisition s'exécute en boucle jusqu'à la fin du programme principal.

b - Thread de Prétraitement : L'image brute acquise par la caméra est parfois constituée de signaux parasites appelés bruits. Sa qualité peut aussi être détériorée à cause du changement brutal de luminosité, caractéristique d'un environnement d'extérieur. En effet, l'acquisition d'une image dans un endroit trop sombre ou trop lumineux peut réduire sa qualité. Tous ces défauts sur l'image doivent être éliminés lors d'une phase que l'on appelle prétraitement. Les prétraitements permettent la préparation des données reçues du thread d'acquisition à la phase suivante d'analyse consacrée à l'extraction des paramètres. A tout moment le thread de prétraitement accède à la zone mémoire partagée numéro 1 et récupère l'image déposée par le thread d'acquisition. Un prétraitement lui est alors appliqué qui est constitué de deux méthodes, à savoir, l'égalisation et l'élargissement. Les étapes du prétraitement sont les suivantes :

- Calcul des seuils
- Elargissement
- Egalisation

Les images suivantes montrent le résultat des pré-traitements effectués sur l'image :

A la fin des opérations, le thread de prétraitement dépose l'image prétraitée dans la zone mémoire partagée numero 2, la synchronisation de l'accès à cette zone est assurée par un sémaphore Mutex2. Le thread de prétraitement s'exécute en boucle jusqu'à la fin du programme principal.

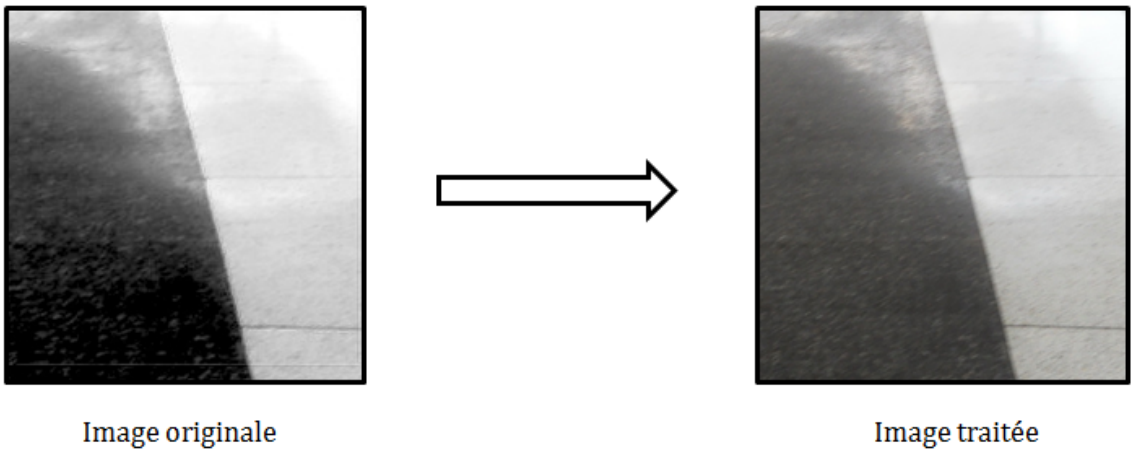

FIGURE 4.39 – Résultat des pré-traitement

c - Thread de détection de contours : La détection de contours, comme nous l'avons déjà défini dans le chapitre 2, peut être obtenue grâce à de multiples méthodes. Celle utilisée est la dérivation par filtrage optimal avec utilisation du filtre de Deriche.

A tout moment le thread de détection de contours accède à la zone mémoire partagée numero 2 et récupère l'image pré-traitée déposée par le thread de pré-traitement.

Le thread de détection de contours applique le filtre de Deriche sur l'image. La figure (Fig : 4.40) montre un exemple d'une image contour.

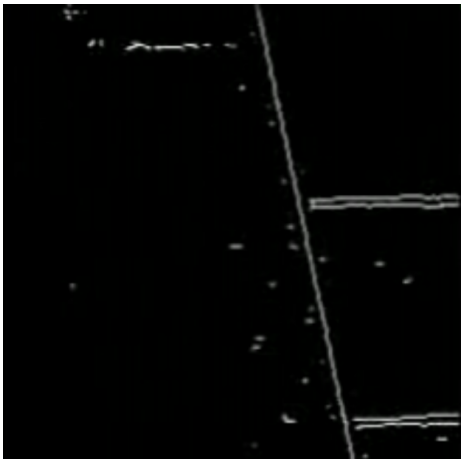

FIGURE 4.40 – Image contour

Vers la fin de l'opération, le thread de détection de contours dépose l'image contour dans la zone mémoire partagée numéro 3. La synchronisation de l'accès à cette zone est assurée par un sémaphore Mutex3.

d - Thread de détection de droites : La transformée de Hough a été utilisé pour la détection de droites dans l'image. C'est un outil puissant dans l'analyse des formes. Notre but est de détecter une droite de paramètres (ρ, θ). Où ρ représente le rayon entre l'origine O et la droite détectée, tel que ρ sera perpendiculaire à la droite. θ l'angle formé entre la droite et l'axe vertical.

Pour cela, le thread de détection de droites commence d'abord par accéder à la zone mémoire partagée numéro 3 et récupérer l'image contour déposée par le thread de détection de contours. Il lui applique un processus de vote qui consiste à détecter les points contours qui appartiennent à la même droite grâce au calcul de ρ en parcourant l'image pixel par pixel pour chaque θ. Une fois que les droites sont extraites, la transformée de Hough détecte les pics pour pouvoir diminuer les fausses droites ou droites sans signification en fixant un seuil.

La figure (Fig : 4.41) représente le résultat de la transformée de Hough appliquée sur une image contour.

FIGURE 4.41 – Droite détectée

Le nombre de droites détectées par la transformée de Hough peut être grand. Notre but est de détecter la droite qui représente la trajectoire à suivire. Pour cela, nous n'avons pris en considération , selon la position de la caméra, que les droites qui ont un angle compris entre une valeur min et max. Quant au seuil (nombre de points qui représente une droite) il a été fixé à 150 après une grande série de tests sur des images.

Le thread de détection de droites s'exécute en boucle jusqu'à la fin du programme principal.

A partir de ρ et θ de la la première droite détectée (considérée comme droite de référence), nos lois de commande consistent à asservir le robot à partir des informations visuelles acquises de chaque droite détectée, ceci afin de suivre la trajectoire définie par ces dernières. la figure (Fig : 4.41) montre un exemple de la position d'une droite détectée (en rouge) par rapport à la droite de référence (en bleu). L'écart entre ces deux droites correspond à une erreur de suivie de la trajectoire par le robot.

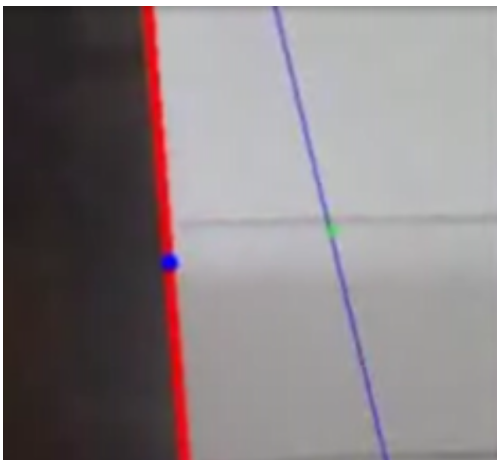

FIGURE 4.42 – Droite détectée et droite de référence

Tâches du client

Pour que le client (PC embarqué) puisse coopérer avec le serveur, un thread de communication est lancé. Celui-ci commence d'abord par établir la connexion entre le client et le serveur, puis récupère les informations qui ont été envoyées par le thread de communication du serveur.

