Table des matières

Liste des tableaux

Liste des figures

Liste d'abréviations et de symboles

Abréviations

RLS	Recursive Least Squares
UMBmark	University of Michigan Benchmark test
EKF	Extended Kalman filter
MCR	les Moindres Carrés Récursifs
LMS	Least Mean Squares
AGV	Automated Guided Vehicles
GPS	Global Positioning System
VR	Virtual Reality
TOF	Time Of Flight

Symboles

V_m	Vitesse linéaire mesurée du robot
V_t	Vitesse linéaire théorique du robot
$\dot{\theta}_m$	Vitesse angulaire mesurée du robot
$\dot{\theta}_t$	Vitesse angulaire théorique du robot
L	L'entre-axe du robot
V	Vitesse linéaire du robot

$\dot{\theta}$	Vitesse angulaire du robot
V_d	Vitesse linéaire de la roue droite
V_g	Vitesse linéaire de la roue gauche
r_d	Rayon de la roue droite
$\dot{\varphi}_d$	Vitesse de rotation de la roue droite
r_g	Rayon de la roue gauche
$\dot{\varphi}_g$	Vitesse de rotation de la roue gauche
$\{\alpha, \beta, \gamma, \psi\}$	Paramètres cinématiques du robot
t	Temps
r	Rayon des deux roues
(p, θ)	Pose du robot
p	Position du robot
(x, y)	Coordonnées du robot
θ	Orientation du robot
S_{pos}	Ensemble des poses de référence
n	Nombre de poses
Δt	Durée de temps
C_{vit}	Ensemble des commandes en vitesses linéaires et angulaires
w_1, w_2	Vecteurs paramètres

u_1, u_2	Vecteurs de régression
J_1, J_2	Fonctions de coût
$\widehat{w}_1, \widehat{w}_2$	Vecteurs paramètres estimés
$\{\hat{\alpha}, \hat{\beta}, \hat{\gamma}, \hat{\psi}\}$	Paramètres cinématiques estimés
G_1, G_2	Gains
ξ_1, ξ_2	Erreurs
P_1, P_2	Matrices de covariance
λ	Facteur d'oubli
δ	Paramètre de régularisation
I	Matrice Identité
k	Temps discret
N	Nombre de valeurs discrètes
E_m	Erreur moyenne
E_i	Erreur mesurée à la $i^{ème}$ point de navigation
E_f	Erreur mesurée au point de navigation final
l	Nombre des points de navigation
$\%C$	Taux de convergence en pourcentage vers une position ou une orientation finale

Chapitre 1 - Introduction

1.1 Contexte général de la robotique mobile

Les robots mobiles sont des systèmes qui agissent physiquement dans leurs environnements afin d'atteindre des objectifs qui leur sont assignés. Ils impliquent de nombreuses disciplines telles que la mécanique, l'électronique et l'informatique. Ces machines sont autonomes et capables de s'adapter à quelques variations de leurs conditions de fonctionnement. Elles possèdent des fonctions de perception, prise des décisions et d'actions. Cette intelligence artificielle permet au robot d'accomplir différentes tâches de plusieurs façons même en rencontrant certaines situations imprévues.

Ces véhicules autonomes deviennent une priorité pour le secteur industriel et la recherche, car ils sont capables de coopérer avec les êtres humains ou même les remplacer afin d'effectuer des tâches qui sont généralement dangereuses, pénibles et répétitives.

Il est très important de mentionner qu'un robot mobile a besoin d'un mécanisme de locomotion qui lui permet de se déplacer dans son espace de travail. On peut trouver des robots à pattes, les robots à chenilles et encore les robots actionnés par des roues qui constituent la partie majeure de ces systèmes.

1.1.1 Les robots à pattes

Les robots à pattes sont des robots mobiles qui utilisent des membres mécaniques pour se déplacer. Ils imitent souvent les animaux à pattes et les humains et peuvent avoir deux pattes ou plus.

Grâce à leurs anatomies à nombreux degrés de liberté, ces robots peuvent traverser des différents terrains. Ils sont aussi polyvalents, car ils sont destinés à réaliser des tâches variées dont l'accès au site est difficile. En revanche, ces avantages nécessitent une complexité de mouvement, un maintien difficile de l'équilibre et de la stabilité du véhicule et une consommation énergétique importante.

La figure 1.1 présente des différents robots mobiles à deux, quatre et six pattes développées par Boston Dynamics.

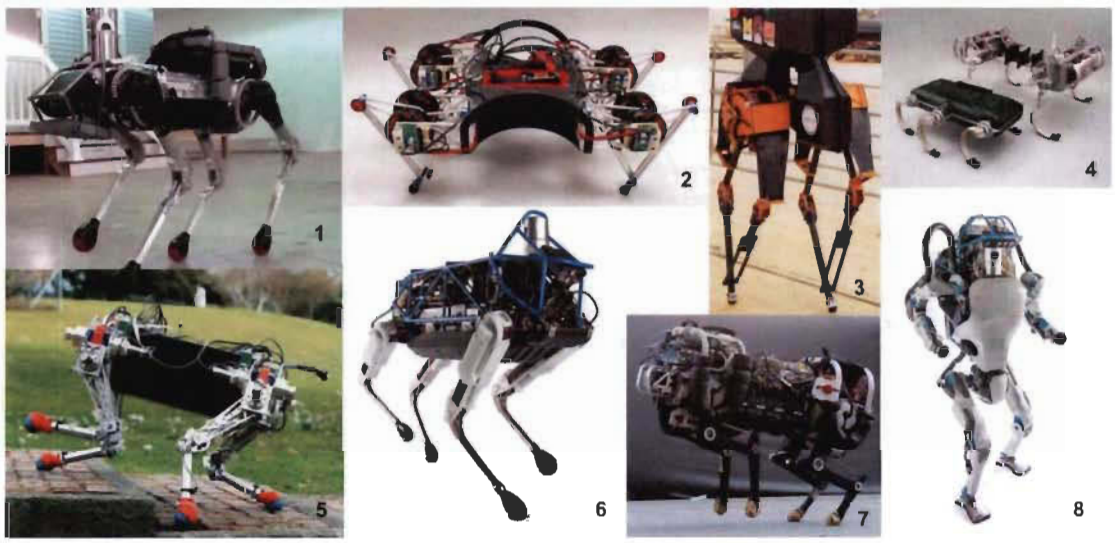


Figure 1.1 Exemples de robots à pattes [1]

1.1.2 Les robots à chenilles

Pour assurer leur locomotion, ces types de robots utilisent des chenilles qui sont des dispositifs mécaniques articulés capables de transmettre le poids du véhicule au sol en la répartissant sur une surface supérieure à la surface de contact des roues. La chenille est une bande de roulement tendue sur une série de roues alignées. Elles sont suspendues indépendamment, sans contact avec le sol et soutiennent le haut de la bande. La surface de contact des chenilles avec le sol est très supérieure à celle des roues ou des pattes ce qui permet une stabilité mécanique et une adhérence au sol remarquable. Ces avantages permettent de franchir des terrains accidentés et des obstacles du relief comme les chars d'assaut. En contrepartie, la grande adhérence et la stabilité entraînent des forces de frictions importantes ce qui causent une augmentation de la consommation énergétique. Dans le secteur industriel, l'utilisation des robots à chenilles est limitée et ne présente pas une solution pratique et rentable.

