République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Ecole Nationale Polytechnique

DEPARTEMENT DE GENIE ELECTRIQUE
Laboratoire de Commande des Processus

MEMOIRE DE MAGISTER

Spécialité : **Automatique**

Option **: Robotique et Productique**

Présenté par

Samir BENBELKACEM

Ingénieur d'Etat en Automatique de l'Université de Tizi-Ouzou

Thème

Stabilisation de systèmes non-holonomes par asservissement visuel : cas des robots mobiles à roues

Soutenu publiquement le 15/02/2009 devant le jury composé de :

F. BOUDJEMA	Professeur à l'E.N.P	Président
M. TADJINE	Professeur à l'E.N.P	Rapporteur
H. CHEKIREB	Maître de conférences à l'ENP	Examinateur
B. HEMICI	Maître de conférences à l'ENP	Examinateur
O. STIHI	Chargé de cours à l'ENP	Examinateur

Laboratoire de Commande des Processus, ENP
10, Avenue Hassen Badi, BP-182, 16200 El Harrach, Alger, Algérie

Dieu merci

Je remercie Dieu, le tout puissant, pour m'avoir donné la santé, le courage, la patience, la volonté et la force nécessaire, pour affronter toutes les difficultés et les obstacles, qui se sont hissés au travers de mon chemin, durant toutes mes années d'études.

Dédicaces

Je dédie ce travail à mes très chers parents, je leurs remercie pour leurs sacrifices, leurs patiences, leurs soutien, l'aide et les encouragements qui m'ont apporté durant toutes ces années d'étude. Sans eux, je ne serais pas ce que je suis aujourd'hui.

Je dédie ce travail également :
A la mémoire de mes regrettés grands parents.
A ma grand mère.
A mon frère.
A tous les membres de ma famille.
A tous mes frères et amis
A toute personne qui porte de l'estime pour moi

Remerciements

Le travail que nous présentons dans ce mémoire a été effectué au Laboratoire de Commande de Processus sous la direction de Monsieur Mohamed TADJINE Professeur à l'Ecole Polytechnique d'Alger.

Je tiens à remercier très vivement Monsieur Mohamed TADJINE pour avoir accompagné mon travail durant mon mémoire. Les judicieux conseils qu'ils ma prodigués, son enthousiasme envers mon travail ainsi que sa grande disponibilité m'ont permis de progresser dans mes études et d'achever ce travail dans les meilleures conditions. Je tiens à lui exprimer toute mon amitié et ma reconnaissance.

Qu'il me soit ensuite permis de remercier très vivement Monsieur Hachemi CHEKIREB Maître de Conférence à l'ENP, Monsieur Boualem HEMICI Maître de Conférence à l'ENP et Monsieur Omar STIHI Chargé de cours à l'ENP pour l'honneur qu'ils m'ont fait en acceptant d'être les examinateurs de ce mémoire. Je les remercie de l'intérêt qu'ils ont montré pour mes travaux.

Que Monsieur Farès BOUDJEMA, Professeur à l'ENP, reçoive l'expression de ma très vive reconnaissance d'avoir accepté de présider ce jury.

Je tiens également à adresser mes sincères reconnaissances à Monsieur O. DJEKOUNE d'avoir mis à ma disposition la documentation nécessaire et de m'avoir orienté pendant l'élaboration de ce travail. Je tiens également à adresser mes sincères remerciements à Monsieur M. HAMERLAIN, Monsieur M. BELHOCINE et Madame N. ZENATI-HENDA pour l'aide et les encouragements qui m'ont apporté.

Je tiens également à adresser mes sincères remerciements à mon amis S. Malek pour sa précieuse aide pour le tirage de ce mémoire. Je souhaite aussi dire un grand merci à tous mes amis pour leur soutien.

Table des matières

Chapitre IV Stabilisation de trajectoire pour un robot mobile

Chapitre V Poursuite de trajectoire stabilisante par vision

Table des figures

ملخص
هذه المذكرة تتناول مراقبة النظم الغيرهولونوميك. بهذا الصدد، َيصُب اهتمامنا على تثبيت جهاز آلي متحرك أحادي الدور و ذلك باستعمال وسائل التحكم الغير خطية و كذا تقنيات الرؤية بواسطة الحاسوب.
اقترحنا في بادئ الأمر خوارزميين لتثبيت الجهاز المتحرك. الأول يشمل تطبيق دارة مغلقة ذات وقت متقطع ثابت على الشكل المتسلسل للنموذج الحركي للجهاز المتحرك. بالنِسبة للخوارزم الثاني، يتعلق الأمر بتقنِية الدارة المغلقة الخطية التي تقوم على مبدأ إقامة إغلاق غير خطي على مستوى النموذج الحركي للجهاز. يتم التحكم في هذه الدارة فيما بعد عن طريق دارة أخرى مغلقة(رجوع الحالة) ذات وقت متقطع ثابت.
طرق التحكم المقترَحة تمكننا من معرفة مسار التثبيت ِللآلة. هذه المعلومة تستخدم في مرحلة التجارب للجهاز المتحرك حيث يتم رسم المسار على الأرض ثم يطلب من الجهاز الآلي تتبع هذا المسار من أجل الوصول إلى مكان التوازن. خلال عملية التتبع، يقوم الجهاز بعملية مُعالجة الصوَر وتحديد نظام التحكم.

كلمات مفاتيح:نظام التحكم الغير خطي، التثبيت بواسطة الدارة المُغلقة، نظام غير هولونومي على الشكل المتسلسل، دارة مغلقة ذات وقت متقطع ثابت، تقنية الدارة المغلقة الخطية، الرؤية بواسطة الحاسوب، معالجة الصوَر.

Résumé
Ce mémoire de magistère est une contribution à la commande des systèmes non-holonomes. Nous nous sommes intéressés à la stabilisation d'un robot mobile de type unicyle en utilisant les méthodes de commande automatique non linéaires et les techniques de vision par ordinateur.
Nous avons, en premier lieu, proposé deux algorithmes de stabilisation. Il s'agit de la commande par retour d'état discontinu appliquée à la forme chaînée du modèle cinématique du robot, et de la commande par retour d'état basée sur la technique « feedback linearisation » où le principe consiste à établir un bouclage non linéaire qui linéarise le modèle du robot. Ce bouclage est ensuite commandé par un retour d'état discontinu.
Les commandes précédentes nous permettent d'obtenir la trajectoire stabilisante. Cette information est utilisée dans l'étape d'expérimentation du robot. Il s'agit de tracer cette trajectoire sur le sol et de demander au robot d'effectuer une opération de poursuite pour atteindre la position d'équilibre. Pendant la poursuite, des opérations de traitement d'images sont effectuées pour identifier la trajectoire et ainsi commander le robot. Dans ce cadre, des algorithmes de traitement d'images et de commande ont été développés et testés sur le robot.

Mots clés : Commande de systèmes non linéaires, stabilisation en boucle fermée, système non-holonome en forme chaînée, commande par retour d'état discontinu invariant, commande par feedback linearisation, vision par ordinateur, traitement d'images.

Abstract
This work deals with the control of non-holonomic systems. We are interested to the stabilization of unicycle mobile robot by using the non-linear control methods and the computer vision techniques.
We propose, at first, two stabilization algorithms. The first one is the discontinuous time-invariant state feedback control applied to the chained form of the robot kinematic model. The second algorithm is the feedback linearization technique. This last establishes a non-linear feedback which linearize the model of the robot. This resulting linearized system is controlled by discontinuous state-feedback.
The previous controllers provide the stabilizing path. This information is used in experimentation test for the mobile robot. In this case, we represent this path on the floor and we ask the robot to follow this path in order to reach a desired static configuration starting from any initial configuration. To allow the mobile robot to track this trajectory, many image processing operations are performed to identify the trajectory. Thus, the control of the robot is applied. In this case, the image processing algorithms and the control techniques have been developed and tested on the robot.

Keywords: Non linear control systems, feedback stabilization, nonholonomic chained system, discontinuous time-invariant state feedback control, feedback linearization control, computer vision, images processing.

Notations et définitions

Notations générales

– $\Re$: ensemble des nombres réels.

– $\Re^+$: ensemble des nombres réels positifs ou nuls.

– $\Re^{+*}$: ensemble des nombres réels strictement positifs.

– $\Re^n$: espace vectoriel de dimension n construit sur le corps des réels.

– N : ensemble des nombres entiers naturels.

– V : voisinage non vide de l'origine dans $\Re^n$.

– [a, b] : intervalle fermé de $\Re$ d'extrémités a et b.

– (a, b) : intervalle ouvert de $\Re$ d'extrémités a et b.

– [a, b) : intervalle semi–ouvert de $\Re$ d'extrémités a et b.

– $C_0(E, F)$: ensemble des fonctions continues de E dans F.

– $C_k(E, F)$: ensemble des fonctions de classe k de E dans F.

– $R_o = (O, i, j, k)$: repère fixe de l'espace lié au point O et de base (i, j, k).

– $t \in \Re$: variable temporelle.

– $t_{initial} \in \Re$: instant initial fixé. Pour les systèmes stationnaires $t_{initial} = 0$.

– $t_{final} \in \Re$: instant final.

– $\dot{x} = \frac{dx}{dt}$: dérivée de la variable x par rapport au temps.

– $\dot{x} = \frac{d^2x}{dt^2}$: seconde dérivée de la variable x par rapport au temps.

– $x^{(j)} = \frac{d^j x}{dt^j}$: j^{ème} dérivée de la variable x par rapport au temps.

– x^T : transposé du vecteur x.

– $x \in \Re^n$: vecteur de composantes x_j.

– $\cdot \wedge \cdot$: produit vectoriel de deux champs de vecteurs.

– $|.|$: valeur absolue d'un nombre réel.

– $||.||$: norme euclidienne sur $\Re^n$.

– det(A) : déterminant de la matrice A.

– $||A||$: norme euclidienne de la matrice A.

– I_n : matrice identité de $\Re^{n\times n}$.

– 0_n : matrice nulle de $\Re^{n\times n}$.

– $\frac{\partial f(x)}{\partial x}$: dérivée partielle de *f* par rapport à *x*

– Soient *f(x)* et *g(x)* des champs de vecteurs de dimension *n*, suffisamment différentiables. Le produit de Lie ou crochet de Lie de *f(x)* et *g(x)* et l'opérateur *ad* sont définis par :

$$[f,g](x) = \frac{\partial g}{\partial x}(x).f(x) - \frac{\partial f}{\partial x}(x).g(x)$$
$$ad_f^k g(x) = [f, ad_f^{k-1} g], k \geq 1$$

avec $ad_f^0 g(x) = g(x)$.

– Soit une fonction réelle différentiable *λ(x)*. En notant $d\lambda(x) = \left(\frac{\partial \lambda}{\partial x_1},...,\frac{\partial \lambda}{\partial x_n}\right)$, la dérivée de λ le long de *f* est donnée par :

$$L_f \lambda(x) = d\lambda(x).f(x) = \sum_{i=1}^{n} \frac{\partial \lambda}{\partial x_i} f_i(x)$$
$$L_f^k \lambda(x) = dL_f^{k-1}\lambda(x).f(x)$$

avec $L_f^0 \lambda(x) = \lambda(x)$

Notations spécifiques au robot

– $u_i \subset \Re^m$: ensemble des commandes admissibles.

– $q_i \in$ Qi : vecteur d'état.

– $q_i(t_{initial}) = q_{i,initial}$: configuration d'état initiale.

– $q_i(t_{final}) = q_{i,final}$: configuration d'état finale.

– $u_i(t_{initial}) = u_{i,initial}$: commande initiale.

– $u_i(t_{final}) = u_{i,final}$: commande finale.

– $f_i : \Re^n \times \Re^m \rightarrow \Re^n$: application décrivant le modèle cinématique du robot.

– $(x_i, y_i, \theta_i) \in \Re^3$: coordonnées du robot.

– *v*, *w* : commande en translation et en rotation du robot

– v_p : vitesse linéaire en un point p

Introduction générale

Introduction générale

Un robot est une machine équipée de capacités de perception, de décision et d'action qui lui permet d'agir d'une façon autonome dans son environnement en fonction de la perception qu'il en a.

La conception des véhicules automatisés, ou robots mobiles, à roues est un domaine de recherche en pleine expansion. Ces robots sont utilisés dans l'industrie comme moyen de transport, d'inspection ou d'opération, et sont particulièrement adaptés à des interventions en environnement hostile [BAY 08], [FIL 04], [VEN 97] : les zones confinées, les construction élevées, les fonds sous-marins, etc.

Un tel cadre d'utilisation requiert que le système robotisé dispose d'un niveau minimum d'autonomie et de facilité de navigation. Pour ce faire, le système doit généralement accomplir trois tâches de base qui sont la localisation, la planification et la navigation. Un robot, doté de capacité de perception et d'information sur son environnement, doit pouvoir se mouvoir en autonomie.

Les robots mobiles à roues sont caractérisés par leur simplicité de conception et de commande. Cependant, ces systèmes ont soulevé plusieurs problèmes difficiles qui ont fait l'objet de plusieurs travaux de recherche. Il s'agit des problèmes de commande, de planification [DEF 07], [JIA 99], [KAB 04], [LAV 06] et de poursuite de trajectoires.

Commander un robot mobile (considéré comme un système non-holonome) pour l'amener d'une configuration de départ à une configuration d'arrivée n'est pas une chose aisée. Le problème a suscité un regain d'intérêt des automaticiens pour la robotique. Dans ce cadre, la commande non linéaire trouvait un champ applicatif considérable permettant de satisfaire un certain besoin lié à là poursuite. Un des problèmes majeurs en ce qui concerne la commande des systèmes non-holonomes [VER 94] est l'existence d'un retour d'état assurant une stabilisation du système [MUR 93], [WEI 67], c'est-à-dire passer d'une configuration initiale à une configuration finale (l'origine).

Pour remédier à ce problème, des méthodes de synthèse de lois de commandes ont été proposées. Elles sont ensuite implémentées dans le robot pour assurer son déplacement. Cependant, cette démarche a montré certaines limites lorsqu'il s'agit de faire interagir le robot avec un milieu inconnu [WEC 98]. Le robot, dans ce cas, se déplace sans avoir suffisamment d'informations sur l'environnement qui l'entoure. Pour cette raison, d'autres techniques ont été proposées. Il s'agit de stabiliser le robot en faisant appel aux méthodes de vision par ordinateur. Le fondement de cette méthode et de permettre au robot, muni d'une caméra, de suivre une trajectoire définie au préalable pour atteindre la position finale. Cette trajectoire est obtenue par le biais des techniques de la commande automatique. Dans cette optique, nous nous sommes intéressés à la problématique de poursuite de trajectoire par vision. Ainsi, deux volets se dégagent de cette problématique. Le premier, concerne l'établissement de lois de commande permettant d'établir la trajectoire (stabilisante) du robot. Le deuxième volet consiste à demander au robot de suivre cette trajectoire en faisant appel aux techniques de traitement d'images et de calibrage de caméra.

Concernant le premier volet, plusieurs méthodes de contrôle ont été proposées. Parmi ces méthodes, on rencontre les retours d'états variant dans le temps, des méthodes basées sur la plénitude, ou encore des techniques récursives. Il est important de noter que peu de commandes assurent une stabilisation rapide du système. Les techniques établies sont théoriquement complexes à développer et difficilement applicables aux robots. Au cours de ces dernières années, d'autres méthodes ont été mises en oeuvre. Pour des raisons de simplicité, elles supposent que le modèle cinématique du robot peut être transformé en un modèle plus simplifié et plus facile à commander. Parmi les modèles qui ont connu un grand succès dans la stabilisation de robots, citons : la forme chaînée, la forme normale,… [ZHO 92]. Ainsi, dans ce travail, nous nous intéressons à la commande de systèmes non-holonomes utilisant des modèles simplifiés dans un objectif de stabilisation.

Pour le deuxième volet, avec l'avènement des robots mobiles capables de percevoir leur environnement et de réagir à celui-ci a fait faire un bond en avant aux théories et techniques de la perception [VAG 93], [VAS 04]. Parmi celles-ci, la vision s'est affirmée d'emblée comme un domaine de recherche privilégié. Des progrès de la technique est née la vision par ordinateur dans laquelle vient s'intégrer le traitement d'images. Ce dernier présente un domaine de recherche très prometteur en robotique mobile. Dans cette optique, nous avons décidé de contrôler notre système en faisant appel aux techniques de traitement d'images. Par

le biais d'une caméra, le but est de permettre au robot de suivre une trajectoire stabilisante tracée sur le sol.

Les recherches présentées dans ce mémoire s'inscrivent dans le cadre de la stabilisation de robot mobile par vision. Dans ce cas, le mémoire que nous présentons est décomposé en deux parties. La première est consacrée à l'état de l'art sur la robotique mobile et la vision par ordinateur ainsi que la problématique de commande de robots. La deuxième partie contient nos contributions.

Plus précisément, la première partie compte trois chapitres. Le chapitre 1 est consacré à la définition des différentes catégories de robots mobiles ainsi que leur modélisation cinématique.

Dans le deuxième chapitre, nous établissons notre problématique. Il s'agit de parler de la stratégie de commande des robots mobiles pour assurer leur déplacement d'une position initiale à une position finale. Nous prenons en compte le cas de la stabilisation où la position finale représente l'origine.

Cette partie s'achève avec le chapitre 3 consacré aux techniques de vision utilisées pour la robotique mobile. Nous présentons, dans un premier temps, les méthodes de base de traitement d'images. Ensuite, nous parlons des techniques de calibrage de caméra qui constituent une étape essentielle dans la vision.

Dans le quatrième chapitre, nous introduisons deux approches de commande stabilisante appliquées à notre robot de type unicycle. Il s'agit de la commande par retour d'état discontinu en forme chaînée et la commande par retour d'état basée sur la technique « feedback linearization ». Pour chacune de ces méthodes nous présentons les différents résultats de simulations montrant les performances de chaque commande. En outre, un test de robustesse est réalisé, à la fin de ce chapitre, pour observer le comportement du robot soumis à des perturbations extérieures.

Dans le cinquième chapitre, nous effectuons des essais de stabilisation sur le robot Pekee. Il s'agit de tracer une trajectoire stabilisante sur le sol et de mettre le robot dans une

configuration initiale quelconque lui permettant d'effectuer la poursuite en se basant sur les différentes techniques de traitement d'images présentées au chapitre 3.

Ce mémoire se termine par une conclusion générale où l'accent sera mis sur la contribution des techniques proposées quant à la résolution des problèmes de stabilisation appliqués au robots mobiles, et sur les travaux s'inscrivant dans le prolongement de ce mémoire et à mener ultérieurement.

Chapitre I
Etat de l'art sur la robotique mobile

Chapitre I Etat de l'art sur la robotique mobile

I. Introduction

Un robot est une machine capable d'agir sur son environnement et de réaliser des tâches diverses. Il est doué d'une capacité de s'auto-adapter à différentes situations. Pour bien mener son fonctionnement, un robot est équipé de capteurs lui permettant de percevoir l'environnement dans lequel il évolue, et d'effecteurs (roue, bras, pince, jambe, etc.) pour agir sur cet environnement.

D'une façon générale, on distingue deux groupes de robots : les robots fixes désignant les robots manipulateurs et les robots mobiles ayant la tendance de se déplacer selon une trajectoire donnée.

On regroupe sous l'appellation « robots mobiles » l'ensemble des robots à base mobile, par opposition notamment aux robots manipulateurs. Le terme « robot mobile » est attribué, dans la plupart des cas, aux robots mobiles à roues. Les autres robots mobiles sont, le plus souvent, désignés par leur type de locomotion, qu'ils soient marcheurs, sous-marins ou aériens.

On peut estimer que les robots mobiles à roues constituent le domaine d'investigation le plus avancé des robots mobiles. Historiquement, leur étude est venue assez tôt, suivant celle des robots manipulateurs. 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, ces systèmes ont soulevé un grand nombre de problèmes difficiles. Nombreux de ceux-ci ne sont d'ailleurs toujours pas résolus. Il s'agit, par exemple, des problèmes de navigation, de localisation, de poursuite et de planification de trajectoire.

Le domaine de la poursuite et de la planification de trajectoire constitue un sujet important dans l'étude du comportement des robots mobiles dans un environnement hostile. C'est pour cette raison qu'on s'attache à mettre l'accent sur les différentes techniques de suivi de trajectoire et leur implémentation dans des cas réels de robotique mobile.

Pour bien mener cette étude, nous consacrons ce chapitre à la présentation d'un état de l'art sur les robots mobiles. Pour ce faire, nous commençons par définir le principe de roulement sans glissement, puis présenter les différents types de robots mobiles à roues à savoir les plus utilisés en pratique, ainsi que leurs caractéristiques cinématiques. Nous

abordons une étude cinématique générale sur un cas de robots mobiles. Il s'agit du robot mobile « Pekee » de la plateforme Wany Robotics.

II. Etude de la cinématique des véhicules à roues

II.1 Hypothèses

La problématique de la commande des robots mobiles étant trop vaste pour pouvoir être présentée de façon exhaustive, nous introduisons dans cette partie un certain nombre d'hypothèses simplificatrices [FIL 04], [BOU 07]:

– les véhicules sont considérés comme rigides (indéformables) et évoluant sur un plan,

– les véhicules sont dotés de roues conventionnelles : le point de contact entre la roue et le sol est réduit à un point *I* et la roue est soumise à la contrainte de roulement sans glissement.

II.2 Roulement sans glissement

La locomotion à l'aide de roues exploite la friction au contact entre la roue et le sol. Pour cela, la nature du 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 *I* est nulle [BOU 07], [BAY 08].

En pratique, le contact se fait sur une surface, ce qui engendre de légers glissements. Nous considérons que ce glissement est faible c'est-à-dire que la vitesse au point de contact est supposée nulle.

Pour exprimer analytiquement la condition de r.s.g, on considère une roue verticale qui roule sans glisser sur un sol plat (figure I.1), dans un repère R (O,X,Y,K). Le repère R_1 (O_1,X_1,Y_1,K_1) est lié à la roue et le roulement sans glissement se traduit par une vitesse nulle au point *I* de la roue en contact avec le sol. Une vitesse qui sera exprimée en fonction de la vitesse de rotation ω de la roue et de la vitesse de son centre O_1.

Dans ce cas, on fait correspondre à tout point P un champ de vecteurs noté v_P. Ce dernier qui représente la vitesse absolue de l'origine par rapport au repère R est donné comme suit [BOU 07], [DEF 07]:

$$\overrightarrow{v_P} = \frac{d}{dt}(\overrightarrow{OP}) \qquad \text{(I.1)}$$

Soit Ω la résultante du torseur représentant le vecteur de rotation du corps par rapport à R.

La connaissance de v_P et de Ω permet de calculer la vitesse de la roue en un point A par la relation fondamentale suivante :

$$\overrightarrow{v_P} = \overrightarrow{v_A} + \vec{\Omega} \wedge \overrightarrow{AP} \tag{I.2}$$

A partir de la relation (I.2), on peut calculer aisément la vitesse du point *I* par rapport au repère R. D'où :

$\vec{V}(I/R) = \vec{0} = \vec{V}(O_1/R) + \vec{\omega}_{(I/O)} \wedge \overrightarrow{O_1 I}$

$\vec{V}(I/R) = \vec{0} = \dot{x}_{re}\,\vec{X} + \dot{y}_{re}\,\vec{Y} + (\dot{\varphi}\,\vec{K} + \theta(-\sin\varphi\vec{X} + \cos\varphi\vec{Y})) \wedge (-r\vec{K})$

$\vec{V}(I/R) = \vec{0} = (\dot{x}_{re} - r\,\dot{\theta}\cos\varphi)\vec{X} + (\dot{y}_{re} - r\,\dot{\theta}\sin\varphi)\vec{Y}$

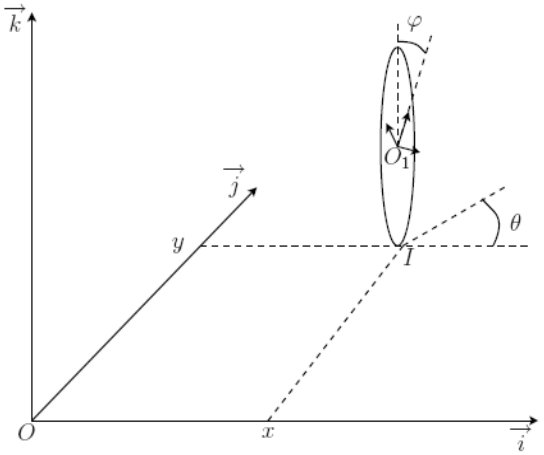


Figure I.1. Description d'une roue.

Avec r le rayon de la roue et (*x*, *y*) les coordonnées du point O_1 dans *R*.

$\vec{V}(I/R) = \vec{0}$, ceci implique que deux types de contraintes peuvent être déduits par l'intermédiaire des équations suivantes :

$$\dot{x}_{re} - r\,\dot{\theta}\cos\varphi = 0 \tag{I.3}$$

$$\dot{y}_{re} - r\,\dot{\theta}\sin\varphi = 0 \tag{I.4}$$

Ces deux équations peuvent être transformées pour faire apparaître les composantes de vitesse dans le plan de la roue d'une part et perpendiculairement à la roue d'autre part. Les nouvelles équations sont données comme suit :

$$-\dot{x}_{re}\sin\varphi + \dot{y}_{re}\cos\varphi = 0 \tag{I.5}$$

$$\dot{x}_{re}\cos\varphi + \dot{y}_{re}\sin\varphi = -r\,\dot{\theta} \tag{I.6}$$

Ce système d'équation traduit les deux propriétés suivantes :

1) la vitesse du centre de la roue est parallèle au plan de la roue

2) la vitesse du centre de la roue est $r\dot{\theta}$.

II.3 Modélisation des robots à roues

D'une façon générale, le comportement d'un robot à roues peut être modélisé sous la forme suivante :

$$\dot{q} = \sum_{i=1}^{m} g_i(q) u_i \qquad \text{(I.7)}$$

Où $q \in R^n$, $m<n$, u_i *(i = 1, . . .,m)* sont des variables de commande et $g_i(q)$ *(i = 1, . . . ,m)* sont des champs de vecteurs différentiables sur R^n.

III. Les grandes classes de robots mobiles

III.1 Disposition des roues et centre instantané de rotation

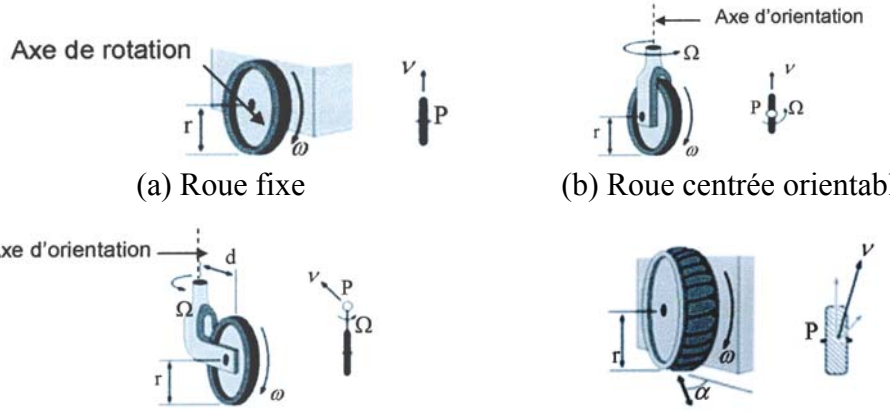
La configuration d'une roue et le nombre de paramètres nécessaires à sa description dépendent du type de roue considérée [BOU 07], [BAY08]. Nous introduisons ici les quatre types de roues principalement utilisés en robotique mobile (figure I.2) :

→ les roues fixes dont l'axe de rotation passe par le centre de la roue, tandis que l'axe d'orientation est constant (figure I.2a).

→ les roues centrées orientables dont l'axe d'orientation, perpendiculaire au sol, passe par le centre de la roue (figure I.2b).

→ les roues décentrées orientables dont l'axe d'orientation, perpendiculaire au sol, ne passe pas par le centre de la roue. Ces roues sont appelées roues folles, (figure I.2c).

→ les roues suédoises dont la bande de roulement a été remplacée par des galets inclinés par rapport à la normale au plan de la roue. C'est la combinaison de la rotation de la roue avec la rotation libre du galet en contact avec le sol qui permet un déplacement sans glissement sur le sol (figures I.2d).



(a) Roue fixe (b) Roue centrée orientable

(c) Roue décentrée orientable (d) Roue suédoise

Figure I.2. Les principaux types de roues dans la robotique mobile

Ces quatre types de roues sont les plus utilisés en robotique mobile. Cependant, ils existent d'autres types comme les roues sphériques qu'on peut trouver dans plusieurs prototypes vu qu'elle offre au robot une propriété très intéressante qui est la propriété omnidirectionnelle (voir figure I.14).

Pour un ensemble de roues données, toute disposition ne conduit pas à une solution viable. Un mauvais choix de roues peut limiter la mobilité du robot ou occasionner d'éventuels blocages. Par exemple, un robot équipé de deux roues fixes non parallèles ne pourrait pas aller en ligne droite.

Pour qu'une disposition de roues soit viable et n'entraîne pas de glissements sur le sol, toutes les roues doivent contenir un point de vitesse nulle unique autour duquel tourne le robot en instantanée. Ce point, lorsqu'il existe, est appelé **C**entre **I**nstantané de **R**otation (CIR).

Le CIR s'obtient par l'intersection des axes de rotation des différentes roues. Ce point est bien unique.

Après avoir donné quelques hypothèses et les différentes configurations de roues existantes, nous présentons dans ce qui suit les principales catégories de robots mobiles à roues [BOU 07], [DEF 07], [LAU 01].

III.2 Robot mobile différentiel ou unicycle

III.2.1 Description

Une des configurations les plus utilisées pour les robots mobiles est la configuration différentielle (differential drive) qui comporte deux roues fixes non orientables commandées indépendamment. Une ou plusieurs roues folles sont ajoutées à l'avant ou à l'arrière du robot pour assurer sa stabilité. Dans certains cas, ils existent certains robots différentiels avec quatre roues commandées indépendamment sauf qu'ils sont modélisés en deux roues. Le schéma de principe du robot différentiel est présenté par la figure I.3.

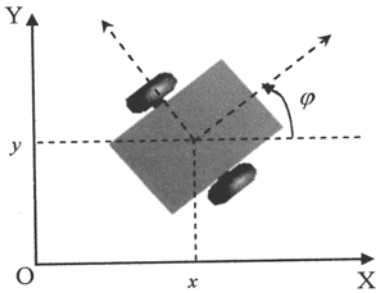
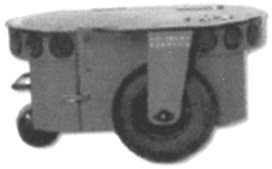


Figure I.3. Schéma de principe d'un robot mobile différentiel

Ce type de robot est très répandu en raison de sa simplicité de développement et de ses propriétés cinématiques intéressantes, comme sa capacité de tourner sur lui même. La figure I.4 présente deux exemples de robots mobiles différentiels.



(a) Robot mobile Pioneer P3-DX

(b) Robot mobile ATRV2

Figure I.4. Exemples de robots mobiles différentiels.

III.2.2 Modélisation

Dans ce cas de robots, les roues motrices ont le même axe de rotation. Le *CIR* dans ce cas est un point de cet axe.

Soit *R* le rayon de courbure de la trajectoire du robot, c'est à dire la distance du *CIR* au point *O'* (figure I.5).

Soit *2L* la distance qui sépare les deux roues et Ω la vitesse angulaire du robot par rapport au *CIR*.

Les vitesses des roues droite et gauche, respectivement notées v_d et v_g vérifient :

$$v_d = (R + L)\Omega \tag{I.8}$$

$$v_g = (R - L)\Omega \tag{I.9}$$

A partir de ces deux équations, on peut déterminer *R* et Ω (en fonction des vitesses des roues) de la façon suivante :

$$\Omega = \frac{v_d - v_g}{2L} \tag{I.10}$$

$$R = L\frac{v_d + v_g}{v_d - v_g} \tag{I.11}$$

La vitesse linéaire *v* du robot au point *O'* est :

$$v = \frac{v_d + v_g}{2} \tag{I.12}$$

La vitesse de rotation du robot est égale à la vitesse de rotation autour du *CIR* :

$$\Omega = \dot{\varphi} = \frac{v_d - v_g}{2L} \tag{I.13}$$

L'équation (I.11) permet de situer le *CIR* sur l'axe des roues. Par ailleurs, ces équations expliquent deux propriétés particulières du mouvement des robots différentiels :

* Si $v_d = v_g$, la vitesse angulaire Ω sera nulle et le rayon de courbure *R* est infini et le robot se déplace donc en ligne droite.

* Si $v_d = - v_g$, $\Omega \neq 0$ et R est nulle, alors le robot effectue une rotation sur lui même.

Cependant, dans le cas où $v_d \neq - v_g$, le déplacement du robot est un virage à gauche ou à droite et ceci en fonction du signe de v_d par rapport à v_g (le virage est dans une direction qui correspond à la vitesse inférieure).

L'utilisation de ce mode de locomotion fournie une solution simple pour déplacer le robot d'une position à une autre. C'est, sans doute, une des raisons du succès de ce type de robot.

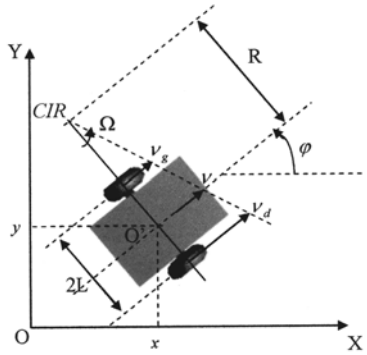


Figure I.5. Le CIR d'un robot mobile différentiel

III.2.3 Modélisation cinématique

Le modèle cinématique du robot différentiel est donné par les équations suivantes :

$$\begin{cases} \dot{x} = v\cos\varphi & \text{(I.14)} \\ \dot{y} = v\sin\varphi & \text{(I.15)} \\ \dot{\varphi} = \Omega & \text{(I.16)} \end{cases}$$

Ces équations relient la dérivée de la position (x,y,φ) du robot à la commande $u = (v,\Omega)^T$, avec φ la rotation instantanée du robot par rapport au repère (O, X, Y). De ce fait, la position du robot est donnée par :

$$x(t) = \int_0^t v(\sigma)\cos(\varphi(\sigma))d\sigma \quad \text{(I.17)}$$

$$y(t) = \int_0^t v(\sigma)\sin(\varphi(\sigma))d\sigma \quad \text{(I.18)}$$

$$\varphi(t) = \int_0^t \Omega(\sigma)d\sigma \quad \text{(I.19)}$$

III.3 Robot mobile tricycle

III.3.1 Description

L'architecture d'un robot mobile tricycle est représentée dans la figure I.6. Ce robot est constitué de deux roues fixes de même axe et d'une roue centrée orientable placée sur l'axe longitudinal du robot. Le mouvement du robot dépend de deux actions : la vitesse longitudinale et l'orientation de la roue orientable.

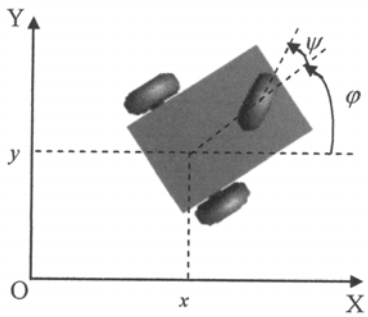


Figure I.6. Schéma de principe d'un robot mobile tricycle

Deux prototypes de robots tricycles sont représentés dans la figure I.7.

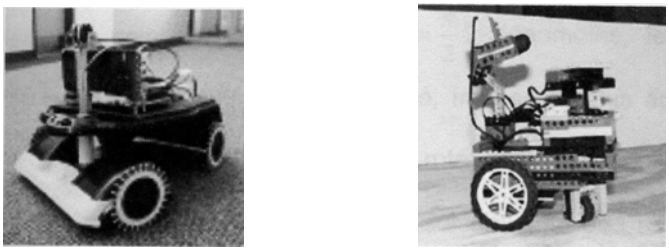


Figure I.7. Exemples de robots mobiles de type tricycle

III.3.2 Modélisation

Le *CIR* du robot se situe à l'intersection des axes des roues fixes et de la roue orientable, (figure I.8).

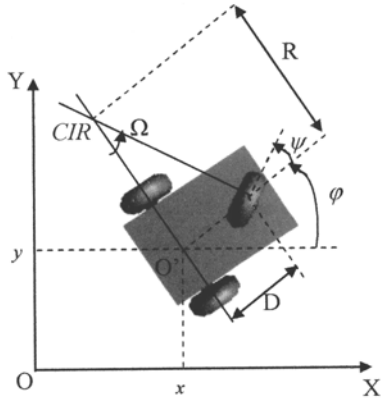


Figure I.8. Le *CIR* du robot mobile de type tricycle

On peut déterminer *R* de manière géométrique à partir de l'angle d'orientation Ψ de la roue avant et la vitesse de rotation autour du CIR (Ω) à partir de la vitesse linéaire *v* du robot (vitesse en *O'*). Les expressions de *R* et Ω sont données comme suit :

$$R = \frac{D}{\tan\psi} \tag{I.20}$$

$$\Omega = \frac{v}{D}\tan(\psi) \tag{I.21}$$

La vitesse linéaire *v* peut être exprimée en fonction de la vitesse linéaire de la roue orientable v_s :

$$v = v_s \cos\psi \tag{I.22}$$

Ce type de robots peut se diriger en ligne droite pour $\Psi = 0$ et peut tourner (en théorie) autour du point *O'* (sur lui-même) pour $\psi = \frac{\pi}{2}$. Néanmoins, le rayon de braquage de la roue orientable impose le plus souvent des valeurs de Ψ tel que $-\frac{\pi}{2} < \psi < \frac{\pi}{2}$, interdisant la rotation du robot autour de lui-même.

III.3.3 Modélisation cinématique

L'écriture des contraintes correspondantes à chacune des roues est similaire à celle établie dans le cas du robot différentiel. Ceci permet de déterminer le modèle cinématique du robot tricycle. Toutefois, par un simple raisonnement géométrique, nous pouvons établir les équations représentant la dérivée de la position du robot de la façon suivante :

$$\begin{cases} \dot{x} = v\cos\varphi & \text{(I.23)} \\ \dot{y} = v\sin\varphi & \text{(I.24)} \\ \dot{\varphi} = \Omega = \frac{v}{D}\tan\psi & \text{(I.25)} \\ \dot{\psi} = \Omega_s & \text{(I.26)} \end{cases}$$

Où $u = (v, \Omega_s)^T$ est le vecteur de commande cinématique, φ la rotation instantanée du robot par rapport au repère (O, X, Y), et Ω_s la vitesse d'orientation correspondante à la roue orientable.

La position est donc donnée par les expressions suivantes :

$$x(t) = \int_0^t v(\sigma)\cos(\varphi(\sigma))d\sigma \tag{I.27}$$

$$y(t) = \int_0^t v(\sigma)\sin(\varphi(\sigma))d\sigma \tag{I.28}$$

$$\varphi(t) = \int_0^t \Omega(\sigma)d\sigma \tag{I.29}$$

III.4 Robot mobile de type voiture

Le cas des robots de type voiture est similaire à celui du tricycle. La différence se situe au niveau du train avant, qui comporte deux roues au lieu d'une seule roue au milieu. En pratique, on rencontre beaucoup plus souvent ce type de système.

A titre d'exemple, voici deux prototypes de robots mobiles de type voiture (figure I.9).

(a) Robot Kanade

(b) Robot Cycab

Figure I.9. Exemples de robots mobiles de type voiture.

III.4.1 Modélisation

Comme nous l'avons vu précédemment, l'existence d'un *CIR* unique impose que les axes des roues du robot soient concourants. Dans le cas du robot de type voiture, cela impose aux roues avants d'avoir une orientation différente, comme illustré dans la figure I.10. Le roulement idéal, assurant que le *CIR* est bien unique, est réalisé par un système de braquage différentiel (dit d'Ackerman). Par ailleurs, les roues n'ayant pas le même rayon de courbure ont des vitesses différentes.

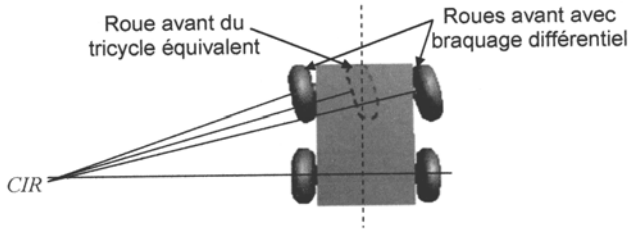


Figure I.10. Schéma de principe d'un robot mobile de type voiture.

III.4.2 Modélisation cinématique

Le modèle cinématique du robot de type voiture est donné par les équations suivantes :

$$\begin{cases} \dot{x} = v\cos\varphi & \text{(I.30)} \\ \dot{y} = v\sin\varphi & \text{(I.31)} \\ \dot{\varphi} = \Omega & \text{(I.32)} \end{cases}$$

Ces équations relient la dérivée de la position (x,y,φ) du robot à la commande $u = (v,\Omega)^T$, avec φ la rotation instantanée du robot par rapport au repère (O, X,Y). De ce fait, la position du robot est donnée par :

$$x(t) = \int_0^t v(\sigma)\cos(\varphi(\sigma))d\sigma \quad \text{(I.33)}$$

$$y(t) = \int_0^t v(\sigma)\sin(\varphi(\sigma))d\sigma \quad \text{(I.34)}$$

$$\varphi(t) = \int_0^t \Omega(\sigma)d\sigma \quad \text{(I.35)}$$

III.5 Robot mobile omnidirectionnel

III.5.1 Description

Un robot mobile est dit omnidirectionnel si l'on peut agir indépendamment sur les vitesses de translation selon les axes *x* et *y*, et la vitesse de rotation autour de ***z***. D'un point de vue cinématique, cela n'est pas possible avec des roues fixes ou des roues centrées orientables. On peut, en revanche, réaliser un robot omnidirectionnel ayant recours à un ensemble de trois roues décentrées orientables ou de trois roues suédoises disposées aux sommets d'un triangle équilatéral (figure I.11).

Trois roues sont donc suffisantes. Cependant, dans certains cas, une quatrième roue offre des possibilités d'optimisation et permet de rendre le système plus robuste en évitant le glissement dans le cas d'un sol qui n'est pas parfaitement plat par exemple.

En ce qui concerne les roues suédoises (figure I.2d), la rotation des galets inclinés permet aux roues du robot de rouler dans une direction perpendiculaire à celle autour duquel elles roulent normalement. C'est ce qui permet au robot de se déplacer dans diverses directions sans avoir besoin de tourner. Dans ce cas, le robot n'effectue pas de rotation mais uniquement des translations.

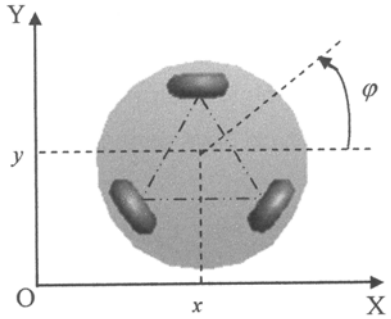


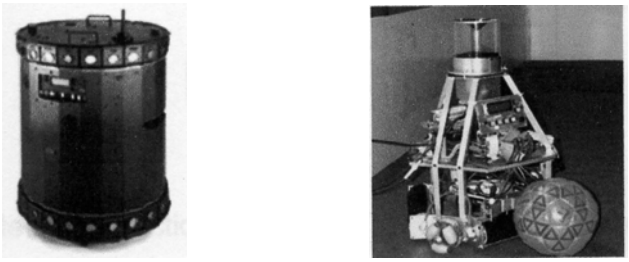
Figure I.11. Schéma de principe d'un robot mobile omnidirectionnel.

III.5.2 Modélisation

Dans ce cas, on peut considérer qu'il est possible d'appliquer directement la commande sur le modèle cinématique qui sera défini par les équations suivantes :

$$\begin{cases} \dot{x} = u_1 & \text{(I.36)} \\ \dot{y} = u_2 & \text{(I.37)} \\ \dot{\varphi} = u_3 & \text{(I.38)} \end{cases}$$

Où $u = (u_1,u_2,u_3)^T$ représentent le vecteur de commande. Nous choisissons d'une façon générale ce type de robot pour se dispenser des problèmes de planification et de commande liés à la non-holonomie. La figure I.12 représente des exemples de ce type de robots.



(a) (b)

Figure I.12. Exemples de robots mobiles omnidirectionnels

(a) Robot omnidirectionnel muni de roues décentrées orientables.

(b) Robot omnidirectionnel muni de roues suédoises.

Une autre technique consiste à utiliser des roues sphériques ou des associations de roues sphériques tronquées, montées orthogonalement les unes par rapport aux autres. Dans ce dernier cas, lors du pivotement de l'axe motorisé, le contact sur le sol se fait en alternance avec l'une ou l'autre des deux roues (sphères) formant un essieu (figure I.13).

Les axes de rotation des deux sphères sont perpendiculaires entre eux et également perpendiculaires et concourants avec l'axe longitudinal de l'essieu.

La mobilité interne que constitue la rotation libre des sphères permet un déplacement en translation de l'essieu (et de la structure qui le porte) dans sa direction longitudinale, sans glissement des roues sur le sol (figure I.14).

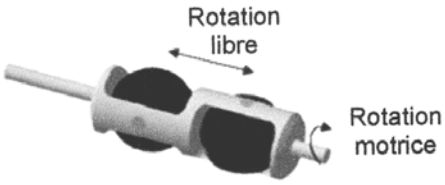


Figure I.13. Représentation 3D d'un essieu

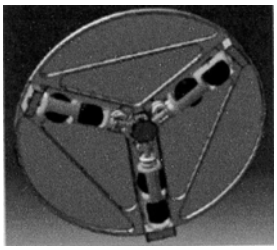


Figure I.14. Disposition des trois essieux

III.6 Robot mobile à traction synchrone

III.6.1 Description

La traction synchrone (synchronous drive) est une technique utilisée pour minimiser l'effet de glissement et augmenter la force de traction.

On rencontre ce type de robot dans l'industrie automobile et dans les robots tout terrain. La configuration du robot à traction synchrone est similaire à un robot à trois ou quatre roues couplées de façon qu'elles soient actionnées en même temps, en ayant la même vitesse et la même orientation. Ce système est réalisé grâce à deux moteurs, un pour la traction et l'autre pour l'orientation. L'ensemble est relié par une chaîne (ou une ceinture) pour s'assurer que

toutes les roues tournent de façon synchrone. La figure I.15 montre un robot à quatre roues couplées.

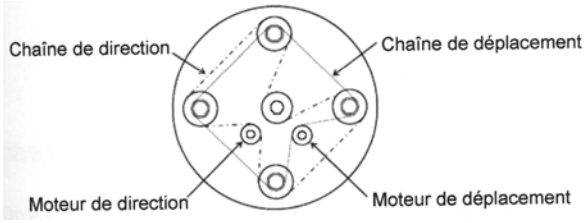


Figure I.15. Architecture d'une plate-forme de robot mobile à traction synchrone.

La figure I.16 montre un exemple de ce type de robots.

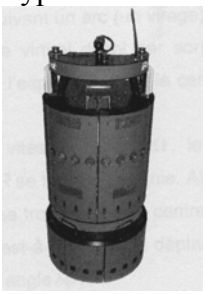


Figure I.16. Robot mobile B21r à traction synchrone.

III.6.2 Structure des roues

Nous avons vu qu'à chaque modèle de robot correspond un type de roues et ceci selon son architecture et ses tâches à accomplir. A titre d'exemple, le robot B21r dispose de quatre roues décentrées orientables qui tournent selon deux axes, une rotation selon l'axe y permettant le roulement des roues pour obtenir la translation, et une rotation selon l'axe k permettant de changer leurs orientations (figure I.17).

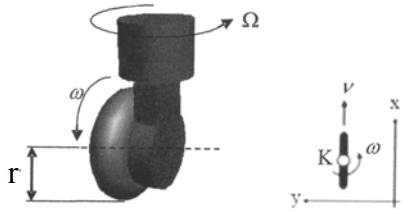


Figure I.17. Roue décentrée orientable

Les paramètres de la roue sont :

r = rayon de la roue.

v = vitesse linéaire de la roue.

ω = vitesse angulaire de la roue.

Ω = vitesse d'orientation.

III.6.3 Modélisation cinématique

Afin de trouver les équations de déplacement d'un robot à traction synchrone, on utilise les contraintes déduites, du principe de roulement sans glissement, présentées dans les équations (I.3) et (I.4).