Une fois les informations récupérées, le client doit les appliquer sur le robot, l'injection des commandes reçues s'effectue de manière **graduelle** en communiquant avec le niveau bas du robot.

l'injection de l'angle de braquage se fait de manière graduelle. Sachant que le maximum de l'angle de braquage du robot mobile *Robucar* est de 20 degrés.

4.3.2 Test de l'algorithme de traitement d'image

Dans un premier temps, un essai de l'algorithme de traitement d'image est nécessaire. A travers ce test on déterminera le temps de calcul nécessaire pour la phase de traitement. De plus, les données recueillies (évolution des erreurs sur le rayon et l'angle) seront sauvegardées dans un fichier.

L'environnement d'essai est illustré sur la figure suivante. Il représente un couloir dont le sol est de couleur blanche contenant une bande noire au milieu, ce qui représente un bon environnement de test pour l'algorithme de traitement d'image.

FIGURE 4.43 – L'enveronnement d'essai

Résultat du test :

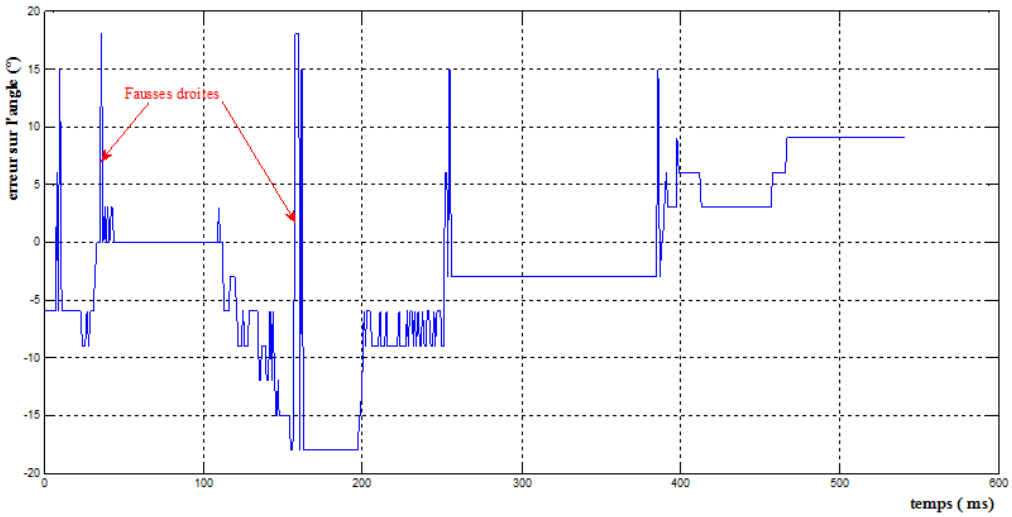

FIGURE 4.44 – Les fausses droites

A travers ce test, on remarque l'existence de forte variation de l'erreur sur l'angle (pics sur la figure Fig :4.44). Ces dernières sont dues à la détection de fausses droites autres que la droite désirée. Ceci va influencer sur le calcul de la commande. De plus la variation de l'angle se fait de manière discontinue avec un pas de 10°. Or une telle variation engendre une forte oscillation de la sortie du régulateur, mais une diminution de cette résolution engendre une augmentation du temps de calcul. La solution consiste donc à trouver un compromis entre la résolution et le temps de calcul. Des tests expérimentaux ont montré qu'une résolution de 3° satisfait ce compromis avec un temps de calcul de $30ms$.

Pour remédier au problème des fausses droites, on a proposé d'ajouter un filtre à la chaine de traitement d'image qui va permettre de lisser les signaux d'erreurs. La figure (Fig :4.45) suivante montre l'intérêt de ce filtre, on remarque bien que la commande de l'angle n'est pas influencée par la présence des fausses droites.

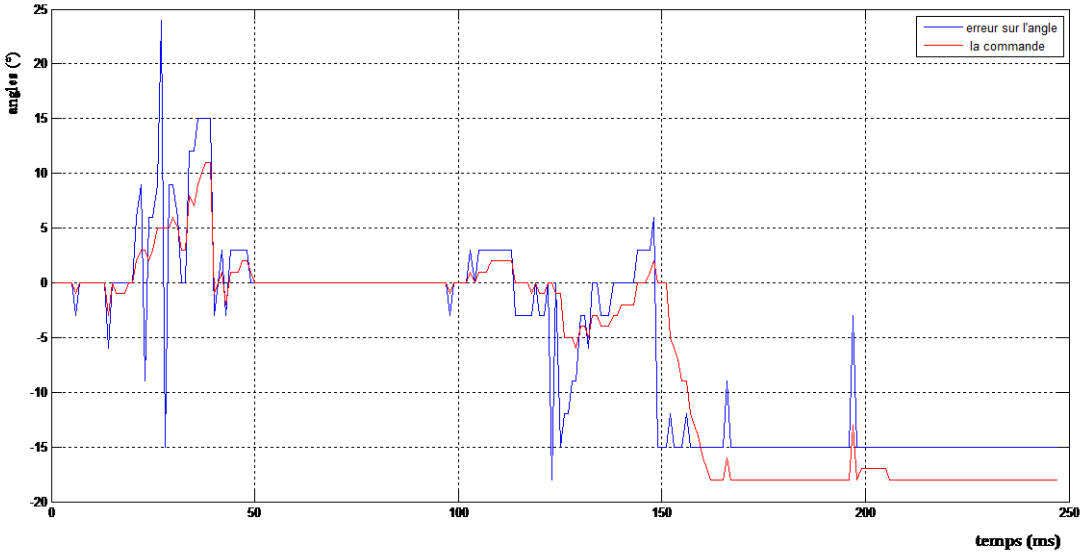

FIGURE 4.45 – Les fausses droites

Le choix de la droite de référence

Le choix de la droite de référence se fait de deux manières différentes selon la tâche à réaliser. Pour une tâche de positionnement, le choix de la droite se fait manuellement en introduisant les valeurs du rayon et de l'angle (ρ, θ).

Pour une tâche d'asservissement, le choix de la droite se fait automatiquement, en prenant comme droite de référence celle qui sera détectée à partir de la position initiale du robot. L'asservissement du robot consistera donc à garder une même configuration celui-ci par rapport à la droite.

L'environnement d'essai étant le couloir précédent, dans lequel le traitement d'image a été testé, nous présentons ci-dessous les résultats des tests pour les différentes commandes implémentées.

4.3.3 Résultats des essais réels sur le Robucar

Essai avec exploitation directe des informations visuelles

Comme nous l'avons précisé dans le chapitre 3, nous avons essayé d'exploiter directement les informations visuelles. Ceci avec un simple régulateur proportionnel qui prend en considération les erreurs sur le rayon et sur l'angle de la droite détectée. La valeurs des gains sont $K_1 = 0.35$, $K_2 = 0.7$. Les figures suivantes montrent les résultats des essais effectués sur le *Robucar* en utilisant ce régulateur.