Figure 1.2 Exemples de robots à chenilles [2]

1.1.3 Les robots à roues

En raison de la simplicité du mécanisme de locomotion, les robots mobiles à roues sont les plus répandus actuellement. Ce choix technologique a été fait, car les roues sont toujours plus faciles à contrôler que des pattes ou des chenilles et elles permettent au robot de se déplacer plus rapidement en dissipant moins d'énergie. Cependant, ces robots ne sont pas pratiques sur des terrains accidentés et sur des surfaces qui ne sont pas dures. Ils opèrent dans des environnements aménagés soit à l'intérieur comme les sites industriels ou à l'extérieur comme les routes. Les robots mobiles à roues peuvent avoir deux roues ou plus et les types les plus utilisés sont les robots de type voiture et les robots à entraînement différentiel. La figure suivante présente quelques catégories des robots mobiles à roues.

Figure 1.3 Exemples de robots à roues [3]

1.1.3.1 Les robots de type voiture

Un robot de type voiture est doté de quatre roues disposées à chacun des coins du châssis. Deux roues fixes placées sur un même axe assurent la traction du robot, alors que les deux autres roues qui sont aussi sur le même axe sont orientables et servent à le diriger.

Grâce à sa structure et ses quatre appuis, le robot de type voiture est considéré stable dans la famille des robots à roues. La plupart du temps, ces robots sont utilisés à l'extérieur comme les voitures sans pilotes qui sont encore en phase de la recherche et le développement. Il faut noter qu'à cause de la nature de la locomotion, la commande d'un robot de type voiture est compliquée. En effet, il est impossible de le déplacer perpendiculairement aux roues fixes ou le tourner sur place, ce qui limite leur utilisation dans des espaces encombrés. Toutes ces limitations nous mènent à favoriser l'utilisation des robots mobiles à entraînement différentiel présentés dans la prochaine section.

Figure 1.4 Exemples de robots de type voiture [4]

1.1.3.2 Les robots mobiles à entraînement différentiel

Un robot à entraînement différentiel est un robot mobile dont le mouvement est basé sur deux roues motrices entraînées indépendamment. Ces deux roues ont généralement le même rayon et sont placées de chaque côté du corps du robot.

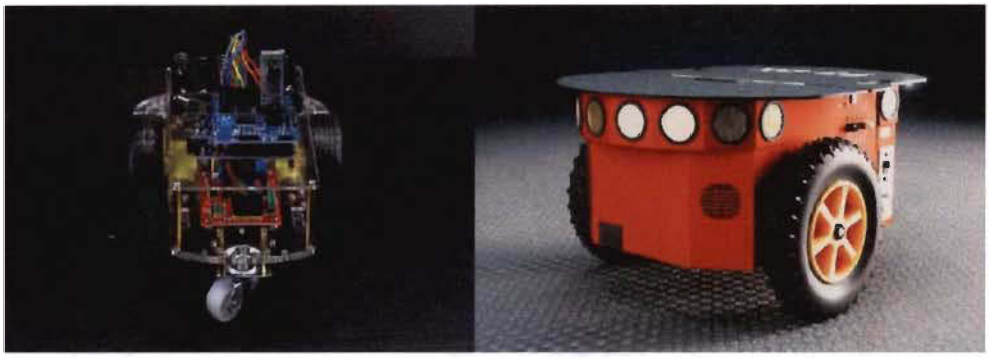


Figure 1.5 Exemples des robots mobiles à entraînement différentiel [5]

La locomotion de ce type de robot est simplifiée, car il est capable de tourner autour de son centre de rotation situé au milieu de l'axe reliant les deux roues, ce qui n'est pas le cas pour les robots de type voiture. De ce fait, le déplacement d'un robot mobile à entraînement différentiel peut être réalisé en exécutant seulement des mouvements de rotation autour de lui-même et des mouvements de translation ce qui rend sa commande plus facile que la majorité des autres robots mobiles.

Le terme « différentiel » signifie que la vitesse du véhicule, qui est en train d'exécuter un virage, est déterminée par la différence de vitesse entre ces deux roues. Par exemple, pour effectuer une translation, il faut actionner les deux roues à la même vitesse et au même sens. Alors que pour une rotation, il faut actionner les deux roues à la même vitesse, mais dans des sens opposés. La figure 1.3 suivante représente les différents mouvements que le robot peut réaliser en fonction des vitesses de ces deux roues.

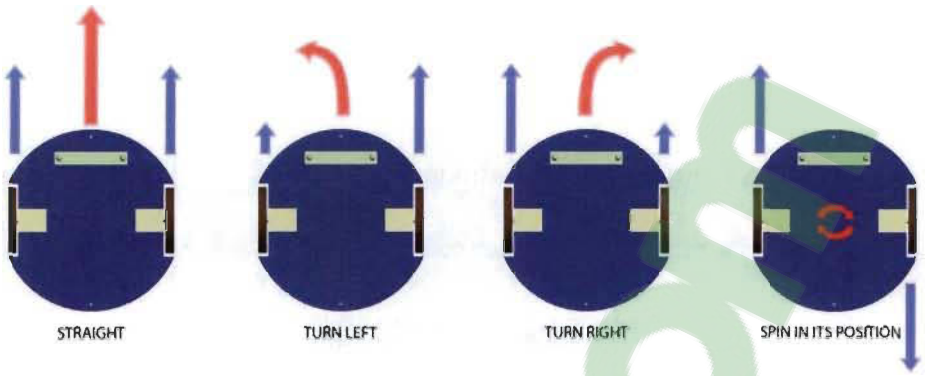


Figure 1.6 Les mouvements d'un robot à entraînement différentiel [6]

Contrairement à un robot de type voiture, ce robot est considéré moins stable puisqu'il contient deux appuis seulement. Pour cela, une ou plusieurs roues folles (pivotantes) peuvent être ajoutées afin d'assurer son équilibre et sa stabilité.