En introduisant la vitesse de roulement de la roue $v = r\dot{\theta}$ et sa vitesse de rotation Ω autour de l'axe k (figure I.18), on forme le modèle cinématique suivant :

$$\begin{cases} \dot{x}_{re} = v\cos\varphi & \text{(I.39)} \\ \dot{y}_{re} = v\sin\varphi & \text{(I.40)} \\ \dot{\varphi} = \Omega & \text{(I.41)} \end{cases}$$

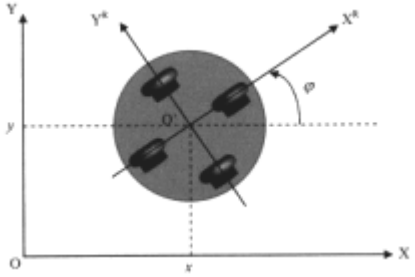


Figure I.18. Modèle cinématique utilisé pour le robot mobile à traction synchrone

La position du robot mobile dans le repère absolu est donnée par le modèle suivant :

$$x(t) = \int_0^t v(t)\cos(\varphi(\sigma))d\sigma \quad \text{(I.42)}$$

$$y(t) = \int_0^t v(t)\sin(\varphi(\sigma))d\sigma \quad \text{(I.43)}$$

$$\varphi(t) = \int_0^t \Omega(\sigma)d\sigma \quad \text{(I.44)}$$

IV. Notre plateforme de travail : le robot Pekee

Le robot Pekee est un robot construit par la société française « Wany Robotics [SLI 05] ». Il dispose d'un mécanisme de déplacement constitué de trois roues (deux roues motrices à l'avant commandées par deux moteurs indépendants et une roue folle directrice à l'arrière pour permettre au robot d'être stable). Sa longueur est de 40 cm, sa largeur de 25.5 cm et son poids de 3.3 kg [DEF 07].

Ce robot est capable d'effectuer des mouvements autonomes selon l'environnement qu'il identifie et ceci grâce à ses différents capteurs qui lui sont intégrées.

« Pekee » est un robot de type unicycle. Il peut être représenté par un modèle cinématique décrit dans la section III.2.3.

Ce robot est une plate-forme ouverte à des professionnels, des chercheurs, des éducateurs, et des étudiants de robotique. Il est utilisé dans différents domaines de recherche comme : la navigation, la stéréovision, l'analyse de signal et l'intelligence artificielle…

Il est muni de plusieurs dispositifs comme : l'infrarouge, les odomètres, un détecteur de chocs, des gyromètres, un module WIFI, et même une caméra vidéo (figure I.19).

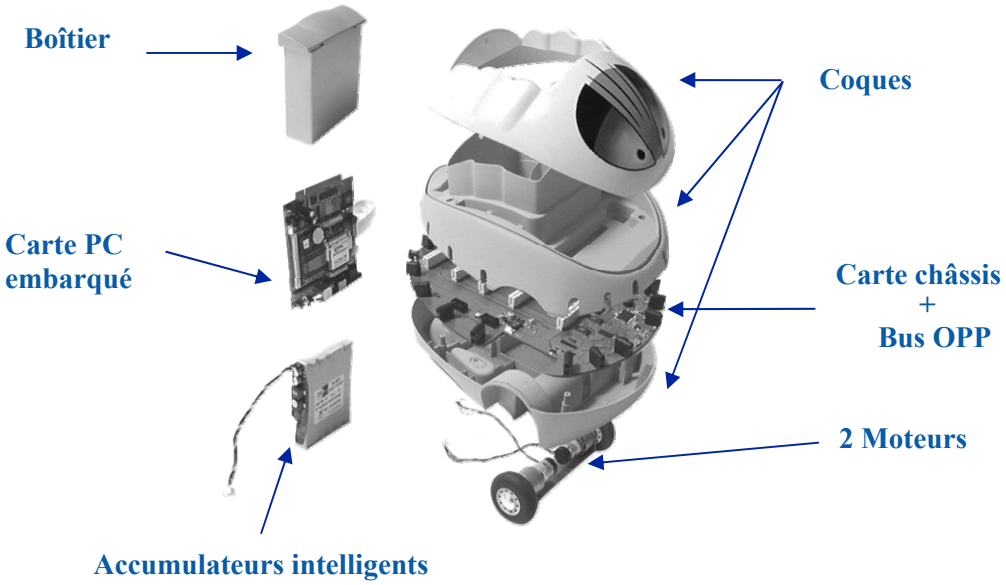


Figure I.19. Robot mobile Pekee utilisé dans notre travail

Notre travail sera focalisé sur ce type de robot dont la description de la problématique, la mise en oeuvre théorique, les essais de simulation et l'expérimentation feront l'objet de développement dans les prochains chapitres.

V. Holonomie et non-holonomie

La capacité d'un robot mobile à pouvoir se déplacer, à partir d'une situation donnée, dans n'importe quelle direction est appelée « holonomie ». En effet, le mécanisme holonomique permet au robot de manœuvrer dans n'importe quelle direction arbitraire à partir de n'importe quelle configuration arbitraire.

Un système holonomique est un système qui comporte le même nombre de déplacements virtuels que le nombre de coordonnées généralisées pour le décrire. Comme nous l'avons vu, nous avons besoin de trois coordonnées généralisées pour décrire d'une façon unique la configuration d'un robot mobile sur le plan. A titre d'exemple, un robot holonomique omnidirectionnel est un robot pour lequel les trois mouvements planaires

indépendants (deux de translation et un de rotation) sont admissibles à partir d'une configuration quelconque.

Par contre, de nombreux robots mobiles sont des systèmes non-holonomes tels que les robots différentiels, les tricycles, les robots voitures, et les robots à traction synchrone. Pour ce type de robots, bien que l'espace de configuration sur le plan soit de trois dimensions, le mouvement est produit par deux déplacements indépendants seulement. Le robot, dans ce cas, ne possède que deux degrés de liberté.

D'un point de vue mathématique, nous pouvons dire que la non-holonomie d'un robot est due au fait que ses équations de contraintes ne sont pas intégrables. Pour cette raison, l'analyse cinématique, dynamique et le contrôle sont plus complexes étant donné que les coordonnées de leur représentation ne peuvent être éliminées en utilisant les équations de contraintes. A cet effet, les systèmes qui possèdent des contraintes non-holonomiques nécessitent toujours un nombre plus grand de coordonnées pour leur description que le nombre de degrés de liberté.

Un robot est dit ***non-holonome*** lorsque son modèle ***cinématique*** est régit par un système non linéaire de la forme [SLI 05] :

$$\dot{x} = u_1 X_1(x) + \cdots + u_m X_m(x), \quad x \in CS, \tag{I.45}$$

CS est l'***espace de configuration*** qui représente l'ensemble de toute les configurations (ou positions) possibles du robot.

La dimension *n* de l'espace *CS* doit être plus grande que la dimension *m* des contrôles.

Dans le cas du robot Pekee, un ou plusieurs de ses degrés de liberté (avancement, position) ne peuvent pas être contrôlés indépendamment. Cela rend difficile la prévision des mouvements car chaque avancée s'accompagne d'une modification de la position latérale. Il s'agit donc d'un système non-holonome (dans ce cas, il est difficile de manoeuvrer dans des espaces réduits).

Ce type de robot possède, donc, un nombre de degrés de liberté supérieur aux contrôles disponibles.

VI. Conclusion

Nous avons présenté dans ce chapitre les différents types de robots mobiles à roues les plus utilisés. Cette variété réside dans leur mode de locomotion qui dépend du type et de la disposition des roues utilisées. Une étude cinématique spécifique était donc nécessaire pour chaque type de robot avant toute étape de développement.

Nous avons montré aussi la différence entre un système holonome et un système non-holonome. Nous avons observé que la majorité des robots mobiles sont des systèmes non-holonomes excepter le cas des robots omnidirectionnels.

Après avoir effectué un tour d'horizon sur les robots mobiles, nous présentons notre problématique qui traite la commande des systèmes non-holonomes pour réaliser une stabilisation en un point d'équilibre. Ceci fera l'objet d'étude dans le chapitre suivant.

Nous nous focalisons dans notre travail sur un exemple de robot mobile de type unicycle. Il s'agit du robot Pekee de la plateforme Wany Robotics.

Chapitre II
Problématique

Chapitre II Problématique

I. Introduction

La recherche d'algorithmes de planification et de stratégies de commande pour des robots mobiles non-holonomes constitue aujourd'hui l'un des principaux axes de recherche de la robotique moderne. Plusieurs raisons contribuent à cet engouement. La première est que les véhicules à roues constituent de nos jours le moyen de transport individuel principal. Leur automatisation, précédemment limitée aux expérimentations en laboratoire, est maintenant envisagée pour des applications grand public (systèmes de transport urbain intelligent, etc.).

Une autre raison plus technique tient au fait que les équations régissant le déplacement des systèmes non-holonomes revêtent un intérêt théorique particulier dans le domaine de l'automatique non linéaire.

Dans ce cadre, plusieurs travaux ont été engagés pour résoudre les problèmes de commande des robots mobiles. Ces travaux considèrent différentes difficultés que rencontre le concepteur de lois de commande automatique, pour essayer d'automatiser sa démarche en fonction des contraintes industrielles. Deux types de problèmes sont considérés : ceux liés à la performance des lois, puis ceux liés à la robustesse du système vis-à-vis des incertitudes sur certains paramètres du modèle non linaire, ou vis-à-vis de modes structuraux basses fréquences pouvant conduire à une instabilité. Dans ce cas, les différents objectifs fixés sont :

- la stabilité du système,
- l'amélioration de la robustesse du système,

L'objectif de ce chapitre est de présenter, dans sa première partie, le problème de stabilisation des systèmes non-holonomes et de résumer quelques techniques de commandes développées. L'autre partie de ce chapitre est consacrée à l'étude de la problématique de poursuite de trajectoire stabilisante en faisant appel aux techniques de vision par ordinateur. Nous présentons, ici, les grandes lignes de cette technique ainsi que notre approche globale.

II. Problème de commande des systèmes non-holonomes

Les obstacles majeurs concernant la commande de systèmes non-holonomes sont :

1. le caractère **non linéaire** du système,
2. la **non commandabilité** de leur approximation linéaire, malgré que le système est commandable

3. la non adéquation avec la condition nécessaire de Brockett [BRO 83] pour l'existence d'un retour d'état continu et indépendant du temps, stabilisant le système.

La difficulté relative du problème de la commande ne dépend pas seulement de la nature non-holonome du système, mais aussi de l'objectif du contrôle. En effet, pour certains objectifs de contrôle tels que :

- la stabilisation vers une variété choisie contenant la variété d'équilibre,
- la stabilisation de trajectoires,
- le suivi de chemins.

Les approches classiques de commande non linéaire sont efficaces. Cependant, pour des objectifs de contrôle tels que la planification de mouvement et la stabilisation vers un état d'équilibre, ces approches peuvent présenter des performances moins satisfaisantes. C'est dans cette optique qu'il est judicieux de penser à d'autres techniques plus élaborées.
Dans ce qui suit, nous étudions le problème de stabilisation des systèmes non-holonomes (robots mobiles dans notre cas) vers un état d'équilibre ainsi que les solutions existantes pour le résoudre.

III. Stabilisation des systèmes non-holonomes

Nous nous intéressons dans ce travail au problème de la stabilisation en un point fixe, c'est-à-dire trouver une loi de commande telle que le point d'équilibre qui représente généralement l'origine, soit stable.

Les principales applications des techniques de stabilisation sont, à titre d'exemple, les manœuvres de parking (créneau, …) ou les applications qui nécessitent une immobilisation du véhicule s'accompagnant d'un positionnement précis.
Plusieurs méthodes issues de l'automatique classique ont été mises en œuvre pour stabiliser les robots mobiles. Cependant, ces méthodes ne donnent pas les résultats inattendus concernant le positionnement précis du robot autour du point d'équilibre. Ce problème est dû principalement à la non prise en compte des propriétés de non-holonomie du système.

III.1 Présentation du problème de stabilisation

Un aspect primordial du problème de la stabilisation des systèmes non-holonomes est lié à la non existence d'un retour d'état continu et indépendant du temps assurant une stabilisation. Ceci découle d'un résultat établi par Brockett [BRO 83] qui est décrit ci-dessous.

Théorème : *Soit le système de commande ($\dot{x} = f(x,u)$, $x \in R^n$, $u \in R^m$) avec f différentiable et (x,y)=(0,0) un point d'équilibre de ce système. Une condition nécessaire pour qu'il existe un retour d'état u continu tel que l'origine du système bouclé :*

$$\dot{x} = f(x,u(x)) \tag{II.1}$$

Soit localement asymptotiquement stable, est la surjectivité locale de l'application (x,u) → f(x,u) au voisinage de (x,u)=(0,0). Plus précisément, pour tout voisinage Ω de (0,0) dans R^{n+m}, l'image par f de Ω doit être un voisinage de 0 dans R^n.

Ce résultat implique que de nombreux systèmes non linéaires ne sont pas asymptotiquement stabilisables par retour d'état continu même s'ils sont commandables. C'est le cas des systèmes non-holonomes et en particulier les robots mobiles à roues.

A titre d'exemple, nous démontrons ce résultat dans le cas d'un robot mobile de type unicycle donné par le modèle cinématique suivant :

$$\dot{x} = f(x,u) \tag{II.2}$$

Avec :

$x=(x_1, x_2, x_3)$,

$u=(u_1, u_2)$,

$f(x,u)=(u_1 cos x_3, u_1 sin x_3, u_2)^T$.

Considérons un vecteur f dans R^3 donné par la forme : $f(x,u) = (0,\delta,0)$ avec $\delta \neq 0$. On constate immédiatement que cette équation n'admet pas la solution au voisinage de $(x,u)=(0,0)$, puisque la première équation à satisfaire, c'est-à-dire $u_1.cos x_3 = 0$ implique $u_1 = 0$. Ainsi, il n'existe pas de solutions à la deuxième équation si δ est différent de zéro. Ceci montre bien que f n'est pas localement surjective au voisinage de $(x, u) = (0, 0)$.

Il est bien connu que l'approximation linéaire des systèmes non holonomes n'est pas commandable. En effet, en prenant le même type de robot mobile, c'est-à-dire celui de l'unicycle, nous remarquons que le linéarisé de ces équations n'est pas commandable. Ainsi les méthodes de commande linéaire ne peuvent pas être utilisées pour stabiliser asymptotiquement (localement) le point d'équilibre.

En conséquence, le robot unicycle ne peut être stabilisé asymptotiquement par un retour d'état continu invariant dans le temps.

Afin de résoudre ce problème de stabilisation, plusieurs approches ont été proposées, à savoir celles basées sur les techniques de commande par retour d'état continu non stationnaire ou discontinu invariant.

Ces approches sont classifiées en trois catégories :

1. Stabilisation par retour d'état continu variant dans le temps [SAM 95], [JIA 99]
2. Stabilisation par retour d'état discontinu invariant [BLO 94],
3. Stabilisation par retour d'état hybride [BEN 92].

Dans notre contexte, nous nous intéressons à la deuxième approche qui consiste à commander le robot par retour d'état discontinu afin de le stabiliser et ceci à partir de n'importe quelle configuration initiale.

Avant de parler de notre approche, nous présentons quelques méthodes de stabilisation existantes dans la littérature.

III.2 Stabilisation par retour d'état non stationnaire continu

L'utilisation de retours d'état non stationnaires continus pour la planification et la stabilisation trouve son origine dans les travaux de Samson au début des années 90 [SAM 90]. Un des premiers résultats d'existence de telles commandes est donné dans [COR 92]. Par la suite, diverses approches basées sur la méthode directe de Lyapunov [POM 92], [SAM 95], [GOD 97], [DIX 00], sur les commandes sinusoïdales et polynomiales [MUR 93], sur les techniques de backstepping [JIA 99] conduisent à la construction de retours d'état continus non stationnaires permettant une planification efficace de trajectoire d'un robot mobile. Bien que connues pour avoir d'assez bonnes propriétés de robustesse, ces commandes présentent généralement des inconvénients d'ordre pratique concernant les taux de convergence, le réglage délicat des paramètres de commande ou les trajectoires générées. Le problème de la génération d'une loi de commande prenant en compte les saturations des entrées est également traité dans [JIA 01]. La convergence de l'état du système vers l'origine est rapide. Cependant, cette commande ne présente pas de bonnes propriétés en termes de robustesse vis-à-vis des perturbations.

III.2.1 Exemple de commande par retour d'état non stationnaire continu

III.2.1.1 Technique de commande « backstepping »

Elle repose sur les techniques de Lyapunov pour établir la loi de commande stabilisante d'un système non-holonomes.

La théorie de Lyapunov initialement présentée comme étant un moyen d'analyse de la stabilité des systèmes dynamiques est, en outre, utilisée pour synthétiser les lois de commande stabilisantes des systèmes.

a) Synthèse des lois assurant la stabilisation autour de l'origine

Soit un système non linéaire représenté par l'équation suivante :

$$\dot{x} = f(x,t) + g(t,x)\,u \tag{II.3}$$

Trouver une loi de commande stabilisante au sens de Lyapunov reviens à chercher une fonction définie positives *V(x,t)* et ensuite calculer sa dérivée. La commande *u* est établie de tel sorte que cette dérivée est définie négative.

b) Technique de backstepping

La technique du backstepping n'est en fait que la construction de la fonction de Lyapunov pour stabiliser un système.

Soit un système non-holonome qui peut s'écrire sous la forme (II.4) :

$$\begin{cases} \dot{\eta} = f(\eta) + g(\eta)\,\varepsilon \\ \dot{\varepsilon} = u \end{cases} \tag{II.4}$$

Où : $\left[\eta^T \ \varepsilon\right]^T \in \Re^{n+1}$ représente l'état du système,

u représente l'entrée de commande.

L'objectif est de déterminer une loi de commande *u* qui stabilisera le système d'équation (II.4) autour de l'origine.

Les travaux développés dans [TAD 03] nous donnent le retour d'état suivant :

$$u = \frac{\partial \phi}{\partial \eta}\left[f(\eta) + g(\eta)\,\varepsilon\right] - \frac{\partial V}{\partial \eta}\,g(\eta) - K\left[\varepsilon - \phi(\eta)\right] \tag{II.5}$$

Cette commande *u* assure la convergence asymptotique du système (II.4) vers l'origine.

III.3 Stabilisation par retour d'état discontinu invariant

Quant à l'utilisation de commandes discontinues pour la robotique mobile, elle trouve son origine dans les travaux de Bloch à la fin des années 80 [BLO 89], [BLO 92]. De nombreux travaux proposent des commandes continues par morceaux [HES 99] ou des commandes discontinues [BLO 92], mais ne traitent pas explicitement le problème de robustesse. Des commandes permettant d'obtenir une convergence exponentielle sont proposées dans

[CAN 92]. Cependant, elles sont sensibles aux erreurs initiales et aux perturbations. Inspirées par ces résultats, des stratégies robustes basées sur une transformation du système en coordonnées polaires [AST 96], [CHW 04] assurent une convergence exponentielle du système mais présentent une singularité à l'origine. D'autres types de commande notamment par modes glissants [FLO 03], [DRA 05] permettent d'obtenir de bons résultats mais font apparaître le phénomène de réticence, qui conduit à l'usure rapide des actionneurs.

III.3.1 Exemple de commande par retour d'état discontinu invariant

III.3.1.1 Commande par mode glissant

Dans les systèmes de contrôle les modes glissants sont caractérisés par des actions discontinues dans le temps.

Dans notre cas, considérons le système dynamique non linéaire, donné par l'équation :

$$\dot{x} = f(x) + g(x).u \tag{II.6}$$

Où :

$x = (x_1,\ldots..x_n)^T \in X$ qui représente le vecteur d'état,

X est une variété différentielle ou un ensemble ouvert de R^n,

$u : R^n \rightarrow$ R qui représente l'entrée du système

f et g sont des champs de vecteurs supposés suffisamment différentiables définis sur X.

La conception d'une commande par modes glissants consiste à établir, dans un premier temps, une fonction de commutation s représentant la dynamique désirée. La loi de commande est établie $u(x,t)$ telle que la surface de commutation $s(x,t) = 0$ est atteinte en temps fini.

La loi de commande $u(x,t)$ est définie comme suit :

$$u(x,t) = \begin{cases} u^+(x,t) & \text{si } s(x,t) > 0 \\ u^-(x,t) & \text{si } s(x,t) < 0 \end{cases} \tag{II.7}$$

avec $u^+(x,t) > u^-(x,t)$

$u^+(x,t)$ et $u^-(x,t)$ étant des fonctions continues.

La mise en oeuvre d'un contrôleur par mode glissant $u(x,t)$ peut se faire en trois étapes :

- Choix de la surface glissante,
- Etablissement des conditions d'existence et convergence,
- Etablissement de la loi de commande.

La technique des modes glissants, du fait de leur caractère discontinu, apparaît être une bonne solution pour stabiliser un système non-holonomes.

Dans le cas des robots mobiles, nous avons l'équation suivante :

$$\dot{x} = g_1(x)u_1 + g_2(x)u_2 \tag{II.8}$$

Ce système peut être transformé en une autre forme (ordre 3) donnée comme suit :

$$\begin{cases} \dot{z}_1 = v_1 \\ \dot{z}_2 = v_2 \\ \dot{z}_3 = z_2 v_1 \end{cases} \tag{II.9}$$

Dans le cas du régime glissant, le retour d'état stabilisant à l'origine correspondant à ce système peut être donné comme suit :

$$\begin{aligned} v_1 &= -z_1 - \alpha_1 z_2 sign(s) \\ v_2 &= -z_2 + \alpha_1 z_2 sign(s) \end{aligned} \tag{II.10}$$

Avec $\alpha_1 > 0$, et $s = z_3 - \frac{z_1 z_2}{2}$ représente la surface de glissement.

III.4 Stabilisation par retour d'état hybride

La stabilisation hybride est principalement constituée de retours d'état variants dans le temps, continu ou non, mais dont la spécificité est que tout ou une partie de la dépendance par rapport à l'état n'est pas réinitialisée que périodiquement et non pas de manière continue, comme c'est le cas des retours d'état variants dans le temps.

IV. Approche adoptée pour la stabilisation des robots mobiles

IV.1 Commande par retour d'état discontinu invariant

Résoudre le problème de stabilisation consiste à trouver un contrôleur feedback adéquat permettant au robot mobile d'atteindre une configuration statique à partir de n'importe quelle configuration initiale.

Pour cela, nous avons choisi un type de commande qui nous semble adapté pour stabiliser un système non-holonome. Il s'agit de la commande par retour d'état discontinu invariant.

Dans ce qui suit, nous élaborons deux techniques génériques de commande par retour d'état discontinu. L'objectif est d'observer le comportement du robot lors de son mouvement et ainsi choisir la commande la plus sophistiquée pour stabiliser le robot.

IV.1.1 Stabilisation d'un système non-holonome en forme chaînée

Dans la plupart des cas, l'application d'une loi de commande non linéaire est devenue complexe et onéreuse lorsqu'il s'agit de contrôler un système non-holonome. Pour ce faire, les automaticiens ont réfléchi à simplifier la modèle du système en lui donnant une forme beaucoup plus simple à contrôler appelée « forme chaînée » [CAN 95], [REY 95], [TSI 93].

L'avantage de cette nouvelle représentation est de faciliter le calcul d'une loi de commande applicable à n'importe quel ordre du système (jusqu'à n-dimension) [TAY 97a].
Dans ce cas, pour stabiliser un robot mobile, la première étape consiste à transformer son modèle cinématique en forme chaînée d'ordre 3. Cette dernière est exploitée pour élaborer une loi de commande stabilisante.
La loi de commande établie est composée de deux principaux éléments : un retour d'état linéaire et un retour d'état discontinu.

IV.1.2 Stabilisation par la technique « feedback linearization »

D'autres techniques de commande peuvent être utilisées pour stabiliser un système non holonome. Parmi ces techniques, nous citons celles issues de « la linéarisation par bouclage ».
Cette méthode consiste à boucler le système par une loi de commande u pour aboutir à un système linéaire.
Le système linéaire résultant est ensuite commandé par un retour d'état pour réaliser la stabilisation.

La linéarisation par bouclage utilise des techniques non linéaires pour établir une loi de commande linéarisante. Citons, à titre d'exemple, les crochets de Lie, les notions de degrés relatifs, la forme normale, etc [GUE 05], [ZER 06], [LAM 02].

Nous développons dans le chapitre IV les différents algorithmes de commandes issues de ces deux techniques pour stabiliser un système non-holonome. Les performances de chacune des techniques seront ensuite étudiées à travers les résultats de simulation obtenus.

V. Stabilisation de robots mobiles par poursuite de trajectoire

Les techniques de commande présentées précédemment permettent de définir les trajectoires admissibles pour stabiliser un système non-holonome. En pratique, ces techniques sont implémentées au niveau du système pour réaliser la stabilisation.
Cependant, cette démarche n'est pas aussi consistante car elle peut provoquer des irrégularités au niveau du comportement du système. Ceci est dû à une absence totale d'un système de perception de l'environnement qui permet une meilleure connaissance de l'environnement dans lequel évolue notre système.
Pour remédier à ce problème, nous avons adopté une nouvelle approche pour stabiliser notre système. Il s'agit d'utiliser les méthodes de vision par ordinateur pour permettre au système d'atteindre la position d'origine. Cette approche nécessite la connaissance, au préalable, de la trajectoire stabilisante.

V.1 Poursuite de trajectoire stabilisante par vision

Avec l'avènement de l'imagerie et de la vision par ordinateur, les spécialistes se sont orientées vers les théories et techniques de perception [TOU 90]. Ces dernières offrent des avantages considérables lorsque nous envisageons de savoir d'une façon instantanée l'état du robot et d'obtenir les informations nécessaires sur son environnement.

Le domaine de l'imagerie nous semble intéressant pour son utilisation dans la robotique mobile [BOU 07], [TOU 90]. Comme il est connu, le sens de la vue est celui qui nous rapporte la plus grande quantité d'informations sur l'environnement qui nous entoure.
Nous utilisons ce principe dans la robotique mobile car il nous offre la possibilité de percevoir notre espace de travail, de reconnaître les différents objets dans la scène, de détecter les différents mouvements, etc.
Dans notre cas, nous faisons appel aux méthodes de traitement d'images pour permettre à notre robot mobile de suivre une trajectoire donnée. Cette trajectoire peut être obtenue par les différentes techniques de stabilisation présentées précédemment.

V.1.1 Problématique

Le problème de stabilisation consiste à calculer pour un robot, la trajectoire admissible joignant une configuration initiale à une configuration finale qui constitue l'origine.
La trajectoire correspondante est calculée à base des techniques de commande par retour d'état présentées dans la section (IV). Une fois la trajectoire connue, il est demandé au robot d'effectuer l'opération de poursuite. Cette dernière est effectuée en faisant appel aux techniques de vision par ordinateur.
Pour résumer l'ensemble des tâches que doit effectuer le robot nous présentons ci-dessous un organigramme général représentant toute les étapes de stabilisation du robot :

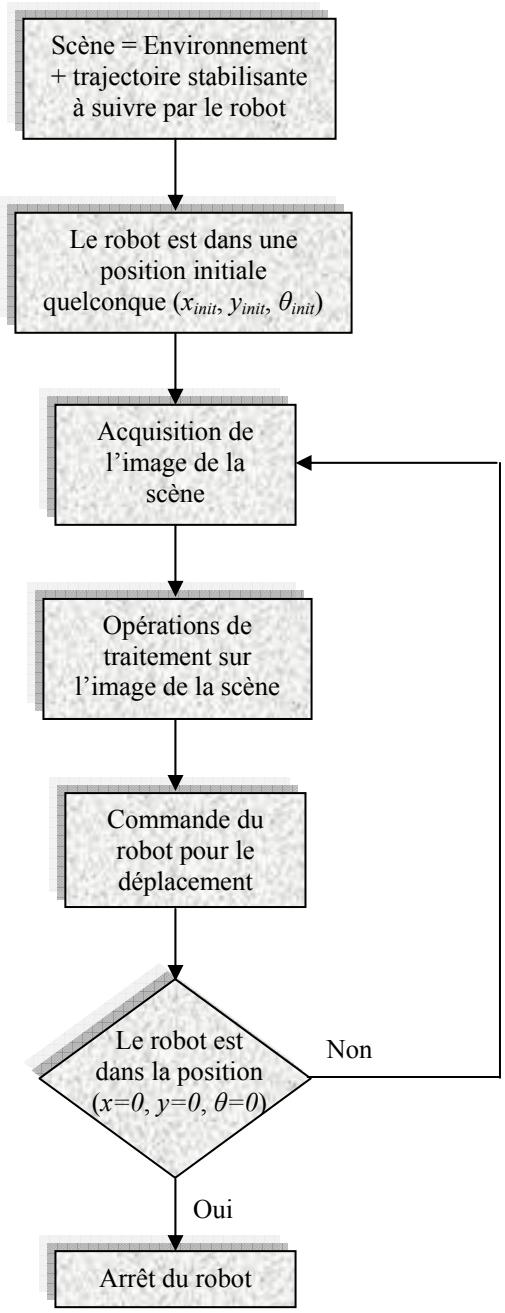


Figure II.1.Organigramme général de l'approche de stabilisation de robot mobile par vision

Dans un premier temps, nous mettons le robot dans une scène constituée de l'environnement qui l'entour en plus de la trajectoire stabilisante (dessinée sur le sol).

Le rebot est muni d'une caméra qui détecte une image de la scène contenant la trajectoire en question.

Différentes opérations de traitement d'images sont ensuite effectuées sur l'image acquise par la caméra pour calculer la commande nécessaire au déplacement du robot. Cette opération est répétée en boucle jusqu'à ce que le robot atteigne le point d'origine ($x=0$, $y=0$, $\theta=0$).

Ainsi, le robot qui été dans une position initiale (x_{init}, y_{init}, θ_{init}) tente de se déplacer le long de la trajectoire pour atteindre le point final.

Nous avons exposé, ici, la démarche générale de l'approche « poursuite de trajectoire par vision ». Le chapitre V présentera plus en détail le contenu de chaque étape et les résultats expérimentaux correspondants.

VI. Conclusion

Dans ce chapitre nous avons présenté notre problématique, à savoir appliquer à un robot mobile un contrôle pour le stabiliser.

Nous avons, en premier lieu, balayer les différentes techniques de stabilisation existantes (commande par retour d'état continu, commande par retour d'état discontinu et la commande hybride). Nous avons, ensuite, opté pour deux techniques appartenant à la commande par retour d'état discontinu.

L'étape suivante consiste à appliquer les techniques précédentes au niveau de notre robot. Nous avons choisi, dans notre cas d'étude, les techniques de vision par ordinateur qui procurent beaucoup plus de performances lors de la stabilisation d'un robot.

Pour mieux comprendre cette étape, nous passons en revue, dans le prochain chapitre, les différentes techniques de traitement d'images les plus utilisées dans la robotique mobile et surtout dans le cas de poursuite de trajectoire.

Chapitre III
Introduction à la vision et son utilisation en robotique mobile

Chapitre III Introduction à la vision et son utilisation en robotique mobile

I. Introduction

L'avènement des robots de la troisième génération capables de percevoir leur environnement et de réagir à celui-ci a fait faire un bond en avant aux théories et techniques de la perception. Parmi celles-ci, la vision s'est affirmée d'emblée comme un domaine de recherche privilégié, et ce pour plusieurs raisons.

L'arrivée sur le marché d'ordinateurs de plus en plus performants a rendu possible ce qui n'était qu'une vue de l'esprit il y a quelques années. Des progrès de la technique est née la vision par ordinateur dans laquelle vient s'intégrer le traitement d'images.

Le traitement d'images présente un domaine de recherche très prometteur en robotique et plus particulièrement la robotique mobile. Ces dernières années, plusieurs chercheurs se sont intéressés à commander un robot mobile en utilisant les techniques de vision. Ceci se justifie par le fait que les méthodes traditionnelles qui consistent à implémenter les techniques de commande automatique ont montré leurs limites lorsque on se réfère à un environnement contraint ou difficilement accessible. En outre, les techniques de traitement d'images donnent des résultats plus performants.

Nous nous intéressons, alors, à intégrer l'aspect traitement d'images dans la commande de robots mobiles. Notre objectif est de permettre au robot de suivre une trajectoire donnée en utilisant une caméra pour le traitement de l'information sous forme d'images.

Nous allons présenter, tout au long de cette partie, quelques notions et principes du traitement d'images utilisés en robotique mobile. On va s'attarder, un peu, sur des notions de détection de contours, de calibrage des caméras, etc, notions très essentielles pour la suite de notre travail.

II. Notion d'imagerie

Dans le monde informatique on s'intéresse à manipuler des images issues de l'environnement réel. Ceci est rendu possible par l'émergence de nouvelles notions d'imagerie appelée « imagerie numérique ».

III. Fondement de l'imagerie numérique

Une image numérique est composée d'unités élémentaires (appelées pixels) qui représentent chacune une portion de l'image.

Une image est définie par :

→ Le nombre de pixels qui la compose en largeur et en hauteur (qui peut varier presque à l'infini),

→ L'étendu des teintes de gris ou des couleurs que peut prendre chaque pixel (on parle de dynamique de l'image).

Toutes les données correspondantes aux informations contenues dans l'image sont structurées d'une certaine façon afin de permettre leur stockage. Ceci donne naissance à plusieurs formats de l'image concernée.

III.1 Numérisation d'image

Le passage du monde réel au monde informatique dans le domaine de l'imagerie, nécessite une transformation appelée « numérisation ». Ceci est indispensable par la suite pour pouvoir manipuler les images résultantes par des calculateurs.

III.2 Principes généraux de la numérisation

La numérisation consiste au passage du monde réel (infini) au monde discret (fini). Le domaine de définition d'une image numérique est donc le monde discret (monde des entiers), contrairement à tout ce qui nous entoure (dont le domaine de définition est le monde des réels).

En effet, les ordinateurs sont incapables de travailler dans le monde des réels (infini). La discrétisation est alors nécessaire. Il est indispensable de garder à l'esprit que cette opération consiste en une simplification de l'information d'origine.

Plusieurs façons existent pour numériser une image. Ceci dépend de son utilisation et des outils matériels et logiciels disponibles.

Dune façon générale, la numérisation dépend de deux paramètres essentiels :

- le nombre de signaux qu'on peut enregistrer dans un espace à deux dimensions x et y,
- la dynamique du signal qu'on peut enregistrer (nombre de niveaux de gris ou de couleurs).

III.3 Matériels utilisés en numérisation

- **Les scanners**

Ce sont des outils qui permettent, en général, de construire une image numérique d'un

document (feuille, revue, livres, photos...) et ceci sous plusieurs formats.

Plusieurs types de scanners existent. A titre d'exemple nous citons : le scanner à main, le scanner à plat, le scanner à diapositives,…

- **Les appareils photos numériques**

Il s'agit d'appareils qui renferment un système de numérisation interne. Pour la plupart, ils sont les équivalents, de point de vue optique, de leurs homologues classiques. Les capteurs arrivent maintenant à un nombre de pixels de l'ordre de 7 millions, ce qui permet une qualité d'image assez surprenante.

- **Les caméras et Webcams**

Il est également possible d'utiliser une caméra analogique reliée à une carte d'acquisition située dans un ordinateur. Cette solution était très utilisée avant l'avènement des appareils photos numériques.

Les caméras sont fréquemment utilisées en robotique mobile. Elles sont embarquées de telle sorte qu'elles peuvent percevoir l'environnement réel et donner les informations nécessaires pour un traitement ultérieur.

III.4 Format d'une image

L'image numérisée peut être enregistrée sous plusieurs formats (en fonction de son utilisation). Le format d'une image comprend en général un en-tête qui contient des données concernant l'image (taille en pixels par exemple). La structuration des données est différente pour chaque format. Un tableau récapitulatif donné en Annexe A illustre les formats les plus couramment utilisés.

Après avoir présenté des notions sur l'imagerie et les dispositifs utilisés pour leur capture et leur numérisation, nous allons nous focaliser dans ce qui suit sur les différentes méthodes de traitement d'images existantes dans la littérature et surtout celles utilisées en robotique mobile.

Le but est d'illustrer les différents traitements que subit une image pendant le mouvement du robot ou lors d'une réalisation d'une tâche de poursuite de trajectoire.

IV. Principales techniques de vision utilisées en robotique mobile

IV.1 Techniques de traitement d'images

IV.1.1 Technique de détection de contours

La détection de contours, également appelée extraction de contours ou de bords, consiste à détecter les frontières qui séparent les régions homogènes de l'image. Ce sont généralement des modifications de caractéristiques comme les textures, les couleurs ou l'intensité.

Dans un cadre général, les contours peuvent être obtenus par une opération de seuillage. Cependant, l'utilisation d'un seuil unique peut faire apparaître des lacunes dues aux bruits. Ce traitement peut être amélioré au moyen d'un seuillage par hystérésis qui réduit ces perturbations (approche par filtre optimal).

Dans un cas général, l'association de filtres lisseurs et dérivateurs à l'opération de seuillage permet d'obtenir une structure classique d'une chaîne de détection de contours (voir figure ci-dessous).

Figure III.1. Schéma de principe d'une chaîne de détection de contours

Plusieurs techniques de détections de contours peuvent être recensées dans notre cas, nous allons présenter celles qui semblent les plus intéressantes à notre travail.

IV.1.1.1 Approche par convolution

L'image étant un échantillon d'une fonction réelle, elle est considérée comme une matrice dont chaque coefficient représente un point de l'image. Chaque point de l'image a donc un indice de ligne ainsi qu'un indice de colonne. Pour obtenir les contours dans l'image, il suffit donc de lui appliquer deux filtres matriciels (selon les axes horizontaux et verticaux) et de calculer une norme.

Cependant, cette méthode est sensible au bruit, ce qui fait apparaître des discontinuités qui seront considérablement accentuées par la transformation.

Rappel

Effectuons tout d'abord un rappel sur le filtrage et la convolution :

Filtrer signifie convoluer une image $I(x,y)$ avec une fonction $h(x,y)$ appelée *réponse impulsionnelle* (ou *opérateur de convolution*) du filtre. Dans le cas continu, l'image filtrée est donnée par :

$$I_h(x,y) = (h * I)(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(x',y')I(x-x',y-y')dx'dy' \qquad \text{(III.1)}$$

Dans le cas discret et pour prendre l'exemple simplifié d'une image carrée, les domaines de I et de h sont bornés. Le domaine de I est $[-N/2,\ N/2]^2$ si N désigne la taille de l'image et le domaine de h est $[-K/2,K/2]^2$ avec nécessairement $K \le N$. La convolution s'écrit alors :

$$I_h(x,y) = (h * I)(x,y) = \sum_{-N/2}^{N/2}\sum_{-K/2}^{K/2} h(i-i',j-j')I(i',j') \qquad \text{(III.2)}$$

On notera que le filtrage linéaire consiste simplement à remplacer chaque niveau de gris par une combinaison de gris des points voisins.

On définit dans ce cas des filtres représentant les différentes opérations que l'on veut réaliser, pour cela il est possible de faire varier la taille de la matrice, ainsi que les coefficients pondérateurs des points de l'image.

L'opération de convolution consiste à faire passer le filtre correspondant par toute l'image (en verticale ou en horizontale). L'image résultante contient l'ensemble des contours en question.

IV.1.1.2 Approche par dérivée première

Les filtres utilisés ici sont les filtres de dérivée première (appelés aussi filtres étroits ou filtre de gradient) et l'on cherche alors le maximum de leur réponse. Cependant, le filtre dérivé accentue le bruit (pixels parasites de répartition aléatoire). Pour remédier à ce problème, des filtres dérivés plus robustes ont été proposés (ils sont présentés plus bas). Le gradient est une dérivation premier ordre, il est donné par la formule :

$$\vec{\nabla} I = \frac{\partial I}{\partial x}\vec{I}_x + \frac{\partial I}{\partial y}\vec{I}_y \qquad \text{(III.3)}$$

Où I_x (resp I_y) est un vecteur unitaire suivant *x* (resp. suivant *y*). Le gradient au niveau d'un pixel d'une image est un vecteur caractérisé par son amplitude et sa direction. L'amplitude est directement liée à la quantité de variation locale des niveaux de gris. La direction du gradient est orthogonale à la frontière qui passe au point considéré.

L'approche la plus classique pour estimer le gradient consiste à choisir deux directions privilégiées (naturellement celles associées au maillage, *i:e:* ligne et colonne) orthogonales sur lesquelles on projette le gradient considéré.

On peut donc obtenir une connaissance parfaite du gradient de l'image qui se calcule comme suit :

$$\nabla_x = \frac{\partial I(x,y)}{\partial x} \text{ et } \nabla_y = \frac{\partial I(x,y)}{\partial y} \qquad \text{(III.4)}$$

Ainsi, en chaque point (*x, y*) de l'image, on définit deux dérivées partielles, suivant *x* et suivant *y*. ensuite, on calcul la dérivée de *I* dans une direction *d* à partir des deux dérivées directionnelles définissant le gradient I_x et I_y de la manière suivante : $L_d = \nabla I.d$

Etudions maintenant trois filtres (dont deux du même type) utilisant la méthode du gradient pour un filtrage des contours.

- **Opérateurs de Sobel et Prewitt (1970)**

Les opérateurs de Sobel et de Prewitt permettent d'estimer localement la norme du gradient spatial bidimensionnel d'une image en niveau de gris. D'une façon générale, les opérateurs de Sobel et Prewitt sont décrits sous forme de paire de masques de convolution 3 x 3 dont une rotation de 90° permet de passer d'un masque de convolution à l'autre.

L'application séparée de chacun des masques donne une estimation des composantes horizontales et verticales du gradient par un simple filtrage linéaire avec un masque 3 x 3.

Il est ensuite possible de calculer la norme et la direction du gradient en chaque point à partir des composantes des gradients ∇_x et ∇_y par la norme euclidienne :

$$\|\nabla\| = \sqrt{\nabla_x^2 + \nabla_y^2} \tag{III.5}$$

Ainsi que son orientation :

$$\theta = \arctan\left(\frac{\nabla_x}{\nabla_y}\right) - \frac{3\pi}{4} \tag{III.6}$$

Pour obtenir un algorithme plus rapide, cette norme est généralement approchée par :

$$\|\nabla\| \simeq |\nabla_x| + |\nabla_y| \tag{III.7}$$

La norme du gradient ainsi estimée correspond à l'intensité attribuée au pixel courant. C'est donc l'image de la norme du gradient que l'on visualise généralement. Dans notre cas les gradients ∇_x et ∇_y peuvent être représentés sous la forme d'une matrice (3x3). Les opérateurs de Sobel et de Prewitt dépendent de la valeur attribuée à la constante *k*, figure II.2.

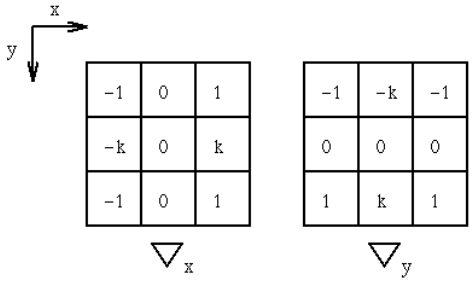


Figure III.2. Masques de convolution des opérateurs de Prewitt (*k*=1) et de Sobel (*k*=2).

- **Opérateur de Roberts (1965)**

Le détecteur de Roberts permet de calculer le gradient bidimensionnel d'une image de manière simple et rapide. Il amplifie les zones (qui correspondent souvent aux contours) où la norme du gradient spatial est importante.

Ici, l'opérateur est constitué de deux masques de convolution (2x2). Le module est calculé par la norme du vecteur composé par les deux composantes de la dérivée (selon des directions diagonales).

Ce principe ne diffère pas beaucoup de celui des opérateurs de Prewitt et de Sobel. De la même manière, chaque masque est obtenu à partir de l'autre par une rotation de 90° (figure III.3).

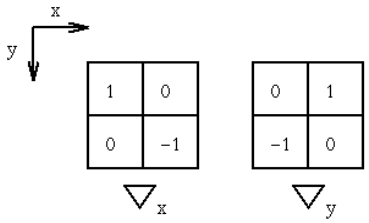


Figure III.3. Masques de convolution de l'opérateur de Roberts.

IV.1.1.3 Approche par dérivée seconde

Nous venons de voir des filtres basés sur la recherche de maxima de la dérivée première. Les filtres larges que nous allons maintenant étudier recherchent les zéros de la dérivée seconde (méthode permettant de bien mettre en évidence les maxima de la dérivée première) ou, plus précisément, le Laplacien qui est une dérivation au deuxième ordre avec :

$$\Delta I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \qquad \text{(III.8)}$$

Dans le cas d'une image, il n'existe pas une dérivée seconde unique mais quatre dérivées partielles (selon x^2, y^2, xy et yx). En pratique, on lève cette ambiguïté en ayant recours à l'opérateur Laplacien qui fait la somme des deux dérivées partielles.

L'estimation de la dérivée seconde étant très sensible aux bruits, il convient de filtrer l'image avant d'en mesurer le Laplacien.

Marr [SLI 05], [TOU 90], [JOL 01] a montré qu'avec de tels filtres gaussiens, on pouvait approcher de très près les effets donnés par le système visuel humain. Le filtre obtenu par convolution avec le Laplacien d'une gaussienne est connu sous le nom de LOG (*Laplacian of Gaussian*).

Un autre filtre utilisé et très proche du LOG est le détecteur Laplacien DOG (*Différence of Gaussians*) de Marr et Hildreth [SLI 05], [JOL 01] (années 80) qui repose sur l'idée que le Laplacien peut être vu comme la différence entre deux lissages gaussiens de tailles différentes. L'image de contour est obtenue comme la différence entre une image peu lissée et une image fortement lissée.

Il est à noter que les filtres LOG et DOG n'ont plus aujourd'hui qu'un intérêt historique et n'ont été présentés ici que pour l'introduction du Laplacien.

IV.1.1.4 Approche par filtrage

Nous présentons, ici, une approche qui permet une bonne détection de contours. On les voit émerger dans les années quatre-vingt-cinq (1985).

Dans un premier temps, Canny [SLI 05], [MAL 07] considère que le signal du contour observé est modélisable comme la somme d'un échelon unitaire (ou fonction de Heavyside) de hauteur A et d'un bruit gaussien n de moyenne nulle et de variance n_0^2 (voir figure ci-dessous) :

$$I(x) = AU(x) + n(x) \tag{III.9}$$

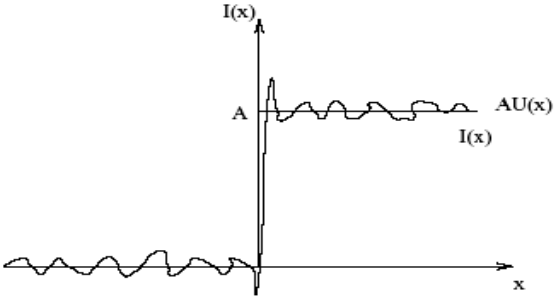


Figure III.4. Modélisation d'un contour par la somme d'un échelon et d'un bruit gaussien de moyenne nulle.

Les qualités attendues d'un filtre de détection de contours sont : une bonne détection des contours c'est-à-dire une réponse forte même à de faibles contours, une bonne localisation de ceux-ci et une faible multiplicité des maxima dus au bruit. Ces trois critères proposés par Canny pour définir le filtre linéaire optimal.

Dans [LAM 01], l'auteur avait définit deux autres critères, toujours monodimensionnels, qui sont la bonne détection des contours proches et une épaisseur des contours de un (1) pixel partout dans l'image.

Trouver un filtrage optimal, reviens à calculer h qui maximise la somme $\sum \Lambda$ sous la contrainte x_{max}, ce qui revient à une équation différentielle (on admettra le résultat) :

$$2h(x) = 2\lambda_1 h''(x) + 2\lambda_2 h'(x) + \lambda_3 = 0 \tag{III.10}$$

De solution générale *h(x)*

$$h(x) = a_1 e^{\alpha x}\sin(\omega x) + a_2 e^{\alpha x}\cos(\omega x) + a_3 e^{-\alpha x}\sin(\omega x) + a_4 e^{-\alpha x}\cos(\omega x) \tag{III.11}$$

Il reste maintenant à déterminer les paramètres a_1, a_2, a_3 et a_4 à partir des conditions limites. Dans ce cas d'étude, deux filtres sont mis en œuvre. Il s'agit du filtre de Canny et celui de Deriche.

- **Filtre de Canny (1986)**

Le filtre de Canny [MAL 07] est un filtre impair à réponse impulsionnelle finie (RIF) défini sur l'intervalle [-*M, M*]. Ce qui signifie que *h* est nul en dehors de cet intervalle et que la dérivée est continue.

Le filtre impose également une pente de valeur *S* à l'origine. Avec ces contraintes, on peut ainsi déterminer les quatre coefficients a_i et *w* à partir de la taille du filtre.

Par une optimisation numérique, Canny détermina que $\sum\Lambda = 1{,}12$ est un compromis optimal pour un filtre RIF. Pour une mise en pratique concrète, l'implémentation se réalise à l'aide de la dérivée d'une gaussienne. On obtient alors $\sum\Lambda = 0{,}92$ et $x_{max} = 0{,}51$. Notons que ce filtre donne des résultats de bonne qualité.