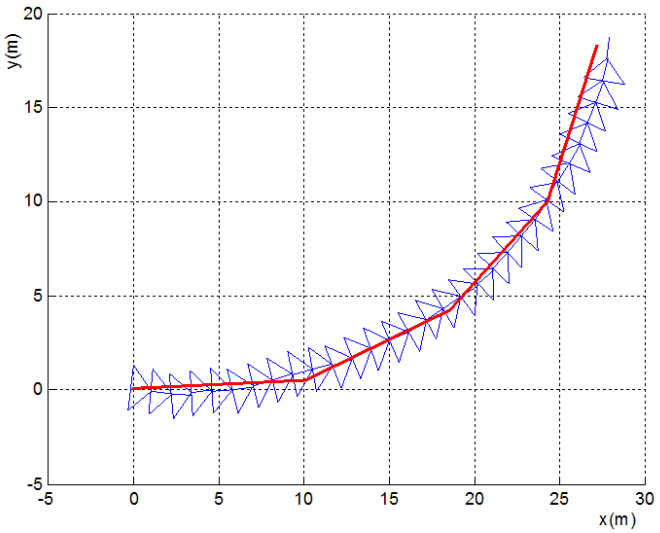

FIGURE 4.46 – La trajectoire du robot

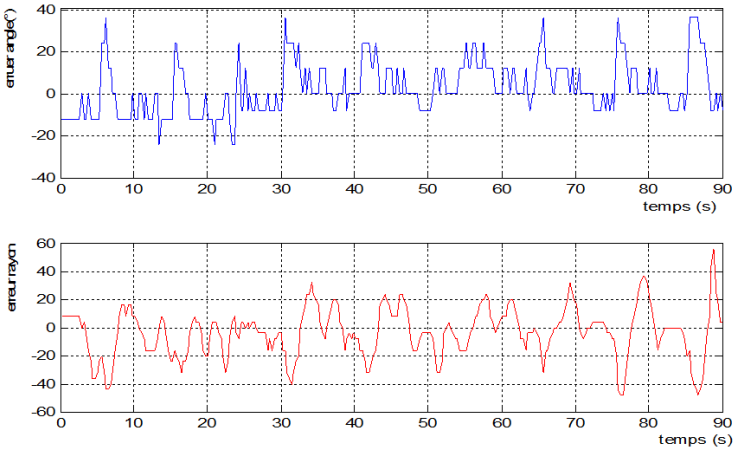

FIGURE 4.47 – Erreurs angle, rayon

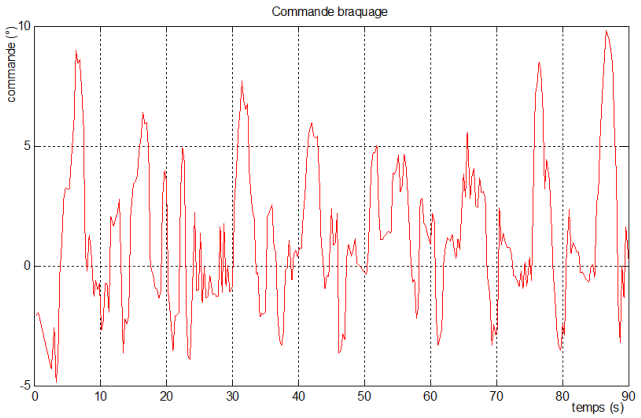

FIGURE 4.48 – La commande de braquage

Interprétation :

A partir des résultats obtenus, on remarque des variations élevées sur les erreurs de l'angle et du rayon, ce qui engendre des variations brusques de la commande. Celles-ci provoquent des oscillations du robot autour de la trajectoire à suivre.

Bien que ce test n'a pas donné de bonnes performances dans le suivi de la trajectoire, il nous a permis de mieux cibler les problèmes de la commande par asservissement visuel, ainsi que d'étudier le comportement du robot et la variation des informations visuelles 2D perçues en fonction de la position du robot par rapport à la droite.

Essais réels de l'asservissement visuel

a - Vitesse moyenne de 0.4 m/s Après validation en simulation de la loi d'asservissement visuel basée sur le formalisme de fonction de tâche, cette dernière est testée en temps réel sur le *Robucar* dans le même environnement (couloir CDTA).

Pour ce premier test, une faible vitesse sera utilisée ($0.4m/s$). Cette vitesse garantie que les informations visuelles (la droite) restent dans le champ de vision du robot.

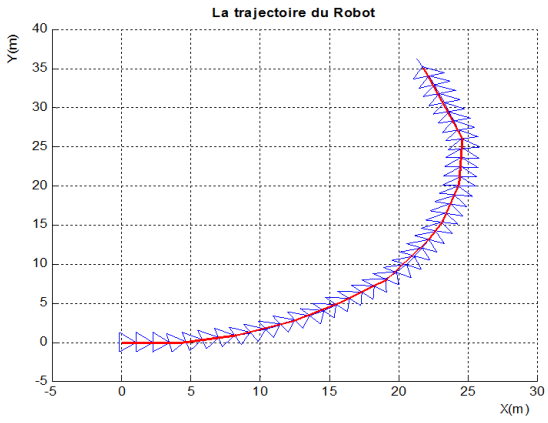

FIGURE 4.49 – La trajectoire du robot

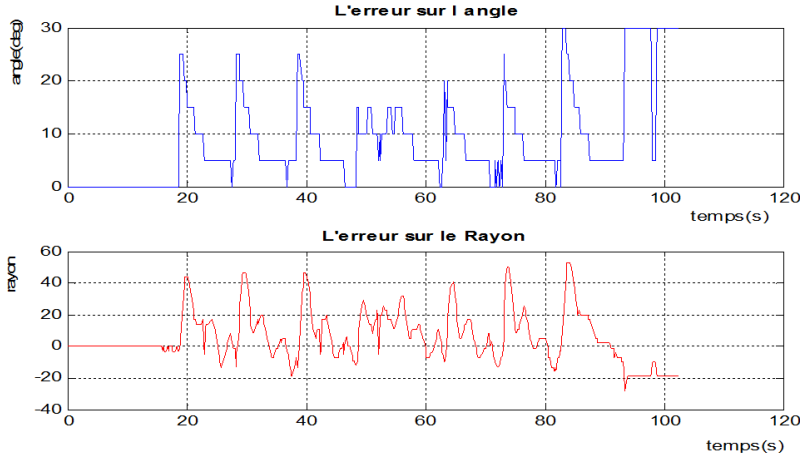

FIGURE 4.50 – Erreurs angle, rayon

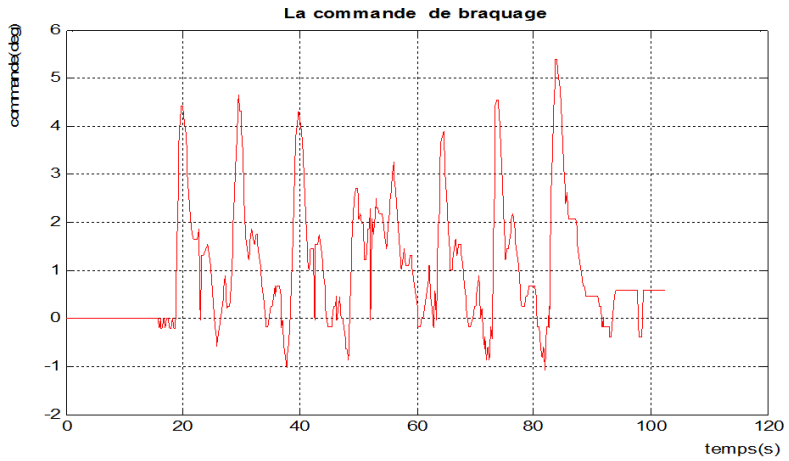

FIGURE 4.51 – La commande de braquage

Interprétation :