Figure 1.7 Roues folles ou pivotantes [7]

Cependant, cette configuration reste parmi les configurations les plus utilisées dans la robotique mobile. Ces robots peuvent servir à l'intérieur comme à l'extérieur et ils sont généralement exploités dans le secteur industriel comme les chariots et les plateformes de navigation autonome. Toutes ces qualités nous incitent à se focaliser sur ce type de robot mobile dans ce travail de recherche.

1.2 Problématique

Bien que les robots à entraînement différentiel possèdent une locomotion simple et une commande facile, l'optimisation et le contrôle de leurs mouvements présentent un défi et un point très important à développer par les roboticiens. Il est évident que l'aspect matériel qui consiste à concevoir la structure mécanique du système, bien choisir sa motorisation et son alimentation est le premier point à traiter, malgré cela, les problèmes spécifiques à la robotique mobile n'apparaissent finalement que lorsqu'on maîtrise le contrôle de mouvement de la plateforme qui revient à calculer et donner les bonnes commandes pour actionner les deux roues et par la suite optimiser le suivi des trajectoires planifiées.

Le robot différentiel nécessite une synchronisation parfaite des mouvements des deux roues indépendantes afin de produire un mouvement rectiligne droit du centre de gravité de la plateforme. La mise en œuvre de cette synchronicité dans la pratique est difficile à réaliser en raison d'une part à l'asymétrie des deux actionneurs (moteurs non parfaitement identiques, asservissements des roues difficiles à coordonner) et l'asymétrie de la résistance au roulement au niveau de chaque roue. En effet, sur une surface plane et rigide, les erreurs de mouvement du robot sont dues généralement à la nature du contact entre le sol et les roues. Dans des conditions parfaites, le contact roue-sol est ponctuel ce qui n'est pas le cas dans la vraie vie, ce qui peut engendrer de légers glissements du robot lors de son mouvement. Ces anomalies causent une déviation du robot de sa trajectoire à suivre et un positionnement erroné surtout quand il se déplace en ligne droite. La figure 1.8 ci-dessous illustre une déviation du robot au cours de son mouvement.

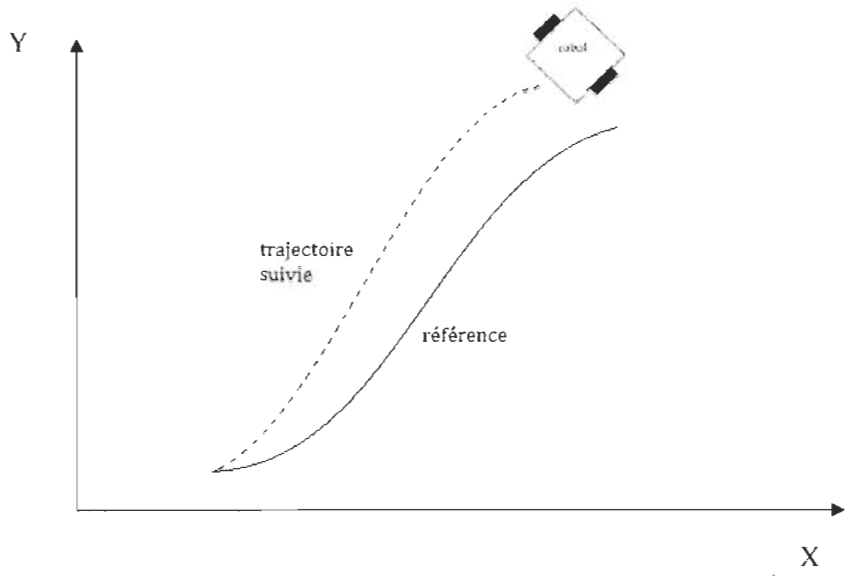


Figure 1.8 Erreur de mouvement d'un robot à entraînement différentiel

En plus de ça, les vitesses linéaire et angulaire mesurées du robot V_m et $\dot{\theta}_m$ ne correspondent pas aux mêmes vitesses théoriques V_t et $\dot{\theta}_t$ à cause des glissements ($V_m \neq V_t$, $\dot{\theta}_m \neq \dot{\theta}_t$). En d'autres termes, les équations cinématiques fondamentales d'un robot mobile à entraînement différentiel ne permettent pas de décrire son mouvement dans des conditions réelles et par la suite elles ne permettent pas de calculer les bonnes commandes et avoir un contrôle optimal du véhicule.

Plusieurs travaux de recherche ont été accomplis et plusieurs techniques ont été proposées pour remédier à cette problématique, cependant il reste encore des optimisations qui peuvent être réalisées afin de maîtriser le contrôle du mouvement de ce type de robot.

1.3 Objectifs

L'objectif de ce travail de recherche est de proposer une méthode adaptative de contrôle du mouvement d'un robot à entraînement différentiel qui permet d'ajuster son mouvement sur une surface rigide afin d'optimiser son suivi des trajectoires planifiées. Pour atteindre ce but, ce travail a été décomposé selon les sous-objectifs suivants :

1. La modélisation des équations cinématiques améliorées qui seront capables de décrire cinématiquement le robot qui subit des erreurs de mouvement.
2. L'élaboration d'une méthode adaptative de contrôle en se basant sur l'algorithme des moindres carrés récursifs.
3. Une validation expérimentale de la méthode proposée.

Ce travail de recherche a fait l'objet d'une publication scientifique et a permis d'ouvrir les portes pour des travaux futurs.

1.4 Méthodologie

Dans ce présent travail, une méthodologie qui consiste à développer les quatre étapes suivantes a été envisagée :

Étape 1 : État de l'Art

Une revue de la littérature est réalisée afin d'approfondir la compréhension du sujet de recherche et la comparaison des méthodes de contrôle du mouvement d'un robot mobile à entraînement différentiel est présentée dans cette partie.

Étape 2 : Modélisation des équations cinématiques améliorées

En partant des équations cinématiques fondamentales ou classiques, une modélisation des nouvelles équations cinématiques qui sont capables de décrire le mouvement du robot dans des conditions réelles est réalisée.

Étape 3 : Élaboration de la méthode de contrôle du mouvement

Cette partie présente les étapes de l'élaboration de la méthode de contrôle de mouvement du robot, commençant par la définition de la trajectoire de référence à suivre par le robot et l'identification des paramètres à estimer du système, et finissant par l'estimation de ces paramètres à l'aide de l'algorithme des moindres carrés récursifs RLS et le calcul des commandes données au robot.

Étape 4 : Validation expérimentale

Finalement, un banc d'essai a permis de mesurer la performance et de valider expérimentalement la méthode de contrôle proposée pour trois types de surfaces différents (tapis, ciment, contreplaqué).