- **Filtre de Deriche (1987)**

Par rapport au filtre de Canny, on préfère souvent le détecteur de Deriche qui est un filtre à réponse impulsionnelle infinie (RII) [MAL 07]. Ces contraintes permettent à nouveau de déterminer les coefficients a_i : $a_1 = a_2 = a_4 = 0$ et $w.a_3 = S$ d'où la solution :

$$h(x) = \frac{S}{\omega} e^{-\alpha|x|} \sin(\omega x) \tag{III.12}$$

On obtient alors :

$$\Lambda = \sqrt{2\alpha},\ \Sigma = \sqrt{\frac{2\alpha}{\alpha^2 + \omega^2}} \text{ et } x_{max} = \sqrt{\frac{\alpha^2 + \omega^2}{5\alpha^2 + \omega^2}} \tag{III.13}$$

Ce filtre répond exactement aux mêmes critères de qualité que celui de Canny. Deriche a montré qu'en prenant α très grand devant *w*, on obtient des résultats optimaux : $\sum\Lambda = 2$ et $x_{max} = 0{,}44$. Il a également montré que pour une valeur de x_{max} identique au filtre de Canny, le filtre de Deriche affiche une valeur de performance $\sum\Lambda$ bien meilleure (1,4 au lieu de 1,12).

Implémenter le filtre de Deriche revient à implémenter la convolution de l'image par le filtre *h*. Il est préférable pour cela d'utiliser l'implémentation récursive proposée par Deriche : cette implémentation privilégie la précision numérique et le nombre de calculs par pixel, quelle que soit la taille du filtre (spécifiée par α).

L'implémentation récursive de Deriche s'effectue en deux phases : la première phase traite les lignes de l'image tandis que la seconde traite les colonnes. Cela est possible car les filtres utilisés ici sont séparables.

IV.1.2 Techniques de seuillage

La dernière étape d'une chaîne de détection de contours bas niveau est la segmentation des pixels de l'image en deux ou plusieurs classes. La limite entre chaque classe est caractérisée par une valeur unique, logiquement appelée seuil. L'utilisation de plusieurs seuils caractérise les méthodes de multi-seuillage ou de classification.

IV.1.2.1 Seuillage simple ou binarisation

Ce type de traitement s'apparente à un filtrage tout ou rien. Son principal inconvénient est la difficulté de fixer le seuil unique. Un seuil trop bas peut entraîner l'apparition de contours parasites dus, entre autre, aux bruits et un seuil trop élevé peut provoquer la disparition pur et simple de contours.

IV.1.2.2 Seuillage par hystérésis

Le principe du seuillage par hystérésis est de segmenter l'image en trois classes : fond, contour probable, contour sûr. Les contours sûrs sont plongés le long des lignes de crête formées à partir des contours probables. Les pixels des lignes de crête plongées sont reclassifiés en contours sûrs. Les crêtes non connectées, au moins, à un contour sont alors supprimées (figure III.5).

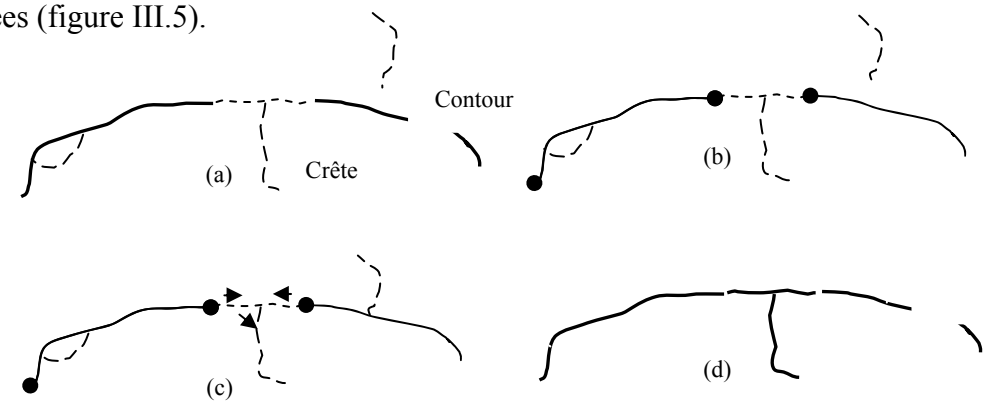


Figure III.5. a) image source, b) détection des extrémités, c) crêtes propagées, d) image après propagation.

Après avoir présenté les différentes techniques de traitement d'images les plus utilisées en robotique mobile, nous abordons le principe de calibrage (ou calibration) de caméra. Ce dernier constitue une étape importante pour la vision par ordinateur.

IV.2 Calibrage de camera

La calibration de caméra s'avère indispensable lorsque l'on souhaite localiser ou reconstruire une scène tridimensionnelle à partir d'une seule image.

Dans la suite de cette section, nous décrivons les différents outils de calibrage. Pour cela, nous

commençons par décrire un type de caméra, le plus utilisé en calibrage, appelé « caméra CCD » ainsi que son fonctionnement. Par la suite, nous présentons le principe de modélisation de la caméra ainsi que son calibrage.

IV.2.1 Caméra CCD

Une caméra CCD (Charge Coupled Device) est, principalement, composée (pour la capture d'image) d'un système optique, d'une matrice CCD et d'un dispositif électronique d'acquisition et de conversion numérique (figure III.6).

Le capteur CCD est composé d'un ensemble de petits éléments, les pixels, de formes carrées ou rectangulaires [SLI 05], [TOU 90].

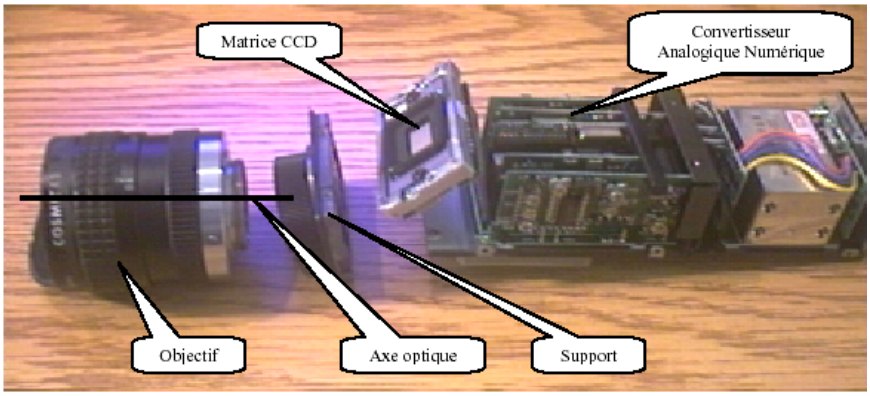


Figure III.6. Caméra CCD

IV.2.2 Modélisation d'une caméra

Plusieurs modèles décrivant le processus de calibrage existent. Le plus simple est le modèle du sténopé ou modèle *pin-hole* dans la littérature anglo-saxonne. Ce dernier est couramment utilisé en traitement d'image car il permet de simplifier considérablement les calculs.

IV.2.2.1 Modèle sténopé

Il s'agit d'une modélisation simple et linéaire du processus de formation des images au sein d'une caméra. Ce modèle suppose que le système optique de la caméra, c'est-à-dire sa lentille respecte les conditions de Gauss. Si l'on utilise la notation matricielle des coordonnées homogènes, il est possible de décrire de manière simple ce processus. Il suffit d'exprimer les relations de passage du repère monde au repère caméra, d'exprimer la projection du repère caméra dans le plan image et d'appliquer la transformation affine qui conduit aux coordonnées de l'image.

Les paramètres employés dans ce modèle sont usuellement divisés en deux catégories : les **paramètres intrinsèques** qui sont internes à la caméra, et les **paramètres extrinsèques** qui peuvent varier suivant la position de la caméra dans l'espace de travail.

Soit deux repère : le repère univers et le repère caméra. Ils sont composés d'un ensemble d'axes orthogonaux deux à deux formant un trièdre.

Soit un point *M* défini dans un espace tridimensionnel par ses coordonnées (x_i, y_i, z_i) qui sont des composantes du vecteur O_iM. Soit *c* l'indice du repère caméra et *w* l'indice du repère univers.

Les coordonnées du point *M* sont : *x* (l'abscisse), *y* (l'ordonnée) et *z* (l'altitude). La modélisation géométrique de la caméra est obtenue par un changement de repère suivi d'une projection perspective conique.

L'image d'un objet est obtenue par une projection perspective (figure III.7). C'est la représentation du modèle sténopé qui consiste à modéliser l'ensemble des lentilles, appartenants système optique, par un point où convergent tous les rayons lumineux.

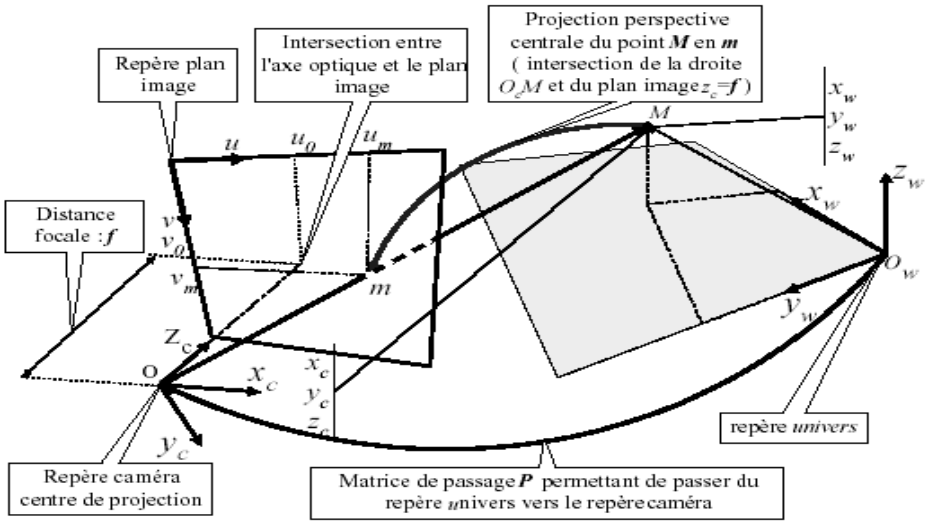


Figure III.7. Changement de repère et projection perspective (principe du Sténopé).

Ainsi, le point M est défini dans plusieurs repères :

- Le repère univers, où se situe le point $\boldsymbol{M}$, a pour coordonnées ($\boldsymbol{o_w}$, $\boldsymbol{x_w}$, $\boldsymbol{y_w}$, $\boldsymbol{z_w}$).
- Le repère caméra, où se situe le point $\boldsymbol{M}$, a pour coordonnées ($\boldsymbol{o_c}$, $\boldsymbol{x_c}$, $\boldsymbol{y_c}$, $\boldsymbol{z_c}$).
- Le repère image, où se situe le point $\boldsymbol{m}$ projection du point $\boldsymbol{M}$, de coordonnées ($\boldsymbol{u_m}$, $\boldsymbol{v_m}$).
- L'axe $\boldsymbol{z_c}$ est l'axe optique de la caméra.

On suppose que l'axe optique ($\boldsymbol{z_c}$) est orthogonal au plan de projection et que le plan image est un parallélogramme. Le plan de projection est donc situé à une distance $\boldsymbol{f}$ (distance focale) de l'origine O_c, centre de projection, du repère caméra.

L'intersection de l'axe optique avec le plan de projection a pour coordonnées ($\boldsymbol{u_0}$, $\boldsymbol{v_0}$) appelées également centre du plan image. L'axe optique n'est pas la normale du plan image car il est généré par le système optique.

Calcul des paramètres extrinsèques de la caméra [SLI 05]

Le point ***M*** est défini par ses coordonnées (x_w, y_w, z_w) dans le repère univers. Il est également défini par ses coordonnées (x_c, y_c, z_c) dans le repère caméra (figure III.8).

Le passage d'un repère à un autre est défini par la notation homogène suivante :

$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = P. \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \qquad \text{(III.14)}$$

P est la matrice de transformation homogène du repère *univers* vers le repère *caméra.* ***P*** est composée d'une matrice de rotation notée **R** et d'une matrice de translation notée **T**. Ainsi trois rotations distinctes forment la matrice de rotation **R** qui décrivent respectivement ***le Roulis, le Tangage et le Lacet***.

- Rotation autour de l'axe x_c par l'angle α,
- Rotation autour de l'axe y_c par l'angle β,
- Rotation autour de l'axe z_c par l'angle γ.

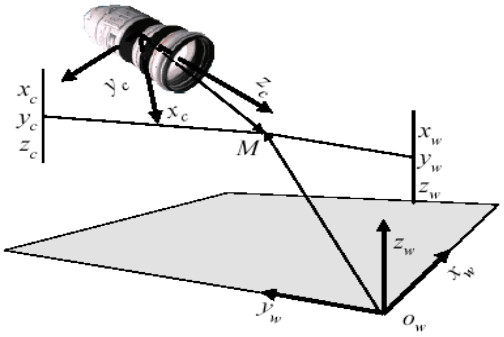
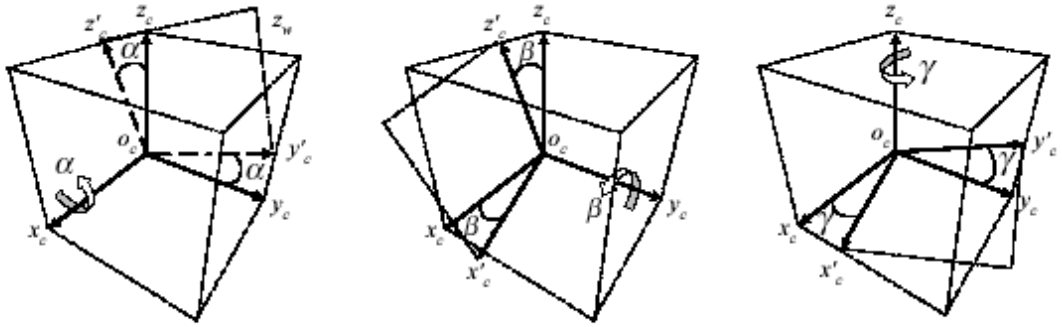


Figure III.8. Changement de repère et projection perspective (Sténopé)

- **Calcul de la matrice de rotation**

Un point dans l'espace peut être défini par des coordonnées cartésiennes, cylindriques ou sphériques. En général le système choisi est le système cartésien permettant de désigner une rotation souvent par le **Roulis** : $0 \le \alpha \le 2\pi$, le **Tangage** $0 \le \beta \le 2\pi$ et le **Lacet** : $0 \le \gamma \le 2\pi$ (Figure III.9). Les angles de Cardan décrivent assez bien la décomposition d'un mouvement naturel et complet.



$$x'_c = x_c \qquad y'_c = y_c\cos\alpha - z_c\sin\alpha \qquad z'_c = y_c\sin\alpha + z_c\cos\alpha$$

soit :

$$R_{(x,\alpha)} = \begin{bmatrix} 1 & 0 & 0 \\ \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \end{bmatrix}$$

$$x'_c = x_c\cos\beta + z_c\sin\beta \qquad y'_c = y_c \qquad z'_c = -x_c\sin\beta + z_c\cos\beta$$

soit :

$$R_{(y,\beta)} = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}$$

$$x'_c = x_c\cos\gamma - y_c\sin\gamma \qquad y'_c = y_c\sin\gamma + y_c\cos\gamma \qquad z'_c = z_c$$

soit :

$$R_{(z,\gamma)} = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Figure III.9. Rotation selon Roulis, Tangage et Lacet

La matrice d'orientation est obtenue par la multiplication des trois matrices :

$$R_{(\alpha,\beta,\gamma)} = R_{(z,\gamma)} + R_{(y,\beta)} + R_{(x,\alpha)} \tag{III.15}$$

On obtient par développement :

$$R_{(\alpha,\beta,\gamma)} = \begin{bmatrix} \cos\beta\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \cos\beta\sin\gamma & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma \\ -\sin\beta & \sin\alpha\cos\beta & \cos\alpha\cos\beta \end{bmatrix} \tag{III.16}$$

- **Calcul de la matrice de translation**

Les translations sont caractérisées par la matrice ***T*** qui est représentée sous la forme suivante :

$$T_{(t_x,t_y,t_z)} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{III.17}$$

La transformation d'une rotation utilisant la matrice de rotation et de translation peut s'écrire sous la forme homogène suivante :

$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\beta\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma & t_x \\ \cos\beta\sin\gamma & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & t_y \\ -\sin\beta & \sin\alpha\cos\beta & \cos\alpha\cos\beta & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \tag{III.18}$$

La matrice ***P*** est composée de paramètres extrinsèques issus des 3 rotations et des 3 translations :

$$P = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} [R] & [T] \\ 0 & 1 \end{bmatrix} \quad \text{(III.19)}$$

Calcul des paramètres intrinsèques de la caméra [SLI 05]

En projection perspective, la relation entre le point **M** du repère monde et sa projection (le point m) sur le plan image est donnée par la figure ci-dessous :

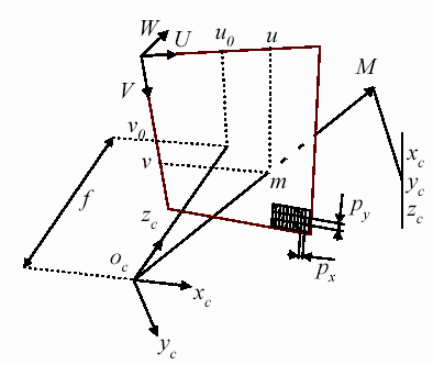


Figure III.10. Relation entre le repère monde et le repère image

La projection perspective d'un point sur le plan image, établie à partir de la loi des triangles semblables, est décrite par l'expression canonique suivante :

$$u = u_0 + f \cdot \frac{x_c}{p_x \cdot z_c} \quad \text{(III.20)}$$

$$v = v_0 + f \cdot \frac{y_c}{p_y \cdot z_c} \quad \text{(III.21)}$$

Soit en notation matricielle :

$$\begin{bmatrix} U \\ V \\ W \end{bmatrix} = \begin{bmatrix} \frac{1}{p_x} & 0 & u_0 \\ 0 & \frac{1}{p_y} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} \quad \text{(III.22)}$$

Avec :

$$u = \frac{U}{W} \text{ et } v = \frac{V}{W} \quad \text{(III.23)}$$

Les coordonnées du repère image sont exprimées en pixel. Pour rendre la transformation homogène, il est nécessaire d'introduire l'unité pixel (p_x, p_y) du système de coordonnées image de la façon suivante :

$$\begin{bmatrix} U \\ V \\ W \end{bmatrix} = \begin{bmatrix} \frac{1}{p_x} & 0 & u_0 \\ 0 & \frac{1}{p_y} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \quad \text{(III.24)}$$

Où (u_0, v_0) sont les coordonnées du centre optique, c'est à dire l'intersection de l'axe optique avec le plan image.

En remplaçant les éléments r_{ij} par leur expression, nous obtenons :

$$\begin{bmatrix} U \\ V \\ W \end{bmatrix} = \begin{bmatrix} \frac{1}{px} & 0 & u_0 \\ 0 & \frac{1}{py} & v_0 \\ 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \cos\beta\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma & t_x \\ \cos\beta\sin\gamma & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & t_y \\ -\sin\beta & \sin\alpha\cos\beta & \cos\alpha\cos\beta & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

(III.25)

En résumé, la transformation du repère univers vers le repère image est donc par la forme matricielle suivante :

$$\begin{bmatrix} U \\ V \\ W \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \cdot \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \quad \text{(III.26)}$$

Avec :

$$\begin{cases} m_{11} = \cos\beta\cos\gamma \cdot \frac{f}{p_x} - \sin\beta \cdot u_0 \\ m_{12} = (\sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma) \cdot \frac{f}{p_x} + \sin\alpha\cos\beta \cdot u_0 \\ m_{13} = (\cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma) \cdot \frac{f}{p_x} + \cos\alpha\cos\beta \cdot u_0 \\ m_{14} = t_x \cdot \frac{f}{p_x} + t_z \cdot u \\ m_{21} = \cos\beta\sin\gamma \cdot \frac{f}{p_y} - \sin\beta \cdot v_0 \\ m_{22} = (\sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma) \cdot \frac{f}{p_y} + \sin\alpha\cos\beta \cdot v_0 \\ m_{23} = (\cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma) \cdot \frac{f}{p_y} + \cos\alpha\cos\beta \cdot v_0 \\ m_{24} = t_y \cdot \frac{f}{p_y} + t_z \cdot v_0 \\ m_{31} = -\sin\beta \\ m_{32} = \sin\alpha\cos\beta \\ m_{33} = \cos\alpha\cos\beta \\ m_{34} = t_z \end{cases}$$

La projection perspective s'obtient par les équations canoniques et homogènes suivantes :

$$u = \frac{U}{W} = \frac{m_{11}.x_w + m_{12}.y_w + m_{13}.z_w + m_{14}}{m_{31}.x_w + m_{32}.y_w + m_{33}.z_w + m_{34}} \quad \text{(III.27)}$$

$$v = \frac{V}{W} = \frac{m_{21}.x_w + m_{22}.y_w + m_{23}.z_w + m_{24}}{m_{31}.x_w + m_{32}.y_w + m_{33}.z_w + m_{34}} \quad \text{(III.28)}$$

Bonneval et Hottier [LAM 01] suggèrent que les équations de colinéarité (III.27 et III.28) dépendent de 11 coefficients inconnus.

$$m_{11}.x_w + m_{12}.y_w + m_{13}.z_w + m_{14} - m_{31}.x_w - m_{32}.y_w - m_{33}.z_w = u.m_{34} \quad \text{(III.29)}$$

$$m_{21}.x_w + m_{22}.y_w + m_{23}.z_w + m_{24} - m_{31}.x_w - m_{32}.y_w - m_{33}.z_w = v.m_{34} \quad \text{(III.30)}$$

La détermination de ces cœfficients nécessite la connaissance des coordonnées d'au moins de six points dans l'image.

Pour la résolution de ce système, il existe plusieurs méthodes (voir [BEN 02]). Parmi ces méthodes citons la technique de Faugeras et Toscani. C'est l'une des techniques de calibrage de caméra la plus souvent utilisée. Son principe peut être décrit comme suit :

A l'aide d'une mire de calibration (voir figures ci-dessous) et à partir des équations de la projection perspective linéaire du modèle sténopé, on extrait les paramètres intrinsèques et extrinsèques par la résolution d'un système de $2\times n$ équations. n est le nombre de points non coplanaires de la mire de calibration. La résolution de ces équations peut être effectué par deux approches :

Une première approche linéaire permet de résoudre le système d'équations en admettant que la distance entre le repère de la caméra et le repère de l'objet de calibration soit ramenée à la valeur "1".

La seconde méthode, très originale, est une approche non linéaire qui utilise, pour résoudre le système d'équations, l'une des contraintes d'orthogonalité de la matrice de rotation qui ne subit pas l'influence des paramètres intrinsèques de la caméra. Cette contrainte se nomme « contrainte de Faugeras-Toscani ».

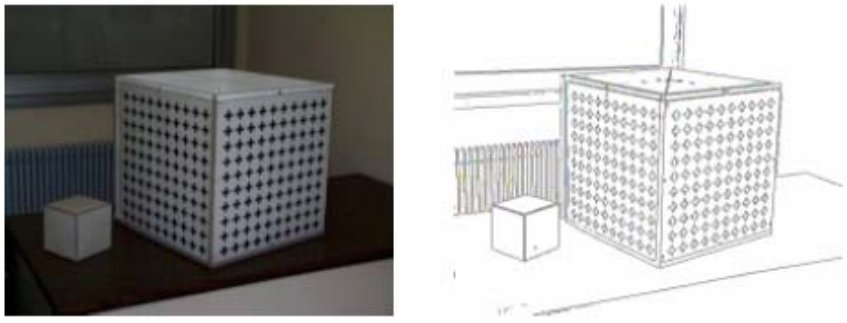


Figure III.11. Mire à base de croix et son image contour correspondante.

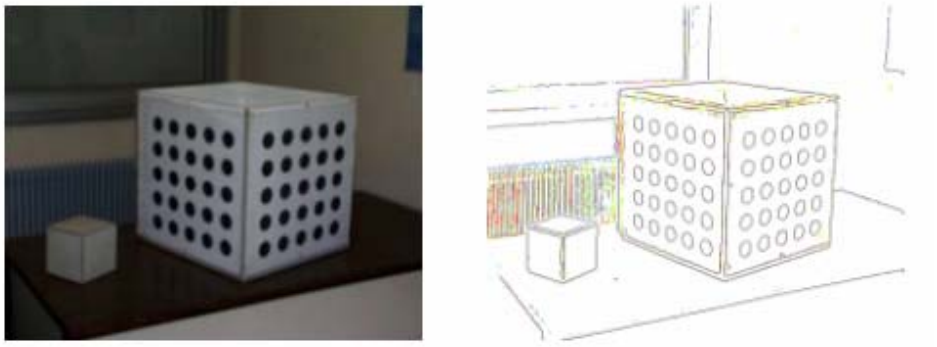


Figure III.12. Mire à base de cercles et son image contour correspondante.

Nous avons décrit dans cette section le principe général d'une caméra numérique, ses composantes ainsi que sa modélisation. Le but n'était pas de définir les différents systèmes existants, leurs capacités et leurs limites mais seulement de présenter, grossièrement, l'une des manière, la plus générale et la plus simple, de modélisation de caméra. Il s'agit dans notre cas du modèle sténopé.

Nous avons, ensuite, présenté l'une des méthodes utilisées pour la définition des paramètres intrinsèques et extrinsèques d'une caméra. C'est la méthode de Faugeras et Toscani.

Il existe, dans la littérature, d'autres techniques comme : le méthode des multi-plans, la méthode de Tsaï, méthode des plans de fuites, méthode à base d'ellipses…La majorité de ces méthodes ont en commun le modèle de représentation géométrique qui est le modèle sténopé mais diffèrent en ce qui concerne l'utilisation du type d'objets (plan, biplan, cube, sphère…) pour réaliser le calibrage.

V. Utilisation des techniques de vision pour la poursuite de trajectoires

Avec l'avènement de l'imagerie et de la vision par ordinateur, les spécialistes de la commande en robotique se sont orientées vers les théories et techniques de perception. Ces dernières offrent des avantages considérables dans le cas des robots mobiles. A travers ces techniques, il est possible de savoir d'une façon instantanée l'état du robot pendant son mouvement.

C'est pour cette raison qu'on s'intéresse à établir une stratégie de poursuite de trajectoire en se basant sur les méthodes de traitement d'images décrites dans ce chapitre.

Notre travail concerne particulièrement la poursuite d'une trajectoire stabilisante par le robot Pekee pour atteindre une position d'équilibre $q_{final}(0, 0, 0)$ à partir de n'importe quelle position initiale $q_{init}(x_{init}, y_{init}, \theta_{init})$.

L'idée générale de notre travail est de déterminer la trajectoire stabilisante, par simulation, en appliquant au robot les techniques de commande présentée précédemment. Une fois connue,

cette trajectoire est tracée sur le sol. Par le biais des méthodes de traitement d'images, le robot, muni d'une caméra, effectue l'opération de poursuite.

VI. Conclusion

Cette partie, de notre rapport, avait été consacrée pour relater quelques notions sur le traitement d'images.

Nous avons commencé par définir, assez rapidement, le principe de l'acquisition et de la numérisation de l'image, les formats existants et les outils de visualisation disponibles.

Nous avons ensuite présenté quelques techniques de traitement d'images que nous jugeons indispensables pour notre travail. Il s'agit des techniques de détection de contours et de seuillage.

Enfin, nous avons abordé le principe de calibrage de caméra en se basant sur quelques outils de modélisation tel que le modèle sténopé.

En somme, nous sommes intéressées au domaine de la vision afin de commander un robot mobile pour effectuer une opération de poursuite de trajectoire. Cette stratégie nécessite la connaissance, au préalable de la trajectoire à poursuivre. Celle-ci est obtenue par les techniques de commande automatique. Ces techniques feront l'objet d'étude dans le prochain chapitre.

Dans le chapitre suivant, nous développons deux techniques de commande stabilisante appliquées au robot mobile unicycle. Les résultats de simulation obtenus nous permettront de choisir la plus performante pour définir la trajectoire et appliquer par la suite le principe de poursuite par vision.

Chapitre IV
Stabilisation de trajectoire pour un robot mobile

Chapitre IV Stabilisation de trajectoire pour un robot mobile

I. Introduction

Nous avons présenté au chapitre II les différentes techniques de commande permettant au robot de se stabiliser à l'origine à partir de n'importe quelle configuration initiale. Nous nous sommes intéressés, à ce stade, à deux techniques de contrôle appartenant à une classe de commande dite à retour d'état discontinu invariant.

Dans la première partie de ce chapitre des algorithmes de stabilisation permettant de générer les trajectoires et les commandes exécutables par les robots mobiles, ont été développés.

Ces algorithmes, applicables en temps réel, ont été conçus dans le but de montrer les performances que présentent les commandes développées pour stabiliser le robot. Aussi, pour observer là réaction du robot, soumis aux effets d'imperfections (erreurs de modélisation) et aux perturbations.

Avant d'introduire les différentes approches de commande stabilisante, nous allons commencer par donner une représentation ou un changement de coordonnées du modèle général représentatif du robot mobile. Une forme qui facilitera l'étude des différentes commandes et leur application au robot.

Pour chacune de ces méthodes, nous présentons des résultats de simulation corroborant nos études théoriques.

II. Stabilisation de trajectoire

II.1 Position du problème

Considérons un robot mobile non holonome, identifié par l'indice i, dont la dynamique est régie par l'équation différentielle suivante [DEF 07] :

$$\dot{q}_i = f_i(q_i, u_i) \tag{IV.1}$$

Où $q_i \in R^n$ représente l'état du système, $u_i \in R^m$ est l'entrée de commande et $f_i : R^n \times R^m \rightarrow R^n$ est une application suffisamment différentiable.

II.2 Description du problème de stabilisation

Le problème de stabilisation consiste à calculer pour le robot, la trajectoire admissible, joignant la configuration initiale $q_i(t_{initial}) = q_{i,initial}$ à la configuration finale $q_i(t_{final}) = q_{i,final}$ = origine (avec des commandes initiales $u_i(t_{initial}) = u_{i,initial}$ et finales $u_i(t_{final}) = u_{i,final}$). La tâche du robot est définie comme l'atteinte de la destination finale $q_i(t_{final})$.

Pour chaque robot, l'objectif est la synthèse d'une commande en boucle fermée $u_i(t)$ assurant sa stabilisation. Cela veut dire, définir un ensemble fini de trajectoires permettant de joindre la position d'origine $q_0(x_0=0,\ y_0=0,\ \theta_0=0)$ à partir de n'importe quelle configuration initiale $q_i(x_i, y_i, \theta_i)$.

III. Synthèse de commande stabilisante

Nous développons, ici, deux techniques de commande stabilisantes pour un système non-holonome. Ces techniques seront appliquées, par la suite, sur un robot mobile unicycle.

III.1 Stabilisation par retour d'état discontinu invariant en forme chaînée

Nous présentons dans cette section une loi de commande stabilisante par retour d'état pour des systèmes non-holonomes. Comme il a été déjà convenu, nous appliquons cette commande à une forme simplifiée et non au modèle cinématique du robot mobile. Cette forme est appelée la forme chaînée [CAN 95], [REY 95], [SAM 95], [TAY 97b].

Nous donnons ci-dessous une brève introduction sur la forme chaînée d'ordre « n » pour ensuite présenter la synthèse de la loi de commande stabilisante correspondante.

III.1.1 La forme chaînée

C'est l'une des manières qui permet de mettre un système sous une certaine forme canonique qui facilite l'analyse et le calcul des contrôles.

L'allure générale de cette forme est donnée par :

$$\begin{cases} \dot{x}_1 = u_1 \\ \dot{x}_2 = u_2 \\ \dot{x}_3 = x_2 u_1 \\ \\ \dot{x}_n = x_{n-1} u_1 \end{cases} \qquad \text{(IV.2)}$$

Où $x = (x_1, x_2, ..., x_n)^T$ représente le vecteur d'état et $(u_1, u_2)^T$ indique le vecteur d'entrée du système (IV.2).

Cette forme est appelée forme **chaînée simple (2, n)** où 2 représente le nombre de contrôle et *n* le nombre d'états.

III.1.2 Algorithme de génération de trajectoire stabilisante

L'objectif de la commande stabilisante est de ramener le robot à une position d'équilibre à partir d'une position quelconque. La trajectoire dans ce cas n'est pas définie au préalable. C'est le type de commande appliquée qui désignera la trajectoire que doit suivre le robot pour atteindre la position d'équilibre souhaitée. Notre but est donc d'établir une loi de commande $(u_1 u_2)^T$ permettant de stabiliser le robot.

Soit un système non-holonome dont la forme chaînée correspondante est donnée par le système (IV.2). Stabiliser le système non-holonome revient à stabiliser sa forme chaînée correspondante.

La stabilisation de notre système, consiste alors à appliquer un contrôleur par retour d'état discontinu invariant $(u_1 u_2)^T$ qui stabilise le système modélisé sous la forme (IV.2).

Dans ce cas, les deux commandes u_1 et u_2 peuvent s'écrire [TAY 97a], [AST 96], [TAY 97b]:

$$\begin{cases} u_1(x) = v_1(x) \\ u_2(x) = v_2(x) + w_2(x) \end{cases} \tag{IV.3}$$

Pour calculer les différents termes de $u_1(x)$ et $u_2(x)$ nous procédons de la manière suivante :

- Posons au départ que $w_2(x)=0$, et considérant le retour d'état linéaire $(v_1(x)\ v_2(x))^T$ tel que :

$$u_1(x) = v_1(x) = -k_1 x_1 \tag{IV.4}$$

$$u_2(x) = v_2(x) = -k_2 x_1 - k_3 x_2 \tag{IV.5}$$

Où $k_1 \in \Re$, k_1 et k_3 sont strictement positifs et $k_1 \neq k_3$.

Le système en boucle fermée peut être intégré, étape par étape, pour obtenir :

$x_1(t) = x_{10} \exp(-k_1 t)$,

$$x_2(t) = \left(x_{20} - \frac{k_2 x_{10}}{K_a} \right) \exp(-k_3 t) + \frac{k_2 x_{10}}{K_a} \exp(-k_1 t),$$

$$x_3(t) = \frac{k_1}{K_b} x_{10} \left(x_{20} - \frac{k_2 x_{10}}{K_a} \right) \exp(-K_b t) + \frac{k_2}{2K_a} {x_{10}}^2 \exp(-2k_1 t) + S_3(x_0),$$

$$x_4(t) = \frac{{k_1}^2 {x_{10}}^2}{K_b(k_1 + K_b)}\left(x_{20} - \frac{k_2 x_{10}}{K_a}\right)\exp(-(k_1 + K_b)t) + \frac{k_2 {x_{10}}^3}{6K_a}\exp(-3k_1 t) + x_{10}S_3(x_0)\exp(-k_1 t) + S_4(x_0)$$

……

$$x_n(t) = \frac{{k_1}^{n-2}{x_{10}}^{n-2}}{K_b(K_b + k_1)...(K_b + (n-3)k_1)}\left(x_{10} - \frac{k_2 x_{10}}{K_a}\right)\exp(-(K_b + (n-3)k_1)t)$$
$$+ \frac{k_2 {x_{10}}^{n-1}}{(n-1)!K_a}\exp(-(n-1)k_1 t) + \sum_{i=1}^{n-3}\frac{{x_{10}}^i}{i!}S_{n-i}(x_0)\exp(-ik_1 t) + S_n(x_0) \qquad \text{(IV.6)}$$

Avec: $K_a=k_1-k_3$, $K_b=k_1+k_3$ et $S_3(x_0)$, $S_4(x_0),...S_n(x_0)$ sont des constantes d'intégration qui peuvent être déterminées, à t=0, comme une fonction des conditions initiales $x_{10}, x_{20},\ldots x_{n0}$.
A partir de (IV.6), nous constatons que ($x_1, x_2, x_3, x_4 \ldots x_n$) tendent vers (0,0, $S_3(x_0)$, $S_4(x_0)$,… $S_n(x_0)$) quand t tend vers l'infini.
Dans ce cas, si nous prenons des conditions initiales tel que $S_j(x_0)=0$ ($3 \leq j \leq n$), tous les états tendent, alors, vers l'origine.

Prenons $S_j(x_0)= S_j$ ($3 \leq j \leq n$). Remplacer t = 0 dans (IV.6) et déterminer $S_j(x_0)$ et ensuite substituer x_0 par x. Les nouvelles expressions de S_j sont données comme suit :

$$\mathrm{S}_3(\mathrm{x}) = x_3 - \frac{k_1}{K_b}x_1 x_2 + \frac{k_2}{2K_b}{x_1}^2,$$

$$\mathrm{S}_4(\mathrm{x}) = x_4 - x_1 x_3 + \frac{k_1}{(k_1 + K_b)}{x_1}^2 x_2 - \frac{k_2}{3(k_1 + K_b)}{x_1}^3,$$

$$\mathrm{S}_5(\mathrm{x}) = x_5 - x_1 x_4 + \frac{1}{2}{x_1}^2 x_3 - \frac{k_1}{2(2k_1 + K_b)}{x_1}^3 x_2 + \frac{k_2}{8(2k_1 + K_b)}{x_1}^4,$$

……

$$S_n(x) = x_n + \sum_{i=1}^{n-3}\frac{(-1)^i {x_1}^i x_{n-i}}{i!} + \frac{(-1)^n k_1 {x_1}^{n-2}}{(K_b + (n-3)k_1)(n-3)!} + \frac{(-1)^{n-1}k_2 {x_1}^{n-1}}{(K_b + (n-3)k_1)(n-3)!(n-1)} \qquad \text{(IV.7)}$$

A partir de (IV.7), il apparaît que si $S_j(x_0)=0$, alors que tout les états tendent à l'origine, puisque x_1 et x_2 décroît exponentiellement vers zéro.
En résumé, la stabilisation de la forme chaînée (IV.2) est équivalent à la stabilisation de ((0,0, S_3, S_4,… S_n) en prenant en compte le paramètre $w_2(x)$.
Dans notre cadre d'étude, $w_2(x)$ est calculé de la façon suivante :

$$w_2(x) = C^T Q(x_1) S \tag{IV.8}$$

Avec :

$$C = \begin{pmatrix} c_3 \\ c_4 \\ \dots \\ c_n \end{pmatrix},\ Q(x_1) = \begin{pmatrix} \frac{1}{x_1} & 0 & .. & .. & 0 \\ 0 & \frac{1}{x_1^{\,2}} & . & . & . \\ . & . & . & . & 0 \\ 0 & . & . & 0 & \frac{1}{x_1^{\,n-2}} \end{pmatrix},\ S = \begin{pmatrix} S_3 \\ S_4 \\ .... \\ S_n \end{pmatrix}$$

Le vecteur C est choisi de telle sorte que la valeur de la matrice $(A+B.C)^T$ possède des parties réelles négatives.

Les expressions de A et B sont données ci-dessous :

$$A = \begin{pmatrix} k_1 & 0 & .. & .. & 0 \\ 0 & 2k_1 & .. & .. & 0 \\ & & & & \\ & & & & 0 \\ 0 & & & 0 & (n-2)k_1 \end{pmatrix},\ B = \begin{pmatrix} b_3 \\ b_4 \\ \dots \\ b_n \end{pmatrix} = \begin{pmatrix} \frac{-k_1}{K_b} \\ \frac{k_1}{k_1 + K_b} \\ \dots \\ \frac{(-1)^n k_1}{(n-3)!(K_b + (n-3)k_1)} \end{pmatrix}$$

III.1.3 Application de l'algorithme de stabilisation au robot mobile Pekee

Le robot mobile considéré est un robot de type unicycle avec deux roues avant motrices et une roue arrière stabilisante comme le montre la figure ci-dessous :

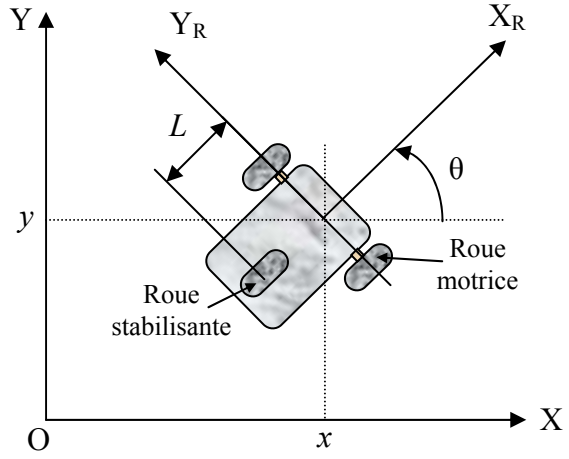


Figure IV.1. Robot mobile de type unicycle.

La commande de ce type de véhicule est établie par l'intermédiaire d'une vitesse linéaire d'un point M notée « *v* » et d'une vitesse angulaire, dans la direction des roues avant, notée « *w* ».

Le robot est supposé dans un environnement de roulement sans glissement dans un sol horizontal.

La configuration de ce robot est décrite par le vecteur (x, y, θ), où (x, y) sont les coordonnées du point M localisées à la mi-distance des deux roues avant. θ représente l'orientation du véhicule, entre l'axe du robot et l'axe des abscisses x.

Le modèle cinématique du robot est donné donc par l'expression suivante :

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \cos\theta \\ \sin\theta \\ 0 \end{pmatrix} v + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} w \qquad \text{(IV.9)}$$

Ici, l'objectif est de commander le robot afin de le ramener à une position d'équilibre en suivant une trajectoire donnée.

Pour appliquer la technique de la commande stabilisante au robot concernée, nous transformons la forme décrite dans l'équation (IV.9) en forme chaînée d'ordre 3.

La forme chaînée, représentant le modèle du robot unicycle, est donnée par l'expression suivante :

$$\begin{cases} \dot{x}_1 = u_1 \\ \dot{x}_2 = u_2 \\ \dot{x}_3 = x_2 u_1 \end{cases} \qquad \text{(IV.10)}$$

L'expression (IV.10) est obtenue en effectuant le changement de variable suivant :

$$\begin{cases} x_1 = x \\ x_2 = \tan\theta \\ x_3 = y \end{cases} \qquad \text{(IV.11)}$$

La relation donnant les entrées de contrôle « v, w » en fonction des nouveaux contrôles « u_1, u_2 » est la suivante :

$$\begin{cases} v = \dfrac{u_1}{\cos\theta} \\ w = u_2 \cos^2\theta \end{cases} \qquad \text{(IV.12)}$$

Pour ramener le robot à une position d'équilibre, nous allons concevoir un contrôleur $(u_1(x), u_2(x))^T$ avec un retour d'état discontinu invariant. Pour ce faire, nous utilisons l'algorithme de génération de trajectoire stabilisante présenté dans la section III.1.2.

Calcul des termes de u_1 et u_2

Les expressions des commandes $u_1(x)$ et $u_2(x)$ sont données comme suit :

$$u_1(x) = v_1(x) \tag{IV.13}$$

$$u_2(x) = v_2(x) + w_2(x) \tag{IV.14}$$

D'abord, posons $w_2(x)=0$ et considérons le retour d'état linaire $(v_1(x), v_2(x))^T$ tels que :

$$u_1(x) = -k_1 x_1 \tag{IV.15}$$

$$u_2(x) = -k_2 x_1 - k_3 x_2 \tag{IV.16}$$

Où $k_2 \in \Re$, k_1 et k_3 sont des constantes strictement positives et $k_1 \neq k_3$.

Le système en boucle fermée peut être intégré pour obtenir les états suivants :

$$x_1(t) = x_{10}\exp(-k_1 t),$$

$$x_2(t) = \left(x_{20} - \frac{k_2 x_{10}}{K_a}\right)\exp(-k_3 t) + \frac{k_2 x_{10}}{K_a}\exp(-k_1 t),$$

$$x_3(t) = \frac{k_1}{K_b} x_{10}\left(x_{20} - \frac{k_2 x_{10}}{K_a}\right)\exp(-K_b t) + \frac{k_2}{2K_a} {x_{10}}^2 \exp(-2k_1 t) + S_3(x_0) \tag{IV.17}$$

Avec: $K_a=k_1-k_3$, $K_b=k_1+k_3$ et $S_3(x_0)$ est une constante d'intégration qui peut être déterminée, à $t = 0$, comme une fonction des conditions initiales x_{10}, x_{20}.

A partir de (IV.17), on observe que (x_1, x_2, x_3) tendent vers (0,0, $S_3(x_0)$) quand t tend vers l'infini. Alors, si nous prenons les conditions initiales de telle sorte que $S_3(x_0)=0$, tous les états tendent vers l'origine.

Prenons $S_3(x_0)= S_3$ de tel sorte que les conditions initiales sont satisfaites (remplacer $t=0$ dans (IV.17) et déterminer $S_3(x_0)$ et ensuite substituer x_0 par x). Nous obtenons l'expression IV.18.

$$\mathrm{S}_3(\mathrm{x}) = x_3 - \frac{k_1}{K_b} x_1 x_2 + \frac{k_2}{2K_b} {x_1}^2 \tag{IV.18}$$

En résumé, la stabilisation de la forme chaînée (IV.10) est équivalente à la stabilisation de ((0,0, S_3) et ceci en prenant en compte le terme restant $w_2(x)$ qui est calculé de la façon suivante : $w_2(x) = C^T Q(x_1) S$

Avec :

$C=c_3$, $Q_1(x) = \dfrac{1}{x_1}$ et $S = S_3$

La valeur de C est choisie en fonction du signe de $(A+BC)^T$. Cette dernière doit posséder des parties réelles négatives.

Dans ce cas nous avons alors :

$A = k_1$

$$B = b_3 = \frac{-k_1}{K_b}$$

$$(A + BC)^T = k_1 - \frac{k_1}{K_b} C = k_1 \left(1 - \frac{C}{K_b}\right)$$

La valeur de $(A+BC)^T$ possède des parties réelles négatives est équivalent à dire que :

$$k_1 \left(1 - \frac{C}{K_b}\right) < 0 \Rightarrow \left(1 - \frac{C}{K_b}\right) < 0 \Rightarrow C > K_b \Rightarrow C > k_1 + k_3 .$$

Au final, l'expression du contrôleur $(u_1(x), u_2(x))^T$ qui stabilise notre robot est donné ci-dessous :

$$u_1(x) = -k_1 x_1 \qquad \text{(IV.19)}$$

$$u_2(x) = -k_2 x_1 - k_3 x_2 + c_3 \frac{S_3}{x_1} \qquad \text{(IV.20)}$$

III.1.4 Résultats

A partir de l'étude réalisée précédemment, nous illustrons le comportement du robot (observer les états x, y, θ) en appliquant la commande stabilisante ($u_1(x)$, $u_2(x)$) donnée par les expressions IV.19 et IV.20. Nous testons les performances du contrôleur à différentes conditions initiales (x_0, y_0, θ_0).

Le robot à stabiliser est un robot unicycle représenté par le modèle cinématique suivant :

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \cos\theta \\ \sin\theta \\ 0 \end{pmatrix} v + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} w \qquad \text{(IV.21)}$$

Etape préliminaire

Pour appliquer le contrôleur ($u_1(x)$, $u_2(x)$), nous fixons d'abord la valeur des différents termes de chaque commande.

Les valeurs des k_i ($1 \le i \le 3$) sont données comme suit : $k_1 = 1$, $k_2 = 0$, $k_3 = 3$.

Nous pouvons, de ce fait, déduire les valeurs de K_a, K_b et C qui valent :

$K_a = k_1 - k_3 = 1-3 = -2$ et $K_b = k_1 + k_3 = 1+3 = 4$.

$C > k_1 + k_3 \Rightarrow C > 4$ (on prend C = 10).

Nous effectuons dans ce qui suit des tests de simulation sur le robot. L'objectif est de visualiser sa trajectoire et sa position instantanée en appliquant la commande stabilisante.

Aussi, nous nous intéressons à observer le comportement des commandes en rotation et en translation pendant le mouvement du robot.

Nous présentons ci-dessous les résultats obtenus pour différentes conditions initiales :

a) Prenons comme conditions initiales (x_0= -2, y_0 = -2, θ_0 = π/4). Les allures x, y, θ sont données par la figure IV.2.

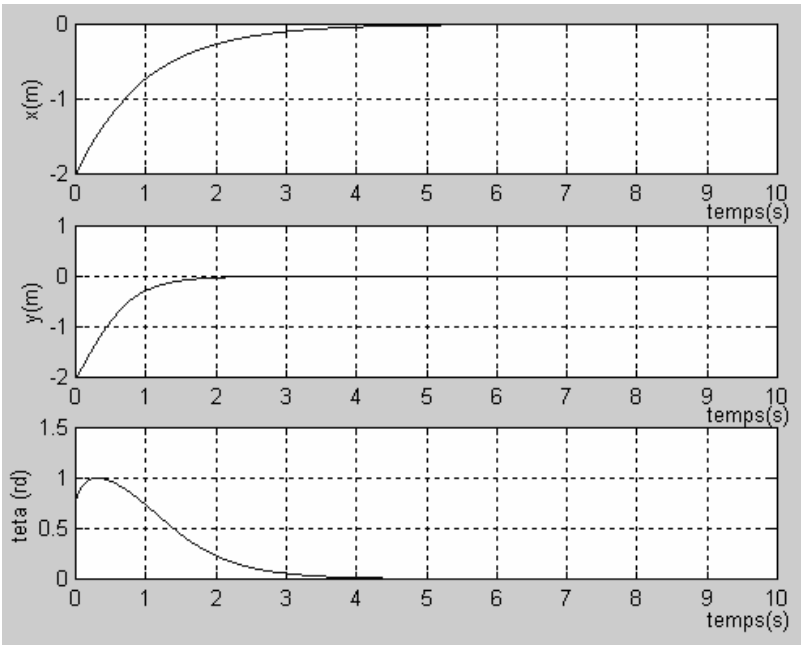


Figure IV.2. Evolution dans le temps des variables d'état (x, y, θ) du robot.