On remarque un bon suivi de la trajectoire par le robot comme le montre la figure (Fig : 4.49). Les pics visibles sur la figure des erreurs sur le rayon et l'angle correspondent aux points d'intersection de deux segments de droites sur le sol qui engendre une grande variation des erreurs. Mais ces dernières sont compensées par la commande et ramenées à zéro d'une façon exponentielle en escalier (signal échantillonné). Ces résultats sont similaires à ceux obtenus en simulation.

Les bons résultats obtenus à travers ce test nous incitent à augmenter la vitesse du robot, et d'évaluer ainsi les performances de cette loi de commande.

b - Vitesse moyenne de 0.75 m/s Les conditions de ce test restent similaires à ceux du test précédent. La vitesse du robot a toutefois été augmentée jusqu'à $0.75m/s$. Le sens de parcours de la trajectoire a été aussi inversé.

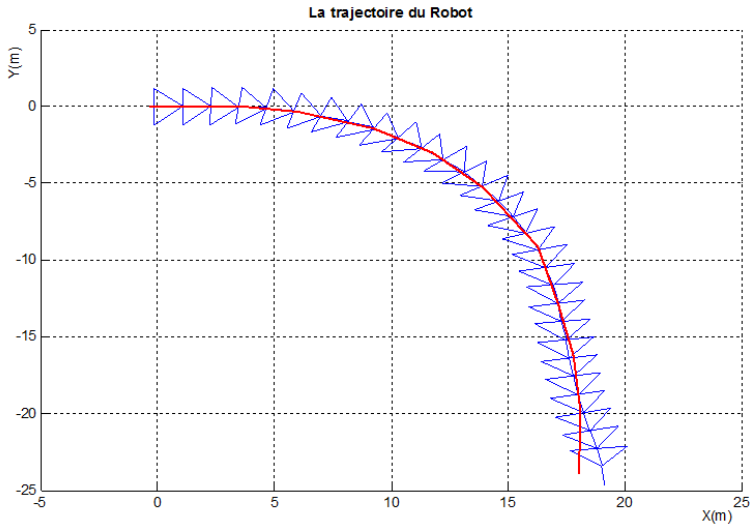

FIGURE 4.52 – La trajectoire du robot

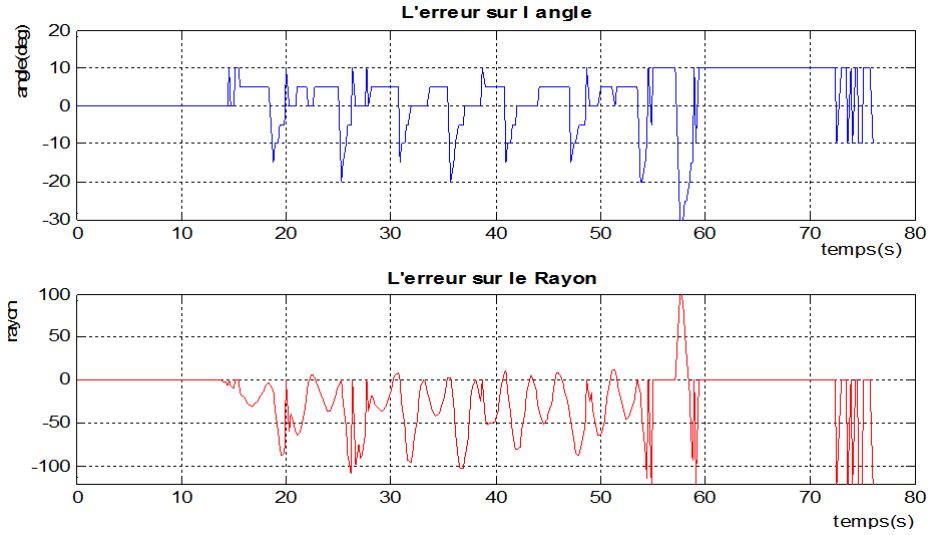

FIGURE 4.53 – Erreurs angle, rayon

Interprétation :

Comme le montre la figure (Fig 4.52), le robot a assuré le bon suivi de la trajectoire malgré l'augmentation de la vitesse. Toute fois pour un angle accentué de la trajectoire, on remarque une perte de la droite dans l'image, visible sur la figure des erreurs (Fig : 4.53) après un temps écoulé de 60 secondes. Cela correspond à une divergence du robot de sa trajectoire au point $y = -20m$.

Pour y remédier, une régulation de la vitesse s'impose, ce sera le point traité au prochain l'essai.

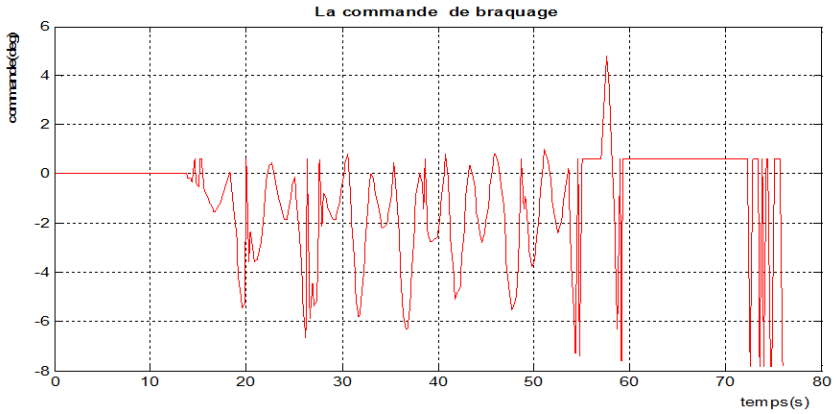

FIGURE 4.54 – La commande de braquage

c - Régulation de vitesse (vmax 1 m/s) Pour éviter la perte de la droite dans l'image pour des trajectoires avec une forte courbure, une régulation de la vitesse est nécessaire. Ainsi la vitesse sera inversement proportionnelle aux erreurs. Ceci permettra au robot d'évoluer à vitesse maximale lorsqu'il est bien asservi et à vitesse minimale en abordant les courbures et virages ou lors d'une perte des informations visuelles.

La formule donnant la vitesse en fonction des erreurs est donnée par :

$$V = V_{max}–(V_{min} - V_{max})(norm(e)/MAX(norm(e))) \tag{4.1}$$

Ou $norm(e)$ représente la norme des erreurs de fonctions de tâche.