1.5 Organisation du mémoire

La suite de ce mémoire est organisée comme suit :

- Le deuxième chapitre présente l'état de l'art qui cite et compare les différentes techniques du contrôle de mouvement.
- Le troisième chapitre présente les équations cinématiques classiques du système, une introduction de la méthode des moindres carrés récursifs, une modélisation des nouvelles équations cinématiques et l'élaboration de la méthode de contrôle.

- Au quatrième chapitre, une validation expérimentale de la méthode proposée et une discussion des résultats obtenus seront présentées.
- Finalement, le cinquième chapitre est dédié aux conclusions générales et les travaux futurs.

Chapitre 2 - État de l'art

Au cours des deux dernières décennies, de nombreuses études ont été réalisées afin de contrôler le mouvement d'un robot mobile à entraînement différentiel. Beaucoup de chercheurs ont travaillé pour une longue période sur cette problématique et différentes techniques de contrôle ont été proposées pour optimiser le suivi des trajectoires de ce type de robot mobile. Ce chapitre présente une revue de littérature sur les méthodes de contrôle de mouvement de ce type de robot.

2.1 Contrôle en position et en orientation

Un robot mobile, comme son nom l'indique, doit se déplacer d'un point initial à une destination finale, tout en satisfaisant les contraintes de vitesse et/ou de position sur son chemin. Le suivi des trajectoires ou encore le contrôle de la pose du robot a été décomposé en deux catégories qui sont traitées séparément ou ensemble. Il faut noter qu'une pose du robot est le couple de sa position et son orientation. Les deux catégories sont les suivants:

- Contrôle de la position du robot
- Contrôle de l'orientation du robot

Parmi les deux catégories mentionnées ci-dessus, le contrôle de la position du robot est considéré comme le problème le plus difficile. Il consiste à suivre des chemins prédéfinis pour atteindre une destination finale en naviguant d'une position du plan à une autre. Ce problème est proposé en définissant une discrétisation sur le chemin souhaité. Des vitesses sont implicitement données au robot à chaque point d'échantillonnage. L'une des solutions les plus courantes pour cette catégorie de problèmes est donnée par

Liapunov [8-10]. Dans cette méthode, deux contrôleurs linéaire et non linéaire sont proposés et l'optimisation est assurée par la fonction de Liapunov [11-13]. Dans cette approche, un modèle cinématique du robot est incorporé pour la conception de la méthode de contrôle. Les études [14-16] sont des exemples d'utilisation des contrôleurs pour les systèmes non holonomes (un système non holonome est une plate-forme mobile qui ne dispose que de deux degrés de liberté sur un plan puisque les translations latérales sont impossibles à réaliser).

Le contrôle de l'orientation du robot est une catégorie qui ne manque pas d'importance. L'objectif est de respecter la série des orientations données au robot au cours de son mouvement. Le contrôle de la vitesse de rotation du robot mobile est indispensable pour atteindre cet objectif. Différentes approches [17-19] ont été proposées comme la stabilisation de Liapunov qui est la méthode la plus ancienne pour résoudre ce problème. Cependant, des études récentes comme [20, 21] ont réussi à simplifier et résoudre ce problème en exploitant les données de mesure des orientations et en utilisant des contrôleurs linéaires. Cette approche simplifie la structure du contrôleur et permet également d'obtenir une meilleure performance.

2.2 Contrôle en estimant le coefficient de résistance au roulement

Il est aussi important de citer les méthodes d'estimation du coefficient de résistance au roulement qui peuvent contribuer énormément à optimiser le mouvement des robots et des plateformes mobiles en diminuant les glissements des roues.

Dans l'étude [22] Omar Trigui présente deux catégories d'estimation de ce coefficient : une méthode indirecte (ou hors ligne) et les méthodes d'estimation en temps réel (ou en ligne).

La méthode indirecte estime les paramètres physiques du véhicule comme les rayons des roues [23]. Ces paramètres et le coefficient de résistance au roulement affectent la dynamique et les performances du véhicule. Ils sont aussi dépendants de la pression des roues. Pour cette raison un système indirect de détection de la présence du gonflage a été utilisé pour réaliser l'estimation des trois dynamiques : verticale, longitudinale et en rotation, les vitesses de rotation des roues et les couples.

La deuxième catégorie consiste à estimer le coefficient de résistance au roulement en temps réel lors du déplacement du véhicule. Elle contient trois méthodes d'estimation : la méthode du filtre de KALMAN étendu (EKF), la méthode de glissement de la pente « Slip-slope method » et la méthode d'estimation individuelle de la roue.

La méthode du filtre de KALMAN étendu (EKF) consiste à estimer l'état du véhicule et l'identification du frottement. Cette méthode dépend de la dynamique longitude et latérale et elle a été appliquée sur un modèle dynamique à 8 degrés de liberté dans [24]. Dans [25], le filtre de KALMAN étendu a permis d'estimer le coefficient de résistance en se basant sur un modèle dynamique plus simplifié qui ne prend pas en compte les trois dynamiques. Cette méthode a un grand inconvénient, car elle dépend de la mesure de la force de la roue qui est généralement coûteuse.

La méthode de glissement de la pente « Slip-slope method » est utilisée dans plusieurs recherches. Dans [26-28], la méthode d'estimation a été appliquée respectivement dans le cas de traction, freinage et accélération et freinage à la fois. Cette méthode nécessite un GPS « Global Position System » pour mesurer la vitesse du véhicule et un système de freinage antiblocage (ABS). L'inconvénient de cette méthode c'est qu'elle dépend de la mesure de la force longitudinale qui n'est pas précise.

La méthode d'estimation individuelle de la roue consiste à estimer en temps réel le coefficient de résistance au roulement pour chaque roue. Dans l'étude [29], cette méthode est exploitée et a été capable d'estimer le taux de glissement et les forces longitudinales à l'aide de l'algorithme des moindres carrés récursifs. Les résultats n'étaient pas satisfaisants. Alors que dans l'étude [30], les auteurs ont proposé une meilleure approche en utilisant le même algorithme. Cette approche est plus robuste et plus rentable en termes de coût et de précision.

2.3 Contrôle en utilisant le modèle cinématique et le modèle dynamique

En raison de la nature non holonomique du système et de ses limites, le suivi des trajectoires a été reconnu comme le problème le plus difficile dans la communauté de la robotique mobile. Plusieurs études ont mis l'accent sur le modèle dynamique et ses effets sur le système dans son ensemble. Dans [31] un modèle détaillé d'un robot mobile, incluant la dynamique et les couples moteurs, a été proposé et la dynamique est contrôlée en utilisant un modèle de référence d'un contrôleur adaptatif au niveau du couple moteur. Bien que ce soit un effort authentique de tenir compte de la dynamique du robot, dans la plupart des systèmes la commande des couples moteurs n'est pas très pratique.