La trajectoire du robot correspondante pour atteindre la position d'équilibre (x_{final}= 0, y_{final} = 0, θ_{final} = 0) depuis la position initiale (x_0= -2, y_0 = -2, θ_0 = π/4) est donné par la figure ci-dessous.

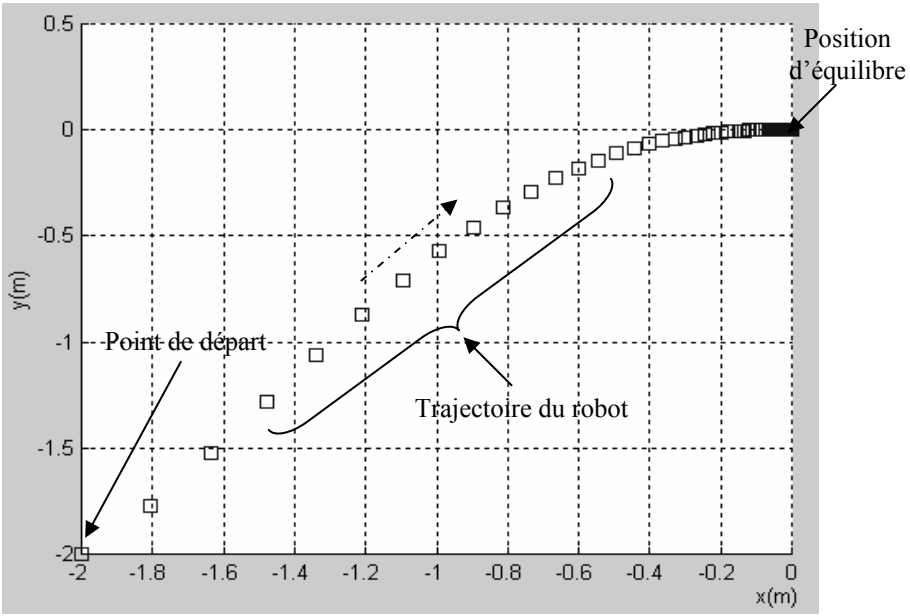


Figure IV.3. Trajectoire du robot.

En outre, nous avons étudié le comportement des commandes stabilisantes (en translation et en rotation) tout au long de la trajectoire. Les résultats obtenus sont illustrés dans la figure IV.4.

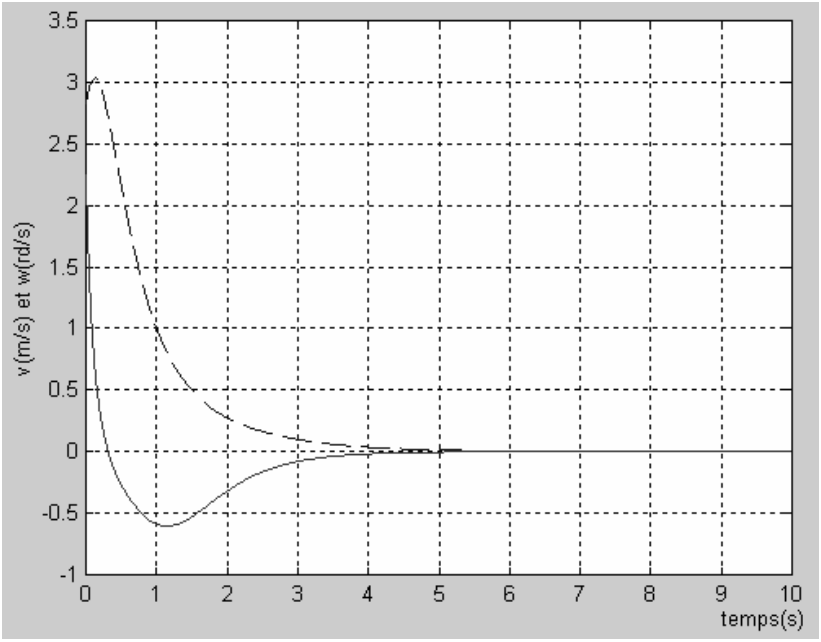


Figure IV.4. Evolution dans le temps des commandes en translation v (---) et en rotation w (—).

b) Pour les conditions initiales (x_0= 4, y_0 = 4, θ_0 = 0), les allures x, y, θ sont données par la figure IV.5.

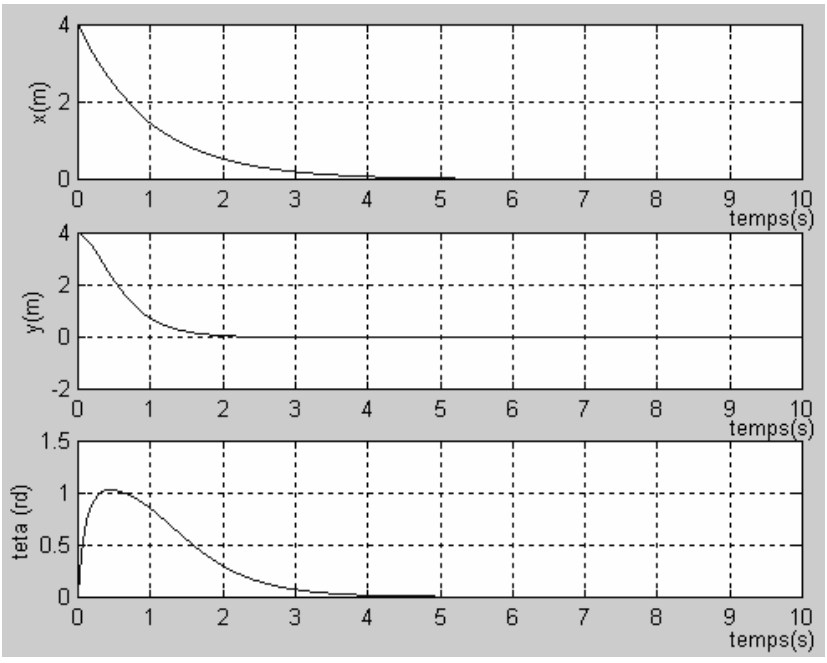


Figure IV.5. Evolution dans le temps des variables d'état (x, y, θ) du robot.

La trajectoire du robot correspondante pour atteindre la position d'équilibre depuis la position initiale (x_0= 4, y_0 = 4, θ_0 = 0) est donnée par la figure ci-dessous.

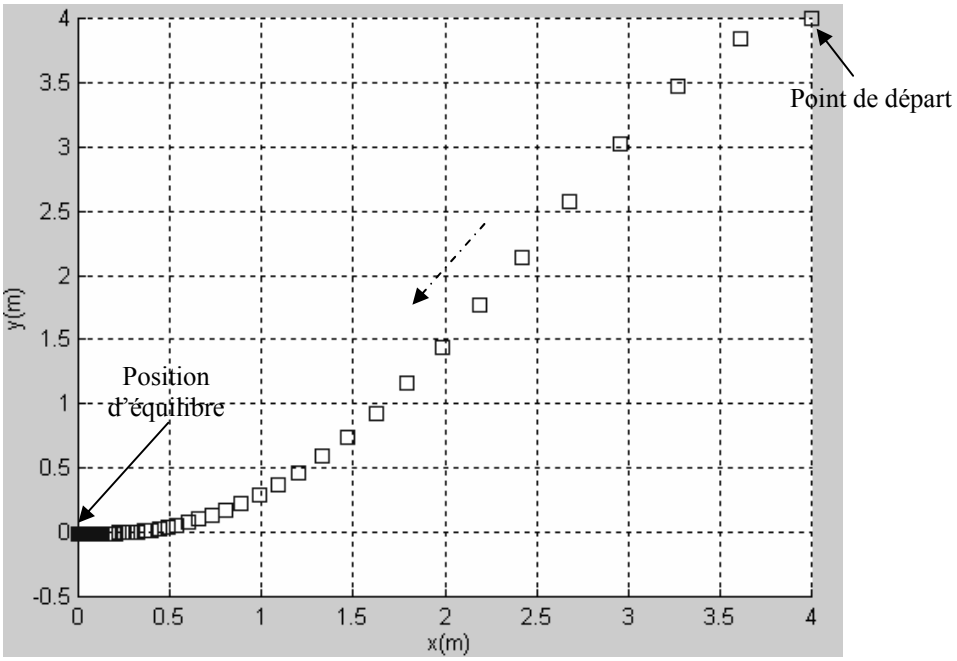


Figure IV.6. Trajectoire du robot.

Le comportement des commandes stabilisantes (v et w) correspondantes est illustré par la figure IV.7.

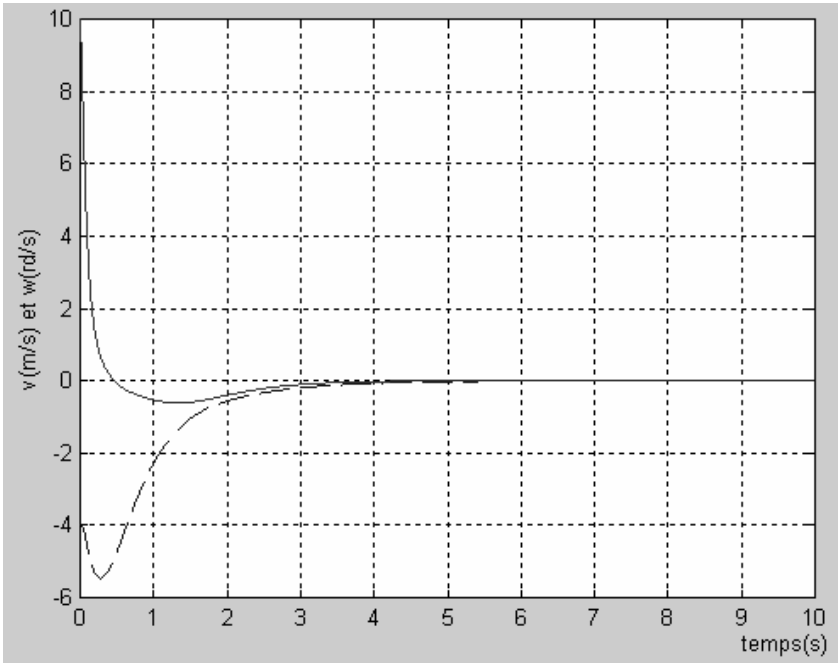


Figure IV.7. Evolution dans le temps des commandes en translation v (---) et en rotation w (—).

c) Prenons maintenant les conditions initiales suivantes : $x_0=4$, $y_0=-5$ *et* $\theta_0=\pi/3$. Les allures de x, y et θ sont données par la figure IV.8.

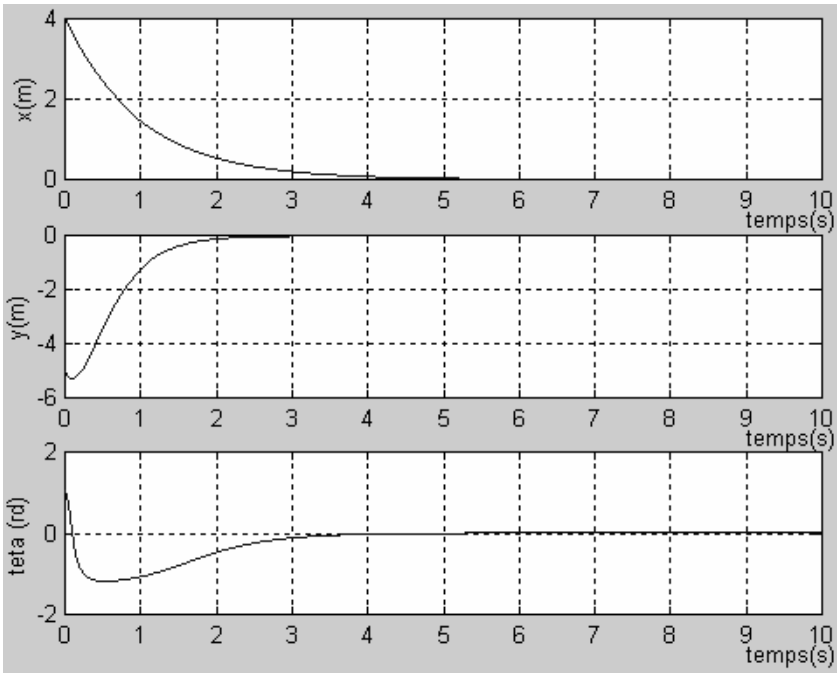


Figure IV.8. Evolution dans le temps des variables d'état (x, y, θ) du robot.

La trajectoire du robot correspondante pour atteindre la position d'équilibre depuis la position initiale (x_0= 4, y_0= -5, θ_0= π/3) est donné par la figure ci-dessous.

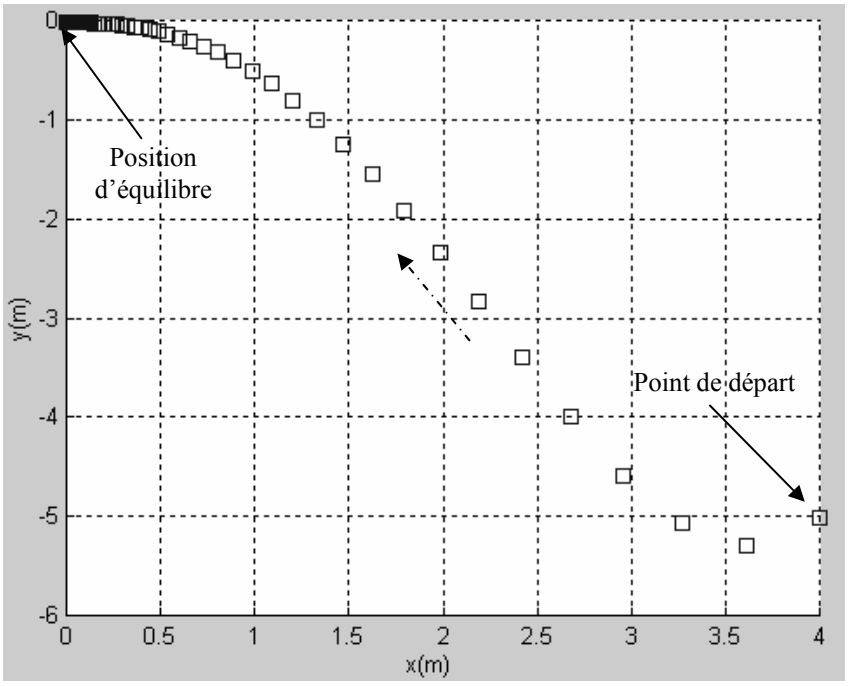


Figure IV.9. Trajectoire du robot.

Le comportement des commandes stabilisantes (v et w) est illustré par la figure IV.10.

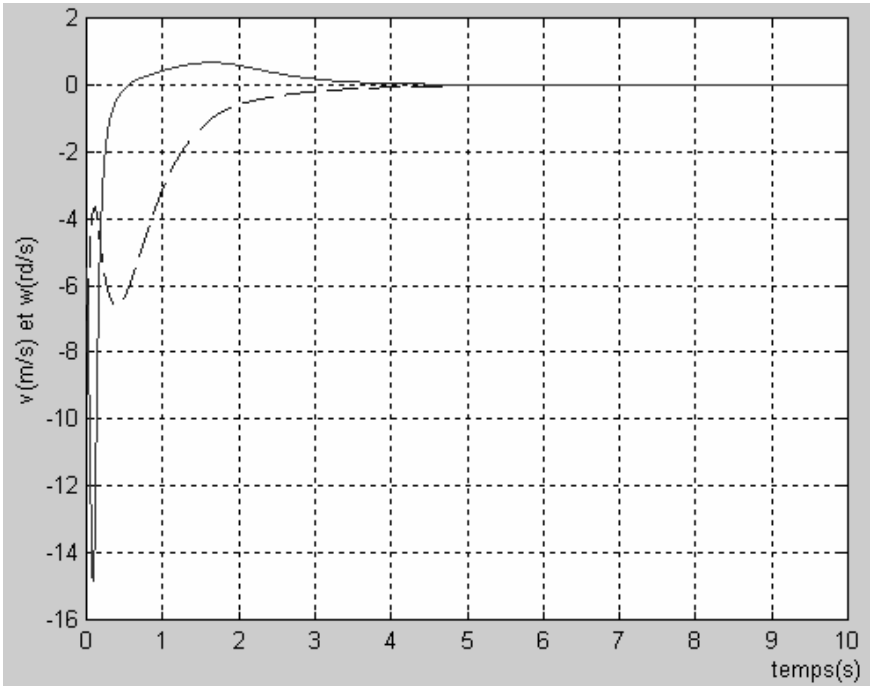


Figure IV.10. Evolution dans le temps des commandes en translation v (---) et en rotation w (—).

En résumé, nous avons effectué des tests de simulations pour observer le comportement du robot pour différentes conditions initiales. La convergence des états du système (IV.21) vers l'origine à partir des états (x_0, y_0, θ_0) est présentée respectivement dans les figures IV.2, IV.5, IV.8. Les figures IV.3, IV.6, IV.9 montrent la trajectoire du robot vers l'origine à partir des conditions initiales précédentes. Enfin, l'allure des commandes « v, w » durant la trajectoire du robot est illustré par les figures IV.4, IV.7, IV.10.

Nous avons proposé une commande par retour d'état discontinu pour stabiliser un système non-holonome. Les résultats ont montré que le système converge bien vers la position d'origine et ceci quelque soit sa position initiale dans laquelle il se trouve.
Pour juger les performances de cette commande, nous appliquons au système un deuxième type de contrôle dont l'étude théorique et les tests de simulation sont décrits dans la section suivante.

III.2 Stabilisation par retour d'état discontinu invariant basée sur la technique « feedback linearization »

Dans cette section, nous nous intéressons à stabiliser notre robot par une loi de contrôle issue de la méthode de « la linéarisation par bouclage ». On s'intéresse alors à décrire les fondements de base de la commande linéarisante ainsi que son application à un système non-holonome.

Nous commençons à décrire le principe de cette commande pour les systèmes mono-entrée/mono-sortie (SISO). Par la suite, les résultats d'étude établis seront étendus aux systèmes multi-entrée/multi-sortie (MIMO). Nous utiliserons souvent, dans ce qui suit, des notions mathématiques de la géométrie différentielle et topologie, tel que la dérivée de Lie, les crochets de Lie, le difféomorphisme et l'involutivité, etc.

III.2.1 Principe du feedback linearization ou de la linéarisation par bouclage

Considérons un système non linéaire décrit par l'expression suivante :

$$\begin{cases} \dot{x} = f(x) + g(x)u \\ y = h(x) \end{cases} \tag{IV.22}$$

Dont le nombre d'entrées et le nombre de sorties sont tous les deux égaux à *m*.

L'idée de la linéarisation par bouclage, est de boucler le système par une commande du type $u = r(x, v)$, où *v* est la nouvelle entrée, aussi de dimension *m*.

Pour effectuer ce bouclage, il nous faut exprimer les dérivées successives de chacun des y_i en fonction de l'état et de l'entrée [GUE 05], [JAU 07]. On s'arrête de dériver y_i, dès que les entrées commencent à intervenir dans l'expression de la dérivée. Nous disposons ainsi d'une équation du type :

$$\begin{pmatrix} y_1^{(k_1)} \\ \vdots \\ y_m^{(k_m)} \end{pmatrix} = A(x)u + b(x) \tag{IV.23}$$

Où k_i désigne le nombre de fois qu'il nous faut dériver y_i pour y voir apparaître une entrée. Sous l'hypothèse que la matrice A(x) soit inversible, le bouclage suivant :

$$u = A^{-1}(x)(v - b(x)) \tag{IV.24}$$

Où v est notre nouvelle entrée (voir figure IV.11), forme un système linéaire S_L de *m* entrées à *m* sorties décrit par les équations différentielles :

$$S_L : \begin{cases} y_1^{(k_1)} = v_1 \\ \vdots \\ y_m^{(k_m)} = v_m \end{cases} \tag{IV.25}$$

Ce système est linéaire et complètement découplé (c'est-à-dire que chaque entrée v_i agit sur une et une seule sortie y_i). Il est donc plus simple à commander par les techniques de contrôles linéaires ou non linéaires.

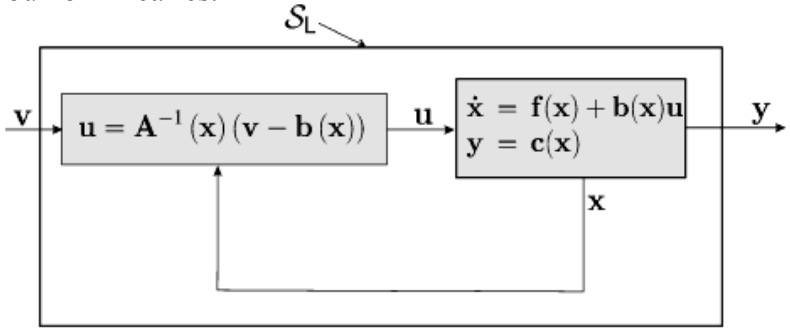


Figure IV.11. Schéma de principe d'un modèle linéaire obtenu par bouclage d'un système non-linéaire.

III.2.2 Linéarisation par bouclage pour un système mono-entrée/mono-sortie (SISO)
Soit un système non linéaire défini par les expressions suivantes :

$$\begin{cases} \dot{x} = f(x) + g(x)u \\ y = h(x) \end{cases} \qquad \text{(IV.26)}$$

Avec :

$x = [x_1, x_2, x_3, \ldots\ldots, x_n]$ qui représente le vecteur d'état.

$u = [u_1, u_2, u_3, \ldots\ldots, u_n]$ qui représente le vecteur de commande.

$y = [y_1, y_2, y_3, \ldots\ldots, y_n]$ qui représente le vecteur de sortie.

f(x) et *g(x)* sont des champs de vecteur et *h(x)* est une fonction analytique.

Ces fonctions sont définies dans un voisinage de x_0 de R^n. En plus, *f(x)* et *g(x)* sont supposées infiniment différentiable.

Le principe consiste à établir un bouclage non linéaire qui linéarise le système après une transformation des coordonnées dans l'espace d'état. Ce bouclage ne doit pas être seulement linéarisant mais surtout stabilisant, cela en vérifiant une certain condition concernant la notion de degré relatif, qui peut dans certain cas faire apparaître une partie non observable dont la dynamique est dite dynamique des zéros.

III.2.2.1 Notion de degré relatif
Un système de la forme (IV.26) est dit de degré relatif r dans une région Ω au voisinage d'un point x_0 si :

$$1)\ l_g l^i{}_f h(x) = 0 \ \text{ pour } \ 0 \le i \le r-2$$
$$2)\ l_g l^{r-1}{}_f h(x) \neq 0$$

Où $l_f h(x)$ est la dérivée de Lie de $h(x)$ selon le champ de vecteur f.

Le degré relatif d'un système non linéaire, représente le nombre de fois qu'il faut dériver la sortie afin de faire apparaître explicitement l'entrée [BEN 06]. En effet, si nous dérivons la sortie du système, nous obtenons :

$$\dot{y} = \frac{\partial y}{\partial x}.\frac{dx}{dt} = \frac{\partial h(x)}{\partial x}.\frac{dx}{dt}$$
$$\dot{y} = \frac{\partial h(x)}{\partial x}.(f(x) + g(x)u)$$
$$\dot{y} = l_f h(x) + l_g h(x)u \qquad \text{(IV.27)}$$

Lorsque le degré relatif du système est supérieur à 1, pour tout x au voisinage de x_0 nous avons $L_g h(x)=0$. La deuxième dérivation de y s'écrit alors :

$$\ddot{y} = \frac{\partial (l_f h(x))}{\partial x}.\frac{dx}{dt} = \frac{\partial (l_f h(x))}{\partial x}(f(x) + g(x)u)$$
$$\ddot{y} = l^2{}_f h(x) + l_g l_f h(x)u \qquad \text{(IV.28)}$$

Encore une fois, si le degré relatif est supérieur à deux, pour tous x au voisinage de x_0 alors : $l_g l_f h(x) = 0$

D'où : $\ddot{y} = l^2{}_f h(x)$

En étendant le raisonnement à l'ordre k (k<r) nous obtenons :

$$y^{(r)} = l_f^r h(x) + l_g l_f^{r-1} h(x)u \qquad \text{(IV.29)}$$

$l_g l_f^{r-1} h(x) \neq 0$

Remarque 1
Pour un système contrôlable, le nombre de dérivation de la sortie y pour faire apparaître explicitement l'entrée u, ne peut pas dépasser l'ordre n du système, c'est à dire que $r \leq n$. Si l'entrée u n'apparaît pas après n dérivation, le système est non contrôlable.

III.2.2.2 La forme normale

Comme dans le cas des systèmes linéaires, les systèmes non linéaires peuvent être mis sous une forme canonique facile à manipuler appelée *forme normale*. Cette dernière est basée sur une transformation des coordonnées, non linéaire, autour d'un point, définie par [GUE 05], [BEN 06] :

$$\begin{cases} z_1 = \Phi_1(x) = h(x) \\ z_2 = \Phi_2(x) = l_f h(x) \\ z_3 = \Phi_3(x) = l_f^2 h(x) \\ ..... \\ z_r = \Phi_r(x) = l_f^{r-1} h(x) \end{cases} \quad \text{(IV.30)}$$

Dans le cas où le degré relatif r est inférieur à l'ordre n du système, nous pouvons toujours trouver (n-r) fonctions ϕ_{r+1},.....ϕ_n telle que l'application $\phi(x)= [\phi_1(x)... \phi_n(x)]$ soit un difféomorphisme [JAU 07]. Ainsi, la représentation du système dans les nouvelles coordonnées se déduit facilement. En effet, les dérivées successives des expressions de l'équation (IV.30) donnent :

$$\begin{cases} \dot{z}_1 = \dfrac{dz_1}{dt} = \dfrac{\partial h(x)}{\partial x}.\dfrac{dx}{dt} = l_f h(x) = z_2 \\ \dot{z}_2 = \dfrac{dz_2}{dt} = \dfrac{\partial l_f h(x)}{\partial x}.\dfrac{dx}{dt} = l_f^2 h(x) = z_3 \\ ..... \\ \dot{z}_{r-1} = \dfrac{dz_{r-1}}{dt} = \dfrac{\partial (l_f^{r-2} h(x))}{\partial x}.\dfrac{dx}{dt} = l_f^{r-1} h(x) = z_r \end{cases} \quad \text{(IV.31)}$$

Avec comme dernière dérivée :

$$\dot{z}_r = \frac{dz_r}{dt} = l_f^r h(x) + l_g l_f^{r-1} h(x) u \quad \text{(IV.32)}$$

Etant donné que : $x = \phi^{-1}(z)$. La dernière équation devient : $\dot{z}_r = b(z) + a(z)u$
Avec :

$$\begin{cases} b(z) = l_f^r h(\phi^{-1}(z)) \\ a(z) = l_g l_f^{r-1} h(\phi^{-1}(z)) \end{cases}$$

Remarque
Le coefficient a(z) est par définition non nul aux voisinages de $z_0=\phi(x_0)$.

$$\begin{cases} q_i(z) = l_f \phi_i(\phi^{-1}(z)) \\ p_i(z) = l_g \phi_i(\phi^{-1}(z)) \end{cases} \text{ pour tout } r+1<i<n$$

Nous obtenons pour les (*n-r*) équations différentielles restantes :

$$\dot{z}_i = \frac{dz_i}{dt} = q_i(z) + p_i(z)u \quad \text{(IV.33)}$$

Finalement, la représentation du système dans l'espace d'état, relativement aux nouvelles coordonnées est :

$$\begin{cases} \dot{z}_1 = z_2 \\ \dot{z}_2 = z_3 \\ \ldots\ldots\ldots\ldots \\ \dot{z}_{r-1} = z_r \\ \dot{z}_r = b(z) + a(z)u \\ \dot{z}_{r+1} = q_{r+1}(z) + p_{r+1}(z)u \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \dot{z}_i = q_i(z) + p_i(z)u \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \dot{z}_n = q_n(z) + p_n(z)u \end{cases} \quad \text{(IV.34)}$$

Et l'équation de sortie $y = h(x) = z_1$ (IV.35)

En outre, dans certains cas il est possible de choisir les (n-r) fonctions complémentaires de manière à avoir :

$$p_i = l_g \phi_i(x) = 0 \quad \text{(IV.36)}$$

Pour tous x aux voisinages de x_0 et $r+1 \leq i \leq n$.

L'équation (IV.36) représente un système d'équation différentielle partielle, lorsque ce choix particulier est possible, le système (IV.34) se réduit à :

$$\begin{cases} \dot{z}_1 = z_2 \\ \dot{z}_2 = z_3 \\ \ldots\ldots\ldots\ldots \\ \dot{z}_{r-1} = z_r \\ \dot{z}_r = b(z) + a(z)u \\ \dot{z}_{r+1} = q_{r+1}(z) \\ \ldots\ldots\ldots\ldots\ldots \\ \dot{z}_i = q_i(z) \\ \ldots\ldots\ldots\ldots\ldots \\ \dot{z}_n = q_n(z) \end{cases} \quad \text{(IV.37)}$$

Et l'équation de sortie $y = h(x)$ (IV.38)

Remarque
Suivant le degré relatif r du système, on distingue deux possibilités :

1. Si *r=n*, la forme normale ne nécessite pas des fonctions additionnelles $\phi_i(r+1 \leq i \leq n)$.

On peut, à travers un retour d'état linéariser le système, c'est à dire trouver une équation différentielle linéaire d'ordre *n* entre la sortie *y* et l'entrée *u*.

2. Si *r<n* les fonctions $\phi_i(r+1 \leq i \leq n)$ sont exigées pour compléter la transformation des coordonnées, on peut, à travers un retour d'état, trouver une équation différentielle linéaire d'ordre r entre la sortie y et l'entrée u, le système est ainsi particulièrement linéarisable.

III.2.2.3 Linéarisation exacte par bouclage statique

Pour obtenir un comportement entrée/sortie linéaire du système, nous avons exprimé la commande *u* en fonction de l'état *x* et une entrée de référence *v* [ZER 06]. Cette relation est donnée par :

$$u = \alpha(x) + \beta(x).v \tag{IV.39}$$

Où α et β sont définis sur R^n, β est supposée non nulle.

Ce qui donne en remplaçant dans (IV.26) :

$$\begin{cases} \dot{x} = f(x) + g(x).\alpha(x) + g(x).\beta(x)v \\ y = h(x) \end{cases} \tag{IV.40}$$

Remarque
La nouvelle commande du système « *v* » peut être obtenue en appliquant au système linéarisé, une régulation classique (placement des pôles, mode de glissement, etc.)

La linéarisation exacte nécessite la condition *r=n*, la transformation des cordonnées donnant la forme normale est donc :

$$z = \phi(x) = \begin{bmatrix} \phi_1(x) \\ \phi_2(x) \\ \phi_3(x) \\ \cdots\cdots \\ \phi_n(x) \end{bmatrix} = \begin{bmatrix} h(x) \\ l_f h(x) \\ l_f^2 h(x) \\ \cdots\cdots \\ l_f^{n-1} h(x) \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ \cdots \\ z_n \end{bmatrix} \tag{IV.41}$$

Aucune fonction additionnelle ϕ_i n'est nécessaire. La forme d'état du système devient :

$$\begin{cases} \dot{z}_1 = z_2 \\ \dot{z}_2 = z_3 \\ \cdots\cdots\cdots \\ \dot{z}_{n-1} = z_n \\ \dot{z}_n = b(z) + a(z)u \end{cases} \tag{IV.42}$$

Avec *a(z)* non nul au voisinage de x_0.

En choisissant la loi de commande comme suit :

$$u = \frac{1}{a(z)}(-b(z) + v) \tag{IV.43}$$

Avec : $a(z) = l_g l_f^{n-1} h(x)$ et $b(z) = l_f^n h(x)$

Le système est alors décrit par les équations suivantes :

$$\begin{cases} \dot{z}_1 = z_2 \\ \dot{z}_2 = z_3 \\ ..... \\ \dot{z}_{n-1} = z_n \\ \dot{z}_n = v \end{cases} \tag{IV.44}$$

La forme (IV.44) correspond à un système linéaire et commandable; elle est appelée la forme canonique de Brunowsky [ISI 89], sous forme condensée nous avons :

$$\dot{z} = Az + Bv \tag{IV.45}$$

Avec : $A = \begin{bmatrix} 0 & 1 & .. & 0 \\ 0 & 0 & 1 & 0 \\ 0 & .. & .. & 1 \\ 0 & .. & .. & 0 \end{bmatrix}$ et $B = [0\ 0\ 0 \ldots .1]$

III.2.3. Linéarisation par bouclage pour un système multi-entrées/multi-sorties (MIMO)

La théorie développée pour les systèmes non-linéaires monovariables peut être étendu pour le cas multivariable. On ne considère que les systèmes carrés, c'est à dire ayant le même nombre de sortie que d'entrée. De ce fait un grand nombre de résultats issus des systèmes monovariables peuvent faire l'objet d'une extension au cas multivariable.

III.2.3.1 Technique de linéarisation au sens des entrées-sorties

Le concept de la linéarisation au sens des entrées-sorties est maintenant très connu. A cet effet, nous allons montrer comment obtenir une relation linéaire entre la sortie *y* et une nouvelle entrée *v*. Le modèle équivalent étant linéaire.

On peut lui imposer une dynamique de stabilisation en se basant sur les méthodes linéaires ou non linéaires.

Considérant un système non linéaire avec p entrées et p sorties donné par la forme suivante :

$$\dot{x} = f(x) + \sum_{i=1}^{p} g_i(x) u_i \tag{IV.46}$$

$$y_i = h_i(x) \tag{IV.47}$$

Avec : i =1,2,3,…..p

$x=[x_1, x_2,...x_n]^T \in R^n$ est le vecteur de états, $u=[u_1, u_2,...u_p]^T \in R^p$ est le vecteur des commandes.

$y = [y_1, y_2,...y_p]^T \in R^n$ représente le vecteur des sorties.

Le problème consiste à trouver une relation linéaire entre l'entrée et la sortie en dérivant la sortie jusqu'à l'apparition d'une entrée et ceci en utilisant l'expression [BEN 06] :

$$y_i^{(r_j)} = L_f{}^{r_j} h_j(x) + \sum_{i=1}^{p} L_{g_i}(L_f{}^{r_j-1} h_j(x))u_i \qquad \text{(IV.48)}$$

Avec : j = 1,2,3,….p

$L_f^i h_j$ et $L_g^i h_j$ sont les i^{éme} dérivées de Lie de h_j dans la direction de f et g respectivement.

r_j est le nombre de dérivées nécessaires pour qu'une entrées apparaisse dans l'expression (IV.48) connue sous le nom du degré relatif correspondant à la sortie y_j.

Le degré relatif total (r) est défini comme étant la somme de tous les degrés relatifs obtenus à l'aide de l'équation (IV.48) et doit être inférieur ou égal à l'ordre du système :

$$r = \sum_{j=1}^{p} r_j \le n \qquad \text{(IV.49)}$$

On dit que le système (IV.46) a pour degré relatif r s'il vérifie :

$$L_{g_i} l_f^k h_j = 0 \qquad 0 < k < r_j\text{-}1,\ 1 \le j \le p,\ 1 \le i \le p$$

$$L_{g_i} l_f^k h_j \neq 0 \qquad k = r_j\text{-}1$$

Dans le cas ou la degré total est égale à l'ordre du système, nous sommes dans le cas d'une linéarisation au sens des entrées-états. Par contre, si le degré relatif total est strictement inférieur à l'ordre du système, la linéarisation sera établie au sens des entrées-sorties

Pour trouver l'expression de la loi linéarisante u qui permet de rende la relation linéaire entre l'entrée et la sortie, on réécrit l'expression (IV.48) sous forme matricielle suivante :

$$\left[y_1^{r_1} ....y_p^{r_p}\right]^T = A(x) + E(x).u \qquad \text{(IV.50)}$$

Où :

$$A(x) = \begin{bmatrix} L_f^{r1} h_1(x) \\ ..... \quad ... \\ L_f^{r_p} h_p(x) \end{bmatrix} \qquad \text{(IV.51)}$$

$$E(x)=\begin{bmatrix} L_{g_1}L_f^{r_1-1}h_1(x) & L_{g_2}L_f^{r_1-1}h_1(x) & \dots & \dots & L_{g_p}L_f^{r_1-1}h_1(x) \\ L_{g_1}L_f^{r_2-1}h_2(x) & L_{g_2}L_f^{r_2-1}h_2(x) & & & L_{g_p}L_f^{r_2-1}h_2(x) \\ \dots & \dots. & \dots & \dots. & \dots. \\ L_{g_1}L_f^{r_p-1}h_p(x) & L_{g_2}L_f^{r_p-1}h_p(x) & \dots. & \dots. & L_{g_p}L_f^{r_p-1}h_p(x) \end{bmatrix} \quad \text{(IV.52)}$$

E(x) est appelée matrice de découplage du système :

Si on suppose que *E(x)* n'est pas singulière, la loi de commande linéarisante a pour forme [JAU 07]:

$$[u]=E(x)^{-1}[-A(x)+[v]] \quad \text{(IV.53)}$$

On note que la linéarisation ne serait possible que si la matrice de découplage E(x) est inversible, le schéma bloc du système est donné dans là figure IV.12.

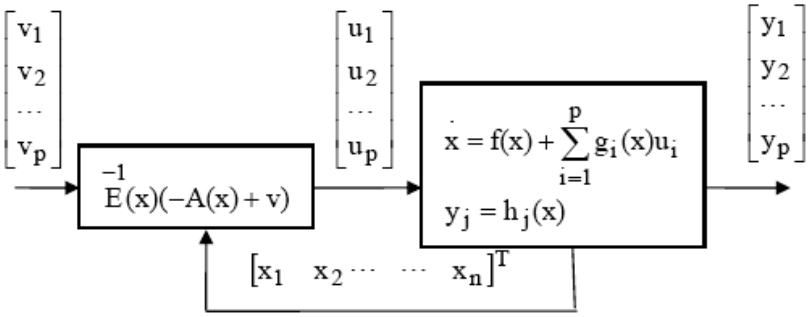


Figure IV.12. Schéma bloc du système linéarisé.

III.2.3.2 Conception du vecteur de commande pour le système linéarisé

Le vecteur *v* est conçu selon les objectifs de commande. D'une façon générale la commande v peut s'écrire :

$$v_j = y_{d_j}^{(r_j)} + k_{r_j-1}(y_{d_j}^{(r_j-1)} - y_j^{(r_j-1)}) + \dots + k_1(y_{d_j} - y_j) \quad \text{i} \le \text{j} \le \text{p}.$$

Où les vecteurs $\{y_{d_j}, y_{d_j}^{(1)}, y_{d_j}^{(2)}, \dots, y_{d_j}^{(r_j-1)}, y_{d_j}^{(r_j)}\}$ définissent les trajectoires de référence imposées pour les différentes sorties.

Dans notre cas, le système linéarisé en boucle fermée peut être représenté par la figure IV.13.

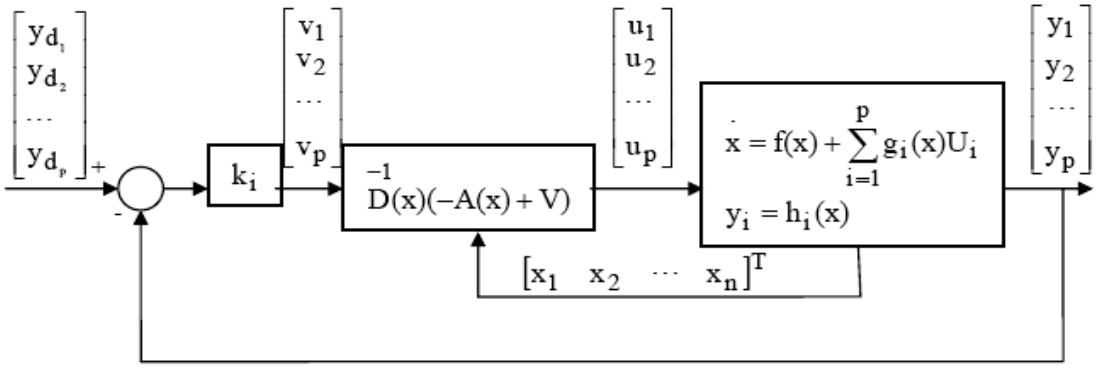


Figure IV.13. Schéma bloc du système linéarisé en boucle fermée.

Nous avons présenté dans la section III.2 une étude théorique sur les notions de la commande non linéaire. L'idée principale de cette commande est de linéariser partiellement ou totalement un système non linéaire.

Le système ainsi linéaire, peut être commandé au moyen des techniques de régulation appropries aux systèmes linéaires (placement de pôles, mode de glissement, etc.).

III.2.4 Application de l'algorithme de stabilisation au robot Pekee

Dans ce qui suit, nous présentons l'application de la technique de la commande linéarisante pour stabiliser le robot mobile décrit par l'équation (IV.9).

III.2.4.1 Modélisation cinématique de robot

Le comportement de notre robot est régi par l'équation d'état suivante :

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} v \\ w \end{pmatrix} \tag{IV.54}$$

Ce modèle peut s'écrire sous la forme suivante :

$$\dot{X} = f(X) + g(X)u \tag{IV.55}$$

Où les champs vectoriels f et g sont :

$$f(X) = \begin{pmatrix} f(X_1) \\ f(X_2) \\ f(X_3) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \text{ et } g(X) = [g_1(X) \quad g_2(X)] = \begin{pmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{pmatrix}$$

III.2.4.2 Loi de commande linéarisante appliquée au robot

Pour appliquer la technique de commande linéarisante, le système (IV.54) doit satisfaire les hypothèses suivantes [HAM 07] :

1) rang $\Delta_0(x)=n$ pour tout $x \in \mathrm{R}^n$,

2) Δ_1 et Δ_2 sont involutives,

3) il existe une fonction $h_1(x)$ telle que $dh_1\Delta_1 = 0$ et $dh_1.g_1 = 1$.

Avec :

$$\Delta_0 = \mathrm{Vect}\{g_1, g_2, \ldots\ldots, ad_{g1}^{n-2} g_2\}$$

$$\Delta_1 = \mathrm{Vect}\{g_2, \mathrm{adg}_1 g_2, \ldots\ldots, ad_{g1}^{n-2} g_2\}$$

$$\Delta_2 = \mathrm{Vect}\{g_2, \mathrm{adg}_1 g_2, \ldots\ldots, ad_{g1}^{n-3} g_2\}$$

$$ad_{g1}^{k} g_2(x) = [g_1, \quad ad_{g1}^{k-1} g_2](x), \; k \geq 1$$

$$ad_{g1}^{0} g_2(x) = g_2(x) \tag{IV.56}$$

Dans ce cas le système peut s'écrire sous la forme suivante (forme normale)

$$z = \phi(x) = \begin{bmatrix} \phi_1(x) \\ \phi_2(x) \\ \phi_3(x) \\ ..... \\ \phi_n(x) \end{bmatrix} = \begin{bmatrix} h_1(x) \\ l_{g1}^{n-2} h_2(x) \\ \\ l_{g1} h_2(x) \\ h_2(x) \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ .... \\ z_n \end{bmatrix} \quad \text{(IV.57)}$$

Et le retour d'état linéarisant s'écrit [JAU 07], [HAM 07] :

$$u = \psi(x)v = \frac{1}{L_{g2} L_{g1}^{n-2} h_2(x)} \begin{pmatrix} L_{g2} L_{g1}^{n-2} h_2(x) & 0 \\ -L_{g1}^{n-2} h_2(x) & 1 \end{pmatrix} \quad \text{(IV.58)}$$

Où h_2 est une fonction indépendante de h_1 telle que : $dh_1 \Delta_2 = 0$

En projetant ces équations au cas de notre robot nous obtenons :

$$\dot{q} = g_1 u_1 + g_2 u_2 \quad \text{(IV.59)}$$

Avec :

$q = (x, y, \theta)$, $g_1 = (0, 0, 1)^t$, $g_2 = (\cos\theta, \sin\theta, 0)^t$

u_1 et u_2 respectivement les vitesses linéaires et angulaires.

En calculant les crochets de Lie :

$[g_1, g_2] = (0 - \cos\theta \ -\sin\theta)^T$

Et en vérifiant bien les deux hypothèses 1 et 2, il vient :

1) rang Δ_0=3,

2) Δ_1=V*ect* $\{g_2,[g_1,g_2]\}$est involutive

Δ_2= V*ect* $\{g_2\}$ est involutive

D'après l'hypothèse 3, les fonctions h_1 et h_2 résultantes sont données par :

$$\begin{aligned} h_1 &= \theta, \\ h_2 &= x\cos\theta + y\sin\theta \end{aligned} \quad \text{(IV.60)}$$

En utilisant le changement de coordonnées suivant :

$$\begin{cases} z_1 = \theta \\ z_2 = x\cos\theta + y\sin\theta \\ z_3 = x\cos\theta - y\sin\theta \end{cases} \quad \text{(IV.61)}$$

Et le retour d'état :

$$\begin{cases} u_1 = v_1 \\ u_2 = v_2 + z_3 v_1 \end{cases} \quad \text{(IV.62)}$$

Le modèle est transformé en forme normale donnée comme suit :

$$\begin{cases} \dot{z}_1 = v_1 \\ \dot{z}_2 = v_2 \\ \dot{z}_3 = z_2 v_1 \end{cases} \quad \text{(IV.63)}$$

III.2.4.3 Loi de commande stabilisante appliquée au robot

Après avoir établie la commande linéarisante du système (IV.54), nous envisageons d'appliquer une loi de commande de type $u = - k z$ (placement de pôle) au système bouclé pour stabiliser le robot.

Ici, nous établirons une commande stabilisante qui ramène le robot à une position d'équilibre. Il s'agit de concevoir un contrôleur $(v_1(z), v_2(z))^T$ avec un retour d'état discontinu. Ce contrôleur est calculé de la même façon que celui décrit dans la section III.1.3.

Dans ce cas, l'expression des commandes $v_1(z)$ et $v_2(z)$ sont données comme suit :

$$v_1(z) = c_1(z) \quad \text{(IV.64)}$$

$$v_2(z) = c_2(z) + w_2(z) \quad \text{(IV.65)}$$

En faisant référence à la section III.1.3, nous pouvons déterminer les expressions finales du contrôle à savoir :

$$v_1(z) = -k_1 z_1 \quad \text{(IV.66)}$$

$$v_2(z) = -k_2 z_1 - k_3 z_2 + C^T Q(z_1) S \quad \text{(IV.67)}$$

Avec : $S(z) = z_3 - \frac{k_1}{K_b} z_1 z_2 + \frac{k_2}{2K_b} z_1^{\,2}$, $Q(z_1) = \frac{1}{z_1}$ et $C > k_1 + k_3$.

III.2.5 Résultats

Nous allons montrer le comportement du robot (observer les états x, y, θ) en appliquant la commande stabilisante $(v_1(z), v_2(z))$ donnée par les expressions IV.66 et IV.67. Nous allons tester les performances du contrôleur à différentes conditions initiales (x_0, y_0, θ_0). Ces dernières sont identiques à celles utilisées dans la section III.1.4.

Le robot à stabiliser est un robot unicycle représenté par le modèle cinématique suivant :

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} v \\ w \end{pmatrix} \quad \text{(IV.68)}$$

Etape préliminaire

Pour appliquer la commande ($v_1(z)$, $v_2(z)$), nous fixons les valeurs des différents termes de chaque contrôle.

Les valeurs des k_i ($1 \leq i \leq 3$) sont données comme suit : $k_1 = 1$, $k_2 = 0$, $k_3 = 3$.

Nous pouvons, de ce fait, déduire les valeurs de K_a, K_b et C qui valent :

$K_a = k_1 - k_3 = 1-3 = -2$ et $K_b = k_1 + k_3 = 1+3 = 4$.

$C > k_1 + k_3 \Rightarrow C > 4$ (on prend C = 10).

Nous effectuons des tests de simulation sur le robot. L'objectif est de visualiser sa trajectoire et sa position instantanée en appliquant la commande stabilisante basée sur la linéarisation par bouclage.

Aussi, nous nous intéressons à observer le comportement des commandes en rotation et en translation pendant le mouvement du robot.

Nous présentons ci-dessous les résultats obtenus pour différentes conditions initiales :

a) Prenons comme conditions initiales ($x_0 = -2$, $y_0 = -2$, $\theta_0 = \pi/4$). Les allures x, y, θ sont données par la figure IV.14.