A présent, nous allons coupler le régulateur précédemment testé avec la régulation de vitesse, la plage de variation de la vitesse du robot sera comprise entre 0.5 et 1 m/s. Les résultats des essais réels sont présentés sur les figures suivantes :

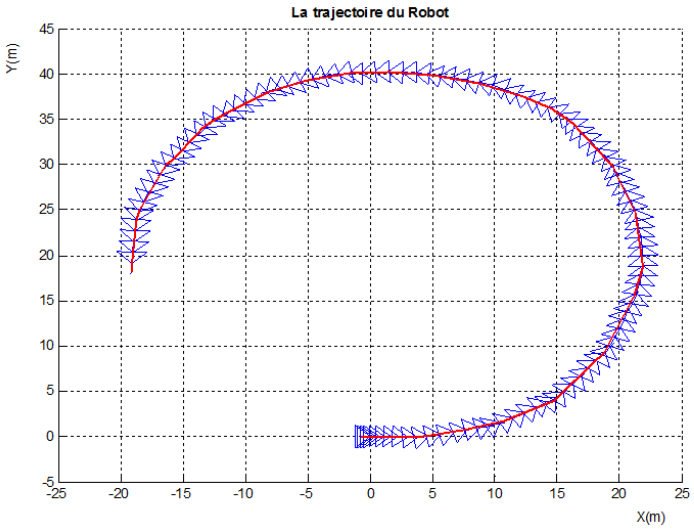

FIGURE 4.55 – La trajectoire du robot

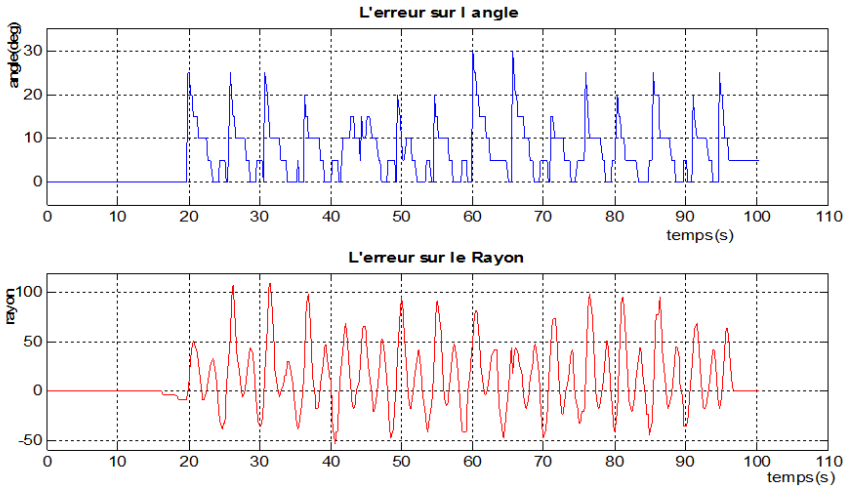

FIGURE 4.56 – Erreurs angle, rayon

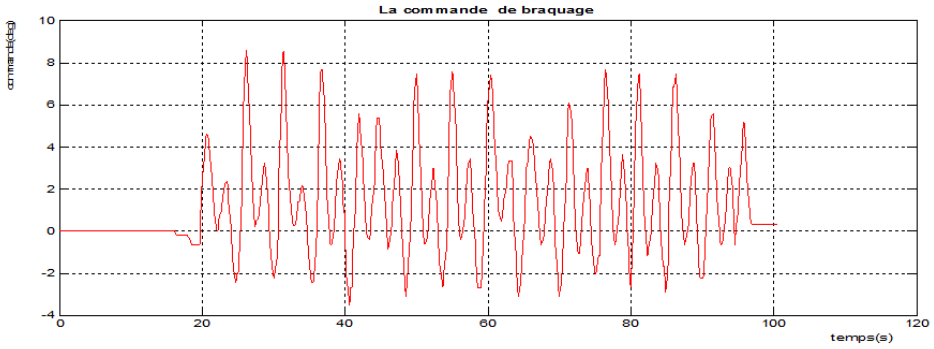

FIGURE 4.57 – La commande de braquage

Interprétation :

Comme le montre la figure (Fig : 4.55), on remarque le bon suivi de la trajectoire par le robot. Ce dernier a parcourue toute la longueur de la trajectoire qui est en forme de cercle. Malgré des vitesses atteignant $1m/s$, aucune perte des informations visuelles n'a été engendrée, ceci grâce à la régulation de la vitesse qui a assuré la bonne stabilité du robot.

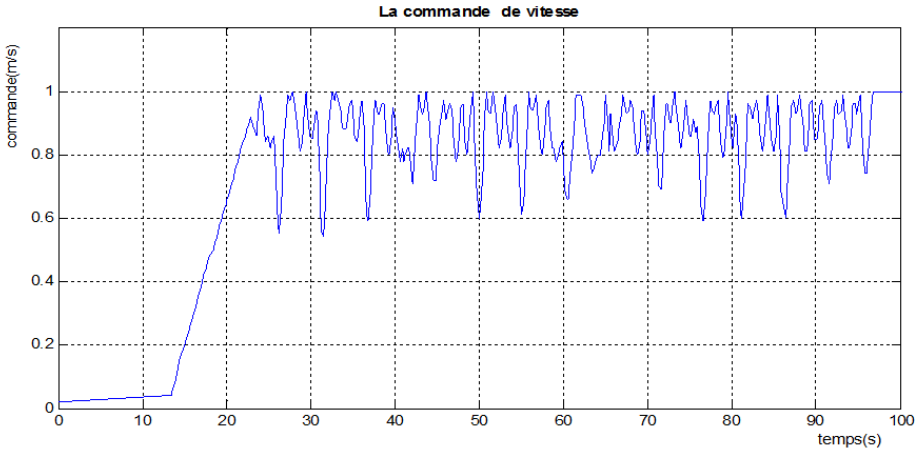

FIGURE 4.58 – La commande de vitesse

d - Essai avec régulateur flou

Après validation du régulateur précédent, nous allons à présent tester notre régulateur flou. Pour des raisons d'optimisation du temps de calcul, c'est le type TSK qui sera implémenté. Ce dernier, comparé à la méthode du centre de gravité, nécessite des temps de calcul moindres en plus de sa facilité à être implémenté. Les résultats des essais sont présentés sur les figures suivantes :