Dans [20, 21], le robot et la dynamique d'un actionneur simplifié ont été considérés et deux contrôleurs sont intégrés comme mentionné précédemment pour résoudre le problème de suivi des trajectoires. Dans cette méthode, les vitesses linéaire et angulaire du robot ne sont pas détectées ou contrôlées explicitement et on s'appuie uniquement sur la détection de position, qui est en général plus sujette à des erreurs que la détection des vitesses, ce qui rend le système plus susceptible aux erreurs.

Autre que [20, 21], dans lesquelles la dynamique du robot est incluse, mais non explicitement contrôlée, des études comme [19, 21, 32, 33] ont été traités le problème de suivi des trajectoires au niveau cinématique. Cela signifie que la dynamique du robot est négligée et que l'on suppose que les commandes de vitesse sont réalisées instantanément. Cette négligence est justifiée à condition que le moteur soit suffisamment puissant ou qu'il soit déjà contrôlé à l'aide des contrôleurs low level. Cela fait ressortir l'importance de l'approche d'utiliser un modèle cinématique. Les études [34, 35] consistent à négliger la dynamique du système en utilisant le retour d'information d'état basé sur la connaissance du modèle cinématique qui décrit le mouvement du robot. Cette méthode est plus simple à exploiter et elle est considérée comme une approche très pratique.

Dans [36] Johann Borenstein et Liqiang Feng ont créé une méthode de calibration pour les robots à entraînement différentiel nommé University of Michigan Benchmark calibration (UMBmark). Cette méthode consiste à effectuer deux tests qui sont la conduite du robot le long d'un carré de 4 mètres de longueur dans le sens des aiguilles d'une montre et dans le sens contraire des aiguilles d'une montre. À la fin des deux tests, les erreurs de positionnement finales sont calculées. Par la suite, des erreurs de mouvement permettant la calibration des paramètres géométrique du robot (diamètre des roues et l'entraxe) seront calculées suite à des équations présentées dans [36]. Cette calibration permettra l'optimisation du modèle cinématique du robot et par conséquent son suivi des trajectoires. Cette méthode est considérée comme une contribution scientifique importante dans le contrôle de mouvement. Elle est aussi unique et innovante pour l'étalonnage des robots mobiles. Un étalonnage tel que proposé dans leur travail augmentera la précision odométrique du robot et réduira les coûts de fonctionnement puisque le robot nécessitera

moins de mises à jour pour les positionnements absolus. En revanche, l'étalonnage UMBmark est systématique et fournit des résultats quasi optimaux puisqu'il est basé seulement sur la détection des erreurs systématiques. Les erreurs non systématiques appelées aussi stochastiques sont les erreurs de mouvement dus à des facteurs extérieurs qui ne sont pas liés au robot. Ces derniers sont principalement l'état du sol.

Dans [37], Zayd et al. ont créé une simulation qui est capable de contrôler le mouvement d'un robot mobile à entraînement différentiel en se basant sur un modèle cinématique. Cette simulation exploite l'algorithme des moindres carrés récursifs, utilisé précédemment dans les techniques de contrôle en estimant le coefficient de résistance au roulement, et le retour d'information des capteurs de vitesse et de position pour estimer les paramètres du modèle et par la suite améliorer son suivi de trajectoires. L'utilisation de l'algorithme RLS a permis d'obtenir des résultats satisfaisants dans le cadre de ce travail ce qui nous amène à le considérer comme un pilier pour réaliser ce travail de recherche

Chapitre 3 - Méthode adaptative de suivi des trajectoires

D'après la revue de littérature faite précédemment, l'algorithme des moindres carrés récursifs RLS a permis d'obtenir des résultats satisfaisants, une convergence rapide et une erreur asymptotique relativement faible lors du contrôle du mouvement d'un robot à entraînement différentiel . Ces avantages nous permettent de choisir la méthode de RLS afin d'aboutir aux objectifs de ce projet. La méthode de contrôle de mouvement conçue dans ce travail de recherche est une méthode adaptative qui permet à un robot mobile à entraînement différentiel de suivre avec plus de précision les trajectoires planifiées. Elle est basée principalement sur l'estimation des paramètres cinématiques du mouvement (ces derniers introduits et expliqués plus tard dans ce chapitre) à l'aide du RLS. Cette méthode est une boucle de contrôle qui consiste à mesurer les erreurs de mouvement en vitesse linéaire et angulaire du robot, estimer les paramètres cinématiques à l'aide du RLS et calculer les commandes à donner au robot à chaque instant k le long de son chemin à suivre. Pour réussir l'estimation de ces derniers paramètres et pour calculer les bonnes commandes, une modélisation des nouvelles équations cinématiques améliorées qui sont capables de décrire le mouvement du robot dans des conditions réelles et une identification des paramètres cinématiques à partir de ces équations sont indispensables. Il faut noter qu'un chemin de référence spécifique à suivre par le robot est défini. À partir de ce chemin, une série de vitesses linéaires et angulaires donnée au robot au cours du temps est générée. Il sert aussi à mesurer les erreurs de mouvement en position et en orientation en comparant le chemin suivi

par rapport au chemin de référence ou bien à suivre. Toutes les étapes de l'élaboration de la méthode de contrôle de mouvement sont présentées et expliquées en détail dans ce chapitre. Aussi, les équations cinématiques fondamentales du robot et une introduction des moindres carrés récursifs seront présentées.

3.1 Équations cinématiques fondamentales d'un robot à entraînement différentiel

Pour un robot à entraînement différentiel, les deux roues motrices sont situées sur le même axe et sont placées de chaque côté du corps du robot. Soit L l'entre-axe, V la vitesse linéaire du robot et $\dot{\theta}$ sa vitesse angulaire. V_d et V_g représentent les vitesses linéaires des roues gauche et droite.

Les deux équations cinématiques suivantes décrivent le mouvement de ce type de robot.

$$V = \frac{V_d + V_g}{2} \tag{3.1}$$

$$\dot{\theta} = \frac{V_d - V_g}{L} \tag{3.2}$$

Où $V_d = r_d \dot{\varphi}_d$ et $V_g = r_g \dot{\varphi}_g$

Avec

- r_d est le rayon de la roue droite
- r_g est le rayon de la roue gauche
- $\dot{\varphi}_d$ est la vitesse angulaire de la roue droite
- $\dot{\varphi}_g$ est la vitesse angulaire de la roue gauche

En remplaçant V_d et V_g par leurs expressions dans 2.1 et 2.2, les équations cinématiques deviennent :

$$V = \frac{r_d\dot{\varphi}_d + r_g\dot{\varphi}_g}{2} \tag{3.3}$$

$$\dot{\theta} = \frac{r_d\dot{\varphi}_d - r_g\dot{\varphi}_g}{L} \tag{3.4}$$

La figure suivante schématise un robot à entraînement différentiel et montre ses différents paramètres géométriques et ses différentes vitesses.