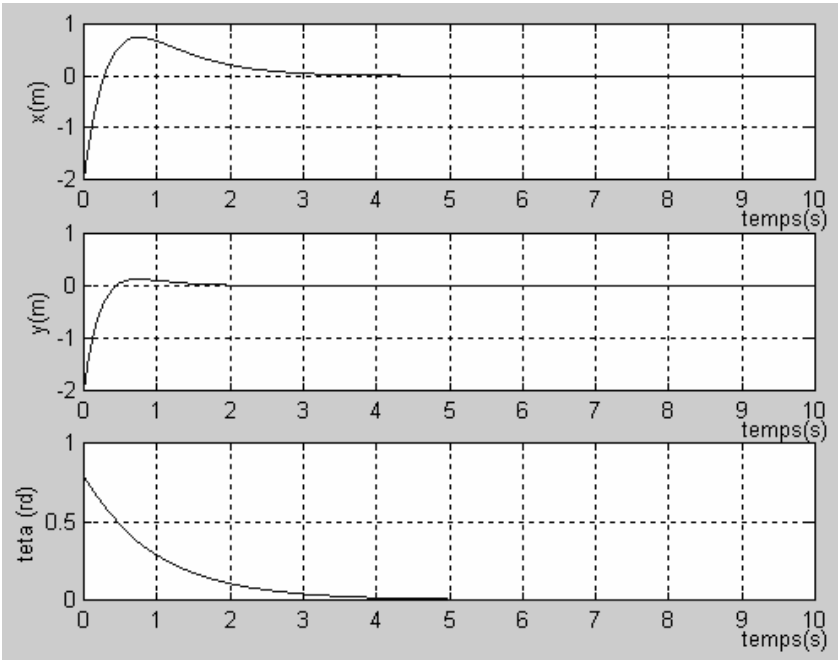


Figure IV.14. Evolution des variables d'état (x, y, θ).

La trajectoire du robot correspondante pour atteindre la position d'équilibre ($x_{final} = 0$, $y_{final} = 0$, $\theta_{final} = 0$) depuis la position initiale ($x_0 = -2$, $y_0 = -2$, $\theta_0 = \pi/4$) est donné par la figure ci-dessous.

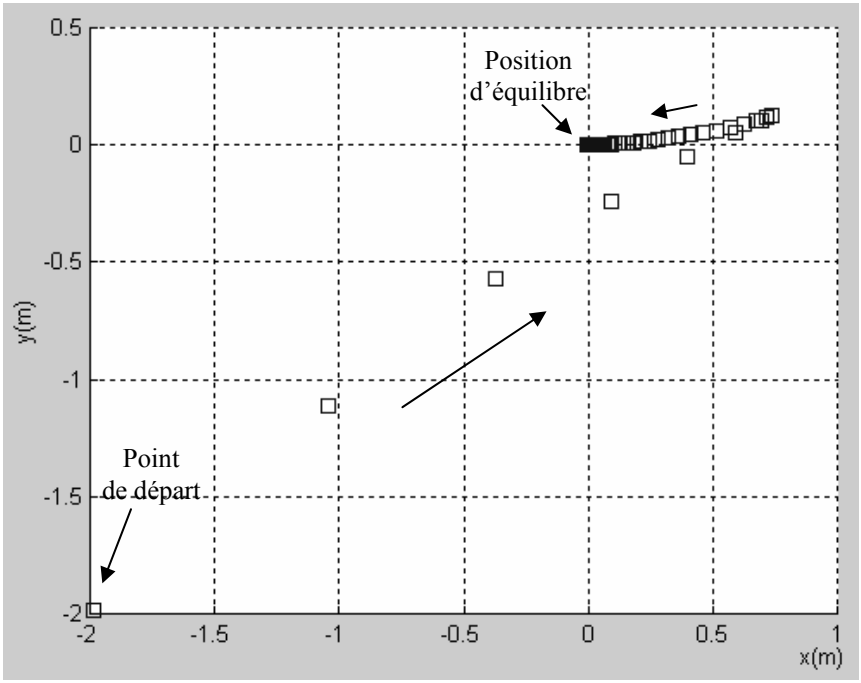


Figure IV.15. Trajectoire du robot.

Nous avons étudié le comportement des commandes stabilisantes (en translation et en rotation) tout au long de la trajectoire. Les résultats obtenus sont illustrés dans la figure IV.16.

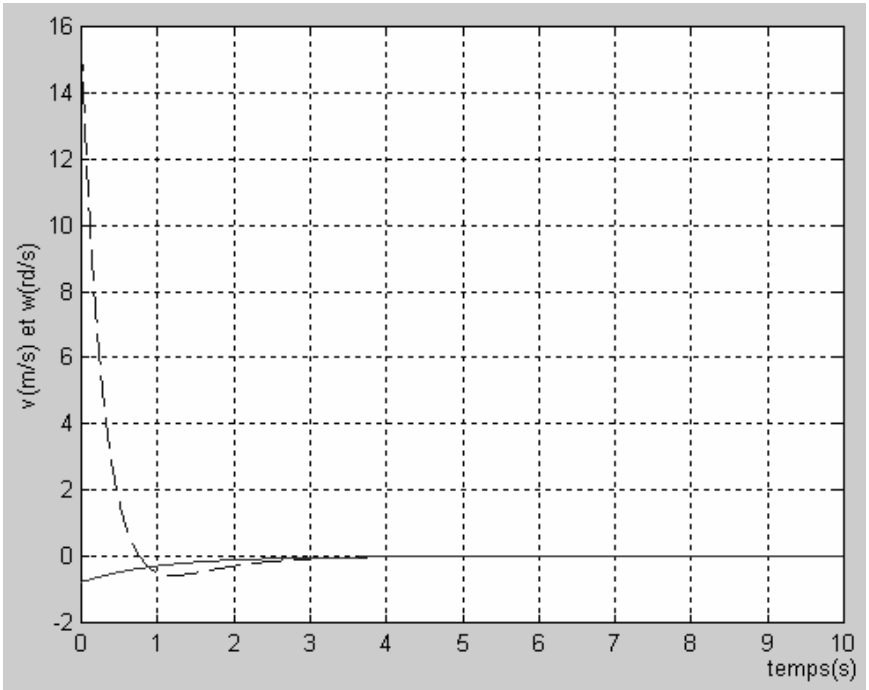


Figure IV.16. Evolution dans le temps des commandes en translation *v* (---) et en rotation *w* (—).

b) Pour les conditions initiales (x_0= 4, y_0= -5 et θ_0= π/3), les allures *x*, *y*, θ sont données par la figure IV.17.

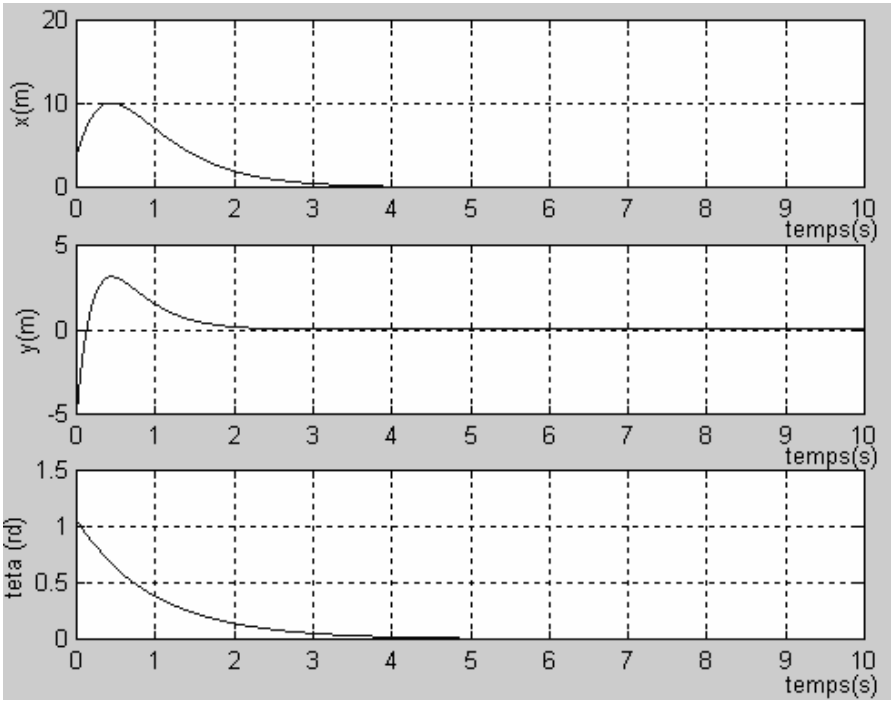


Figure IV.17. Evolution des variables d'état (x, y, θ).

La trajectoire du robot correspondante pour atteindre la position d'équilibre depuis la position initiale (x_0= 4, y_0 = -5, θ_0 = π/3) est donné par la figure ci-dessous.

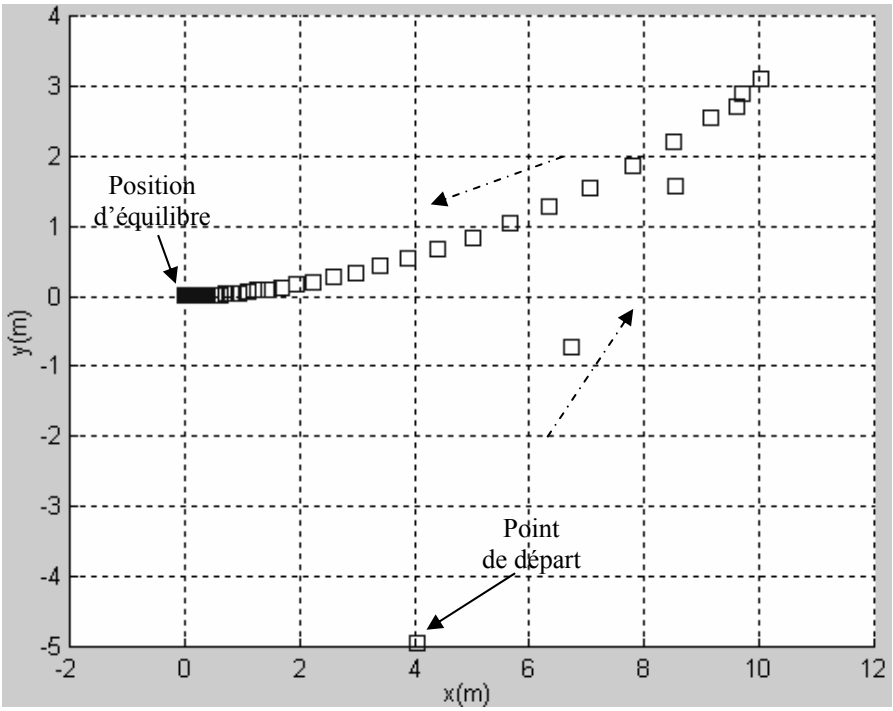


Figure IV.18. Trajectoire du robot.

Le comportement des commandes stabilisantes dans ce cas est illustré par la figure IV.19.

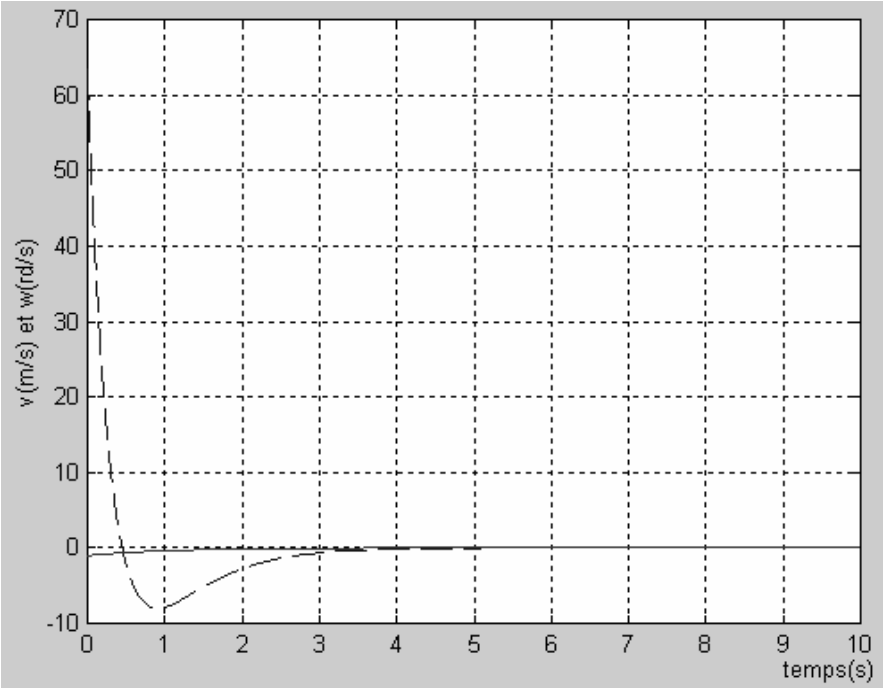


Figure IV.19. Evolution dans le temps des commandes en translation *v* (---) et en rotation *w* (—).

III.2.6 Comparaison des résultats issus des deux techniques de commande

Nous avons observé le comportement du robot mobile soumis à deux types de commande stabilisante. Les trajectoires correspondantes nous renseignent sur les performances que représente chaque contrôle.

La commande par « feedback linearization » donne des résultats peu satisfaisants par rapport à la commande par retour d'état appliquée à la forme chaînée.

Lorsque le robot est soumis à la commande par retour d'état appliquée à la forme chaînée, il effectue une trajectoire plus courte (figure IV.3) (pour atteindre la position d'équilibre) que dans le cas où il est soumis à la commande issue de la linéarisation par bouclage (figure IV.15).

Le robot se stabilise dans un délai qui est identique pour les deux cas de contrôle. Cependant, pour le deuxième cas de contrôle, le robot nécessite une commande d'amplitude beaucoup plus importante pour l'amener à la position désirée (figure IV.4 et IV.16). Ceci est dû au comportement du robot qui s'éloigne de temps en temps de la position d'équilibre. Ce qui nécessite une amplitude de commande beaucoup plus importante pour le faire rapproche de sa cible.

En faisant référence à ces résultats, il est recommandé pour stabiliser notre robot de choisir la commande par retour d'état discontinu appliquée à la forme chaînée. Cette commande donne de meilleures performances et des résultats satisfaisants.

IV. Robustesse des systèmes non-holonomes soumis à une loi de commande stabilisante

Cette section est consacrée à l'étude du comportement du robot soumis aux perturbations et ceci en appliquant les commandes présentées précédemment.

L'objectif de cette démarche est d'étudier les performances de notre système vis-à-vis aux différents types de bruits lorsqu'il suit une trajectoire stabilisante.

IV.1 Position du problème

Reprenons le système (IV.26) que l'on suppose maintenant soumis à des perturbations p [HAM 07] :

$$\dot{x} = f(x) + g(x)u + p(x,t) \tag{IV.69}$$

Ces perturbations peuvent représenter :

- des incertitudes paramétriques sur le terme nominal de la dérivée f : $p = \delta f(t,x)$
- des perturbations externes indépendantes de l'état p= b(t).

Ici, nous supposons que le système peut être perturbé par un bruit externe. Ainsi, nous allons tester les performances de notre robot en lui appliquant les lois de commande stabilisante.

Remarque
En annexe, nous présentons les tests de robustesse sur le robot lorsqu'il est soumis aux incertitudes paramétriques.

D'une façon générale, si on suppose que b(t) est le bruit qui agit sur le système, l'équation (IV.54) peut s'écrire alors :

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} v \\ w \end{pmatrix} + b(t) \tag{IV.70}$$

Avec : $b(t) = \begin{pmatrix} b_1(t) \\ b_2(t) \\ b_3(t) \end{pmatrix}$

Le système IV.70 peut s'écrire ainsi :

$$\begin{cases} \dot{x} = \cos\theta.v + b_1(t) \\ \dot{y} = \sin\theta.v + b_2(t) \\ \dot{\theta} = w + b_3(t) \end{cases} \tag{IV.71}$$

Ici, le bruit peut intervenir sur l'état du système, c'est-à-dire sur la position et l'orientation du robot.

Nous distinguons deux types de bruits : le bruit stochastique (bruit blanc,…) et le bruit déterministe (échelon,…). Dans ce cas, nous testons les performances du robot en appliquant les deux types de bruits.

Remarque
L'étude effectuée précédemment concernant la mise en œuvre d'une loi de commande stabilisante est valable pour le cas d'un système perturbé.

IV.2 Etude de robustesse dans le cas du robot Pekee

IV.2.1 Bruit de type stochastique

Dans ce cas, nous avons choisi un bruit stochastique de type « bruit blanc » de moyenne nulle et de variance $m=0.003$.

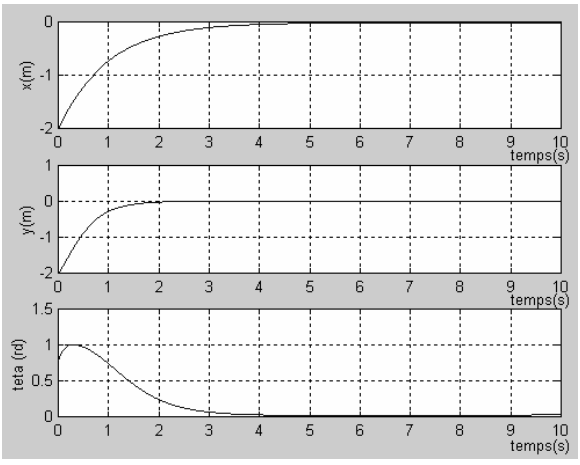
Nous supposons que le robot est perturbé au niveau de ses trois états (les positions x, y et l'orientation θ).

Application de la commande stabilisante

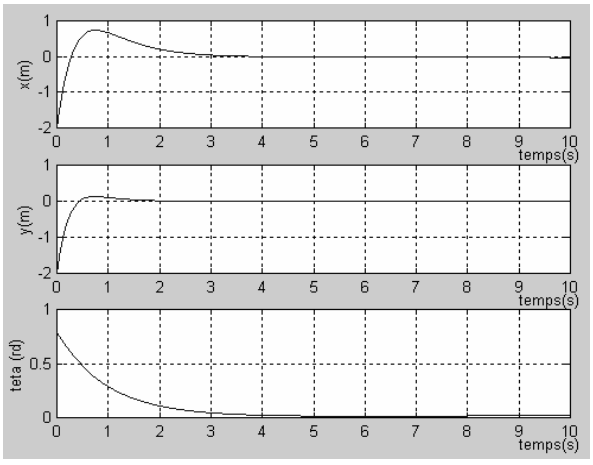
Nous appliquons au modèle (IV.71) les deux lois de commande stabilisante présentées dans les sections précédentes et ceci à des conditions initiales identiques.

Prenons les conditions initiales suivantes : x_0= -2, y_0 = -2, $\theta_0 = \pi/4$.

Les figures IV.20 et IV.21 montre l'évolution de la trajectoire du robot soumis à un bruit blanc.

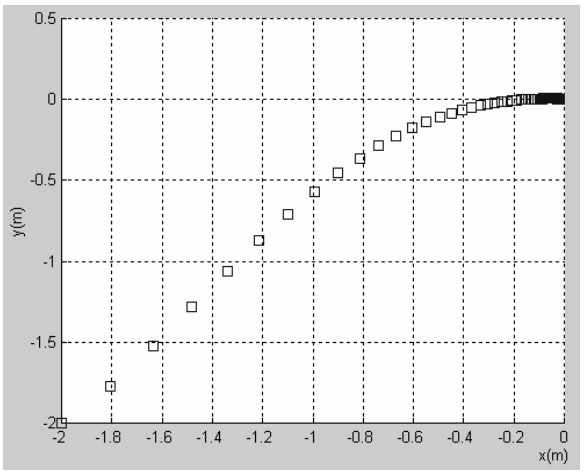


(a) commande en forme chaînée

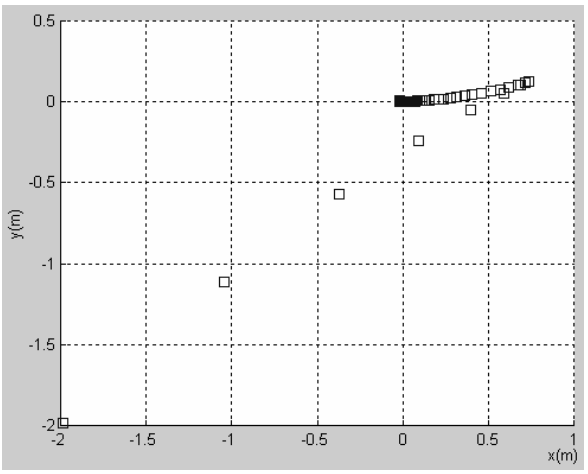


(b) commande en « feedback linearisation »

Figure IV.20. Evolution des états du robot soumis à un bruit blanc.



(a) commande en forme chaînée



(b) commande en « feedback linearisation »

Figure IV.21. Trajectoires du robot soumis à un bruit blanc.

Pour les deux cas de contrôle, on peut remarquer que l'évolution de la trajectoire du robot est analogue en présence ou en absence du bruit blanc. Ainsi, on peut conclure que le robot se stabilise à l'endroit voulu malgré la présence de perturbations au niveau de ses états (x, y, θ).

IV.2.2 Bruit de type déterministe

Nous supposons que le robot est perturbé par un bruit déterministe de type échelon d'amplitude $m=(m_1, m_2, m_3)^T$. Admettons que le bruit existe au niveau de ses deux positions x et y (l'orientation θ est supposée non bruitée).

Le bruit b(t) s'écrit alors :

$b_1(t) = m_1$ pour t ≥0

$b_2(t) = m_2$ pour t ≥0

$b_3(t) = 0$ pour t ≥0

Pour l'application numérique : $m_1 = 0.002$, $m_2 = 0.003$.

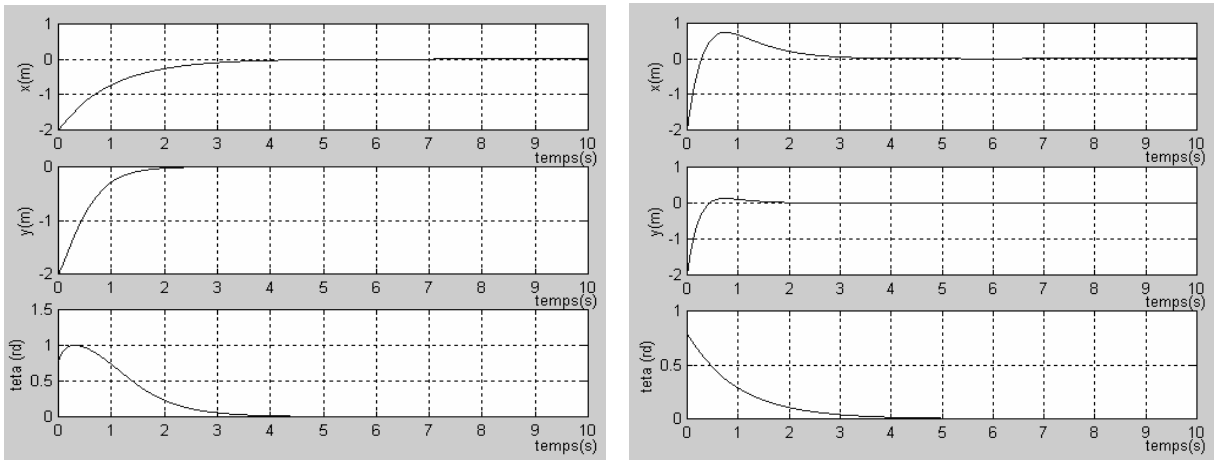
Le modèle cinématique du robot devient alors

$$\begin{cases} \dot{x} = \cos\theta.v + m_1 \\ \dot{y} = \sin\theta.v + m_2 \\ \dot{\theta} = w \end{cases} \qquad \text{(IV.72)}$$

Application de la commande stabilisante

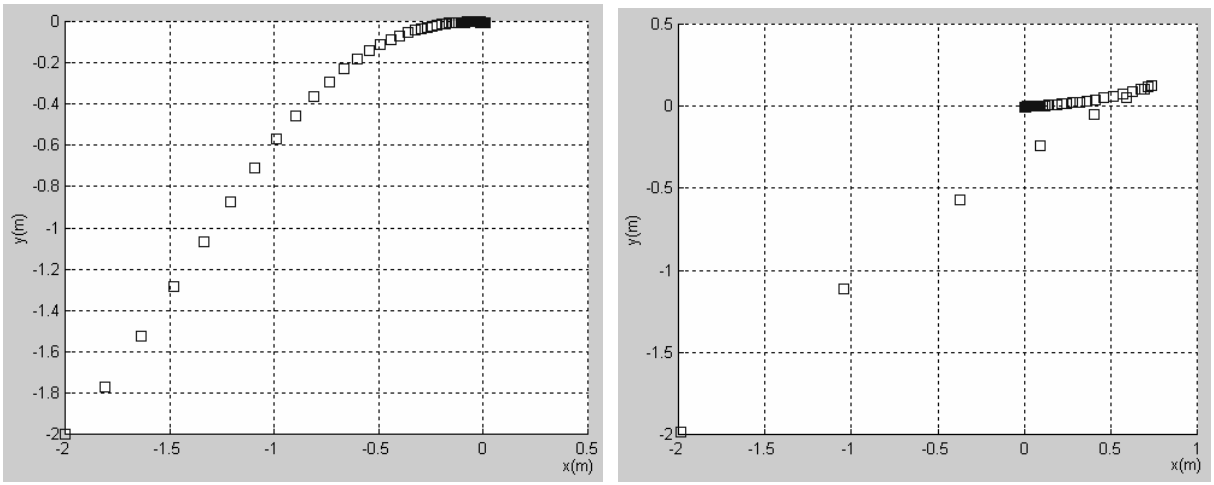
Nous appliquons au modèle (IV.72) les deux lois de commande stabilisante en prenant en considération les conditions initiales suivantes : x_0= -2, y_0 = -2, $\theta_0 = \pi/4$.

Les figures IV.22 et IV.23 montre l'évolution de la trajectoire du robot soumis à un bruit de type échelon.



(a) commande en forme chaînée (b) commande en « feedback linearisation »

Figure IV.22. Evolution des états du robot soumis à un bruit de type échelon.



(a) commande en forme chaînée (b) commande en « feedback linearisation »

Figure IV.23. Trajectoires du robot soumis à un bruit de type échelon.

Pour les deux cas de contrôle, on peut remarquer que l'évolution de la trajectoire du robot est analogue en présence ou en absence du bruit de type échelon. Ainsi, les positions x et y sont insensibles aux perturbations.

V. Conclusion

Dans ce chapitre, deux méthodes de commande ont été synthétisées et testées afin de garantir la stabilisation d'un robot mobile de type unicycle.

La première méthode consiste à ajouter un terme discontinu à une commande par retour d'état linéaire qui stabilise le robot modélisé sous la forme chaînée.

En ce qui concerne la deuxième méthode, afin de synthétiser la loi de commande, le modèle du robot mobile est mis sous la forme « normale » pour aboutir à un système linéaire par bouclage. De même, ce système est contrôlé par un retour d'état discontinu.

Pour chacune de ces commandes, nous avons effectué des tests de simulation sur le robot en absence et en présence des perturbations.

La commande par retour d'état discontinu appliquée à la forme chaînée donne des performances meilleures pour stabiliser un robot unicycle. Dans ce cas, l'évolution du robot est réalisée suivant une trajectoire optimale (contrairement à la commande basée sur la linéarisation par bouclage).

Après avoir défini l'ensemble des trajectoires stabilisantes, nous allons effectuer des essais expérimentaux sur le robot Pekee.
Il s'agit de tracer la trajectoire (obtenue après simulation) sur le sol et de demander au robot, muni d'une caméra, de suivre cette trajectoire. Cette poursuite sera effectuée en faisant appel aux techniques de traitement d'images.
Dans le chapitre suivant nous illustrons les différentes étapes nécessaires pour stabiliser le robot Pekee en faisant référence aux techniques de vision par ordinateur.

Chapitre V
Poursuite de trajectoire stabilisante par vision

Chapitre V Poursuite de trajectoire stabilisante par vision

I. Introduction

Ce chapitre présente un exemple de poursuite de trajectoire appliqué au robot Pekee (les différents composants matériels et logiciels de ce robot sont donnés en Annexe A).

Il s'agit, ici, de stabiliser le robot dans une position d'équilibre $q_{final}(0, 0, 0)$ à partir de n'importe quelle position initiale $q_{init}(x_{init}, y_{init}, \theta_{init})$. La trajectoire stabilisante est déjà déterminée, par simulation, en appliquant au robot une loi de commande par retour d'état discontinu en forme chaînée (voir chapitre IV). Une fois connue, la trajectoire est tracée sur le sol. Cette information est indispensable pour permettre au robot, muni d'une caméra, de poursuivre cette trajectoire.

Dans la première partie de ce chapitre, nous développons les différents algorithmes de vision et de commande permettant de stabiliser le robot Pekee. La deuxième partie est consacrée à l'implémentation de ces algorithmes sur le robot. Un test expérimental est détaillé à la fin de ce chapitre.

II. Stratégie générale de poursuite de trajectoire stabilisante par vision

Pour permettre au robot Pekee de se stabiliser, il est indispensable de lui fournir les informations nécessaires pour son déplacement le long d'une trajectoire qui mène à la position d'équilibre.

Dans cette optique, nous présentons ci-dessous un organigramme général qui décrit les différentes étapes conduisant à la stabilisation du robot Pekee.

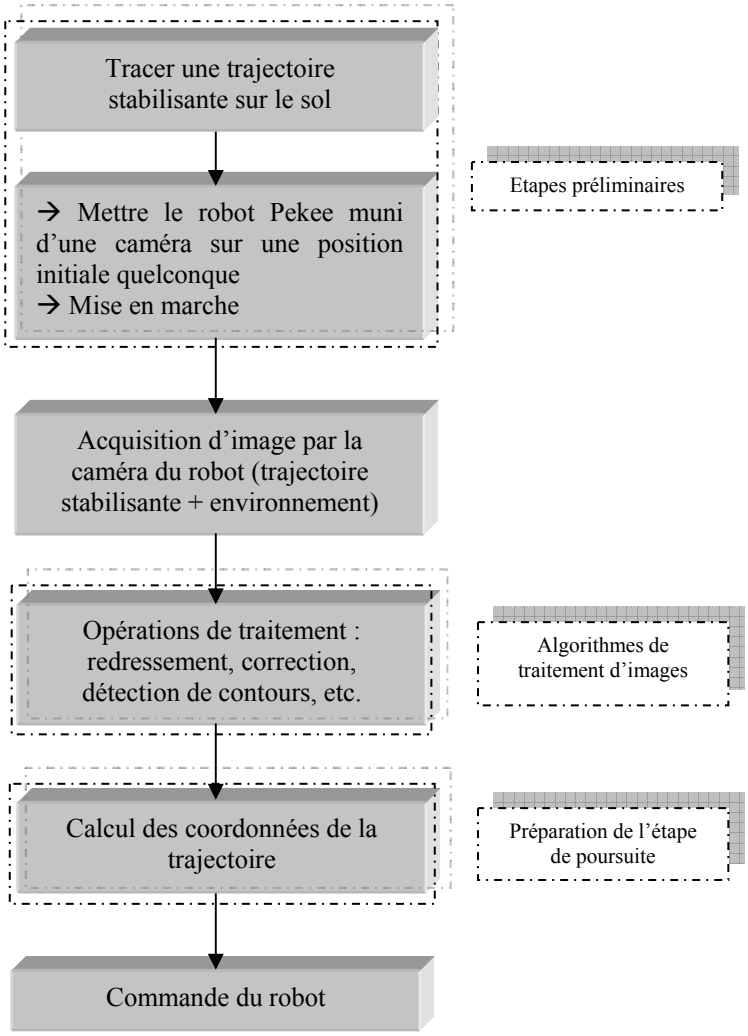


Figure V.1. Organigramme général de poursuite de trajectoire stabilisante pour le robot Pekee.

La figure V.1 décrit la procédure de stabilisation d'un robot mobile dans un environnement particulier. Cette procédure contient les algorithmes nécessaires pour illustrer l'évolution du robot.

Avant d'appliquer ces algorithmes, une étape préliminaire est indispensable pour réaliser la stabilisation. Il s'agit, en premier lieu, de tracer sur le sol la trajectoire stabilisante obtenue après simulation. Pour sa mise en marche, le robot est ensuite placé dans une position initiale quelconque.

La caméra installée sur le robot prend une image de l'environnement contenant la trajectoire. Cette image subit un ensemble de transformations nécessaires pour calculer le contrôle qui déplace le robot en suivant, bien évidement, la trajectoire tracée. Ces opérations sont répétées en boucle jusqu'à l'arrivée du robot à la position d'équilibre.

Les différents algorithmes illustrés dans l'organigramme précédent se subdivisent en trois catégories :

- algorithmes de traitement d'images,
- algorithmes de poursuite de trajectoire,
- algorithmes de commande du robot.

II.1 Algorithmes de traitement d'images

Cette étape constitue les différentes opérations de traitement effectuées sur l'image reçue par la caméra du robot. Ces opérations sont :

a) Etalonnage de l'image

En traitement d'image, l'opération d'étalonnage d'une caméra revient à modéliser le processus de formation des images, c'est-à-dire trouver la relation entre les coordonnées spatiales d'un point de l'espace avec le point associé dans l'image prise par la caméra.

Le modèle sténopé

C'est une représentation simple et linéaire du processus de formation des images au sein d'une caméra (matrice CCD, système optique, convertisseur A/N). Il s'agit d'exprimer les relations de passage du repère monde au repère caméra, d'exprimer la projection du repère caméra dans le plan image et d'appliquer la transformation affine qui conduit aux coordonnées de l'image.
Les paramètres employés dans ce modèle sont usuellement divisés en deux catégories : les **paramètres intrinsèques** qui sont internes à la caméra, et les **paramètres extrinsèques** qui peuvent varier suivant la position de la caméra dans l'espace de travail.

Parmi les **paramètres intrinsèques** nous comptons :

- les coordonnées de l'origine ($\boldsymbol{x_0}$ *et* $\boldsymbol{y_0}$) du repère de la matrice CCD, figure V.2,
- la distance focale ($\boldsymbol{f}$),
- les facteurs multiplicateurs (pour obtenir les coordonnées en pixels : $\boldsymbol{A}$, $\boldsymbol{B}$),

Les **paramètres extrinsèques** sont :

- R_{3x3} : qui est la matrice de rotation permettant de passer du repère lié à l'espace de travail au repère lié à la caméra,
- t_x, t_y et t_z : qui sont les composantes du vecteur de translation permettant de passer du repère lié à l'espace de travail au repère lié à la caméra.

Dans notre cas, nous simplifions le problème en supposant que le repère de la matrice CCD n'est pas orienté. Nous ne cherchons alors que les paramètres intrinsèques de la caméra.

Calcul des paramètres intrinsèques

Pour calculer les paramètres intrinsèques de la caméra, nous cherchons d'abord les points d'intersections entre deux droites et le plan où se situe les points : P_{ccd} (une cellule sur la matrice CCD), P_f (le point focale), P_s (un point du sol), figure V.2.

Les coordonnées de ces point sont : P_{ccd} $(\breve{x}, \breve{y}, 0)$, P_f $(0,0, f)$, P_s $(X, -h, Z)$

Pour le point Ps, ses coordonnées peut être exprimées de la façon suivante :

$$X = (\frac{B}{A}) * (\frac{(x - Larg/2 - x_0)}{(y - Haut/2 + y_0)}) * h \quad \text{(V.1)}$$

$$Z = \frac{B * f * h}{(y - Haut/2 + y_0)} + f \quad \text{(V.2)}$$

Où :

→ A et B sont des facteurs multiplicateurs,

→ *Larg* (respectivement *Haut*) est la largeur (respectivement la hauteur) de l'image en pixels,

→ f la distance du point focale,

→ h la hauteur de la caméra par rapport au sol.

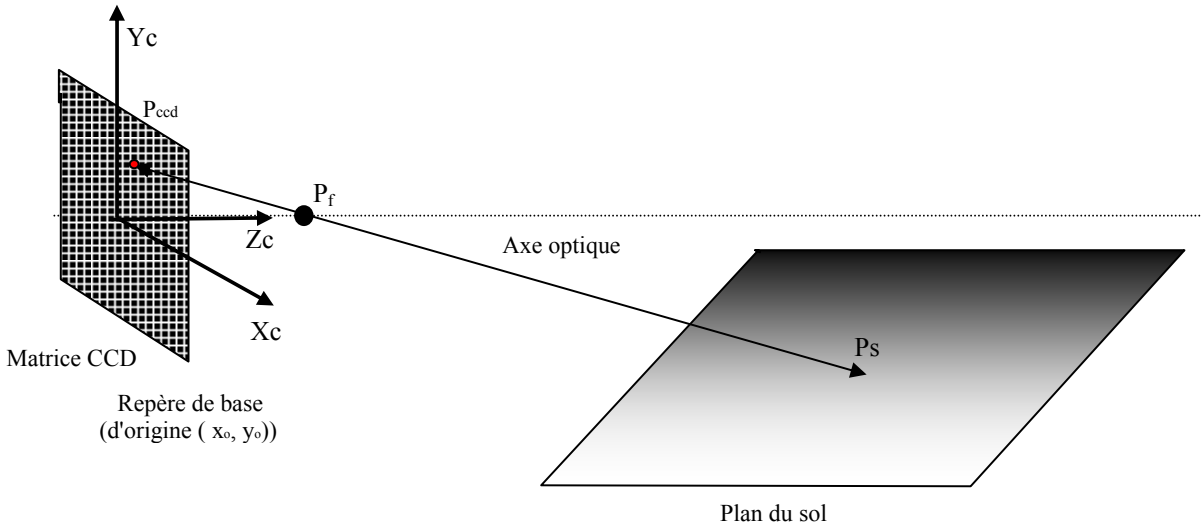


Figure V.2. Illustration du principe de sténopé.

Pour trouver ces paramètres intrinsèques, nous allons résoudre un système non-linéaire formé par les deux équations (V.1, V.2). Nous obtenons après calcul les expressions suivantes :

$$A * f = \frac{Z * (\Delta x)}{(\Delta X)} \quad \text{(V.3)}$$

$$B * f = \frac{Z}{h * \Delta y} \quad \text{(V.4)}$$

$$y_0 = -y + Haut / 2 + \frac{B * f * h}{Z} \quad \text{(V.5)}$$

$$x_0 = -x + L\,\text{arg}/\,2 - \frac{X * \alpha * f * h}{Z} \quad \text{(V.6)}$$

Où :
→ ΔX (respectivement ΔZ) représentent la différence entre deux points sur le sol suivant l'axe des x (respectivement l'axe des z)
→ Δx (respectivement Δy) représentent la différence entre deux points sur l'image suivant les colonnes (respectivement les lignes).

Pour trouver $A*f$ et $B*f$ on utilise un objet que l'on pose par terre (forme carrée, rectangulaire, triangulaire...) ensuite il suffit d'avoir les coordonnées de deux points en pixels (x,y) et dans l'espace (X,Z).
Pour le calcul de y_0 il suffit de prendre un point en pixels et un point dans l'espace (y et Z)
Pour trouver x_0 on prend deux points dans l'espace et un point en pixels (x, X, Z).

II.1.1 Redressement de l'image

Après calcul des paramètres : $A*f$, $B*f$, x_0 et y_0, il nous est, maintenant, possible de calculer les différentes coordonnées de l'image redressée.
Généralement, on travaille sur la zone des 40 premiers pixels en commençons par le bas de l'image. Cependant, à chaque fois qu'on s'éloigne vers le haut de l'image la distance entre les points dans l'espace augmente. Cette situation conduit à l'apparition de lignes de pixels noires. L'image résultante nécessite alors une étape de correction afin d'éliminer les pixels noirs.

Les figures V.3 et V.4 illustre un exemple de l'opération de redressement d'image.

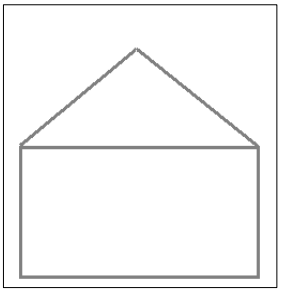


Figure V.3. Images test : une maison

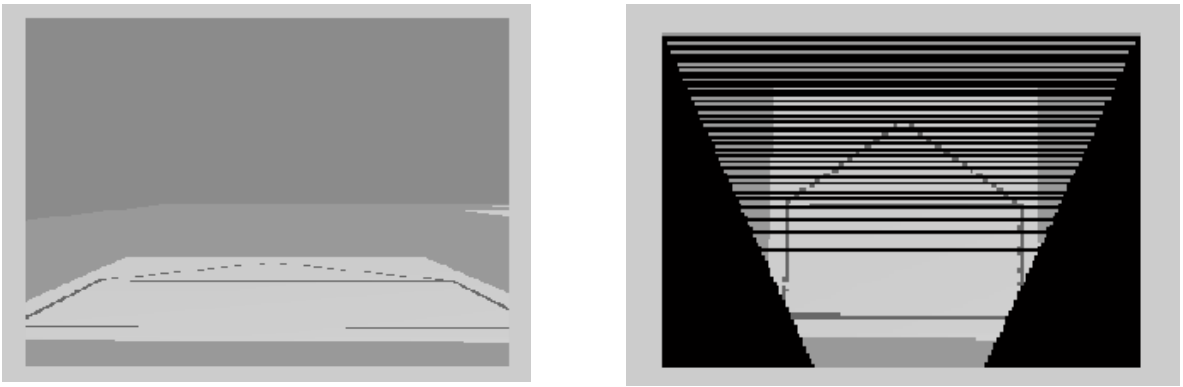


Figure V.4. Image test (maison) sur le sol (gauche) et image redressée (droite)

II.1.2 Correction

Comme on peut le voir sur les images précédentes (figures V.3 et V.4), nous constatons l'apparition de lignes de pixels noires. Ce phénomène est dû aux arrondissements effectués sur les positions des pixels de l'image redressée.

Pour remédier à ce problème, nous avons réfléchi à une étape supplémentaire qui est l'étape de correction de l'image redressée. Il s'agit d'appliquer à cette dernière une forme d'interpolation pour faire disparaître les lignes noires.

Nous présentons ci-dessous l'algorithme de correction d'une image redressée :

```
Pour i allant de 1 à Haut
    Si Im_Redr (i,Larg/2)=0
        Si Im_Redr (i+1,Larg/2)≠ 0
            Pour j allant de 1 à Larg

Im_Redr_Cor(i,j)=Im_Red(i+1,j)
            Fin
        Sinon
            Si Im_Redr (i+2,Larg/2)≠ 0
            Pour j allant de 1 à Larg

Im_Redr_Cor(i,j)=Im_Red(i+2,j)
            Fin
            ...etc
            Fin Si
        Fin Si
    Sinon
        Im_Redr_Cor(i,j)=Im_Red(i,j)
    Fin Si
Fin
```

Tableau V.1. Algorithme de correction de l'image redressée.

Où Im_Redr est l'image redressée, *Haut* est la hauteur de cette image, *Larg* sa largeur.

La figure suivante montre le résultat de la correction de l'image de la maison (figure V.4). Ici, nous remarquons la disparition des lignes noires.

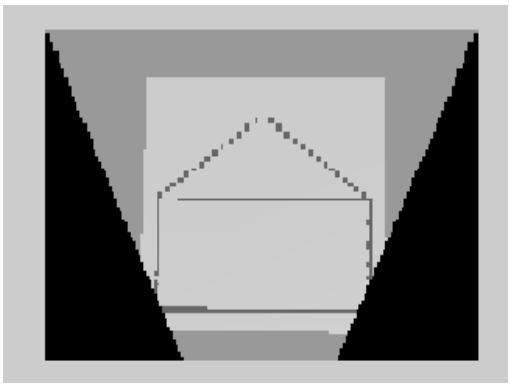


Figure V.5. Application de la correction sur l'image redressée

II.1.3 Détection de contours

La détection de contours [SLI 05], [TOU 90] est étape importante pour la détermination des coordonnées de la trajectoire à suivre par le robot. Cette étape consiste à isoler, dans l'image redressée et corrigée, le contour de la trajectoire du robot.

Dans ce cas, nous utilisons des toolbox de Matlab qui offre un choix entre les approches par convolution comme les techniques de Sobel, Prewitt, Roberts,… et les approches par filtrage optimal comme la technique de Canny.

La syntaxe générale de cette détection est :

Im_Cont = Méthode (Im_Red_Cor,s)

Où *Im_Red_Cor* est l'image redressée et corrigée et « *s* » un seuil qui permet la prise en compte d'un contour.

La figure suivante illustre les résultats de détection de contours de l'image redressée et corrigée de la figure V.5.

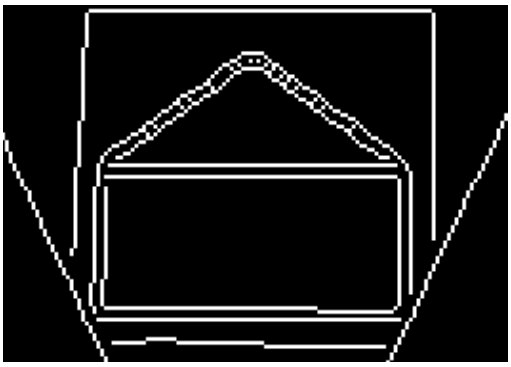


Figure V.6. Détection de contours sur l'image redressée

II.2 Algorithmes de poursuite de trajectoire

Cette étape consiste à définir les coordonnées de la trajectoire que doit suivre le robot en se basant sur l'image obtenue après l'application d'un détecteur de contours.

L'algorithme de planification de trajectoire du robot est donné ci-dessous :

```
Pour i allant de 1 à Haut
    Pour j allant de 1 à Larg
        Si Im_Cont(i,j)=1
            Traj_X(k)=[j+Xmin-1]
            Traj_Z(k)=[i-Zmax-1]
            Traj=[Traj_X ; Traj_Z]
             k=k+1
         Fin Si
    Fin
Fin
```

Tableau V.2. Algorithme qui calcul des coordonnées spatiales de la trajectoire.

Où

Im_Cont est l'image contours,

Haut est la hauteur de cette image, *Larg* sa largeur,

X_{min} l'ordonnée spatiale minimale

Z_{max} l'abscisse spatiale maximale.

On définit par la suite un certain seuil sur lequel on divise le nombre total des points de l'image contours pour obtenir les coordonnées réelles de la trajectoire.

Au final, l'algorithme qui détermine les coordonnées de la trajectoire est décrit ci-dessous :

```
Pour i allant de 1 à ( (k)/seuil )
    Coord_x( ((k)/NbPt)+1-i ) =  Traj_X(i*seuil)
    Coord_z( ((k)/NbPt)+1-i ) =  Traj_Z(i*seuil)
    Coord_Traj(i)=[Traj_X(i*seuil) ; Traj_Z(i*seuil) ]
Fin
```

Tableau V.3. Détermination des coordonnées de la trajectoire.

Où « *k* » est le nombre total de points de l'image contour.

II.3 Algorithmes de commande du robot

L'établissement d'une loi de commande du robot repose sur la connaissance de deux types d'informations :

1) les distances entre les différents points de la trajectoire.
2) les angles entre les points.

Dans ce sens, nous présentons ci-dessous l'algorithme général de la commande du robot

Pour i allant de 1 à ((k)/seuil)

Coord_Alpha0(i+1) = $\arctan((Coord_x(i+1)-Coord_x(i))/(Coord_z(i+1)-Coord_z(i)))$

Coord_Alpha(i+1) = Coord_Alpha0(i+1) - Coord_Alpha0(i)

Coord_Z(((k)/NbPt)+1-i) = $\sqrt{(Coord_x(i+1)-Coord_x(i))^2-(Coord_z(i+1)-Coord_z(i))^2}$

Fin

Tableau V.4. Algorithme de calcul des angles et distances des points de la trajectoire.

Dans la plupart des cas, le robot est limité par des contraintes physiques liées aux caractéristiques de la commande. Dans ce cas, la commande en rotation est comprise entre $-\pi/2$ et $\pi/2$ et la commande en translation est comprise entre -600 et 600 mm/s. Ceci impose un ajustement de l'algorithme précédent (Tableau V.4) pour pouvoir commander le robot.

Le principe d'ajustement est donné par l'algorithme suivant :

```
Pour i allant de 1 à n
    Si (mod(Coord_Alpha(i))>90) et (mod(Coord_Alpha(i))<α1)
        Coord_Ajust0=[Coord_Alpha(i)/m ... Coord_Alpha(i)/m; 0 ...0 Coord_Z(i)]
    Fin Si
    Si ((Coord_Z(i))>60) et (mod(Coord_Z(i))<z1)
        Coord_Ajust=[0...0 Coord_Alpha(i); Coord_Z(i)/m1...Coord_Z(i)/m1 ]
    Fin Si
Fin
```

Tableau V.5. Algorithme d'ajustement.

Après avoir calculé les distances entre les points de la trajectoire et les angles entre eux, le robot peut être commandé pour se déplacer entre ces points.