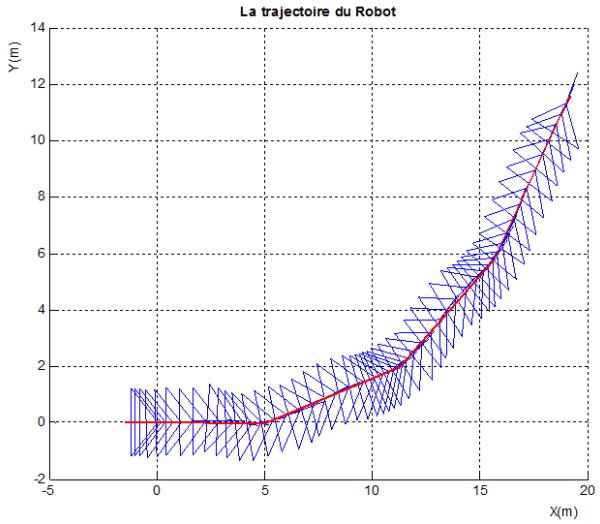

FIGURE 4.59 – La trajectoire du robot

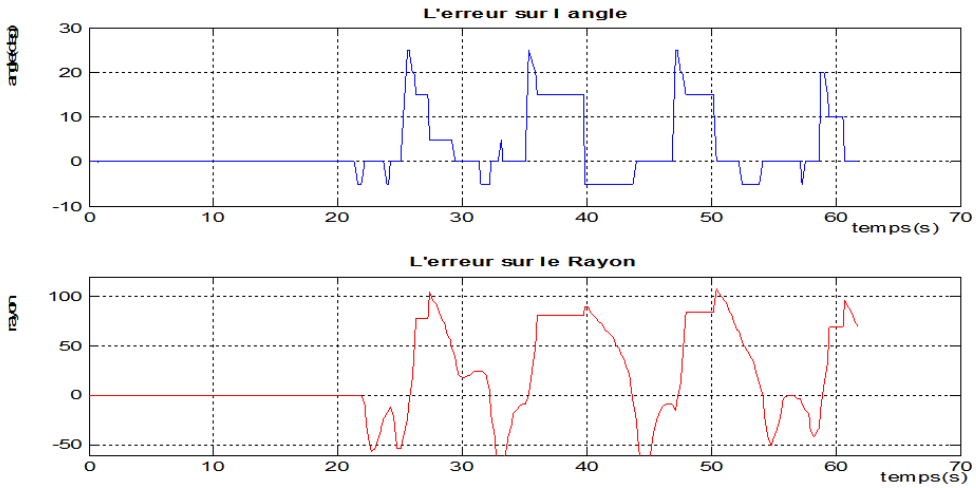

FIGURE 4.60 – Erreurs angle, rayon

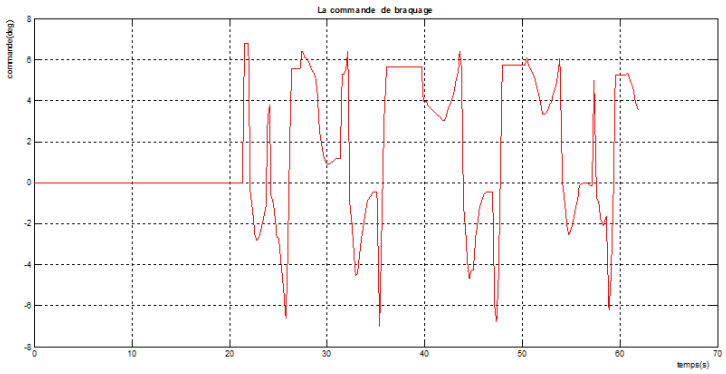

FIGURE 4.61 – La commande de braquage

Interprétation :

Le suivi de la trajectoire s'est effectué avec des oscillations bien visibles sur la figure (Fig :4.59). Le profil de la commande varie d'une façon assez brusque, ceci s'explique par le fait qu'il faudra enrichir la base des règles.

e - Essai en milieu externe

Nous allons maintenant procéder à un essai dans un environnement extérieur représenté sur la figure (Fig :4.62). La trajectoire de référence est représentée par un fil, ce dernier nous permettra d'envisager des lignes droites et des courbures (virages).

Cet essai nous permettra d'évaluer non seulement les performances de notre loi de commande, mais aussi la robustesse de la chaine de traitement d'image. Notons que la loi de commande qui sera utilisée est le régulateur PID. Avec une régulation de vitesse variant dans une plage de 0.6 à 1 m/s.

FIGURE 4.62 – La trajectoire du robot

Les différents résultats obtenus sont représentés sur les figures suivantes :

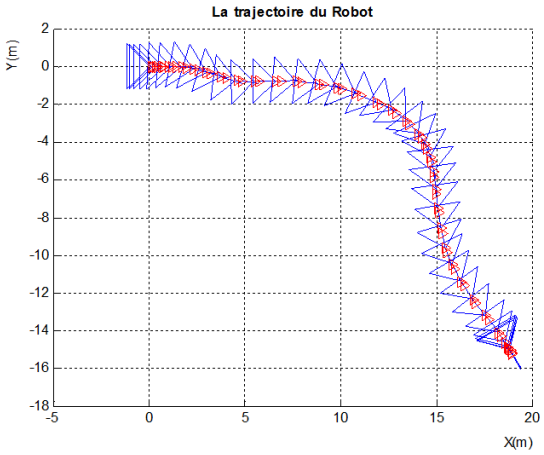

FIGURE 4.63 – La trajectoire du robot

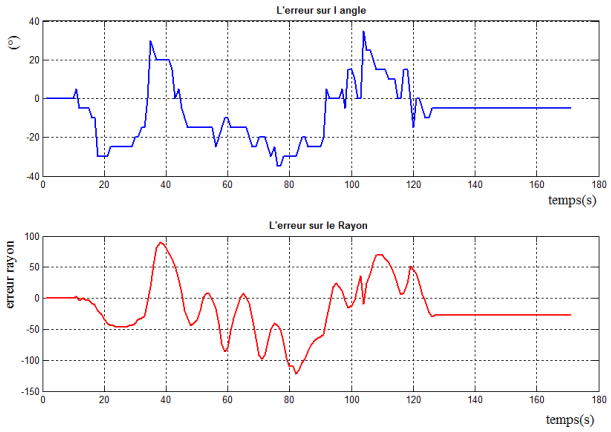

FIGURE 4.64 – Erreurs (rayon, angle)

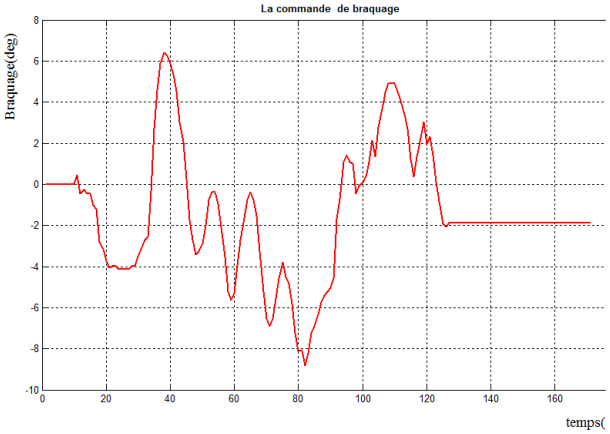

FIGURE 4.65 – La commande de braquage

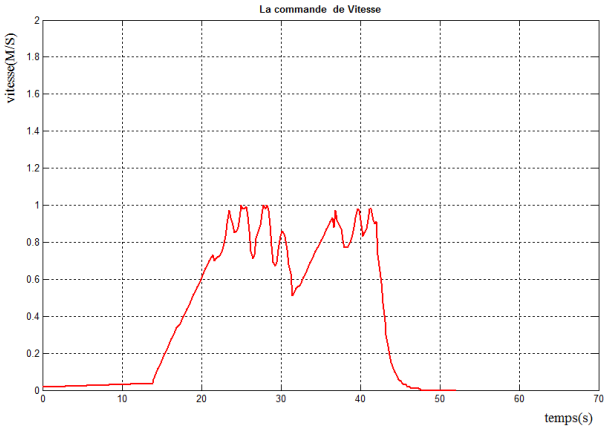

FIGURE 4.66 – La commande de la vitesse)

Interprétation :

Nous remarquons que le robot a bien effectué la tâche de suivie comme l'illustre la figure (Fig : 4.63). Les commandes injectées ont été bien inférieures aux valeurs de saturation (angle de braquage inférieur à 10°). De plus, malgré les perturbations liées à l'environnement extérieur (Ensoleillement, sol irrégulier...etc) auxquelles notre système est soumis , nous pouvons noter la robustesse de notre loi de commande, ainsi que du programme de traitement d'image.