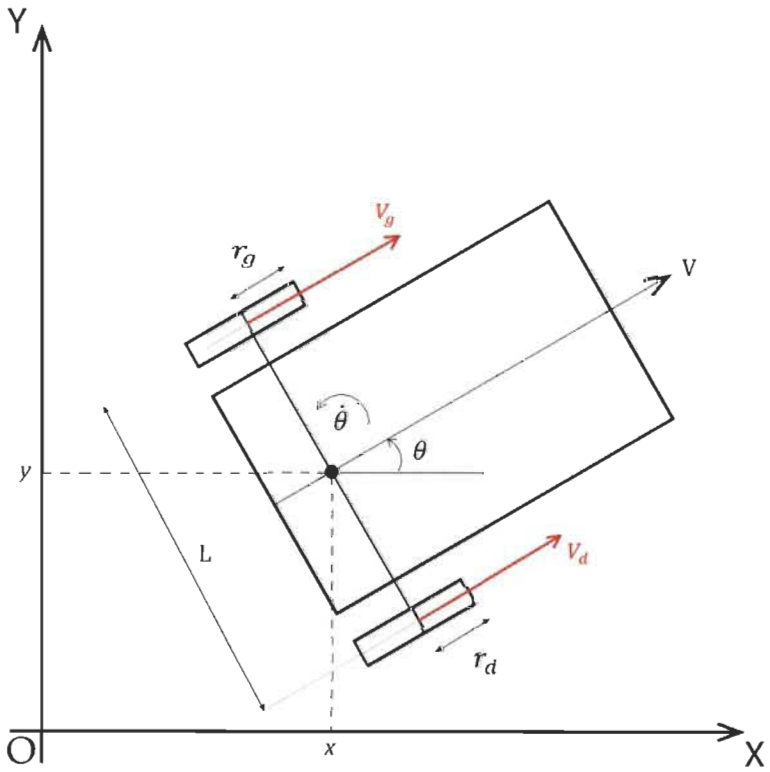


Figure 3.1 Schéma d'un robot à entraînement différentiel

3.2 Modélisation des équations cinématiques améliorées

Il faut bien noter que les équations cinématiques fondamentales présentées précédemment décrivent le mouvement d'un robot mobile à entraînement différentiel dans des conditions parfaites là où les glissements et les frottements n'existent pas, ce qui n'est pas le cas dans la vie réelle.

Afin de concevoir une méthode de contrôle efficace, il est indispensable d'améliorer le modèle cinématique classique en modélisant un nouveau modèle qui est capable de décrire le mouvement du robot au cours du temps et dans des conditions réelles. En d'autres termes, modéliser des équations qui sont capables de calculer les vraies valeurs des vitesses linéaire et angulaire Vet $\dot{\theta}$ en tenant compte des erreurs de mouvement causées par l'interaction entre les roues du robot et le sol (glissements, frottements ...).

À cause de ces erreurs, les roues du robot tendent à tourner plus ou moins que prévu en fonction de l'état de sol ce qui fausse le calcul des Vet $\dot{\theta}$ puisque ces derniers sont en fonction des vitesses de rotation des deux roues droite et gauche $\dot{\varphi}_d$ et $\dot{\varphi}_g$.

Pour remédier à ce problème, l'idée est de multiplier $\dot{\varphi}_d$ et $\dot{\varphi}_g$ dans les deux équations du modèle classique par des coefficients $\{\alpha, \beta, \gamma, \psi\}$ qui vont caractériser l'état de sol à chaque instant k. L'injection de ces coefficients dans les équations classiques permet de calculer les bonnes valeurs de Vet $\dot{\theta}$ et par la suite avoir des équations cinématiques plus robustes. On appellera les coefficients $\{\alpha, \beta, \gamma, \psi\}$ les paramètres cinématiques du robot.

En multipliant $\dot{\varphi}_d$ et $\dot{\varphi}_g$ par les paramètres cinématiques $\{\alpha, \beta, \gamma, \psi\}$ dans 2.3 et 2.4, on obtient le nouveau modèle cinématique du robot représenté par les équations suivantes :

$$V(t) = \frac{r_d \alpha(t) \dot{\varphi}_d(t) + r_g \beta(t) \dot{\varphi}_g(t)}{2} \tag{3.5}$$

$$\dot{\theta}(t) = \frac{r_d \gamma(t) \dot{\varphi}_d(t) - r_g \psi(t) \dot{\varphi}_g(t)}{L} \tag{3.6}$$

Dans ce travail de recherche, on va considérer l'hypothèse suivante :

Hypothèse 1 : Les deux roues du robot ont le même rayon ($r_d = r_g = r$).

L'hypothèse 1 nous permet de simplifier les équations précédentes et les remplacer par les suivantes :

$$V(t) = \frac{r}{2}\Big(\alpha(t) \dot{\varphi}_d(t) + \beta(t) \dot{\varphi}_g(t)\Big) \tag{3.7}$$

$$\dot{\theta}(t) = \frac{r}{L}\Big(\gamma(t) \dot{\varphi}_d(t) - \psi(t) \dot{\varphi}_g(t)\Big) \tag{3.8}$$

Les équations cinématiques améliorées permettent d'estimer des paramètres cinématiques $\{\alpha, \beta, \gamma, \psi\}$ avec l'algorithme RLS et le calcul des commandes $\dot{\varphi}_d$ et $\dot{\varphi}_g$ pour actionner les roues et déplacer le robot.

3.3 Définition du chemin de référence

Le chemin de référence est l'ensemble des trajectoires à suivre par le robot lors sa navigation d'un point à un autre. On appelle les extrémités d'une trajectoire les points de navigation.

Dans cette étude, l'hypothèse suivante est considérée :

Hypothèse 2 : Le robot se déplace seulement en ligne droite.

Cette hypothèse nous mène à constater que pour naviguer le long de son chemin, le robot doit effectuer des mouvements rectilignes d'un point de navigation à un autre et des mouvements de rotation autour de son centre de rotation à ces derniers points. On constate aussi que le chemin de référence est composé des trajectoires rectilignes seulement.