On utilise, dans ce cas, une dll PekeeMat avec comme paramètres 'S', 'u' et 'v' où 'u' et 'v' sont les contrôles du robot en translation et en rotation.

L'algorithme de commande appliqué au robot pour suivre la trajectoire demandé est alors donné comme suit :

```
Pour i allant de 1 à indice
  Impuls=Coord_Ajust(2,i)
    pour j allant de 1 à Impuls
      Si j<=Impuls
        PekeeMat( 'S',u,Coord_Ajust(1,i)/Impuls)
          pause(Délais)
      sinon
        PekeeMat( 'S',0,0)
      Fin
    Fin
Fin
```

Tableau V.6. Algorithme de commande.

Où

→ « indice » est la taille du tableau du programme,

→ Coord_Ajust est le nombre de points de la trajectoire,

→ « Délais » est la période d'envoie des impulsions.

III. Résultats expérimentaux

Pour illustrer les différents algorithmes présentés précédemment et permettre au robot d'effectuer une poursuite de trajectoire stabilisante, nous effectuons des tests sur le robot Pekee dans un environnement contenant une trajectoire tracée sur le sol.

Pour rappel, le robot Pekee est un robot mobile de type unicycle constitué de trois roues : une roue folle pour permettre au robot d'être stable et deux roues motrices qui ne peuvent pas braquer et sont commandées par deux moteurs indépendants.

Les différents algorithmes de traitement d'images, de planification de trajectoire et de commande sont déportés sur un ordinateur distant. Un test est alors effectué sur un logiciel de simulation propre à la plateforme Wany Robotics appelé « RSL ». Une fois les tests concluants, les différents algorithmes sont transférés vers le robot pour effectuer les essais de stabilisation par poursuite.

Déroulement
La démarche entreprise pour effectuer des tests sur le robot Pekee se divise en trois étapes :

Etape 1 : mise en marche du robot.

Etape 2 : traitements préliminaires pour le calcul numérique de la trajectoire.

Etape 3 : commande du robot.

III.1 Mise en marche du robot
Nous commençons par définir une *dll* appelée « ***PekeeMat*** » qui permet d'établir une connexion avec le robot et d'avoir accès à tous ses composantes : capteurs, moteurs...
La connexion avec le robot est faite à partir d'un ordinateur externe qui travaille sous Windows 2000. Les différents traitements seront effectués sous ***Matlab 6.5.***

Connexion et déconnexion avec le robot
Nous utilisons dans ce cas la *dll* « ***PekeeMat*** » pour créer une passerelle avec le robot. Ainsi, nous pouvons définir, à partir de cette *dll*, trois types de fonctions :

a) Fonctions de connexion
Elles servent d'établir une connexion entre un PC externe et le robot.

b) Fonctions de lecture
Elles permettent l'accès à certaines composantes du robot. Parmi les fonctions de lecture, citons :

→ Acquisition de l'image à partir d'une caméra montée sur le robot (capteur d'image),

→ Lecture des coordonnées du robot (capteur de position),

→ Acquisition d'informations diverses : température du robot (capteurs de température),…

c) Fonctions d'écriture
Elles permettent d'agir sur le comportement du robot. A titre d'exemple, on peut commander le déplacement du robot en imposant directement les vitesses de déplacement et de rotation.

Etape préliminaire
Pour pouvoir travailler avec cette dll il faut :

→ Soit la rajouter aux bibliothèques de ***Matlab*** pour cela on procède de la manière suivante:

1. On lance ***Matlab***;
2. On clic sur ***File*** ensuite ***Set Path*** ;
3. On ajoute un fichier (***Add Folder***) où on sélectionne le chemin vers l'emplacement de la DLL;
4. Enfin on enregistre (***Save***) et on ferme la fenêtre (***Close***).

Ceci nous permet de faire appel aux fonctions de la DLL ***PekeeMat***.

→ Soit être sûr de l'avoir dans l'espace de travail quand on envisage de faire appel à ses fonctions.

Pour connaître la syntaxe des fonctions de la dll PekeeMat, il suffit de taper PekeeMat ('H') sous ***Matlab*** et cela va nous afficher :

```
*******************************************************
*   Help on PekeeMat Interface
*******************************************************
*  - Parameters significations :
*
*    > Additionnal help function:
*    'h':  Additionnal help on a specific parameter
*
*    > Connexion functions:
*    'Z':  Pekee robot connexion
*    'z':  Pekee robot disconnexion
*
*     > Reading functions
*    'l':  Light Info Request
*    'p':  Position Info Request
*    't':  Temperature Info Request
*    'i':  All IR Telemeters Sensors Values Request
*    's':  Speed and Steering Infos Request
*    'o':  Odo Infos Request
*    'm':  Motors Speed Info Request
*    'c':  Cam Image Size Info Request
*    'v':  Cam Image Request
*
*     > Writting functions
*    'B':  Beep sound Request
*    'L':  LED Setting
*    'P':  Position Setting
*    'S':  Speed and Steering Setting
*    'M':  Motors Setting
*
*******************************************************
```

Pour avoir plus de détails sur les différentes fonctions, il suffit de taper PekeeMat('h','paramètre') où 'paramètre' est l'une des lettre indiquées ci-dessus.

Remarque

Avant de faire appel aux fonctions de lecture ou d'écriture il est nécessaire de commencer par établir une connexion avec le robot et après avoir fini de travailler avec le robot il faut se deconnecter avant de quitter ***Matlab***.

Connexion :

Pour établir une connexion avec le robot, on procède comme suit :

- Commencer par être sur que la DLL ***PekeeMat*** existe soit dans les bibliothèques de

Matlab ou dans l'espace de travaille (étape préliminaire);

- Ecrire dans l'espace de travaille de Matlab la fonction qui va établir la connexion avec le robot et qui a syntaxe suivante : PekeeMat('Z','Adresse IP', 'Nom du Robot') suivi d'un appui sur la touche Entrée. Le premier paramètre permet de définir la fonction à utiliser. Dans ce cas c'est la fonction de connexion. Le second paramètre est l'adresse IP du robot (ou du PC du robot).

Si on travaille sur le simulateur qui se trouve sur le même poste à l'endroit où on applique nos commandes, il suffit de mettre '***localhost***'. Alors que si on veut, par exemple, avoir accès à un simulateur se trouvant sur un autre ordinateur, il suffit d'indiquer l'adresse IP de cet ordinateur. Le dernier paramètre est le nom du Robot. Si on laisse ce champ libre, la connexion va se faire avec le premier robot rencontré.

Si la connexion aboutit, on va obtenir le message suivant : «***Connexion (AdresseIP)***» dans l'espace de travail de Matlab suivi de «***ans = 1***». Dans le cas contraire, on aura le message suivant : «***Warning: Connexion initialization Failed!***» suivi de «***ans =0***».

Déconnexion

Pour pouvoir déconnecter l'ordinateur avec lequel est relié du Robot, il faut écrire : «PekeeMat('z')» dans l'espace de travail de ***Matlab*** suivi d'un appui sur la touche Entrée. Si la tâche aboutée on va voir le message suivant : «***ans = 1***» dans l'espace de travail de ***Matlab***. Dans le cas contraire, on aura le message suivant : «***ans =0***».

III.2 Traitements préliminaires pour le calcul numérique de la trajectoire

Cette partie sera consacrée aux différents traitements que l'on va effectuer sur les images acquises par la caméra du robot. Nous commençons par décrire la fonction qui réalise l'acquisition. Cette image est utilisée par la suite pour effectuer l'opération d'étalonnage et de redressement. Enfin, des techniques de détection de contours sont appliquées sur l'image redressée pour isoler la trajectoire stabilisante.

III.2.1 Acquisition d'images

Nous utilisons la fonction PekeeMat('v') pour effectuer l'acquisition de l'image. Cette dernière contient des informations sur l'environnement du robot (image d'un sol sur lequel est tracée une trajectoire), figure V.7.

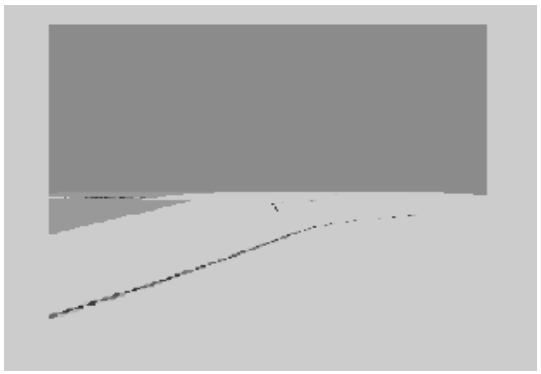


Figure V.7. Image de l'environnement (sol) contenant la trajectoire du robot

III.2.2 Etalonnage et redressement d'images

L'application des algorithmes d'étalonnage et de redressement, présentés dans la section II.1, à l'image d'origine (Figure V.7) nous donnent le résultat suivant (figure V.8).

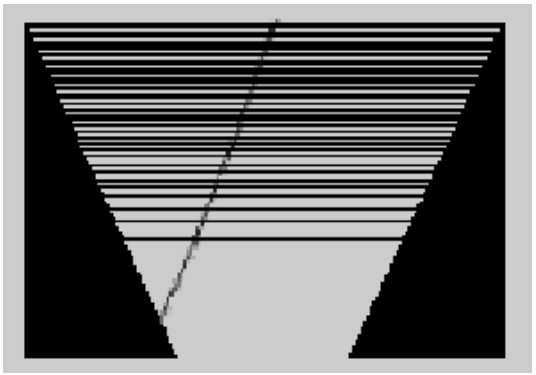


Figure V.8. Image redressée de l'image d'origine

Cependant, cette image ne peut être traitée telle qu'elle est présentée, car elle contient plusieurs informations erronées qui affectent la qualité de traitement par la suite.

En effet, nous observons sur l'image redressée l'apparition de pixels noirs. Ce qui provoque l'apparition de faut contours. Pour remédier à ce problème, une étape de correction est mise en ouvre.

La figure suivante montre le résultat issu de la correction de l'image redressée. Ici, nous remarquons la disparition des lignes noires.

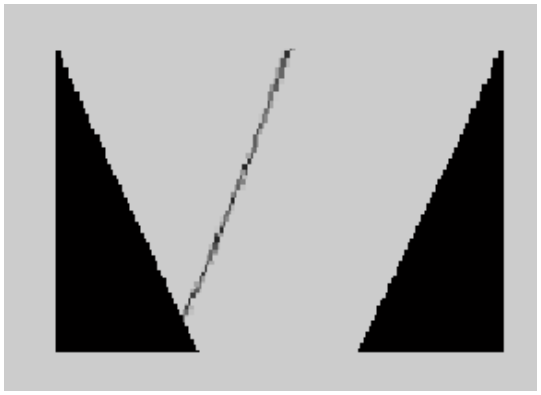


Figure V.9. Application de la correction sur l'image redressée

III.2.3 Détection de contours

Après avoir corrigé l'image redressée, l'étape de détection de contours est nécessaire pour isoler la trajectoire du robot [COU 05].

Des toolbox disponibles dans Matlab sont utilisées pour effectuer la détection adéquate. La figure suivante illustre les résultats de détection de contours appliquée à l'image corrigée.

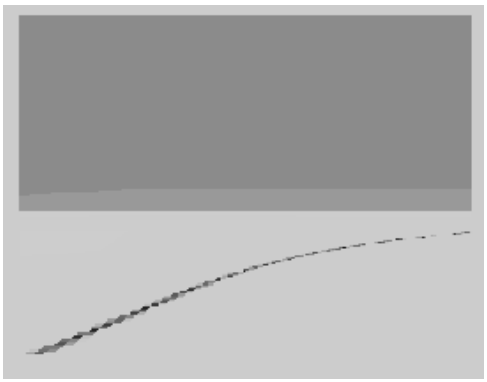
Figure V.10. Application de l'opération de détection de contours à l'image corrigée, apparition de la trajectoire du robot.

III.3 Application des algorithmes de poursuite de trajectoire

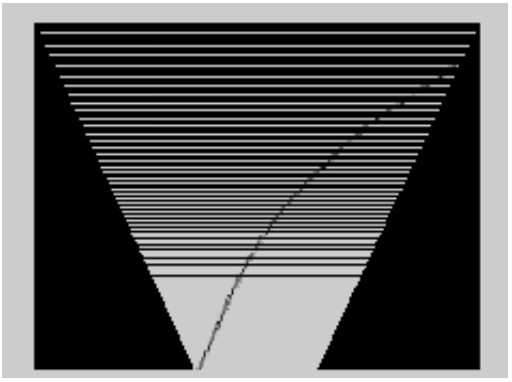
Cette étape consiste à définir les coordonnées de la trajectoire que doit suivre le robot en se basant sur l'image contour obtenue lors de l'étape précédente. L'application des algorithmes (tableau V.2 et tableau V.3) sur l'image contour de la figure V.10 nous donne l'image suivante :

Figure V.11. Coordonnées des différents points de la trajectoire

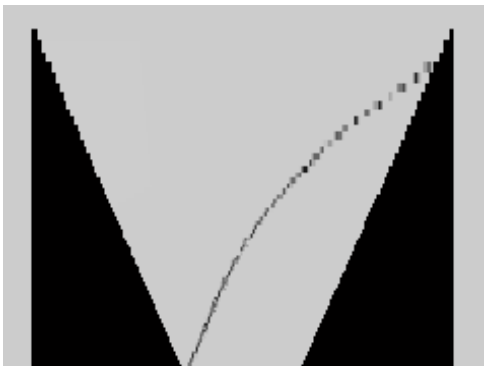
En résumé, la figure ci-dessous synthétise les différentes étapes de traitement nécessaires pour définir les coordonnées de la trajectoire. Cette information est nécessaire pour établir une loi de commande au robot pour suivre la trajectoire en question.



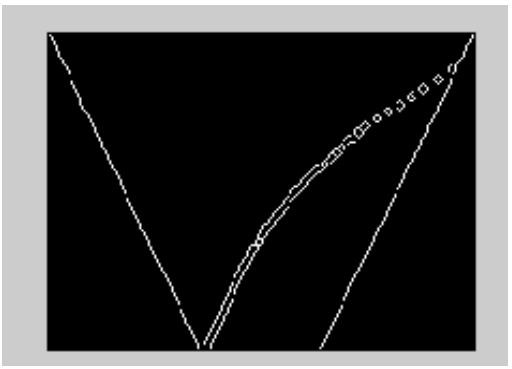
a) Image initiale (sol + trajectoire)



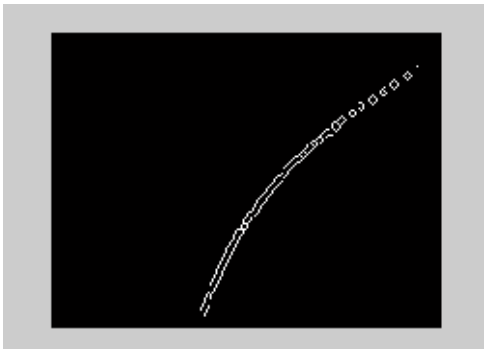
b) Image redressée



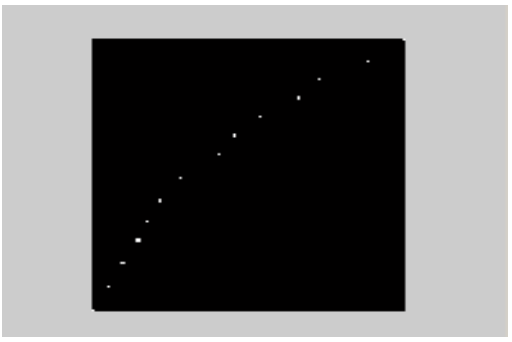
c) Image corrigée



d) Détection de contours



e) Détection du contour de la trajectoire



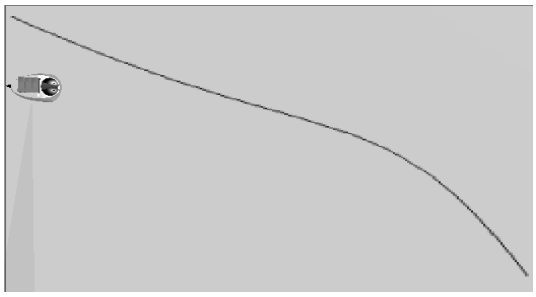
f) Calcul de coordonnées de la trajectoire

Figure V.12. Synthèse des différentes étapes de traitements pour définir les coordonnées de la trajectoire du robot.

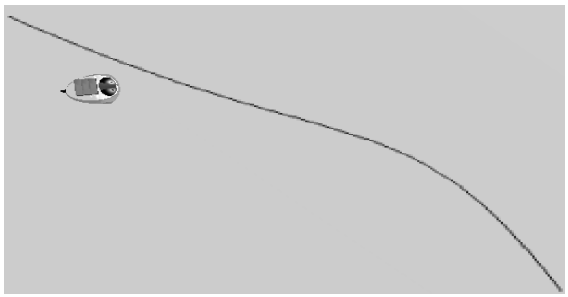
III.4 Application des algorithmes de commande

Après avoir défini l'ensemble des coordonnées de la trajectoire du robot, l'étape qui suit consiste à lui appliquer l'algorithme de commande pour suivre la trajectoire correspondante.

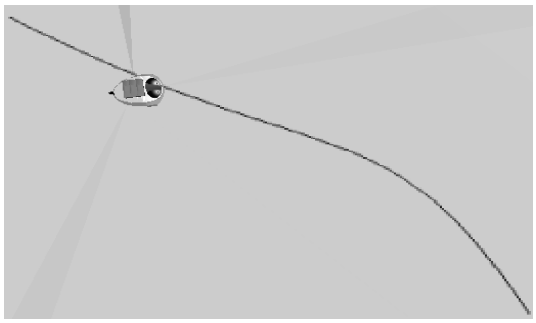
La figure V.13 illustre le mouvement du robot après application de la loi de commande.



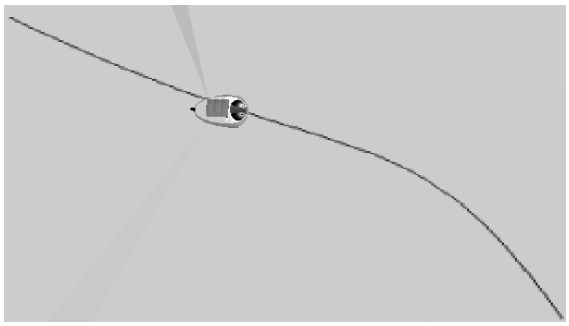
a) le robot est dans une position initiale quelconque



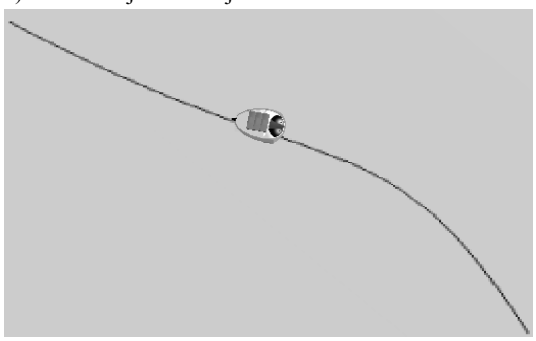
b) le robot cherche à joindre la trajectoire de référence



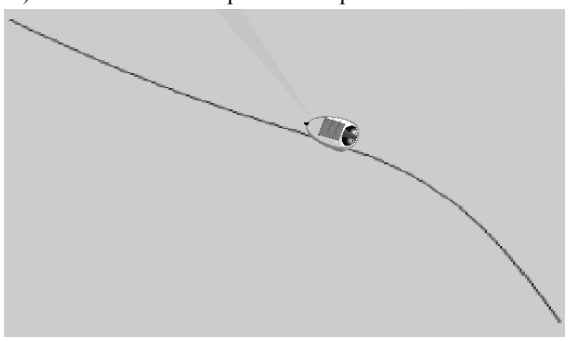
c) le robot rejoint la trajectoire



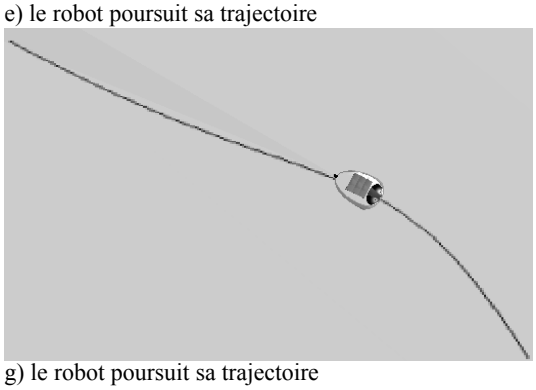
d) le robot effectue l'opération de poursuite



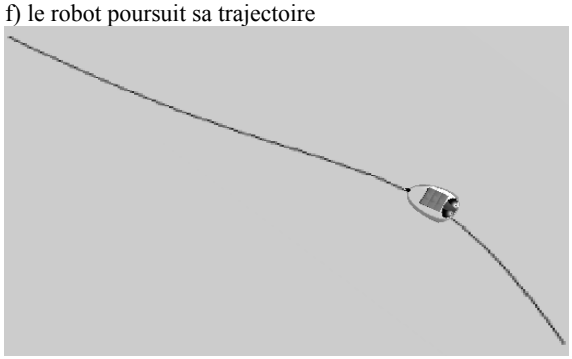
e) le robot poursuit sa trajectoire



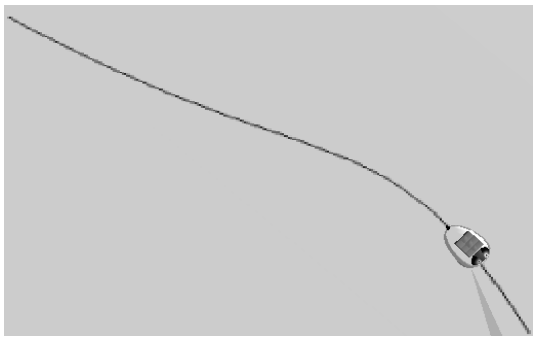
f) le robot poursuit sa trajectoire



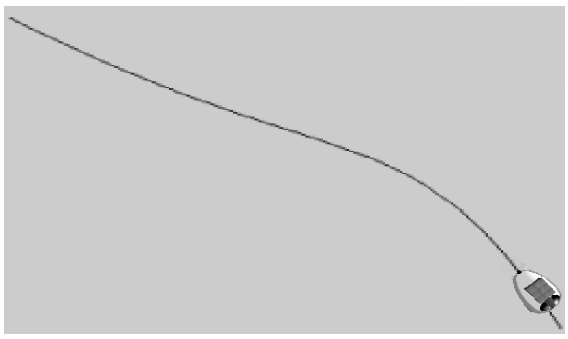
g) le robot poursuit sa trajectoire



h) le robot poursuit sa trajectoire



i) le robot poursuit sa trajectoire



j) le robot atteint la position finale

Figure V.13. Exemple de simulation montrant le robot Pekee qui suit une trajectoire donnée.

Nous remarquons que, quelque soit la position initiale du robot, il commence par rattraper la trajectoire ensuite il effectue l'opération de poursuite.

III.5 Test sur le robot Pekee

Tout ce que nous avons décrit dans les sections précédentes sur les techniques de poursuite de trajectoire avait été testé sur le simulateur RSL. Cette démarche restera valable sur lorsqu'on veut faire des tests sur le robot Pekee.

La principale difficulté lorsqu'on travaille sur le vrai robot est l'isolation de la trajectoire par rapport au milieu extérieur. Quand on travaille sur le simulateur il suffit de créer un environnement adéquat où la trajectoire est isolée. Ceci n'est pas le cas dans le monde réel.

Pour détourner ce problème, nous avons pensé à utiliser des formes géométriques, par exemple des cercles. Ces derniers seront détectés en utilisant la transformée de Hough.

Dans ce cas, nous pouvant isoler des cercles par rapport à l'environnement extérieur ensuite nous calculons les centres des cercles qui seront les coordonnées de la trajectoire à suivre.

Le tableau ci-dessous donne l'idée générale de l'algorithme de la transformée de Hoogh :

```
Définir un masque Mask1 (la forme q'on veut détecter);
Dilater le masque  Mask2
Créer un nouveau masque Mask3 (qui représente la différence entre les deux
masques précédents)
Pour i allant de 1 à Haut
  Pour j allant de 1 à Larg
       Im1=Mask3 + Im_Cont1(i,j)
       Im2=Mask2 – Im1
          Pour i1 allant de 1 à Haut1
            Pour j1 allant de 1 à Larg1
               Si Im2 ≠ 0
                   Seuil1 = Seuil1 +1
               Fin Si
            Fin
          Fin
          Si Seuil1<Seuil
             Coord1(k)=i+R/2
             Coord2(k)=j+R/2
              k=k+1
          Fin Si
  Fin
Fin
```

Tableau V.7. Algorithme de la transformée de Hough

→ Haut (respectivement Larg) est la hauteur (respectivement la largeur) de l'image contours (on recherche les cercles dans l'image contours).

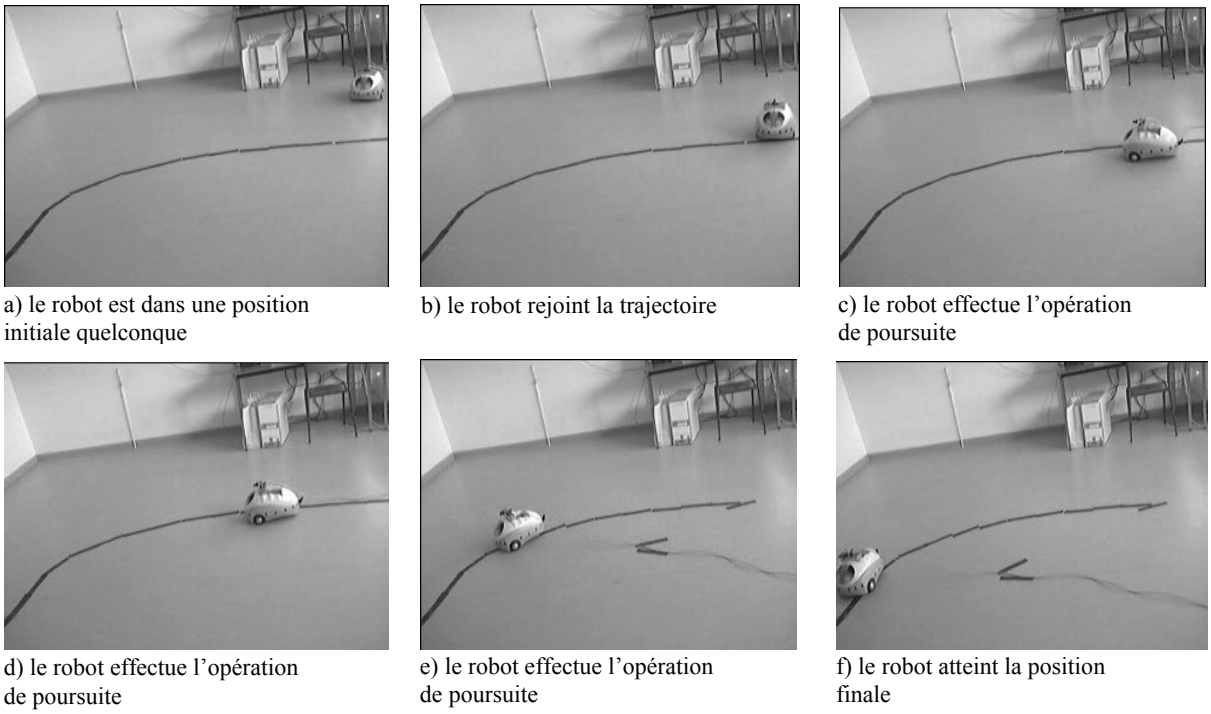
→ Im_Cont1 est une portion de l'image contour de même taille que le masque

→ Haut1 (respectivement Larg1) est la hauteur (respectivement la largeur) du masque.

→ « Seuil » est une marge d'erreur qui nous permet d'accepter ou non la forme détectée,

→ Coord1 et Coord2 sont les vecteurs qui contiennent les coordonnées des différents cercles.

Pour appuyer notre démarche, nous avons développé un exemple réel de poursuite de trajectoire stabilisante tracée sur le sol (figure V.14).



a) le robot est dans une position initiale quelconque

b) le robot rejoint la trajectoire

c) le robot effectue l'opération de poursuite

d) le robot effectue l'opération de poursuite

e) le robot effectue l'opération de poursuite

f) le robot atteint la position finale

Figure V.14. Exemple montrant le robot Pekee suivant une trajectoire (stabilisante) tracée sur le sol.

Nous remarquons que le robot qui est dans une position initiale arbitraire essaye de rejoindre la trajectoire. Ensuite, il essaye de la poursuivre jusqu'à qu'il arrive à la position finale qui est la position d'équilibre.

IV. Conclusion

Nous avons, tout au long de cette partie, présenté les éléments nécessaires pour réaliser une poursuite de trajectoire en utilisant l'information sur les images acquises par une caméra placée sur le robot.

Nous avons commencé par expliquer quelques éléments logiciels (les dll par exemple) utilisés pour réaliser les différents traitements (connexion, acquisition des images, application de la

commande...) sur le robot. Ensuite nous avons effectué des opérations de traitement d'images sur les images capturées par la caméra (détection de contours, redressement, discrétisation). A partir de ces résultats nous avons appliqué la commande correspondante au robot.

Cette démarche présente des performances remarquables pour commander un robot mobile de type voiture afin de poursuivre une trajectoire donnée. Cette trajectoire est calculée mathématiquement en utilisant les techniques de commande par retour d'état discontinu invariant dans le temps.

Conclusion générale

Conclusion générale

Dans ce travail, nous nous somme intéressés au problème de la commande des systèmes non-holonomes dans un but de stabilisation et ceci en faisant appel simultanément aux techniques de contrôle non-linéaires et aux techniques de vision par ordinateur.

L'objectif de notre travail a été d'élaborer une technique de commande pour la stabilisation en un temps fini d'une classe particulière de systèmes non-holonomes (robots mobiles à roues).

Afin d'atteindre cet objectif, nous avons présenté, dans un premier temps, les bases théoriques relatives aux systèmes non-holonomes. Pour cela nous avons établi un état de l'art sur les différents robots mobiles. Ici, nous nous sommes intéressés aux robots dits « unicycles ou différentiels » (exemple : robot Pekee).

Nous avons ensuite présenté notre problématique qui consiste à établir une stratégie de contrôle des systèmes non-holonomes pour assurer leur stabilisation en un point fixe, c'est-à-dire trouver une loi de commande tel que le point d'équilibre qui représente généralement l'origine, soit stable. Dans ce cadre, nous avons recensé trois approches de stabilisation qui sont : commande par retour d'état continu variant, commande par retour d'état discontinu invariant et commande hybride. Nous avons opté pour la première approche qui semble la mieux adaptée à notre problème.
Concernant toujours notre problématique, nous nous sommes penché à la question de stabilisation par poursuite de trajectoire en faisant appel aux techniques de vision. Cette démarche nous semble intéressante dans le cas où le robot se déplace dans un environnement contraint. Dans ce cas, la vision permet au robot d'acquérir les informations pertinentes sur son environnement.

Pour mieux cerner l'approche de poursuite de trajectoire par vision, nous avons établi une synthèse des différentes techniques de traitement d'images utilisées en robotique mobile.

Pour la deuxième partie de ce mémoire, nous l'avons consacré aux différentes contributions concernant la mise en œuvre d'une stratégie de stabilisation de robot unicycle en faisant appel aux lois de commande automatique et aux méthodes de vision par ordinateur.

A propos de la partie commande automatique, nous avons développé deux algorithmes de stabilisation autour d'un point fixe (l'origine). Le premier algorithme, propose une loi de commande par retour d'état discontinu appliquée à la forme chaînée du robot. Le deuxième algorithme se penche vers la technique de commande dite « linéarisation par bouclage ». Les deux algorithmes ont été testés, en simulation, sur le robot Pekee.
En outre, nous avons effectué des essais de robustesse en supposant que le robot est soumis à des perturbations extérieures (bruits stochastiques et bruits déterministes). Les résultats obtenus montrent bien l'insensibilité du robot aux bruits lorsqu'on lui applique une commande stabilisante.

Après s'être concentré sur le problème de stabilisation de systèmes non-holonomes, nous avons synthétisé et implémenté sur le robot Pekee un algorithme de commande qui garantit la stabilisation par suivi de trajectoire. Nous avons utilisé la trajectoire stabilisante, générée par simulation, comme information préliminaire dans l'étape d'implémentation. Par la suite, nous avons tracé la trajectoire sur le sol et placé le robot dans une position initiale quelconque. Par l'intermédiaire de la caméra montée sur le robot, des opérations de traitement d'images sont effectuées pour identifier la trajectoire et ainsi commander le robot pour effectuer l'opération de poursuite.

A l'issu de cette étude, des résultats expérimentaux sur le robot Pekee ont montré l'efficacité de l'approche proposée pour réaliser la stabilisation.

A l'issue de ce travail de magistère, plusieurs problèmes demeurent toutefois en suspens et d'autres méthodes restent à développer. Nous présentons ici ce qui nous semble être le cadre d'investigations où des avancées importantes sont tout à fait envisageables.

Les concepts théoriques introduits dans ce mémoire peuvent conduire à plusieurs extensions ou applications futures. En effet, un très grand nombre de problèmes en ingénierie (robotique, systèmes électriques, problèmes de routage et de congestion, . . .) fait appel aux techniques de commande stabilisante ainsi qu'aux techniques poursuite.

En plus, dans ce mémoire, nous nous sommes intéressés au problème de poursuite de trajectoires admissibles et sans collision. Il serait intéressant d'étudier la possibilité de réaliser une stabilisation en assurant l'évitement de collision en présence d'obstacles. Ici, de nouveaux algorithmes peuvent être étudiés et implémentés sur le robot Pekee.

Références bibliographiques

Références bibliographiques

[**BAY 08**] Bayle B, "Robotique mobile". Rapport de l'Ecole Nationale Supérieure de Physique de Strasbourg, 2008.

[**FIL 04**] Filliat D, "robotique mobile", Cours C10-2 ENSTA, 2 octobre 2004.

[**VEN 97**] Vendittelli, M, Laumond, J. P, "Visible positions for a car-like robot amidst obstacles". In: Laumond, J.-P., & Overmars, M. (eds), Algorithms for robotic motion and manipulation. A.K. Peters, 1997.

[**DEF 07**] Defoort M, "Contributions a la planification et a la commande pour les robots mobiles coopératifs". Thèse doctorat de l'école centrale de Lille, 22 octobre 2007.

[**JIA 99**] Jiang K, Seneviratne L-D, Earles S. W. E, "A Shortest Path Based Path Planning Algorithm for Nonholonomic Mobile Robots". Journal of Intelligent and Robotic Systems 24: 347–366, 1999.

[**KAB 04**] Kabanza F, "Planification en IA et Planification des trajectoires". Support de cours, département d'informatique, Université de Sherbrooke, IFT 702, ETE 2004.

[**LAV 06**] Lavalle S. M, "Planning Algorithms". Cambridge University Press, Cambridge, 2006.

[**VER 94**] Vershik A M., Gershkovich V Ya., "Nonholonomic dynamical systems, geometry of distributions and variational problems". Dynamical Systems, volume 16 of Encyclopaedia of Mathematical Sciences. Springer, 1994.

[**MUR 93**] Murray R, Sastry S, "Nonholonomic motion planning: Steering using sinusoids". IEEE Transactions on Automatic Control, 38 :700–716, 1993.

[**WEI 67**] Weiss L, et Infante E, "Finite time stability under perturbing forces and on product spaces". IEEE Transactions on Automatic Control, AC-12(1):54–59, 1967.

[**WEC 98**] Weckesser P, Dillmann R, "Modelling unknown environments with a mobile robot". Robotics and Automation systems, vol 23, pp.293-300, Germany, 1998.

[**ZHO 92**] Zhao Y, Bement S. L, "Kinematics, Dynamics and Contol of Wheeled Mobile Robots". IEEE International Conference on Robotics and Automation, pp. 91-96, Nice, France, may 1992.

[**VAG 93**] Vaganay J., "Conception d'un système multisensoriel de localisation dynamique 3D pour robot mobile". Thèse de doctorat de l'Université Montpellier II, juillet 1993.

[**VAS 04**] Vasquez Govea D. A, Large F, Fraichard T, Laugier C, "High-speed autonomous navigation with motion prediction for unknown moving obstacles". In *IEEE-RSJ Int. Conf. on Intelligent Robots and Systems, IROS'04*, pages 82–87, Sendai, Japan, October 2004.

[**BOU 07**] Bouraine Sara, "Contribution à la localisation dynamique du robot mobile d'intérieur B21r en utilisant la plateforme multi sensorielle". Mémoire de Magistère de l'Université de Blida, octobre 2007.

[**LAU 01**] Laumond J. P, " la robotique mobile", Edition Hermès, Paris, 2001.

[**SLI 05**] Slimani S, "Commande de Robots Cinématiques par Asservissement Visuel ". Mémoire de Master de l'Université de Bourgogne, Dijon, 2005.

[**BRO 83**] Brockett R.W, "Asymptotic stability and feedback stabilisation". In Differential Geometric Control Theory, eds Boston: Birkhauser, pp.181-191, 1983.

[**SAM 95**] Samson C, "Control of chained systems: Application to path following and time-varying point-stabilization of mobile robots". IEEE Transactions on Automatic Control, 40 :64–77, 1995.

[**JIA 99**] Jiang Z. P, Nijmeijer H, "A recursive technique for tracking control of nonholonomic systems in chained form". IEEE Transactions on Automatic Control, 44:265–279, 1999.

[**BLO 94**] Bloch A. M, Drakunov S. V, "Stabilisation of Nonholonomic Systems via Sliding Modes". In Proceedings of the 33rd Conference on Decision and Control, Orlando, Florida USA, pp 2961-2963, 1994.

[**BEN 92**] Bennani M. K, Rouchou P, "Robust Stabilisation of Flat and Chained Systems", Proceedings of the European Control, vol 37. N°11, pp1746-1757, 1992.

[**SAM 90**] Samson C, "Velocity and torque feedback control of a nonholonomic cart". In International Workshop Nonlinear Adaptive Control: Issues in Robotics, France, 1990.

[**COR 92**] Coron J, "Global asymptotic stabilization for controllable systems without drift". Mathematics of Control, Signals and Systems, 5 :295–312, 1992.

[**POM 92**] Pomet J. B, "Explicit design of time-varying stabilizing control laws for a class of controllable systems without drift". Systems and Control Letters, 18 :147–158, 1992.

[**GOD 97**] Godhavn J, Egeland O, "A lyapunov approach to exponential stabilization of nonholonomic systems in power form". IEEE Transactions on Automatic Control, 42 :1028–1032, 1997.

[**DIX 00**] Dixon W. E, Jiang Z. P, Dawson D. M, "Global exponential set point control of wheeled mobile robots: a lyapunov approach". Automatica, 36:1741–1746, 2000.

[**JIA 01**] Jiang Z. P, Lefeber E, Nijmeijer H, "Saturated stabilization and tracking of a nonholonomic mobile robot". System and Control Letters, 42:327–332, 2001.

[**TAD 03**] Tadjine M, "Stabilisation d'un robot mobile tricycle par la technique Backstepping". Support de cours, Ecole Nationale Polytechnique, El-harrach, 2003.

[**BLO 89**] Bloch A, McClamroch N, "Control of mechanical systems with classical nonholonomic constraints". In IEEE Conference on Decision and Control, Tampa, USA, 1989.

[**BLO 92**] Bloch A, Reyhanoglu M, McClamroch N, "Control and stabilization of nonholonomic dynamic systems". In IEEE Transactions on Automatic Control, 37 :1746–1757, 1992.

[**HES 99**] Hespanha J, Liberzon D, Morse A, "Logic-based switching control of a nonholonomic system with parametric modeling uncertainty". Systems and Control Letters, pp. 167–177, 1999.

[**CAN 92**] Canudas de Wit C, Sordalen O, "Exponential stabilization of mobile robots with nonholonomic constraints". IEEE Transactions on Automatic Control, pp.1791–1797, 1992.

[**AST 96**] Astolfi A, "Discontinuous control of nonholonomic systems. System and Control Letters", 27 :37–45, 1996.

[**CHW 04**] Chwa D, "Sliding-mode tracking control of nonholonomic wheeled mobile robots in polar coordinates". IEEE Transactions on Control Systems Technology, pp. 637–644, 2004.

[**FLO 03**] Floquet T, Barbot J. P, Perruquetti W, "Higher-order sliding mode stabilization for a class of nonholonomic perturbed systems". Automatica, pp. 1077–1083, 2003.

[**DRA 05**] Drakunov S. V, Floquet T, Perruquetti W, "Stabilization and tracking control for an extended heisenberg system with a drift". Systems and control Letters, 54 :435–445, 2005.

[**CAN 95**] Canudas de Wit C, Khennouf H, "Quasi-continuous stabilizing controllers for nonholonomic systems: Design and robustness consideration". In Proceedings of 3rd European Control Conference, pp. 2630-2635, Rome, Italy, 1995.

[**REY 95**] Reyhanoglu M.,"On the stabilisation of a class of non-holonomic systems using invariant manifold technique". In Proceedings of the 34th IEEE conference on Decision and Control, pp. 2125-2136, New Orleans, LA, 1995.

[**TSI 93**] Tsiotras P, Corless M, Longuski J. M, "Invariant manifold techniques for attitude control of symmetric spacecraft". In Proceedings of the 32nd IEEE Conference on Decision and Control, pp. 1470-1475, San Antonio, TX, 1993.

[**TAY 97a**] Tayebi A, Tadjine M, Rachid A, ”Invariant manifold approach for the stabilization of nonholonomic systems in chained form: application to a car-like mobile robot”. In Proceedings of the 36th IEEE Conference on Decision Control, (CDC’97), pp. 4038-4043, San Diego, 1997.

[**GUE 05**] Guemghar K, ”On the use of input-output feedback linearization techniques for the control of nonminimum-phase systems”. Thèse de doctorat de l’É cole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Suisse, 2005.

[**ZER 06**] Zerar M, ”Contribution à la caractérisation LPV d’une classe de systèmes non linéaires pour la synthèse de lois poursuite robuste”. Thèse de doctorat de l’Université Bordeaux 1, France, 2006.

[**LAM 02**] Lamnabhi-Lagarrigue F, Rouchon P, ” Systèmes non linéaires”, Edition Lavoisier, Paris, 2002.

[**TOU 90**] Toumazet J. J, “ Traitement de l’image par l’exemple”. Edition Sybex, Paris, 1990.

[**JOL 01**] Jolion J. M, ” les systèmes de vision”, Edition Hermès, Paris, 2001.

[**MAL 07**] Malek S, “Gestion et contrôle des connaissances dans un système multi-agents dédie a l’analyse d’images”. Mémoire de magister de l’Université de Blida, février 2007.

[**LAM 01**] Lamaty P, “Opérateurs Matériels de Niveau Intermédiaire pour le Traitement Temps Réel des Images”. Thèse de Doctorat en Sciences Traitement de l'Image et du Signal, Février 2001.

[**BEN 02**] Benallal M, “Système de calibration de Caméra Localisation de forme polyédrique par vision monoculaire”. Thèse Pour l'obtention du titre de Docteur de l'É cole des Mines de Paris, 29 novembre 2002.

[**TAY 97b**] Tayebi A, Tadjine M, Rachid A, “Discontinuous Control Design for the Stabilization of Nonholonomic Chained Systems : Application to a car-like mobile robot Mobile Robot”. In proceeding of the 36th IEEE Conference on Decision and Control, (CDC’97), San Diego,1997.

[**JAU 07**] Jaulin L, ”Robotique mobile”. Support de cours, Laboratoire d’Ingénierie des Systèmes Automatisés, Université d’Angers, France, Octobre 2007.

[**BEN 06**] Ben Ferdia A, ”Commande non linéaire d’un moteur synchrone à aimants permanents”. Mémoire de magistére de l’Université de Batna, Algerie, 2006.

[**ISI 89**] Isidori A, ”Nonlinear control systems: communication and control”. Engineering Serie, Berlin, Springer, Verlag, second edition, 1989.

[**HAM 07**] Hamerlain F, ”Stratégies de commande par modes glissants d'ordre supérieur appliquées à des robots mobiles à roues’’. Thèse de doctorat de l’école centrale de Lille, 5 décembre 2007.

[**COU 05**] Coutant A, “La méthode des contours actifs en traitement des images”, Mémoire pour l’examen probatoire en Calcul Scientifique, 02 février 2005.

[**PRE 91**] Preciado Ruiz A, “Sur la modélisation, la localisation et le contrôle d’un robot mobile ”. Thèse de doctorat de l’Université Technologique de Compiègne, France, Juillet 1991.

[**DJE 05**] Djekoune A.O, “ATRV2 : définition et cinématique ”. Rapport Interne N°1, Division Robotique et productique, Centre de Développement des Technologies Avancées, 2005.

[**KLE 02**] Klein E, ”Machine à courant continu : fonctionnement et structure interne”. Support de cours, Laboratoire de Recherches sur les Sciences de la Matière, Saclay, France, 2002.

[**MAD 08**] Madani S, “Synthèse et simulation des lois de commande d’un robot mobile unicycle”. Mémoire de projet, Centre de Développement des Technologies Avancées, Alger, Octobre 2008.

Annexe A

Annexe A Plateforme de développement : WANY Robotics

I. Introduction

Nous allons présenter tout au long de cette section les composantes matérielle et logicielle du Robot Pekee [SLI 05]. En outre, nous s'étalons sur les différents accessoires internes et externes qui accompagnent le robot.

II. Partie hardware de la plateforme

Le Robot Pekee (figure A.1) comporte deux roues motrices avants, contrôlées indépendamment par deux moteurs à courant continu, et une roue libre arrière. Cette disposition permet une mobilité très large. A titre d'exemple, le robot peut effectuer un demi tour ou un tour complet sur place.

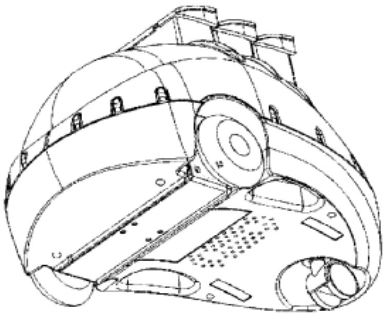


Figure A.1. Vue de dessous du robot Pekee.

II.1 Spécifications

Le robot Pekee est une machine caractérisée par :

- Les dimensions
  - Longueur totale: 40 cm;
  - Longueur sans la queue : 37 cm;
  - Largeur totale: 25.5 cm;
  - Largeur entre les roues motrices: 22.5 cm;
  - Hauteur sans les accessoires: 21 cm.
- Le poids
  - 2.9 kg sans les accessoires.
- L'autonomie
  - 15 heures au maximum (avec des petits mouvements et sans les accessoires);
  - Une heure et demie avec des mouvements continus.
- La rotation
  - 360° dans un cercle de 70 cm de diamètre.

II.2 Les composants internes

Le robot est constitué d'un ensemble d'accessoires définies comme suit :

- 15 capteurs infrarouges;
- 2 odomètres;
- 1 capteur d'impacts;
- 2 gyromètres;
- 1 capteur de lumière;
- 2 capteurs de température
- 2 batteries rechargeables 12v;
- 1 buzzer à fréquence variable;
- 1 indicateur de charge des batteries (Led rouge);
- 1 led rouge;
- 1 liaison infrarouge pour la communication entre le robot et des périphériques externes;
- 1 liaison série infrarouge pour le transfert de donnée entre le robot et un PC;
- 1 microcontrôleur Mitsubishi 16Mhz (16 bits), avec une ROM de 256 KB et une RAM de 20 KB;
- 5 slots pour les accessoires avec bus I2C (voir figure A.2);
- 5 slots (slots OPP) pour les accessoires.

En outre, le robot est constitué de cinq slots qui sont utilisés pour brancher les différents accessoires permettant de commander la machine (figure A.2).

On peut trouver aussi quatre prises destinées à des périphériques compatibles I2C et une autre prise pour le module de programmation du robot.

En plus, le robot est équipé d'un ensemble de connecteurs pour établir le lien entre les différents composant. Nous avons dans ce cas :

- 5 (*120 broches) connecteurs pour le bus OPP;
- 4 (*10 broches) connecteurs males (type HE10), pour les accessoires du PC;
- 1 (*10 broches) connecteur femelle (type HE10), pour re-programmer le microcontrôleur du Robot;
- 1 connecteur rectangulaire avec deux broches males pour le mode de charge rapide des batteries (figure A.3);
- 1 connecteur avec 1 broche circulaires pour le mode de charge lent (figure A.3)

- 2 contacts métalliques situés au dessous du robot du coté de la roue arrière pour un mode de charge rapide en utilisant une station de charge appropriée (figure A.4).