4.4 Conclusion

Ce chapitre a été consacré aux résultats des tests de lois de commande précédemment synthétisées. En effet, après leur validation en simulation, celles-ci ont été implémentées sur le *Robucar*. Dans ce cadre, nous avons pu tiré les performances ainsi que les limitations de ces lois de commande.

Conclusion générale et perspectives

Dans le présent mémoire, nous nous sommes intéressés à la problématique de la commande référencée vision du robot mobile *Robucar* qui est de tye voiture. Ce dernier constitue une plateforme expérimentale de la division Productique et Robotique du *Centre de Développement des Technologies Avancées* (CDTA). Ainsi, nous nous somme focalisés plus précisément sur une application d'asservissement visuel 2D de ce robot mobile. Ceci par rapport à une trajectoire assimilée à des segments de droites.

Dans le premier chapitre de ce travail, nous avons, en premier lieu, rappelé des notions générales mais essentielles sur les robots mobiles autonomes. Ainsi, nous avons bien expliqué leurs aspects matériels et logiciels. Pour ensuite donner un aperçu général sur la perception en robotique mobile en citant les différents capteurs utilisés. Un autre point essentiel abordé est la navigation en robotique mobile. Une tâche qui permet au robot d'évoluer de manière autonome dans son environnement, et cela sans assistance. Dans ce contexte, les différentes stratégies de navigation ont été présentées, tout en détaillant les architectures de contrôle permettant au robot mobile de réaliser les tâches de navigation.

En second lieu, nous avons présenté le robot mobile *Robucar*, sujet de notre étude. Nous avons donc commencé par une présentation générale de celui-ci, ainsi que ces caractéristiques techniques, sa modélisation et son architecture de contrôle qui inclue différents modules intégrés.

Le chapitre 2 a été consacré à un état de l'art sur la commande référencée vision. Pour ce faire, nous avons commencé par des généralités sur la vision en robotique mobile. Ceci avec un rappel sur la notion d'image et les différentes techniques utilisées pour leurs traitements. Par la suite, nous avons détaillé les différentes techniques d'asservissement visuel, notamment l'AV 3D, AV D2, AV 2D 1/2 et AV d2D/dt. En effet, pour chaque technique nous avons ainsi précisé son contexte d'utilisation, ses avantages et ses limitations. A la fin de ce chapitre, nous avons approfondi l'étude du cas de l'asservissement visuel 2D basée sur le formalisme de la fonction de tâche. Cette dernière constitue en effet, la principale approche que nous avions adopté pour la synthèse de nos lois de commande dans le chapitre suivant.

Dans le chapitre 3, nous avons, dans un premier temps, essayé d'exploiter directement les données visuelles issues de la caméra pour commander l'angle de braquage du robot mobile « *Robucar* », et cela pour effectuer le suivi de la trajectoire. Les résultats de ces essais nous ont permis d'apercevoir rapidement les limites d'une telle approche, en effet, aucune prise en compte de l'interaction entre les mouvements du robot et ceux des primitives visuelles n'a été effectuée dans ce cas, ce qui nous a donc montré la nécessité d'une étude théorique approfondie dans le domaine

de l'asservissement visuel basée sur la notion de formalisme de fonction de tâche que nous avons détaillé dans le chapitre précédent.

Une modélisation complète et rigoureuse de notre plateforme robotique a donc été nécessaire. Ceci, pour aborder l'interaction entre les mouvements des primitives visuelles dans l'image et ceux de la caméra du robot. Cela nous a permis la synthétise de nos lois de commande, en se basant sur le formalisme de fonction de tâche.

Une seconde loi de commande a été développée à la fin de ce chapitre. Celle-ci est basée sur les principes de la logique floue. L'expertise acquise dans le cadre des différents tests sur le robot nous ont permis la construction de nos bases de règles ainsi que des fonctions d'appartenance utilisées.

Enfin, dans le chapitre 4, après avoir présenter l'environnement de simulation (*MATLAB*) que nous avons adopté, nous avons d'abord validé nos lois de commande pour des tâches de positionnement, ainsi que des taches d'asservissement sur des trajectoires assimilées par morceaux à des segments de droites.

Une fois nos lois validées en simulation, ces dernières ont été implémentées en temps réel réel sur le *Robucar*. Pour ce faire nous avons ainsi présenté la chaine de traitement d'image utilisée pour l'extraction des primitives visuelles. Ainsi que l'environnement dans lesquels les essais réels ont été effectués.

Les résultats obtenus dans le cadre de notre travail ont été significatifs vis-à-vis des applications considérées. Ils ont notamment permis de doter le *Robucar* de nouvelles fonctionnalités lui permettant le suivi de trajectoire.

Plusieurs perspectives peuvent se dégager de notre travail, on peut citer :

- L'utilisation de deux capteurs de vision permettant l'obtention d'informations encore plus riche. D'autres techniques d'asservissement visuel (2D 1/2, d2D/dt..) peuvent être ainsi employées.
- Dans notre cas, nous avons traité un seul type de primitives visuelles qui est une droite. Une amélioration du programme de traitement d'image pour la détection de primitives plus complexes, à savoir des cercles, ellipses et courbes permettra le développement de tâches robotiques encore plus évoluées.
- Extension de nos résultats pour des applications permettant la détection des voies de circulation dans le but d'une conduite automatiques des véhicules dans des milieu de type autoroute.
- Étant donné les résultats prometteurs obtenu en utilisant le régulateur flou, l'enrichissement des bases des règles de ce dernier à travers l'expertise acquise va améliorer considérablement les résultats.
- La modélisation du lien entre l'espace du capteur et l'espace de configuration du Robucar dans le cadre de notre travail a permis le développement de lois dans l'espace capteur. Le problème de commande étant bien définit d'autres lois de commande robustes peuvent être envisagées (modes glissents, H_∞ ...).

Bibliographie

[Abdou 97] S. Abdou, *"Spécification, Vérification et Implémentation de Missions Appliquées à des Véhicules Automatiques"*,Thèse de Doctorat, Institut National Polytechnique de Grenoble, Grenoble, France, 1997.

[Aknin 02] P. Aknin, *"Capteurs et traitement du signal dans les transports guidés"*, Editions Hermès/Lavoisier, 2002.

[Alendre et al 06] T. Alendre & N. Carayon & S. Soissons. *"Egalisation d'histogramme"*. Projet Télécom, Paris 2006.

[Andreff 07] N. Andreff, F. Chaumette, *"Quoi de neuf en asservissement visuel depuis les JNRR'03 ?"*, Journées Nationales de la Recherche en Robotique, France, 2007.

[Bayle 09] B. Bayle, *"Support de cours (Robotique mobile)"*, Ecole Nationale Supérieure de Physique de Strasbourg, Université de Louis Pasteur, 2009.