Pour naviguer d'un point à un autre, le robot doit effectuer une série de transition en passant d'une pose dans un point de navigation à une autre.

On rappelle qu'une pose du robot est le couple (position, orientation) noté aussi (p, θ). Une position p du robot est ses coordonnées (x, y) dans le plan alors que son orientation θ est l'angle qu'il fait par rapport à l'axe des x.

Cette série qu'on note S_{pos} est l'ensemble des poses de transition et elle est représentée de la manière suivante :

$$S_{pos} = \{(p_0, \theta_0), (p_1, \theta_1), \dots, (p_i, \theta_i), \dots, (p_n, \theta_n)\} \tag{3.9}$$

La figure suivante illustre un exemple d'un chemin de référence et l'ensemble de poses à suivre par le robot.

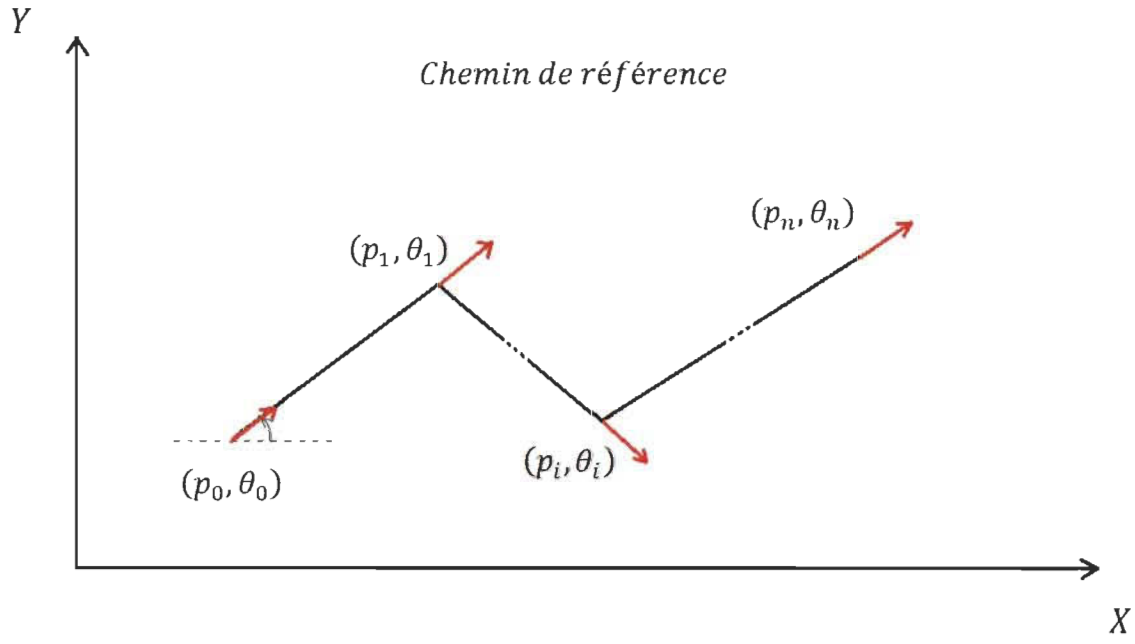


Figure 3.2 Chemin et poses de référence

À l'aide de cette série de poses, on est capable de mesurer les erreurs de mouvement en comparant les poses mesurées du robot par les poses de référence du chemin à suivre.

À partir de la même série, on peut calculer les commandes en vitesse linéaire V et angulaire $\dot{\theta}$ données au robot. Pour passer d'une pose à une autre, il faut calculer le couple de vitesse $(V, \dot{\theta})$ à exécuter par le robot pendant une durée de temps Δt. On peut représenter l'ensemble des commandes noté C_{vit} par :

$$C_{vit} = \{(V_0, \dot{\theta}_0, \Delta t_0), (V_1, \dot{\theta}_1, \Delta t_1), \dots, (V_i, \dot{\theta}_i, \Delta t_i), \dots, (V_{n-1}, \dot{\theta}_{n-1}, \Delta t_{n-1})\} \quad (3.10)$$

3.4 Les moindres carrés récursifs

La méthode des moindres carrés récursifs (MCR) ou bien en anglais Recursive least squares (RLS) [38] est un algorithme pour les filtres adaptatifs qui estime récursivement les coefficients qui minimisent une fonction de coût, plus précisément qui minimisent les moindres carrés pondérés.

Cette approche est une extension d'autres algorithmes tels que les moindres carrés moyens (LMS) [39] qui visent à réduire l'erreur quadratique moyenne. Pour le RLS, les signaux d'entrée peuvent être considérés comme déterministes, tandis que pour le LMS et d'autres algorithmes similaires, ils sont souvent considérés comme stochastiques. Comparé à la plupart de ses concurrents, le RLS présente une convergence extrêmement rapide. En général, le RLS peut être utilisé pour résoudre n'importe quel problème qui peut être résolu par les filtres adaptatifs. Cependant, cet avantage se fait au prix d'une grande complexité de calcul.

La méthode RLS a été découverte par Carl Friedrich Gauss [40]. Il a déjà élaboré les concepts fondamentaux en 1795 lorsqu'il avait l'âge de 18 ans. Sa méthode a été publiée la première fois en 1809 lorsqu'elle parut dans le tome 2 de ses travaux sur la Mécanique céleste, cependant elle est restée inutilisée ou ignorée jusqu'à 1950 quand Plackett [41] a redécouvert l'originalité de l'œuvre de Gauss.

3.5 Identification des vecteurs paramètres et des vecteurs de régression

Pour procéder à l'estimation des paramètres cinématiques $\{\alpha, \beta, \gamma, \psi\}$ du robot à l'aide de l'algorithme RLS, deux modèles linéaires du système étudié sont nécessaires. La linéarité des équations 3.3 et 3.4 par rapport à $\{\alpha, \beta, \gamma, \psi\}$ nous permet d'obtenir deux modèles linéaires, soit un modèle linéaire pour chaque équation. Par la suite, une estimation avec RLS est réalisée pour chaque modèle. Ces modèles nous permettent d'identifier les vecteurs paramètres w_1 et w_2 et les vecteurs de régression u_1 et u_2 du système.