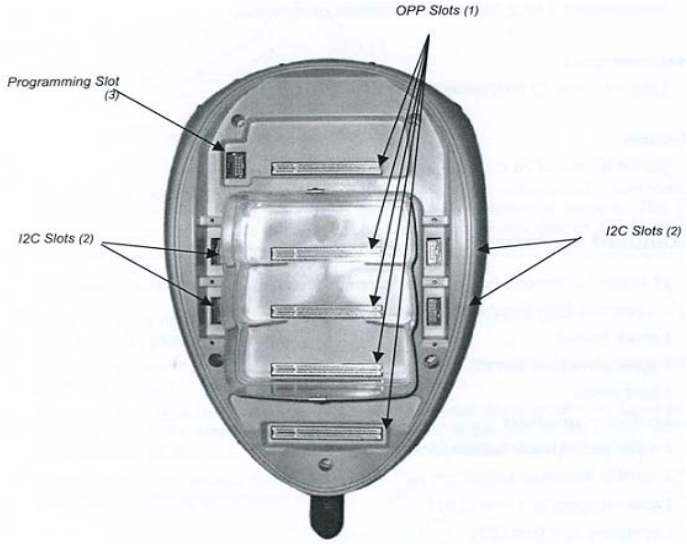


Figure A.2. Vue aérienne (sans le couvercle) des connecteurs du robot

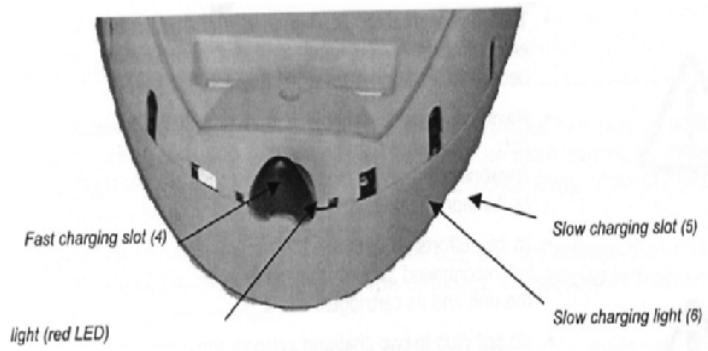


Figure A.3. Les connecteurs de charge et les indicateurs

III. Mise en marche du robot

Pour de meilleures performances, les batteries doivent être chargées complètement avant chaque utilisation.

Pour allumer le robot, on utilise un bouton qui est situé au dessous du robot (voir figure A.4). Un appui léger sur ce bouton met le robot en marche. Ceci est confirmé par un bip.

Remarque

Un appui long (plus de 4 secondes) sur le bouton conduit à l'extinction du robot.

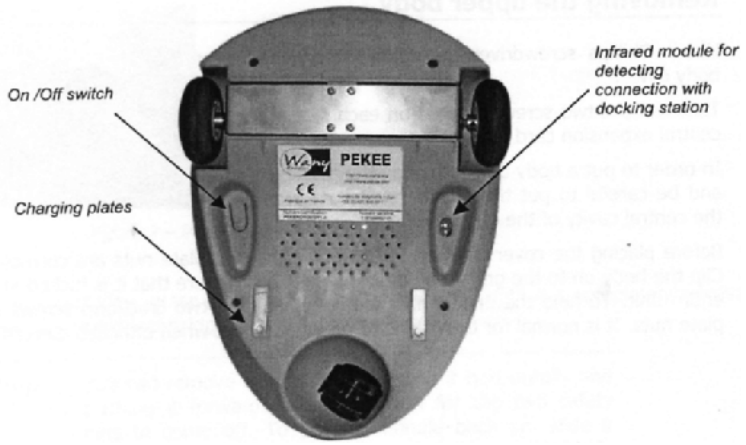


Figure A.4. Vue de dessous du robot Pekee.

II.2 Les composants externes du robot

Le Robot Pekee est conçu de telle sorte qu'il peut supporter des composants externes qui peuvent être branchés en utilisant les bus OPP. Ainsi, le robot est accompagné d'un ou de plusieurs composants comme le PC de bord.

Remarque

Pour placer un composant, il faut le faire glisser dans l'une des cases et appuyer dessus jusqu'à ce que l'on sente une bonne résistance (voir figure ci-dessous).

Pour enlever un composant, il faut le tirer et ensuite le faire glisser verticalement vers le haut.

Pour éviter de provoquer des décharges électriques ou d'endommager l'unité du Robot, il faut toujours éteindre le robot avant d'insérer ou d'enlever un composant.

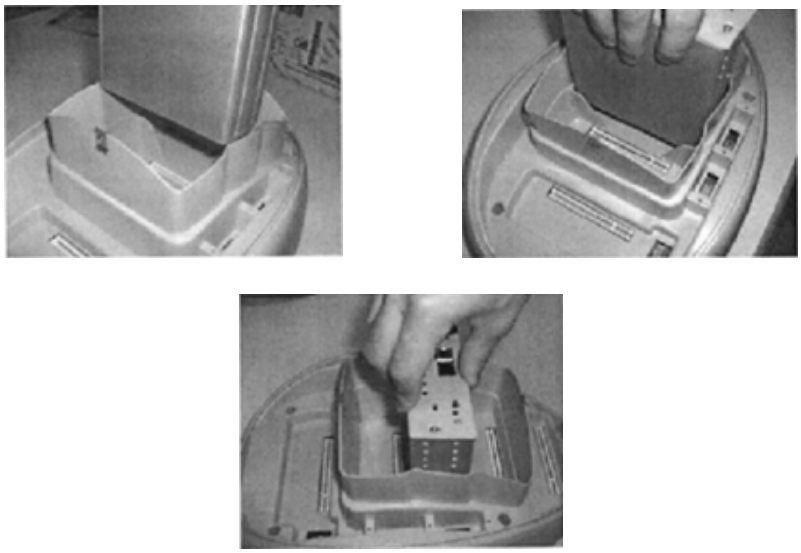


Figure A.5. Insérer et retirer un composant

II.2.1 Le PC de bord

Le PC de bord (figure A.6) est un ordinateur complètement autonome qui est capable de lancer quelques systèmes d'exploitation compatibles.

Il est constitué d'un processus compatible Intel 486, de mémoires RAM et ROM, d'un connecteur SVGA ainsi que d'autres connecteurs pour clavier, souris et réseau Ethernet (10/100Mbps).

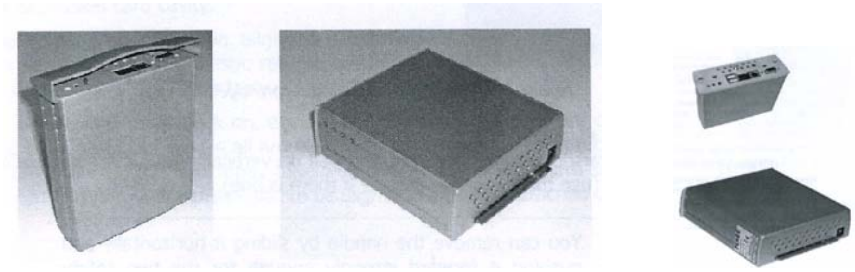


Figure A.6. Pc de bord associé au Robot

Spécifications techniques du Pc de bord (figure A.7) :

- ***Caractéristiques***
  - Microprocesseur Intel 486 (75 Mhz);
  - SDRAM (32MB PC100);
  - Disque dur (128MB);
  - 1 contrôleur SVGA avec 4 MB de mémoire partagée;
  - 1 contrôleur Ethernet (10/100 Mbps): IEEE 802.3;
  - 1 contrôleur USB V1.1 (12Mbps);
  - 1 contrôleur PS/2 (clavier/souris);
  - 1 bus compatible OPP;
  - 1 bouton de mise sous tension;
  - 1 LED qui indique l'état de l'unité (allumée/éteinte);
  - 1 bouton de réinitialisation.
- ***Connecteurs***
  - 1 port femelle (Sub-D 15 points);
  - 1 connecteur femelle (mini-DIN) PS/2 (Clavier/souris);
  - 1 connecteur Ethernet (RJ45);
  - 1 connecteur USB double (type A);
  - 1 connecteur coaxial male qui permet une recharge rapide et autonome de la batterie;
  - 1 connecteur OPP.

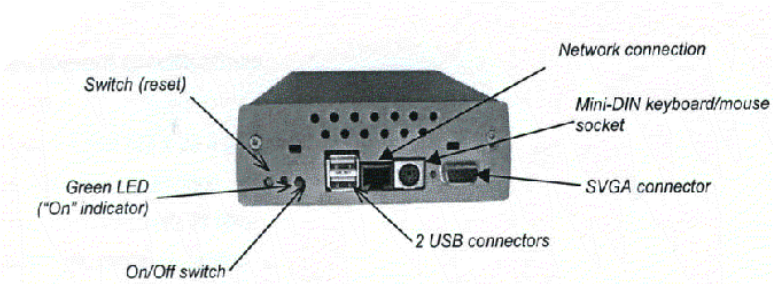


Figure A.7. Vue de dessus de l'unité

On peut brancher au robot un écran externe, deux périphériques USB et un connecteur Ethernet. En plus, le connecteur mini-Din permet de brancher un clavier et une souris en utilisant l'adaptateur fourni.

II.2.2 La caméra vidéo

La caméra vidéo est branchée sur l'ordinateur de bord du robot (figure A.8) et la carte d'acquisition vidéo est incluse à l'intérieur de l'unité. Cette caméra est munie de son propre câble d'alimentation.

Cet outil ajoute de nouvelles caractéristiques au Robot et élargit, ainsi, le champ des applications que l'on pourra développer. La caméra peut communiquer avec le Robot à travers l'interface OPP et avec d'autres périphériques connectés au Robot.

La caméra est branchée à l'unité via un connecteur coaxiale et un câble d'alimentation 5V. Elle est aussi munie d'une batterie interne qui peut être chargée directement en utilisant un connecteur spécifique ou en utilisant le chargeur du Robot lorsque l'unité est branchée sur le Robot à travers le bus OPP.

Figure A.8. Caméra du robot

Spécifications techniques

- ***Caractéristiques***
  - Elle fait partie ou accompagne l'unité centrale du Robot;
  - Possède une carte d'acquisition vidéo.
- ***Connecteurs***
  - 1 connecteur coaxial male (SMB) pour se brancher sur l'unité;
  - 1 connecteur (mono) jack femelle pour l'alimentation de la caméra.

Spécifications de la caméra

- ***Caractéristiques***
  - Dimension (22x22x28 mm);
  - Faible consommation;
  - Lentille intégrée avec un filtre IR;
  - Contrôle de prise d'images automatique;
  - Résolution: de 160x120 jusqu'à 640x480 pixels;
  - Caméra CMOS (1/3").

- ***Connecteurs***
  - 1 sortie vidéo;
  - 1 câble d'alimentation 5v.

Pour les applications vidéo, nous utilisons le connecteur coaxial (CVBS) qui délivre le signal venant de la caméra.

II.2.3 L'antenne WIFI

Cet outil ajoute la possibilité d'établir une communication sans fil avec le Robot. L'antenne comporte un pont Ethernet sans fil (WI-FI 802.11), figure A.9. Elle est alimentée directement à travers le bus OPP du Robot.

Spécifications techniques

Câble RJ45;

Antenne externe;

CD d'installation avec un guide d'utilisation;

Transfert de données haut débit (>11Mbps).

- ***Caractéristiques***

Ne nécessite pas de pilote spécial;

Facile à installer;

Portée;

A l'intérieur

> 50 m avec un débit de 11 Mbps;

> 80 m avec un débit de 5.5 Mbps ou plus bas;

A l'extérieur

> 150 m avec un débit de 11 Mbps;

> 300 m avec un débit de 5.5 Mbps ou plus bas.

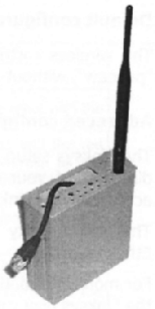


Figure A.9. L'antenne WI-FI

II.2.4 L'alimentation

Pour la mise en marche du robot, l'opération de chargement est indispensable. Ceci est utilisé par l'intermédiaire d'un boîtier d'alimentation fournie (figure A.10).

Le chargement se fait de la façon suivante : on branche la prise d'alimentation dans un secteur. Ensuite, on brancher l'autre extrémité du câble dans la prise sur le Robot (figure A.11). Le chargeur s'arrête automatiquement après que les batteries soient complètement chargées après environ 1h20.

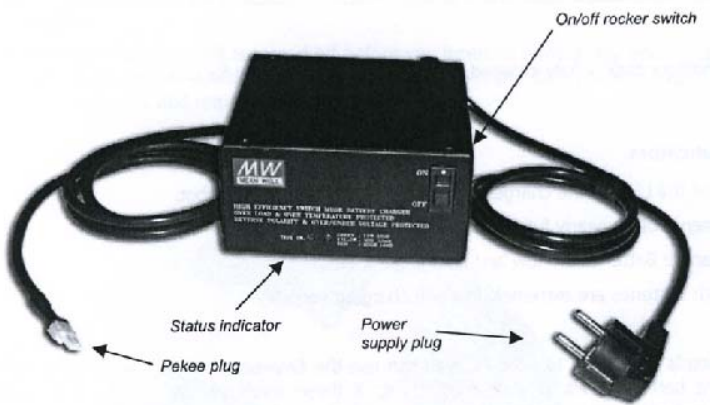


Figure A.10. Alimentation du Robot

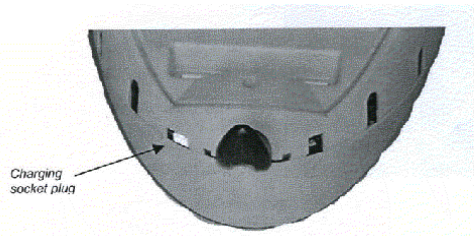


Figure A.11. Point de charge

L'état de la batterie est donné par un indicateur représenté par une LED lumineuse. La couleur de la LED indique l'état des batteries du Robot:

- Verte : Batteries pleines,
- Orange : Batteries faibles et elles se chargent normalement,
- Rouge : Batteries extrêmement faibles et elles se chargent normalement.

Si le Robot est branché sur un PC, on peut utiliser l'application C:\wany\binaries\up002c pour visualiser l'état des batteries, figure A.12.

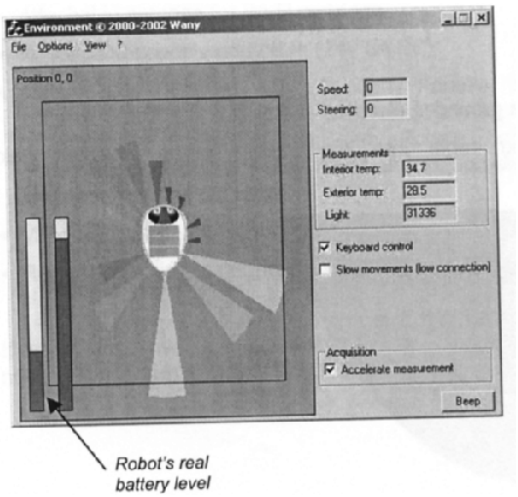


Figure A.12. Indicateur d'état des Batteries

III Partie software de la plateforme

La plateforme de développement du Rebot comporte une variété de logiciels comme:

RSL (Robotic Software Laboratory 2.0). Ce dernier est un logiciel de développement d'environnements pour écrire, tester et simuler des applications sur un Pc avant de les exécuter sur le vrai robot.

Deux éléments sont essentiels pour la mise en marche du robot. Il s'agit d'un :

Kit de développement vidéo

Il nous permet la capture et la manipulation des données vidéo qui proviennent de la caméra du Robot.

Flasher du microcontrôleur

Il nous permet de programmer la mémoire flash du microcontrôleur du robot. Il donne aussi la possibilité de modifier des fonctions OPP (Open Platform Parallel).

Il contient aussi des versions d'essais d'outil de Mitsubishi (comme Task Manager) nécessaires pour compiler le code C en code assembleur et le télécharger sur le microcontrôleur du Robot.

III.1 Programmation du Pekee

Cette section décrit les différentes méthodes de programmation fournies par la plateforme WANY Robotics utilisées pour personnaliser le comportement du robot Pekee.

III.1.1 Programmer le robot directement

Les équipements du Robot contiennent des librairies nécessaires écrites en C. A l'aide de ce code, on peut avoir accès aux : capteurs infrarouge, moteurs, les capteurs d'impacts, les

gyromètres ainsi que les capteurs de température. Ceci nous permettra d'obtenir en instantané les informations nécessaires sur le comportement du robot.
Le robot peut être programmé directement en utilisant un microcontrôleur. Ce dernier traduit les programmes de commandes écrits en C en un langage bas niveau (assembleur).

III.1.2 Simuler les programmes sur le simulateur RSL puis les télécharger sur le microcontrôleur du Robot

Il s'agit d'effectuer des tests sur un simulateur en exécutant le programme de commande écrit en C. Dans le cas où la simulation donne les résultats attendus. Le programme est ensuite transféré à l'unité du robot où il sera traduit en langage assembleur du microcontrôleur.

III.1.3 L'interaction entre les composantes logicielle et matérielle du Pekee

Le diagramme de la figure A.13 illustre comment les différents composants du Robot communiquent entre eux.

La figure A.14 illustre l'interface du simulateur RSL ainsi que l'interaction entre les différents composants des logiciels.
Il faut se rappeler qu'on peut écrire des programmes microcontrôleur ainsi que des programmes haut niveau, et les deux peuvent être utilisés sur le vrai robot ainsi que sur le simulateur.

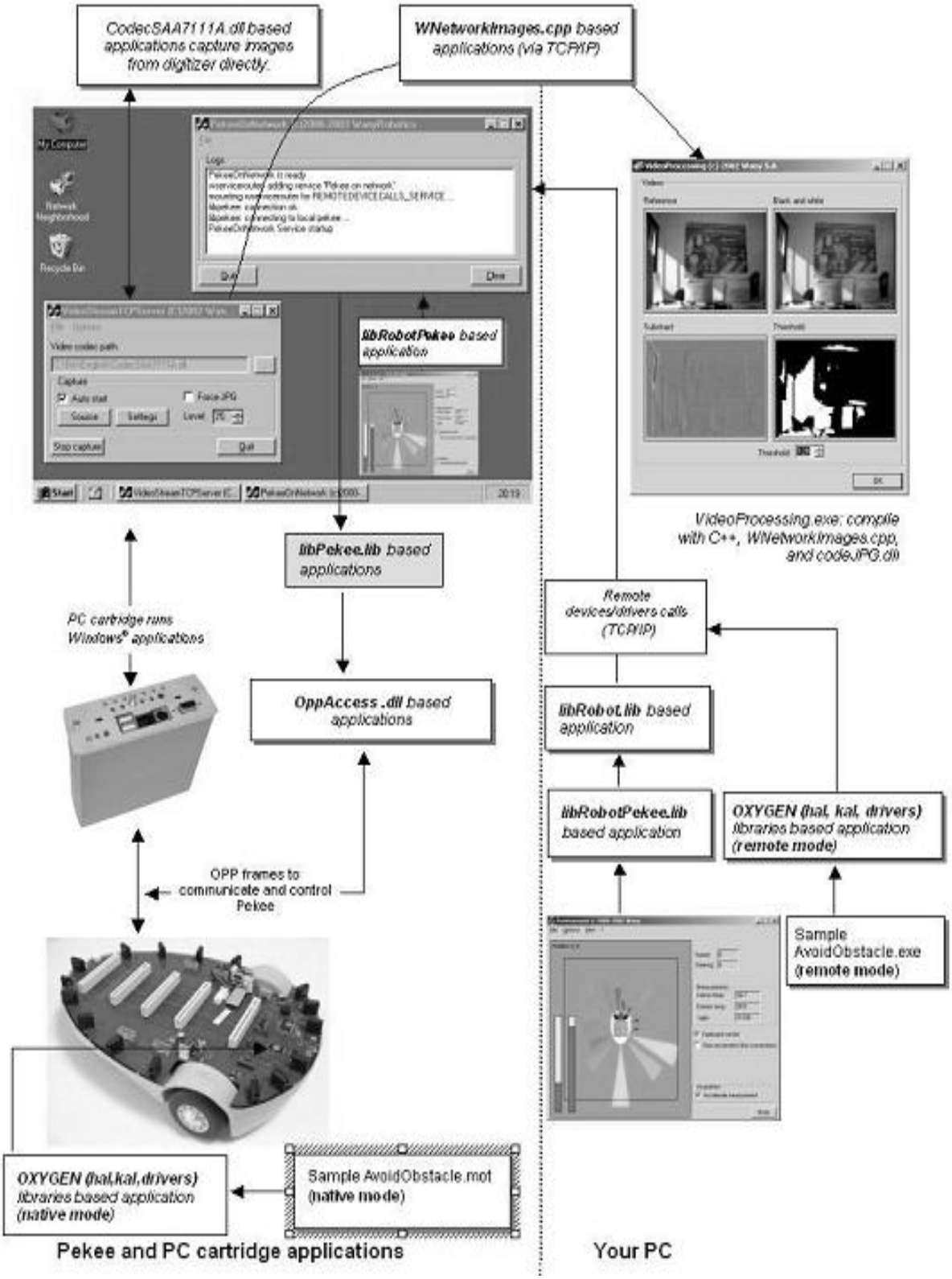


Figure A.13. Interaction entre les composantes matérielle et logicielle du vrai Pekee

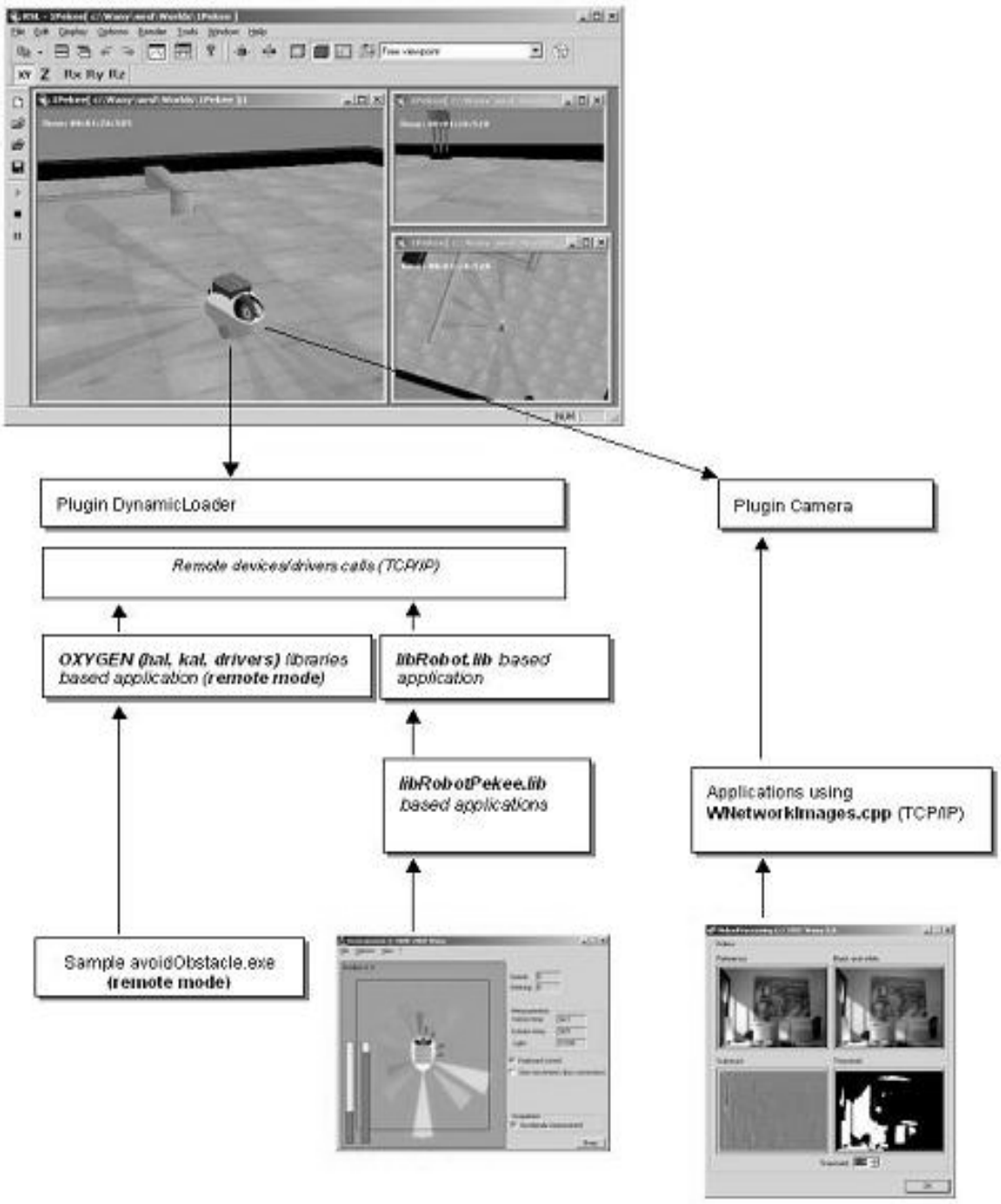


Figure A.14. Interaction entre les composantes matérielle et logicielle du simulateur de Pekee

Annexe B

Annexe B Asservissement de vitesse des moteurs à courant continu pour la commande du robot mobile Pekee

I. Introduction

Nous avons déjà réussi à trouver l'ensemble des commandes en vitesse (v et w) nécessaires pour stabiliser le robot (voir chapitre IV).

Cependant, nous envisageons d'étudier le comportement des moteurs qui actionnent les deux roues avant pendant le mouvement du robot et ceci lorsqu'on applique les commandes v et w.

Les moteurs qu'utilise le robot « Pekee » sont des moteurs à courant continu.

Nous étudions, dans notre cas l'allure de la tension appliquée aux moteurs pour générer les commandes adéquates.

Dans ce qui suit nous effectuons une étude sur le comportement des moteurs et la relations existante entre les vitesses de contrôle du robot et leurs tensions correspondantes.

II. Etude préliminaire

Etant donné un robot mobile de type unicycle (figure B.1) auquel est associé un repère local (O_R, X_R, Y_R) dont l'origine est placé sur le point médian de l'axe des deux roues avant.

Le robot mobile se déplace dans l'espace plan de repère (O, X, Y). Sa configuration complète dans ce repère peut être représentée par les trois valeurs (x, y, θ). En fonction de la variation de ses coordonnées dans le repère R, sa vitesse linéaire peut être déduite :

$$v = \sqrt{\dot{x}^2 + \dot{y}^2} \tag{B.1}$$

Le changement d'orientation du robot, c'est-à-dire, sa vitesse de rotation $\Omega = \dot{\theta}$, est caractérisé par :

$$\Omega = \dot{\theta} = \frac{d\theta}{dt} \tag{B.2}$$

Où θ est l'orientation instantanée du robot par rapport à R.

Le mouvement est crée par la rotation des trois roues motrices du robot mobile. Les deux paires de roues correspondant aux côtés gauche et droit du robot possèdent une vitesse de rotation instantanée.

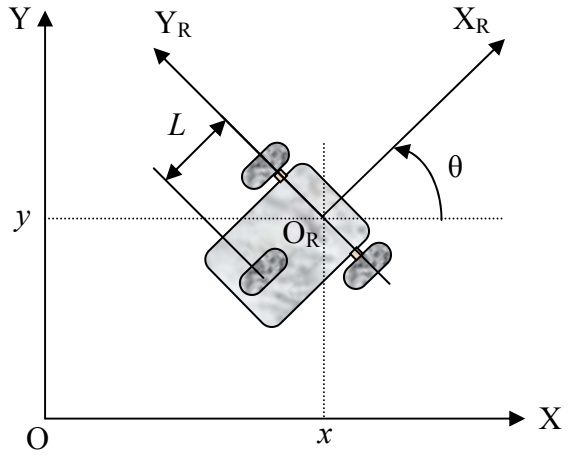


Figure B.1. Repère du robot mobile Pekee dans le repère R.

Afin de trouver les équations de déplacement du robot mobile, nous pouvons reprendre l'étude présentée dans [PRE 91]. Dans celle-ci, on utilise la description suivante du mouvement de deux points d'un même corps [DJE 05].

A chaque instant *t*, il existe un pseudo vecteur Ω(t) tel que pour tout point *P* du solide nous avons :

$$v_p = v_a + \Omega \wedge \overrightarrow{AP} \tag{B.3}$$

A est un point quelconque du solide Ω(t).

Puisque les deux roues motrices virtuelles sont liées mécaniquement par un axe virtuel, et que leur vitesse est connue (v_g et v_d), alors nous pouvons calculer la vitesse *v* du centre *C* de l'axe de traction virtuel (figure 2) de la façon suivante :

$$v = v_g + \Omega \wedge \overrightarrow{CA} \tag{B.4}$$

$$v = v_d + \Omega \wedge \overrightarrow{CB} \tag{B.5}$$

Si nous effectuons la somme des deux équations (B.4) et (B.5), nous obtenons l'expression suivante :

$$2v = v_g + v_d + \Omega \wedge (\overrightarrow{CA} + \overrightarrow{CB}) \tag{B.6}$$

La figure B.2 qui montre l'espace des vitesses du robot Pekee nous permet de déduire que les vecteurs $\overrightarrow{CA}$ et $\overrightarrow{CB}$ sont égaux en amplitude et opposés en sens. Ceci nous mène directement vers l'équation suivante :

$$v = \frac{v_g + v_d}{2} \tag{B.7}$$

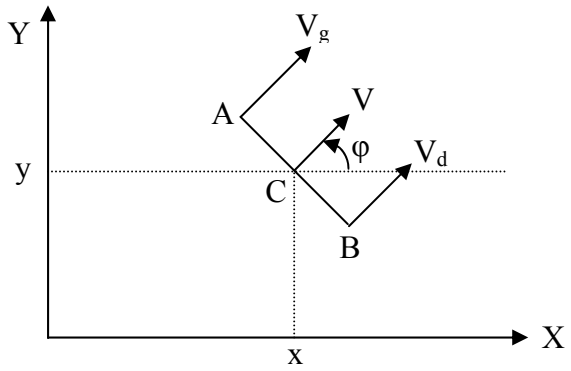


Figure B.2. Espace de vitesses du robot Pekee

Nous remarquons que cette vitesse a la même direction que v_g et v_d. Cette direction correspond à la direction de l'axe X_R du repère du robot mobile.

D'une autre part, si nous faisons la soustraction de l'équation (B.4) à l'équation (B.5), nous pouvons remarquer que le vecteur Ω est le résultat d'une paire de torsion orthogonale au plan du mouvement.

$$v_d - v_g + \Omega \wedge (\overrightarrow{CB} - \overrightarrow{CA}) = 0 \tag{B.8}$$

L'expression de la vitesse de rotation du robot mobile en fonction des vitesses linéaires des roues sera donc :

$$\Omega = \dot{\theta} = \frac{v_d - v_g}{2d} \tag{B.9}$$

Pour les cas de changement de position dans la direction de l'axe x et y du repère R, les expression des dérivées de x et de y sont donnés par :

$$\dot{x} = \frac{v_g + v_d}{2}\cos\theta \tag{B.10}$$

$$\dot{y} = \frac{v_g + v_d}{2}\sin\theta \tag{B.11}$$

Pour rappel, les expressions B.9, B.10 et B.11 constituent les équations cinématiques du robot Pekee.

En résumé, les deux moteurs à courant continu qui permettent le déplacement du robot Pekee génèrent l'ensemble des vitesses suivantes :

→ Vitesse de translation de la roue gauche : v_g

→ Vitesse de translation de la roue droite : v_d

→ Vitesse de rotation de la roue gauche : ω_g

→ Vitesse de rotation de la roue droite : ω_d

v_g et v_d sont reliés respectivement à ω_g et ω_d de la façon suivante :

$$v_g = \text{r}.\ \omega_g$$
$$v_d = \text{r}.\ \omega_d \qquad \text{(B.12)}$$

r : rayon de la roue.

Les résultats obtenus dans les sections III.1.4 et III.2.5 du chapitre IV donnent l'allure des commande v et w (ex : figure IV.4 et IV.16) appliquées au robot afin de le stabiliser.

Pour trouver les commandes correspondantes appliquées à chaque moteur, nous devons cherchons en premier lieu les expressions de (v_g, ω_g) et (v_d, ω_d) en fonction des v et w. Ceci est possible en se référant aux équations (B.7) et (B.9).

Soit v et w les commandes appliquées au robot Pekee. Nous avons :

$$\text{(B.7)} \Rightarrow v = \frac{v_g + v_d}{2}$$
$$\Rightarrow v_d + v_g = 2v \qquad \text{(B.13)}$$
$$\text{(B.9)} \Rightarrow w = \frac{v_d - v_g}{2d}$$
$$\Rightarrow v_d - v_g = 2d.w \qquad \text{(B.14)}$$

En faisant la somme des deux nouvelles équations (B.13) et (B.14) nous obtenons :

$$v_d = v + d.w \qquad \text{(B.15)}$$

$$v_d = \text{r}.\ \omega_d \Rightarrow \omega_d = \frac{v_d}{r}$$

$$\omega_d = \frac{v_d}{r} = \frac{v + d.w}{r} \qquad \text{(B.16)}$$

En remplaçant la valeur de v_d donnée dans l'équation (B.15) dans l'équation (B.13), nous obtenons l'expression de v_g donnée comme suit :

$$v_g = v - d.w \qquad \text{(B.17)}$$

$$v_g = \text{r}.\ \omega_g \Rightarrow \omega_g = \frac{v_g}{r}$$

$$\omega_g = \frac{v_g}{r} = \frac{v - d.w}{r} \qquad \text{(B.18)}$$

Les équations B.15, B.16, B.17, B.18 permettent de donner l'allure des vitesses de translation et de rotation générées par les deux moteurs.

Notre objectif consiste à trouver l'allure des tensions appliquées aux moteurs afin d'obtenir les vitesses définies précédemment. Pour ce faire, nous modélisons le fonctionnement des deux moteurs et nous effectuons une étude d'asservissement pour définir les tensions de commande correspondantes.

Et comme les moteurs utilisés sont identiques, nous effectuons des études de modélisation et de commande sur un seul d'entre eux. Le principe reste le même pour l'autre moteur.

III. Modélisation du moteur

Le moteur à courant continu est une machine électrique tournante qui fonctionne, comme son nom l'indique, à partir de tensions et de courants continus.

Les moteurs à courant continu sont constitués de deux parties principales. Le **stator** est la partie fixe du système. Il entoure la partie tournante, appelée **rotor** [KLE 02].

Le fonctionnement linéaire d'un moteur à courant continu est caractérisé par les équations électrique et mécanique suivantes :

- L'équation électrique du moteur :

$$v(t) = e(t) + R.i(t) + L\frac{di(t)}{dt} \tag{B.19}$$

Avec :

→ v(t) la tension de commande du moteur

→ e(t) la f.e.m. induite

→ R la résistance d'induit

→ i(t) le courant d'induit

→ L l'inductance de l'induit

- L'équation mécanique du moteur :

$$C_m(t) - C_r(t) = J.\frac{d\omega(t)}{dt} + f.\omega(t) \tag{B.20}$$

Avec :

→ $C_m(t)$ le couple moteur

→ $C_r(t)$ le couple résistant

→ J le moment d'inertie du moteur

→ f le coefficient de frottement visqueux

La transformation de Laplace des équations (B.19) et (B.20) donne respectivement :

$$I(p) = \frac{V(p) - E(p)}{R + Lp} \tag{B.21}$$

$$\Omega(p) = \frac{C_m(p) - C_r(p)}{f + Jp} \tag{B.22}$$

La constante de flux et de couple qui sera notée K relie les paramètres électriques aux paramètres mécaniques par les relations suivantes :

$$e(t) = K.\omega(t) \text{ et } C_m(t) = K.i(t) \tag{B.23}$$

L'expression de Ω peut s'écrire alors :

$$\Omega(p) = \frac{K.I(p) - C_r(p)}{f + Jp} = \frac{1}{f + Jp}\left(\frac{K}{R + Lp}\left(V(p) - E(p)\right) - C_r(p)\right) \quad \text{(B.24)}$$

A partir de l'équation (B.23) nous pouvons calculer E(p) de la façon suivante :

$$E(p) = K.\Omega(p) \quad \text{(B.25)}$$

En remplaçant l'expression de E(p) dans l'équation (B.24) nous obtenons :

$$\Omega(p) = \frac{1}{f + Jp}\left(\frac{K}{R + Lp}\left(V(p) - K.\Omega(p)\right) - C_r(p)\right) \quad \text{(B.26)}$$

Cette équation peut se traduire sous le schéma bloc suivant :

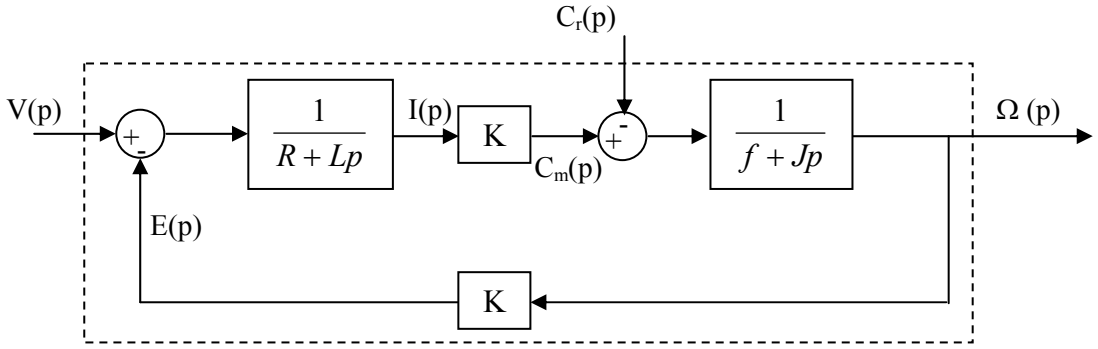


Figure B.3. Schéma bloc du modèle du moteur à courant continu (en boucle ouverte).

V(p) qui désigne la tension appliquée au moteur représente la consigne et Ω(p) qui désigne la vitesse du moteur représente la sortie du système.

Remarque
Dans notre cas d'étude, nous disposons de deux moteurs à courant continu. Pour faciliter les calculs, nous effectuons l'étude de modélisation sur le moteur associé à la roue droite. Le principe restera identique pour le moteur associé à la roue gauche.

III.1 Pour le moteur de la roue droite
Nous cherchons l'allure de la tension *V(p)* qui doit être appliquée au moteur pour obtenir la vitesses ω_d.

Pour cela, nous effectuons un asservissement de vitesse de tel sorte que l'erreur $\varepsilon(p) = \Omega(p) - \omega_d(p)$ tend vers zéro.

$\Omega(p)$ la vitesse désirée du moteur.

$\omega_d(p)$ la vitesse mesurée du moteur.

Le nouveau schéma bloc du moteur en boucle fermée est donné comme suit :

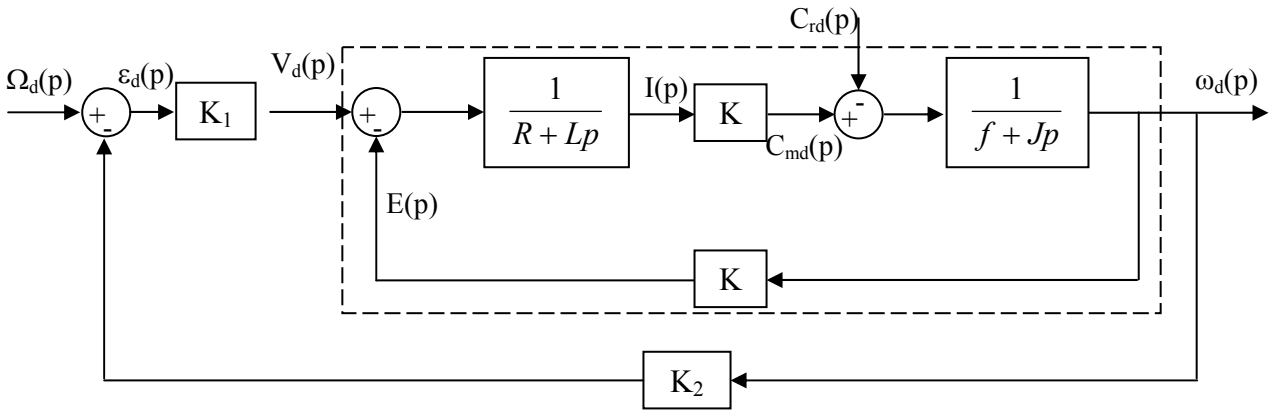


Figure B.4. Schéma bloc du moteur à courant continu en boucle fermée.

A partir de la figure B.4 nous pouvons déduire la tension $V_d(p)$ appliquée au moteur afin d'atteindre la vitesse désirée $\Omega_d(p)$.

IV. Simulation et résultats

Pour commander les deux roues du robot Pekee, nous prenons deux moteurs à courant continu dont les caractéristiques sont les suivantes (identiques pour les deux moteurs):

R = 1 Ω	L = 1 H	K = 10 N.m/A
f = 0.5 N.m/rad/s	J = 2 kg.m^2	k = 0.1 N.m/A
K1 = 10 N.m/A	K2 = 1	

Pour déterminer les vitesses résultantes associées à chaque moteur, nous avons effectué un traitement mathématique donnant l'expression des vitesses angulaires (Ω_d, Ω_g) en fonction de *v* et *w* (voir équations B.27).

$$\omega_d = \frac{v + d.w}{r} \text{ et } \omega_g = \frac{v - d.w}{r} \qquad \text{(B.27)}$$

Nous cherchons l'allure des tensions $v_d(t)$, $v_g(t)$ pour obtenir les vitesses ω_d, ω_g correspondantes. Dans ce cas, nous effectuons un asservissement de vitesse donné par la figure 4.

IV.1 Etape préliminaire

Nous utilisons les résultats obtenus dans le chapitre IV concernant les commandes appliquées au robot Pekee. Nous choisissons les résultats issus de la section III.1.4 où *v* et *w* désignent les commandes stabilisantes qui permettent au robot de suivre une trajectoire donnée pour atteindre la position ($x_0 = 0$, $y_0 = 0$, $\theta_0 = 0$) à partir d'une position initiale ($x_0 = -2$, $y_0 = -2$, $\theta_0 = \pi/4$).

Avant de simuler le comportement des moteurs, nous allons d'abord définir l'allure des vitesses Ω_d et Ω_g à asservir en fonction des allures de v et de w données par la figure B.5 (voir section III.1.4).

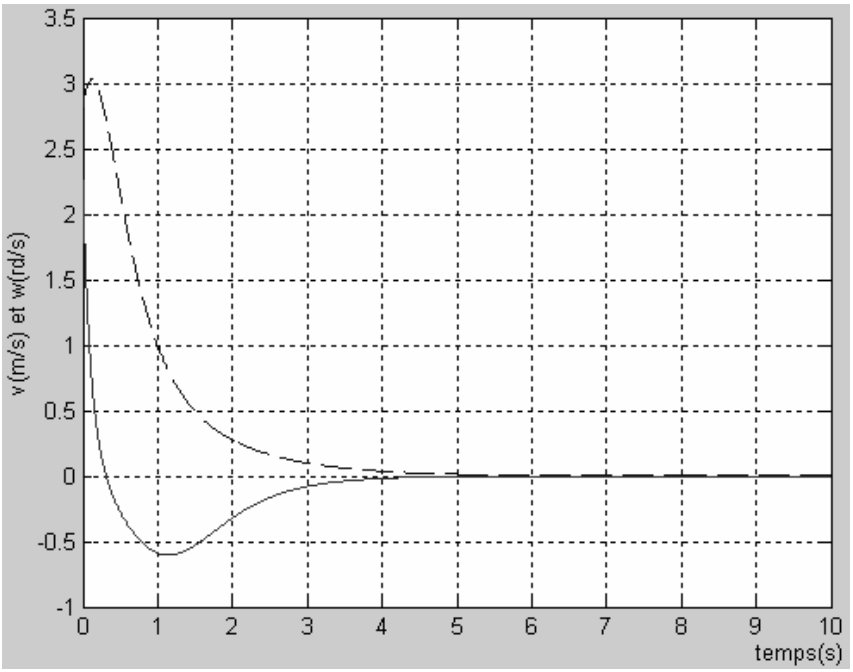


Figure B.5. Vitesses de translation v(---) et de rotation w(—) appliquées au robot

A partir de l'équation B.27, on peut obtenir graphiquement les vitesses angulaires des deux moteurs, à savoir $\Omega_d(t)$ et $\Omega_g(t)$. La figure B.6 montre l'allure de ces vitesses.

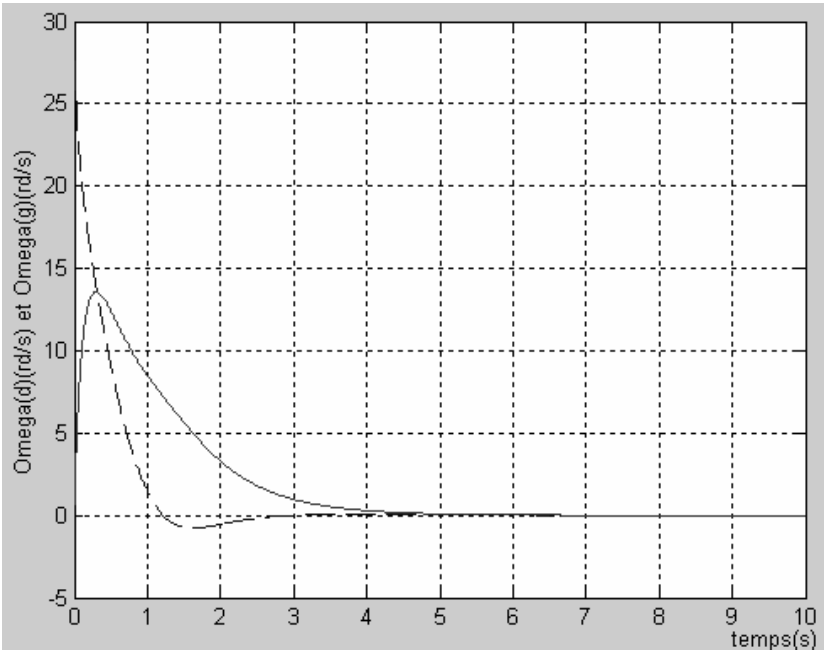


Figure B.6. Les deux vitesses angulaires désirées des moteurs associés à la roue gauche Ω_d (---) et à la roue droite Ω_g(—).

Dans ce cas, nous cherchons la tension *v(t)* qu'on doit appliquer à chaque moteur afin d'atteindre les vitesses données par la figure 6.

La figure ci-dessous montre le schéma de simulation en Matlab-Simulink d'un asservissement de vitesse du moteur de la roue droite. La vitesse désirée $\Omega_d(t)$ du moteur est celle donnée par la figure B.6.

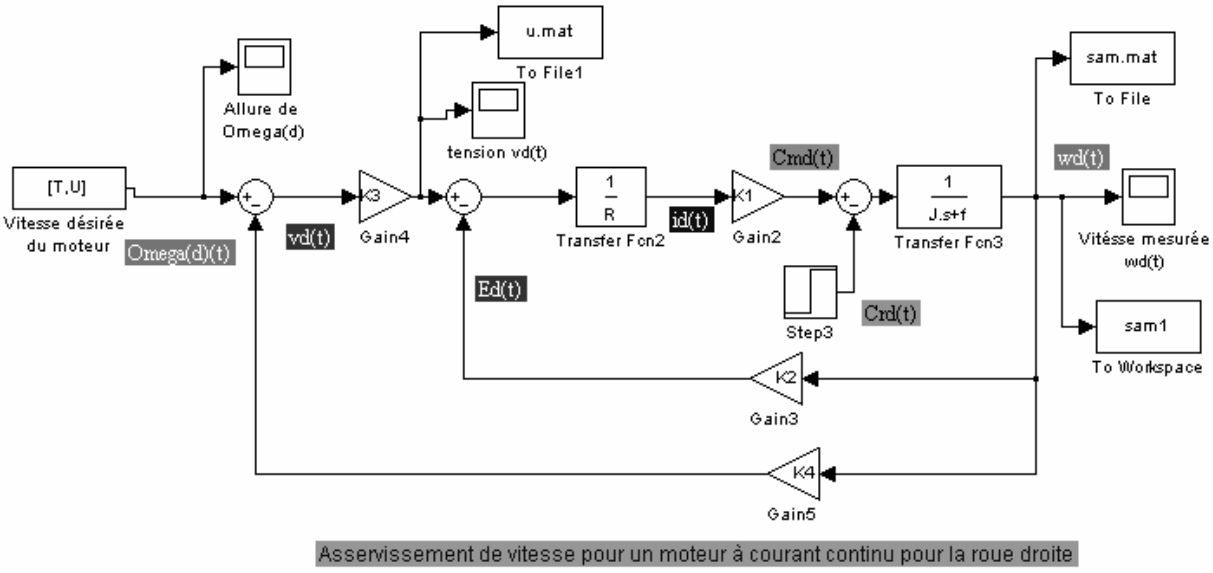


Figure B.7. Schéma de simulation sous Matlab-Simulink d'un asservissement de vitesse du moteur de la roue droite.

Après simulation, nous avons obtenu les courbes $w_d(t)$ et $v_d(t)$ désignant respectivement la vitesse mesurée du moteur et la tension appliquée correspondante et ceci en fonction du gain K_3 qui représente la correction de notre système.