[Bayle 11] B. Bayle, *"Robotique mobile"*, Ecole Nationale Supérieure de Physique de Strasbourg, Université de Louis Pasteur, 2011.

[Belaid 91] A. Belaid, *" Reconnaissance des formes"*, Edition InterEditions, 1991.

[Benseddik 11] M.L Benseddik, *"Commande et navigation visuel d'un robot à roues"*, Thèse de Magistère, Ecole Nationale Polytechnique, Alger, Algérie, 2011.

[Boucher] A. Boucher. *" Cours de vision par ordinateur"*. Institut de la Francophonie pour l'Informatique (IFI).

[Boudar 10] O. Boudar, *"Contribution au développement d'un système de détection des bords de la chaussée par vision monoculaire pour un robot mobile autonome "*, Projet de fin d'étude, Université des Sciences et de la Technologie Houari Boumediene, Alger, Algérie, 2010.

[Cabodevila 08] G. Cabodevila, *"Automatique linéaire échantillonnée"*, Support de cours, École Nationale Supérieure de Mécanique et de Micro-technique, 2008.

[Cadenat 99] V. Cadenat, *"Commande référencée multi-capteurs pour la navigation d'un robot mobile"*, Thèse de Doctorat, Université Paul SABATIER, Toulouse, France, 1999.

[Chaumette 90] F. Chaumette, *"La relation vision-commande : théorie et application des tâches robotiques"*, Thèse de Doctorat, Université de Rennes I, Rennes, France, 1990.

[Chaumette 98] F. Chaumette, *"De la perception à l'action : l'asservissement visuel, de l'action à la perception : la vision active"*, Habilitation à diriger des recherches, Université de Rennes I, Rennes, France, 1998.

[Cocquerez et Philip 95] J. P. Cocquerez et Philip. *"Analyse d'images : Filtrage et segmentation"*, Edition Masson, 1995.

[Coiffet 86] P. Coiffet, *"La robotique - Principes et applications – Robots"*, Traité des Nouvelles Technologies, Série Robotique, 1986.

[Debain 96] C. Debain, *"Lois de commande pour le contrôle et la mobilité de machines agricoles"*, Thèse de Doctorat, Université Blaise Pascal - Clermont II, Clermont-Ferrand, France, 1996.

[Djekoune 98] A. O Djekoune, *" vision dynamique "*, Rapport d'activité, 1998.

[Dufourd 05] D. Dufourd,*"Des cartes combinatoires pour la construction automatique de modèles d'environnement par un robot mobile"*, Thèse de doctorat, Institut National Polytechnique de Toulouse, 2005.

[Duval 03] T. Duval,*"Robotique Mobile, 68hc11 Et Os Dédié"*, Edition DUNOD, 2003.

[Elougli 09] A. Elougli, *"Intégration des techniques floues à la synthèse de contrôleurs adaptatifs "*, Thèse de doctorat, Université de Sidi Mouhamed. Fès, 2009.

[Filliat 11] D. Filliat, *"Support de cours (Robotique mobile)"*, Ecole Nationale Supérieure des Techniques Avancées (ENSTA) ParisTech, 2011.

[Folio 07] D. Folio, *"Stratégies de commande référencées multi-capteurs et gestion de la perte du signal visuel pour la navigation d'un robot mobile"*, Thèse de Doctorat, Université Paul SABATIER, Toulouse, France, 2007.

[Gangloff 04] J. Gangloff, *"Asservissements visuels"*, Notes de cours, Université de Strasbourg, Strasbourg, France, 2004.

[Guechi 10] El-Hadi Guechi,*"Suivi de trajectoires d'un robot mobile non holonome : approche par modèle flou de Takagi-Sugeno et prise en compte des retards"*, Thèse de doctorat, Université de Valenciennes et du Hainaut Cambrésis, France, 2010.

[Haralick et Paindavoinee 00] Haralick M. Paindavoinee, *"Traitement des images en temps réel "*, Technique de l'ingénieur, Traité mesures de contrôle, R6720, 2000.

[Harchaoui 08] W. Harchaoui, *" Débruitage d'images "*, Cours traitement d'images, Février 2008.

[Jeanpierre 04] L. Jeanpierre, *"Navigation d'un robot mobile"*, Rapport d'activité, 2004.

[Khadraoui 96] D. Khadraoui, *"La commande référencée vision pour le guidage automatique de véhicules"*, Thèse de Doctorat, Université Blaise Pascal - Clermont II, Clermont-Ferrand, France, 1996.

[Large 03] F. Large , *"Navigation autonome d'un robot mobile en environnement dynamique et incertain "*, Thèse de doctorat, Université de Savoie, 2003.

[Laumond 01] J. Laumond , *"La robotique mobile"*,Edition Hermès, 2001.

[Lefebvre 06] O. Lefebvre, *"Navigation autonome sans collision pour robots mobiles nonholonomes"*, Thèse de doctorat, Institut National Polytechnique de Toulouse, 2006.

[Lerond 06] D. Lerond, A. Mathieu, *"La photo numérique"*, 2eme Edition Micro application, 2006.

[Malis 98] E. Malis, *"Contribution à la modélisation et à la commande en asservissement visuel"*, Thèse de Doctorat, Université de Rennes I, Rennes, France, 1998.

[Map-toul] http ://www.map.toulouse.archi.fr, *" Support de cours"*, Ecole Nationale Supérieure d'Architecture de Toulouse, 2002.

[Marey 08] M. Marey, F. Chaumette, *"Analyse de lois de commande d'asservissement visuel"*, IEE int. Conf on Robotics and Automation, ICRA'08, Pasadena, CA, Mai 2008.

[Martinet 99] P. Martinet, *"Asservissement visuel"*, Habilitation à diriger des recherches, Université Blaise Pascal - Clermont II, Clermont-Ferrand, France, 1996.

[Mudry 06] F. Mudry, *"Ajustage du paramètres d'un régulateur PID "*, Notes de cours, Institut d'Automatisation Industriel,Canton de Vaud, 2006.

[Ouadah 04] N. Ouadah, *"Implémentation d'un régulateur floue pour la navigation d'un robot mobile de type voiture"*, Rapport de recherche CDTA, Centre de Développement des Technologies Avancées. Alger, Algérie, 2004.

[Ouadah 11] N. Ouadah, *"Le contrôle visuel appliqué dans la robotique mobile"*, Thèse de Doctorat, Ecole Nationale Polytechnique, Alger, Algérie, 2011.

[Parrain 00] F. Parrain et B. Charlot, *"The TIMA LAB, Research Reports"*,TIMA, Gronoble, France, 2000.

[Pichon 91] J. Pichon, *"Guidage visuel d'un robot mobile autonome d'inspiration bionique"*, Thèse de doctorat de l'Institut National Polytechnique de Grenoble, France, 1991.

[Pruski 96] A. Pruski,*"robotique mobile, la planification de trajectoire"*, Techniques de l'ingénieur, Traité mesure et contrôle, R7850, 1996.

[Roger 07] M. Roger, *" Traitement de l'image Détection de droite "*, Rapport de stage, 2007.

[Rouilly 00] V.Rouilly, *" Détection d'ellipses par transformation de Hough "*, http ://www.tsi.enst.fr/, 2000.

[Ufrima 07] Ufrima, *" Détection de points d'intérêts "*, http ://devernay.free.fr/, 2007.