Pour *K3 = 1*, la tension du moteur $v_d(t)$, la vitesse mesurée et la consigne du moteur sont données comme suit :

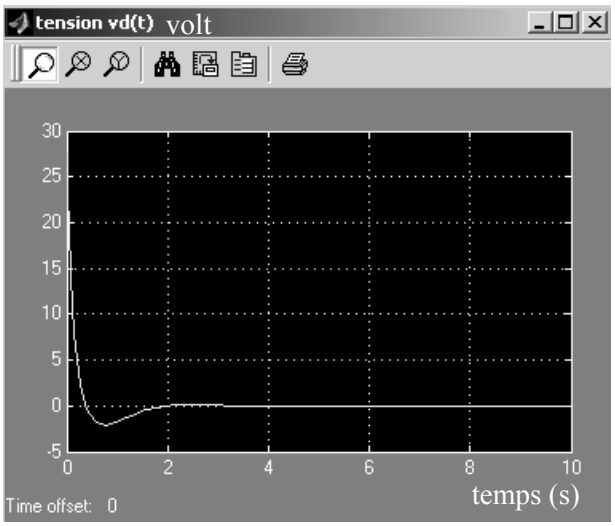


Figure B.8. Courbe de la tension $v_d(t)$ appliquée au moteur

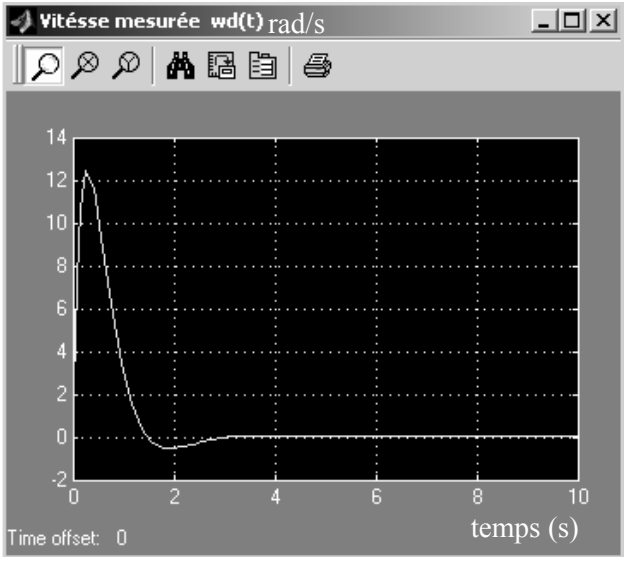


Figure B.9. Allure de la vitesse mesurée du moteur

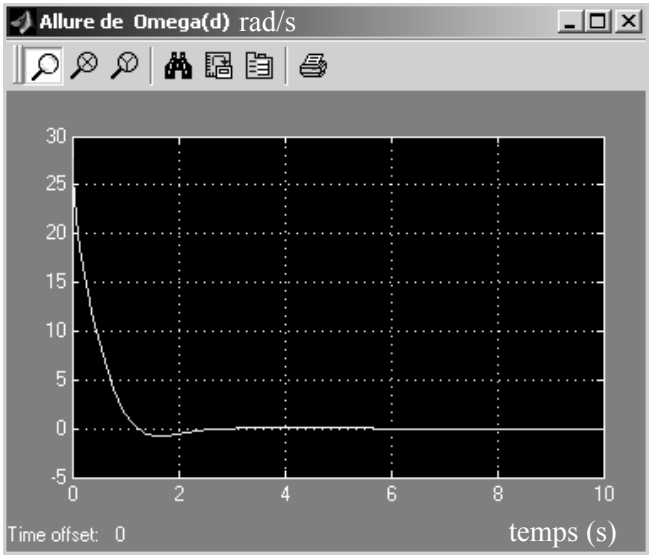


Figure B.10. Consigne du moteur $\Omega_d(t)$ ou vitesse désirée

Idem pour *K3 = 3*

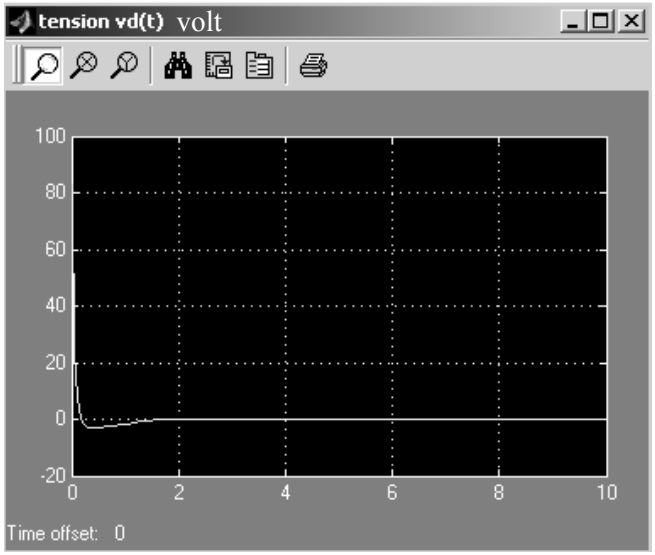


Figure B.11. Courbe de la tension $v_d(t)$ appliquée au moteur

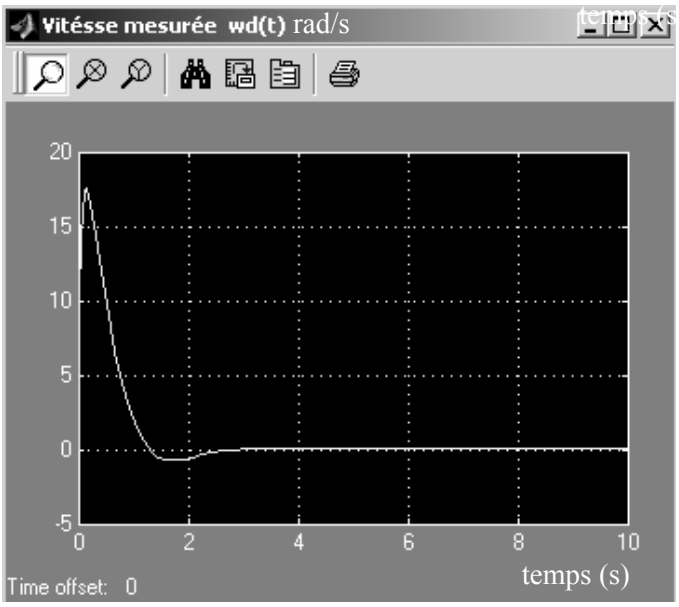


Figure B.12. Allure de la vitesse mesurée du moteur

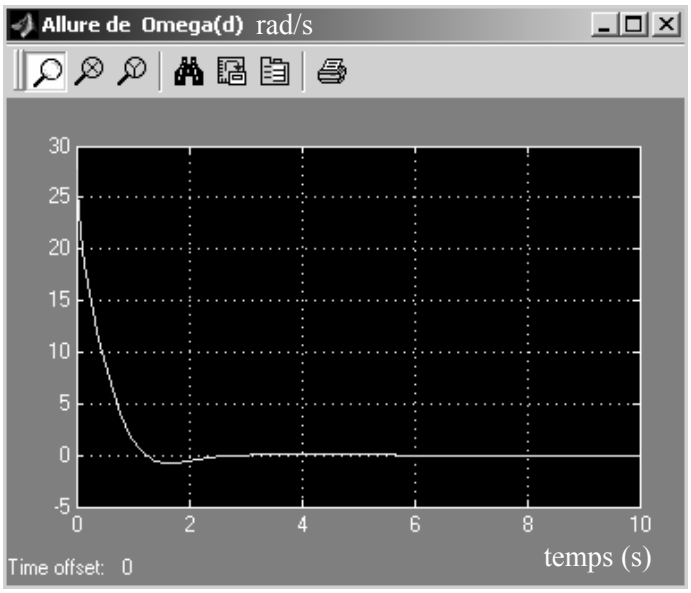


Figure B.13. Consigne du moteur $\Omega_d(t)$ ou vitesse désirée

De la même manière que nous réalisons un asservissement de vitesse du moteur associé à la roue gauche.

La figure ci-dessous montre le schéma de simulation en Matlab-Simulink d'un asservissement de vitesse du moteur de la roue gauche. La vitesse angulaire désirée $\Omega_g(t)$ du moteur dans ce cas est celle donnée par la figure B.6.

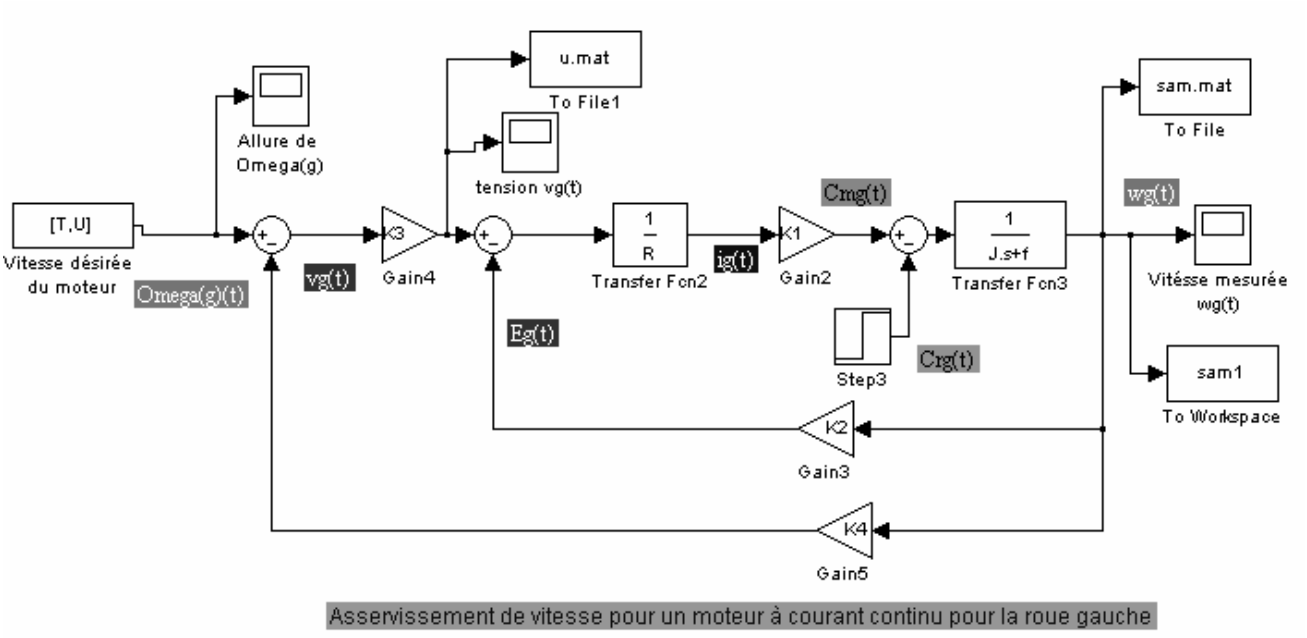


Figure B.14. Schéma de simulation sous Matlab-Simulink d'un asservissement de vitesse du moteur de la roue gauche du robot

Après simulation nous avons obtenu les courbes $w_g(t)$ et $v_g(t)$ désignant respectivement la vitesse mesurée du moteur et la tension appliquée correspondante et ceci en fonction du gain K_3 qui représente la correction de notre système.

Pour *K3 = 1*, la tension du moteur $v_g(t)$, la vitesse mesurée et la consigne du moteur sont données comme suit :

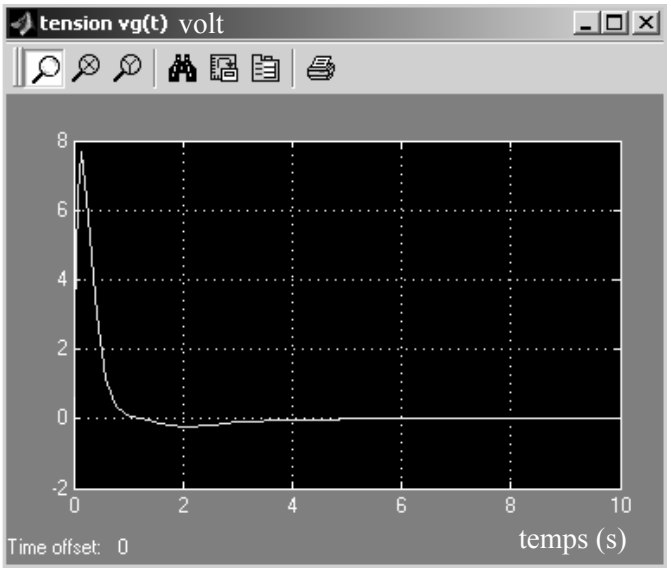


Figure B.15. Courbe de la tension $v_g(t)$ appliquée au moteur

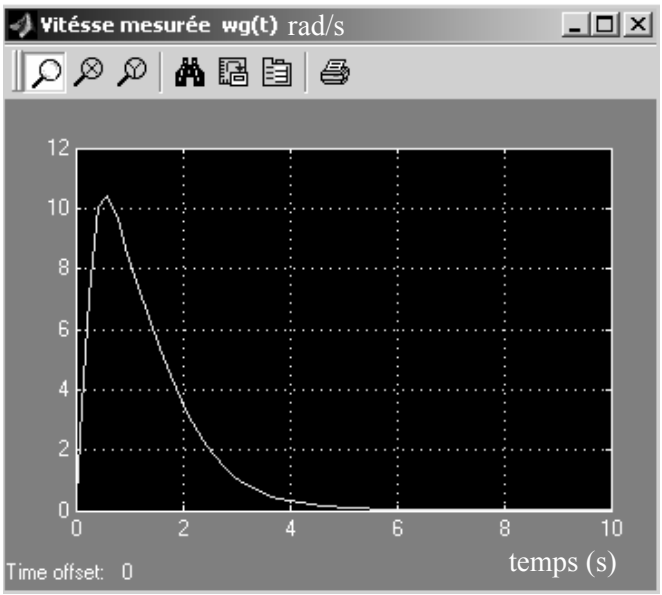


Figure B.16. Allure de la vitesse mesurée du moteur

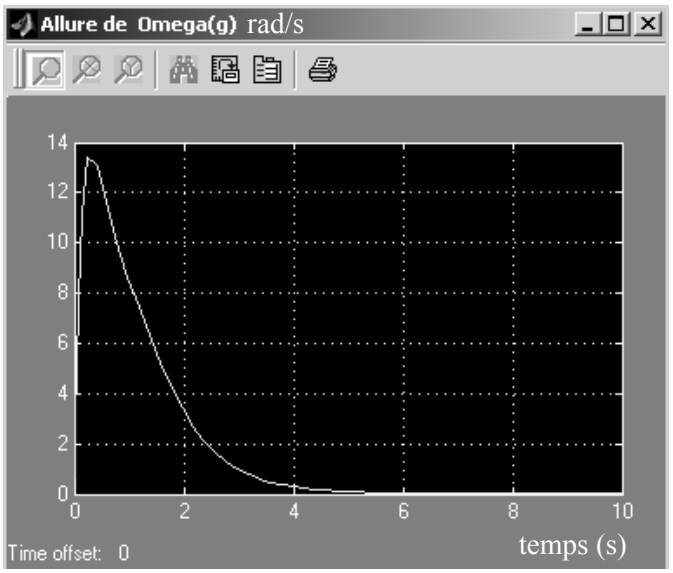


Figure B.17. Consigne du moteur $\Omega_g(t)$ ou vitesse désirée

Pour K3 = 5

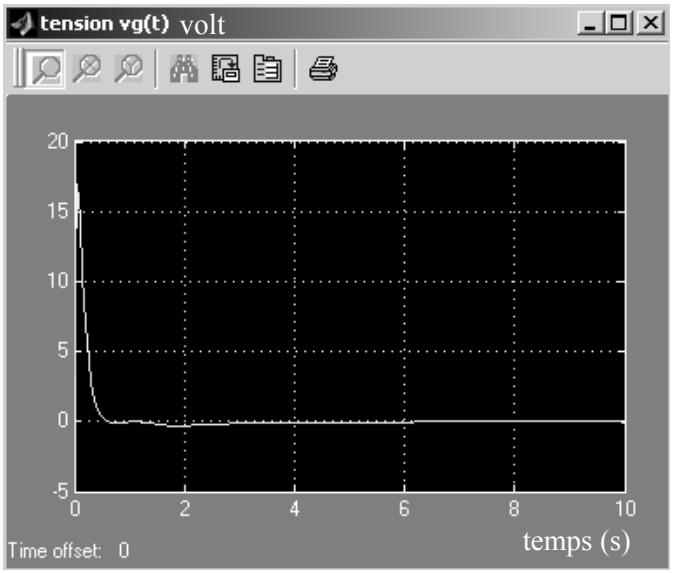


Figure B.18. Courbe de la tension $v_g(t)$ appliquée au moteur

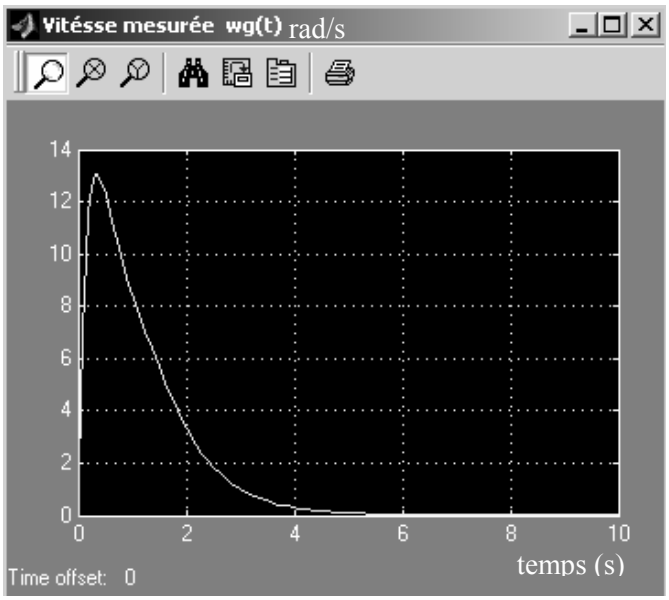


Figure B.19. Allure de la vitesse mesurée du moteur

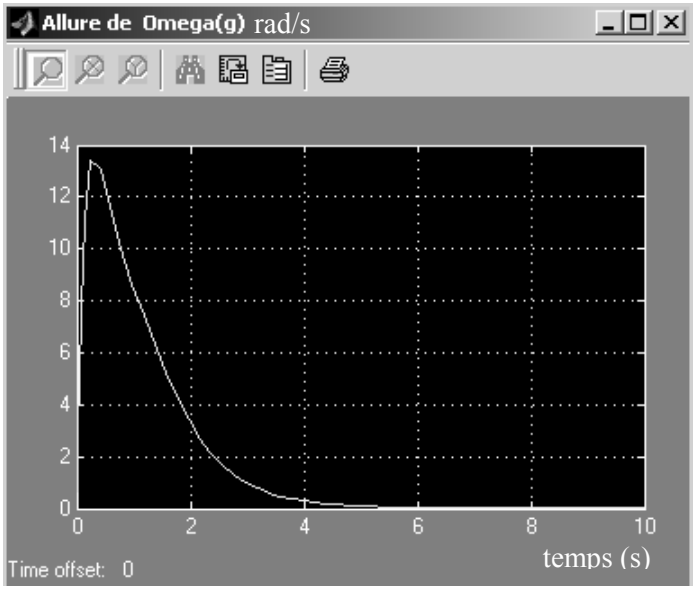


Figure B.20. Consigne du moteur $\Omega_g(t)$ ou vitesse désirée

Les résultats de simulation illustrés précédemment donnent l'allure des différentes tensions appliquées aux deux moteurs pour atteindre les vitesses désirées du robot en translation « *v* » et en rotation « *w* ».

Les performances de chaque moteur peuvent être améliorée en appliquant une correction de type « proportionnelle » au système en boucle fermée.

A titre d'exemple, pour un correcteur $K_3=5$, le moteur de la roue gauche a tendance d'atteindre la vitesse désirée plus rapidement que dans le cas précédent $K_3=1$.

La commande appliquée au moteur est plus conséquente au départ lorsqu'on augmente la valeur de K_3.

En simulation, pour une valeur de K_3 plus grande, l'erreur entre les vitesses mesurées et les vitesses désirées est très faible. Cependant, au niveau pratique cette hypothèse n'est pas tout à fait respectée. Le moteur dans ce cas ne peut pas dépasser une certaine vitesse limite même si la valeur de la tension appliquée est plus importante (dû à l'augmentation de K_3). Le choix de K_3 est alors limité pas les performances de notre moteur.

V. Conclusion

Nous avons établi la relation existante entre la tension appliquée à un moteur à courant continu et la vitesse de rotation résultante. Cette étude nous a été utile pour déterminer la tension nécessaire à chaque moteur (correspondant à chacune des deux roues : gauche et droite) permettant au robot Pekee d'effectuer une trajectoire stabilisante.

Annexe C

Annexe C Influence du bruit de modèle sur les performances du robot Pekee soumis aux commandes stabilisantes

Nous effectuons une étude concernant le comportement du robot Pekee soumis aux bruits paramétriques.
Notre but est d'étudier les performances de notre système lorsqu'il suit une trajectoire stabilisante.

Soit le système (C.1) décrit dans la section III.2.1 du chapitre IV que l'on suppose maintenant soumis à des perturbations p [HAM 07] :

$$\dot{x} = f(x) + g(x)u + p(x,t) \tag{C.1}$$

Ces perturbations peuvent représenter :

- des incertitudes paramétriques sur le terme nominal de la dérivée f : $p = \delta f(t,x)$
- des perturbations externes indépendantes de l'état p= b(t).

Ici, nous supposons que le système peut être perturbé par un bruit paramétrique ou bruit de modèle. Ainsi, nous allons tester les performances de notre robot en lui appliquant les lois de commande stabilisante.

D'une façon générale, si on suppose que $b(t,x)=\delta(t,x)$ est le bruit qui agit sur le système, l'équation (C.1) peut s'écrire alors :

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \cos\theta + \delta_1(t,x,y,\theta) & 0 \\ \sin\theta + \delta_2(t,x,y,\theta) & 0 \\ 0 & 1+\delta_3(t,x,y,\theta) \end{pmatrix} \begin{pmatrix} v \\ w \end{pmatrix} \tag{C.2}$$

Le système C.2 peut s'écrire ainsi :

$$\begin{cases} \dot{x} = \cos\theta.v + \delta_1(t,x,y,\theta) \\ \dot{y} = \sin\theta.v + \delta_2(t,x,y,\theta) \\ \dot{\theta} = w + \delta_3(t,x,y,\theta) \end{cases} \tag{C.3}$$

Pour l'application numérique nous prenons :

$\delta_1(t,x,y,\theta) = 0.0029*\sin(10*x)$

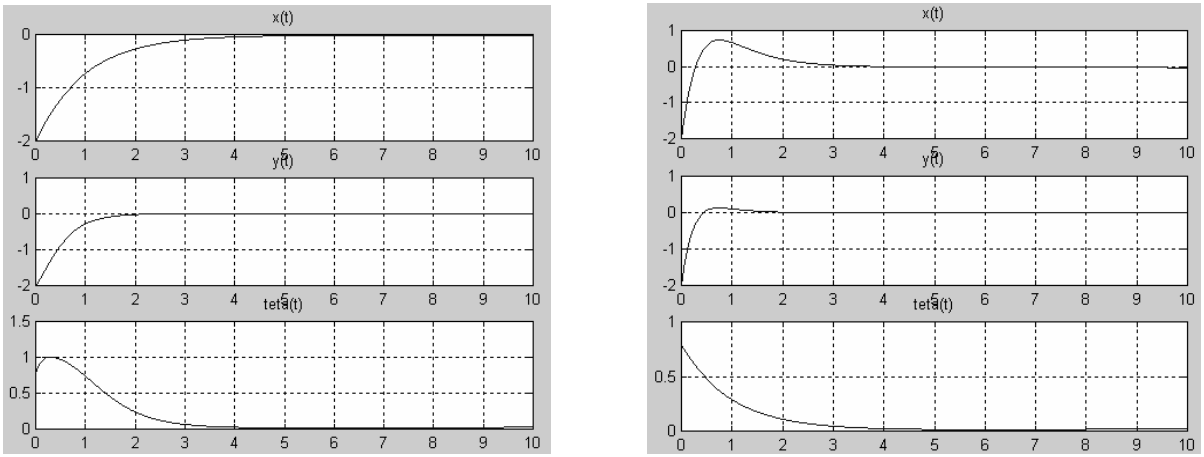
$\delta_2(t,x,y,\theta) = 0.0013*\cos(5*y)$

$\delta_3(t,x,y,\theta) = 0.0028*\sin(\theta)$

Application de la commande stabilisante

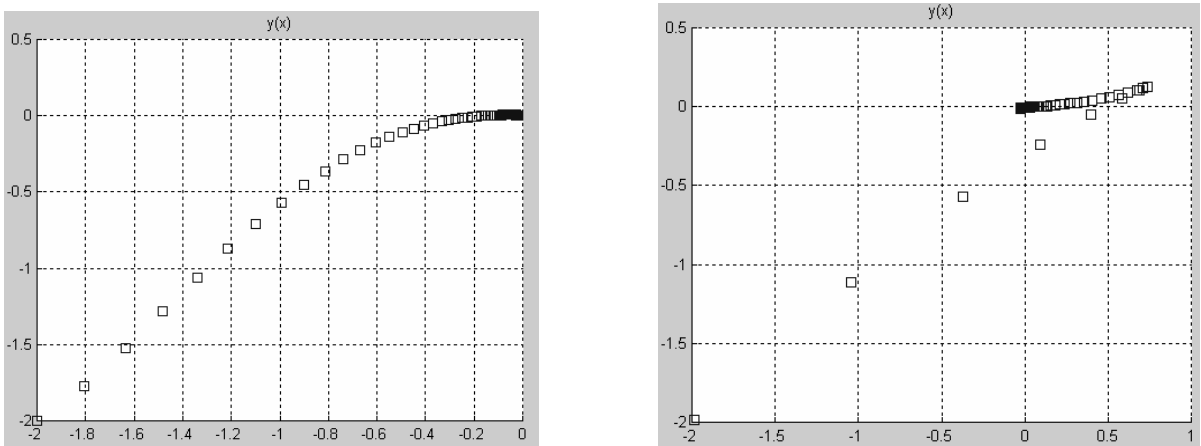
Nous appliquons au modèle (C.3) les deux lois de commande stabilisante en prenant en considération les conditions initiales suivantes : x_0= -2, y_0 = -2, $\theta_0 = \pi/4$.

Les figures C.1 et C.2 montrent l'évolution de la trajectoire du robot soumis à un bruit de modèle.



(a) commande en forme chaînée (b) commande en « feedback linearisation »

Figure C.1. Evolution des états du robot soumis à un bruit de modèle



(a) commande en forme chaînée (b) commande en « feedback linearisation »

Figure C.2. Trajectoires du robot correspondantes

Pour les deux cas de contrôle, nous remarquons que l'évolution de la trajectoire du robot est analogue en présence ou en absence du bruit paramétrique. Ainsi, les positions x, y et l'orientation θ sont insensibles aux perturbations lorsqu'on veut stabiliser le robot.

Annexe D

Annexe D Technique de backstepping appliquée à un robot mobile unicycle

I. Technique de backstepping

La technique de backstepping repose sur la construction de la fonction de Lyapunov pour stabiliser un système non-holonome écrit sous la forme suivante :

$$\begin{cases} \dot{\eta} = f(\eta) + g(\eta)\,\varepsilon \\ \dot{\varepsilon} = u \end{cases} \tag{D.1}$$

Où $[\varepsilon\ \eta]^T \in R^{n+1}$ et u l'entrée de la commande.

La loi de commande u qui stabilise ce système est donnée par l'expression (D.2) [TAD 03], [MAD 08]:

$$u = \frac{\partial \phi}{\partial \eta}\left[f(\eta) + g(\eta)\,\varepsilon\right] - \frac{\partial V}{\partial \eta}\,g(\eta) - K\left[\varepsilon - \phi(\eta)\right] \tag{D.2}$$

Cette commande u assure la convergence asymptotique du système D.1 vers l'origine.

Nous appliquons cette technique sur le robot Pekee afin d'observer les performances de la commande proposée.

II. Application de la technique « backstepping » au robot mobile Pekee

Nous appliquons dans ce cas la théorie de Lyapunov pour déterminer les lois de commande qui assure la convergence des variables d'état vers l'origine.

Prenons le robot Pekee dont le modèle cinématique est donné comme suit :

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \cos\theta \\ \sin\theta \\ 0 \end{pmatrix} v + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} w \tag{D.3}$$

Nous proposons de mettre le système sous la forme chaînée en effectuant les changements de variable suivant :

$$\begin{cases} x_1 = x \\ x_2 = \tan\theta \\ x_3 = y \end{cases} \tag{D.4}$$

A partir de l'équation (D.4) nous dérivons les trois états x_1, x_2, x_3 de la façon suivante :

$$\begin{cases} \dot{x}_1 = v.\cos\theta \\ \dot{x}_2 = \dfrac{\dot{\theta}}{(\cos\theta)^2} \\ \dot{x}_3 = v.\sin\theta \end{cases} \tag{D.5}$$

En faisant la transformation :

$$\begin{cases} v = \dfrac{u_1}{\cos\theta} \\ w = u_2 \cos^2\theta \end{cases} \tag{D.6}$$

Notre système s'écrit :

$$\begin{cases} \dot{x}_1 = u_1 \\ \dot{x}_2 = u_2 \\ \dot{x}_3 = x_2 u_1 \end{cases} \tag{D.7}$$

II.1 Synthèse de la loi de commande réalisant la stabilisation

Apres avoir mis le système sous la forme (D.7), on lui applique l'approche de backstepping qui nous assure la convergence des états du robot vers l'origine.

On constitue d'abord une fonction de Lyapunov $v(x_1)$. Prenant :

$$v(x_1) = \frac{1}{2} x_1 \tag{D.8}$$

La dérivée de cette fonction s'écrit :

$$\dot{v} = \dot{x}_1 . x_1 = x_1 . u_1 \tag{D.9}$$

En prenant $u_1 = -k.x_1$ l'expression (D.9) s'écrit dans ce cas :

$$\dot{v} = -k.x_1^{\,2} \tag{D.10}$$

La commande u_1 assure la convergence asymptotique de x_1 vers l'origine.

Il reste à chercher l'expression de la commande u_2. Pour cela on procède de la manière suivante :

On s'intéresse a stabiliser le sous système suivant :

$$\begin{cases} \dot{x}_2 = u_2 \\ \dot{x}_3 = -k.x_1.x_2 \end{cases} \tag{D.11}$$

Le système (D.11) peut s'écrire sous la forme (D.1) avec :

$\eta = x_3,\ \varepsilon = x_2,\ f(\eta) = 0,\ g(\eta) = -k.x_1.$

Dans ce cas, on propose la fonction de Lyapunov $v(x_3)$ suivante :

$$v(x_3) = \frac{1}{2} x_3^{\,2} \tag{D.12}$$

$$\dot{v}(x_3) = \dot{x}_3 . x_3 = -k . x_1 . x_2 . x_3$$

Prenant :

$$\phi(x_3) = x_2 = \frac{k_1}{k} . \frac{x_3}{x_1} \tag{D.13}$$

Le système (D.1) deviens alors

$$\begin{cases} \dot{\eta} = 0 - k . x_1 . x_2 \\ \dot{\varepsilon} = u_2 \end{cases} \tag{D.14}$$

La loi de commande u_2 permettant de stabiliser ce système est :

$$u_2 = \frac{\partial \phi(x_3)}{\partial x_3} \left[f(x_3) + g(x_3)\, x_2 \right] - \frac{\partial v(x_3)}{\partial x_3} g(x_3) - K\left[x_2 - \phi(x_3) \right] \tag{D.15}$$

La commande u_2 s'écrit :

$$u_2 = \frac{k_1}{k . x_1} . (0 - k . x_1 . x_2) + k . x_1 . x_3 - k_2 \left(x_2 - \frac{k_1}{k} . \frac{x_3}{x_1} \right)$$

D'où les lois de commande permettant de stabiliser le robot sont données par l'expression :

$$\begin{cases} u_1(x) = -k_1 . x_1 \\ u_2(x) = -(k_1 + k_2) . x_2 + \left(k_3 . x_1 + \dfrac{k_1 . k_2}{k_3 . x_1} \right) x_3 \end{cases} \tag{D.16}$$

k_1, k_2 et k_3 sont des constantes strictement positives.

III. Simulations et résultats

Le robot mobile considéré est un robot de type unicycle avec deux roues avant motrices et une roue arrière stabilisante (figure D.1).

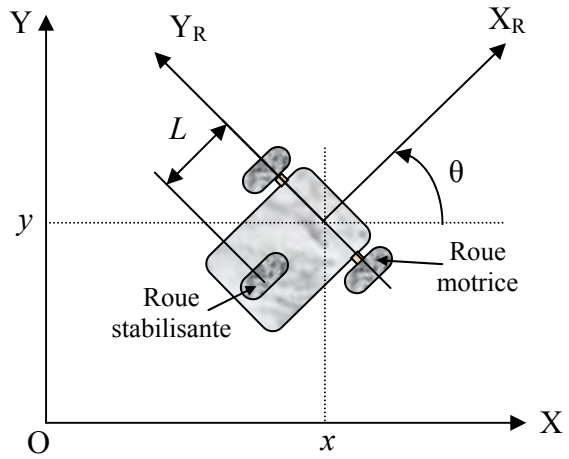


Figure D.1. Schéma de principe d'un robot mobile de type unicycle.

A partir de l'étude réalisée précédemment, nous illustrons le comportement du robot (observer les états x, y, θ) en appliquant la commande stabilisante ($u_1(x)$, $u_2(x)$) donnée par l'expression D.16. Nous testons les performances du contrôleur à différentes conditions initiales
(x_0, y_0, θ_0).

Pour appliquer le contrôleur ($u_1(x)$, $u_2(x)$), nous fixons d'abord la valeur des différents termes de chaque commande.
Les valeurs des k_i ($1 \leq i \leq 3$) sont données comme suit : $k_1 = 5$, $k_2 = 10$, $k_3 = 2{,}5$. Les k_i sont des constantes strictement positives.

a) Prenons comme condition initiale ($x_0 = -2$, $y_0 = -2$, $\theta_0 = \pi/4$). Les allures x, y, θ sont données par la figure D.2.

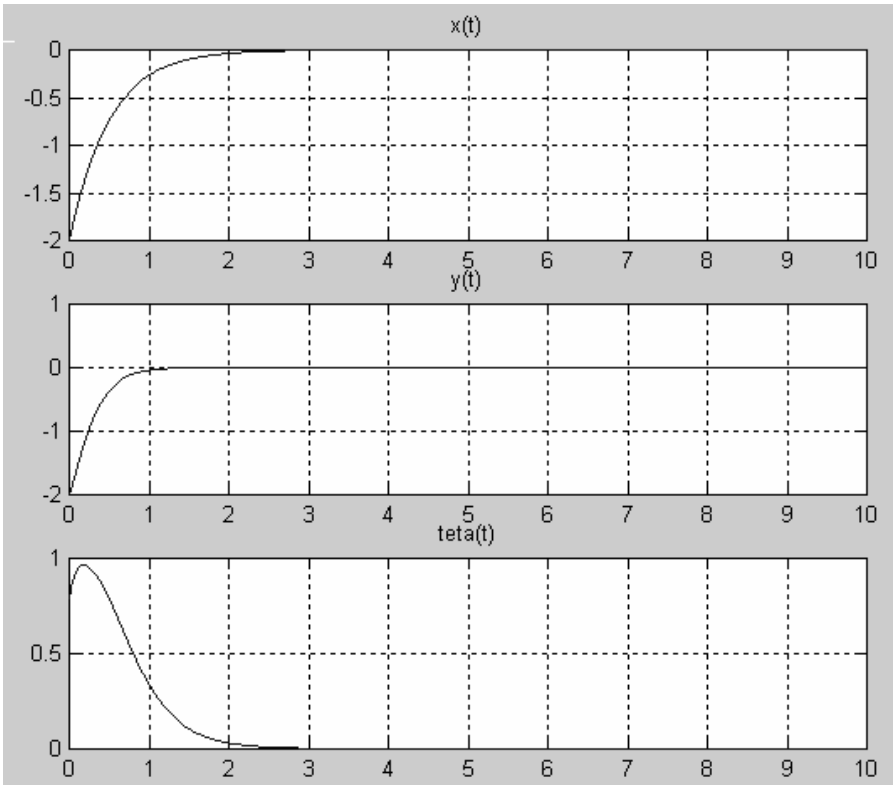


Figure D.2. Evolution dans le temps des variable d'états (x, y, θ) du robot.

Nous remarquons que toutes les variables d'états se stabilisent autour de l'origine au bout de cinq (5) secondes.
La trajectoire du robot correspondante pour atteindre la position d'équilibre ($x_{final} = 0$, $y_{final} = 0$, $\theta_{final} = 0$) depuis la position initiale ($x_0 = -2$, $y_0 = -2$, $\theta_0 = \pi/4$) est donné par la figure ci-dessous.

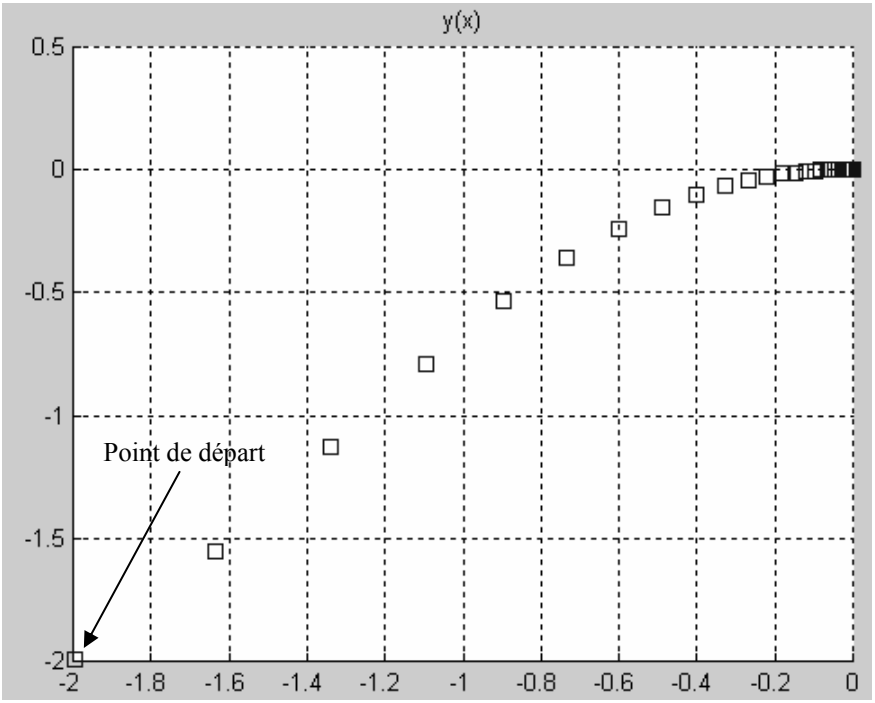


Figure D.3. Trajectoire du robot

En outre, nous avons étudié le comportement des commandes stabilisantes (en translation et en rotation) tout au long de la trajectoire. Les résultats obtenus sont illustrés dans la figure D.4.

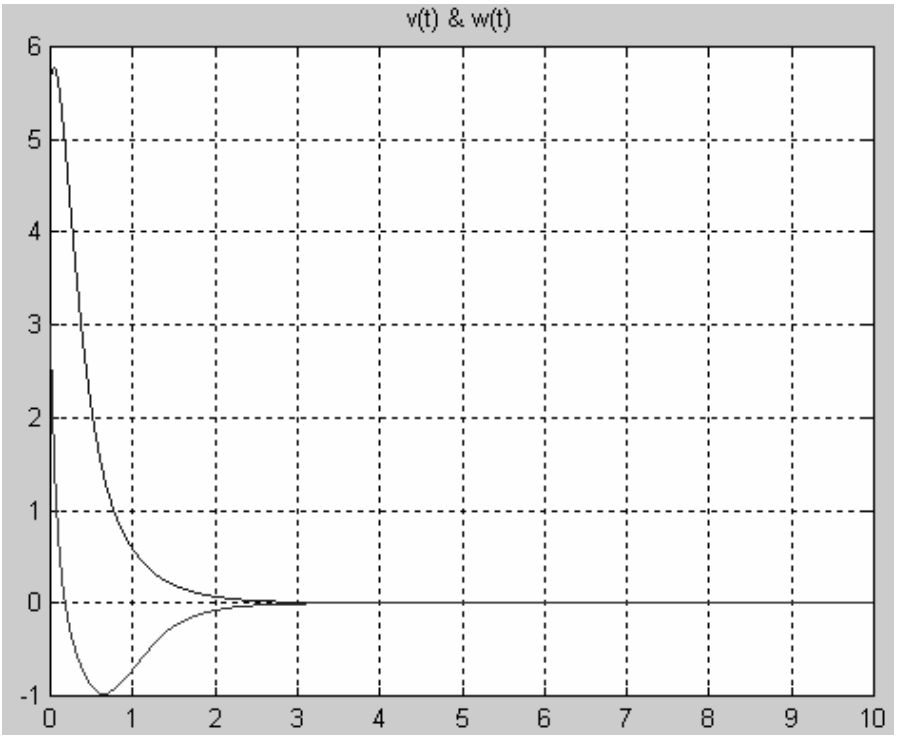


Figure D.4. Evolution dans le temps des commandes en translation *v* (—) et en rotation *w* (—)

b) Pour les conditions initiales (x_0= 4 y_0 = -5, θ_0 = π/3), les allures x, y, θ sont données par la figure D.5.

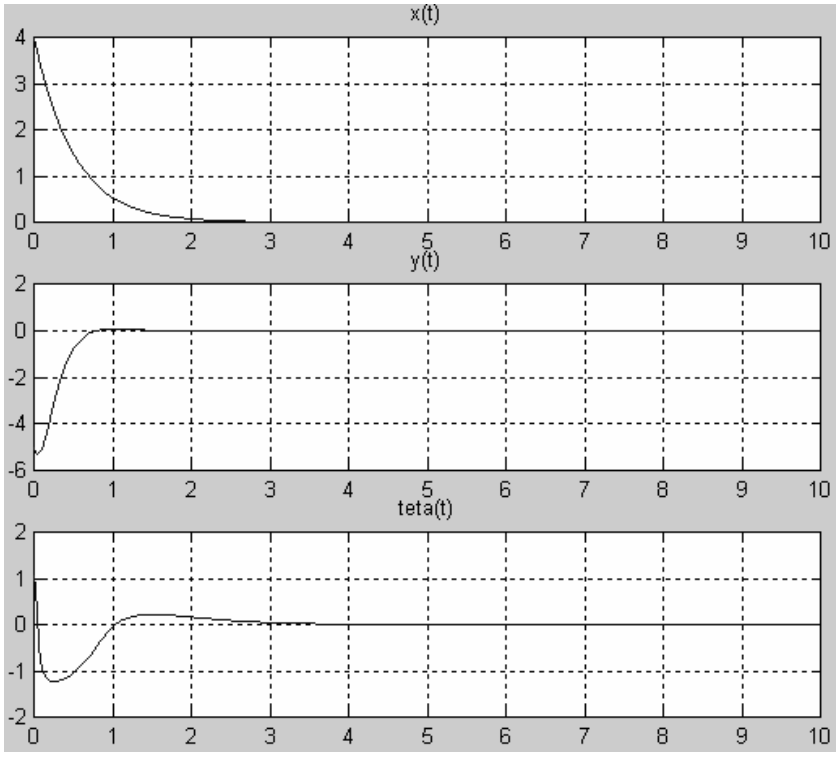


Figure D.5. Evolution dans le temps des variables d'états (x, y, θ) du robot.

La trajectoire du robot correspondante pour atteindre la position d'équilibre depuis la position initiale (x_0= 4, y_0 = -5, θ_0 = π/3) est donné par la figure ci-dessous.

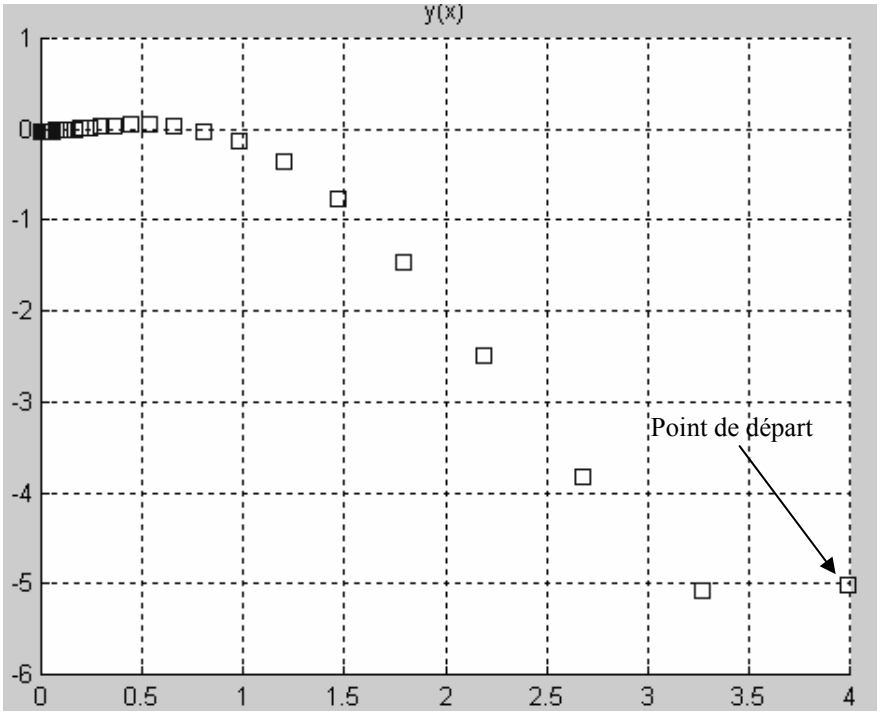


Figure D.6. Trajectoire du robot.

Le comportement des commandes stabilisantes (v et w) correspondantes est illustré par la figure D.7.

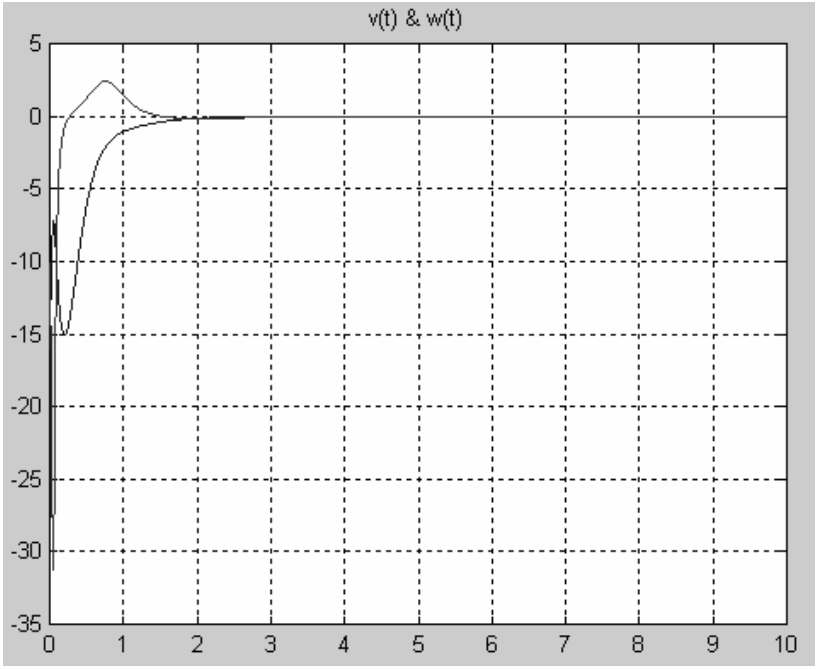


Figure D.7. Evolution dans le temps des commandes en translation v (—) et en rotation w (—).

La figure précédente montre que le robot effectue deux virages, ceci est dû à la l'amplitude des commandes v et w qui pousse le robot à s'éloigner de la cible avant qu'elle essaye de la rejoindre par la suite.

IV. Conclusion

Des tests de simulations ont été effectués pour observer le comportement du robot pour différentes conditions initiales en utilisant la technique de commande par backstepping. La convergence des états du système (D.3) vers l'origine à partir des états (x_0, y_0, θ_0) est présentée respectivement dans les figures 2, 5. Les figures 3, 6 montrent la trajectoire du robot vers l'origine à partir des conditions initiales précédentes. Enfin, l'allure des commandes « v, w » durant la trajectoire du robot est illustré par les figures 4, 7.

Les résultats ont montré que le système converge vers la position d'origine et ceci quelque soit sa position initiale dans laquelle il se trouve.

Cependant, le robot nécessite une commande d'amplitude beaucoup plus importante pour amener le robot à la position désirée.