Si on réécrit les équations 3.3 et 3.4 sous une forme matricielle on obtient :

$$V(t) = w_1(t)u_1(t) \tag{3.11}$$

$$\dot{\theta}(t) = w_2(t)u_2(t) \tag{3.12}$$

Où :

$$w_1(t) = [\alpha(t), \beta(t)]$$

$$u_1(t) = \frac{r}{2}\left[\dot{\varphi}_d(t), \dot{\varphi}_g(t)\right]^T$$

$$w_2 = [\gamma(t), -\psi(t)]$$

$$u_2(t) = \frac{r}{L}\left[\dot{\varphi}_d(t), \dot{\varphi}_g(t)\right]^T$$

Les vecteurs paramètres w_1 et w_2 doivent être estimé de manière qu'ils minimisent des fonctions de coût J_1 et J_2 qui représentent les erreurs quadratiques entre les modèles linéaires et les valeurs expérimentaux.

J_1 et J_2 sont de la forme :

$$J_1(\widehat{w}_1) = \frac{1}{2}\sum_{t=1}^{N}\left(V(t) - w_1(t)u_1(t)\right)^2 \tag{3.13}$$

$$J_2(\widehat{w}_2) = \frac{1}{2}\sum_{t=1}^{N}\left(\dot{\theta}(t) - w_2(t)u_2(t)\right)^2 \tag{3.14}$$

N représente le nombre des valeurs discrètes, $\widehat{w}_1$ l'estimation du vecteur w_1 et $\widehat{w}_2$ l'estimation du vecteur w_2. On peut noter les paramètres cinématiques estimés par $\{\hat{\alpha}, \hat{\beta}, \hat{\gamma}, \hat{\psi}\}$, par la suite $\widehat{w}_1$ et $\widehat{w}_2$ sont représentés par :

$$\widehat{w}_1(t) = \left[\hat{\alpha}(t), \hat{\beta}(t)\right] \tag{3.15}$$

$$\widehat{w}_2(t) = \left[\hat{\gamma}(t), -\hat{\psi}(t)\right] \tag{3.16}$$

3.6 Processus de contrôle du mouvement

Le processus de contrôle du mouvement est une boucle de régulation qui vise à minimiser à chaque instant k les erreurs de mouvement du robot. Cette boucle est basée sur trois étapes importantes, la mesure des vitesses linéaires et angulaires du robot en utilisant respectivement un système de positionnement en intérieur et un compas électronique, l'estimation des paramètres cinématiques à l'aide de l'algorithme des moindres carrés RLS et le calcul des vitesses de rotation des deux roues en utilisant la cinématique inverse. Les trois étapes sont détaillées et présentées dans ce chapitre. Un algorithme et un schéma fonctionnel de la boucle de régulation sont présentés dans ce chapitre et dans l'annexe A.

3.6.1 Mesure des vitesses linéaires et angulaires

La mesure des vitesses linéaires et angulaires du robot à chaque instant k, est indispensable pour réussir la prochaine étape. Ces vitesses mesurées vont servir à calculer les erreurs entre ces dernières et les vitesses données comme commandes au robot. À partir des erreurs calculées, une estimation des paramètres du système $\{\hat{\alpha}, \hat{\beta}, \hat{\gamma}, \hat{\psi}\}$ va être réalisée. La vitesse linéaire du robot est mesurée à l'aide d'un système de positionnement en intérieur GPS, alors que la vitesse angulaire est mesurée à l'aide d'un compas électronique. On peut noter ces deux vitesses mesurées respectivement par $V_m(k)$ et $\dot{\theta}_m(k)$.

3.6.2 Estimation des paramètres cinématiques

Dans cette partie, une estimation des paramètres cinématiques du robot $\{\hat{\alpha}, \hat{\beta}, \hat{\gamma}, \hat{\psi}\}$ en utilisant l'algorithme des moindres carrés récursifs RLS est présenté par une série d'équations. On rappelle que l'estimation des vecteurs paramètres $\widehat{w}_1(k)$ et $\widehat{w}_2(k)$ revient à estimer $\{\hat{\alpha}(k), \hat{\beta}(k), \hat{\gamma}(k), \hat{\psi}(k)\}$. Les équations qui permettent l'estimation de $\widehat{w}_1(k)$ et $\widehat{w}_2(k)$ sont défini comme suit :

$$\widehat{w}_1(k) = \widehat{w}_1(k-1) + G_1(k)\xi_1(k) \tag{3.17}$$

$$\widehat{w}_2(k) = \widehat{w}_2(k-1) + G_2(k)\xi_2(k) \tag{3.18}$$

$\widehat{w}_1$ et $\widehat{w}_2$ sont calculés à partir de G_1, G_2 et ξ_1, ξ_2 qui représentent respectivement les vecteurs gains et les erreurs calculées à partir des mesures prises.

$\widehat{w}_1$ et $\widehat{w}_2$ sont initialisés par :

$$\widehat{w}_1(0) = [\hat{\alpha}(0), \hat{\beta}(0)]$$

$$\widehat{w}_2(0) = [\hat{\gamma}(0), -\hat{\psi}(0)]$$

Les vecteurs gains G_1 et G_2 et les erreurs ξ_1 et ξ_2 sont définis par les équations suivantes :

$$G_1(k) = \frac{P_1(k-1)u_1(k)}{\lambda + u_1{}^T(k)P_1(k-1)u_1(k)} \tag{3.19}$$

$$G_2(k) = \frac{P_2(k-1)u_2(k)}{\lambda + u_2{}^T(k)P_2(k-1)u_2(k)} \tag{3.20}$$

$$\xi_1(k) = V_m(k) - \widehat{w}_1(k-1)u_1(k) \tag{3.21}$$

$$\xi_2(k) = \dot{\theta}_m(k) - \widehat{w}_2(k-1)u_2(k) \tag{3.22}$$

G_1 et G_2 sont calculés à partir de P_1 et P_2 qui représentent les matrices de covariance, les vecteurs de régression u_1 et u_2, et λ représente le facteur d'oubli ($0< \lambda \leq 1$) qui sert à donner moins de poids aux estimations précédentes. Dans ce projet de recherche, des surfaces qui sont relativement planes contenant parfois des difformités seront utilisées dans la partie expérimentale. Ces types de surfaces causent une réactivité stochastique du robot lors de son mouvement ce qui nous pousse à oublier le poids des paramètres cinématiques précédents. Par conséquent, le facteur d'oubli λ doit s'approcher le plus possible de 1. Après plusieurs essais expérimentaux, λ est fixé à 0.99.

ξ_1 et ξ_2 sont calculées à partir des vitesses linéaire et angulaire mesurées du robot, des estimations précédentes de $\widehat{w}_1$ et $\widehat{w}_2$ et des vecteurs de régression u_1 et u_2.

Les vecteurs de régression u_1 et u_2 sont définis en fonctions du rayon de la roue *r*, l'entraxe *L* et les vitesses de rotation des deux roues $\dot{\varphi}_d$ et $\dot{\varphi}_g$ qui représentent les commandes à donner au robot pour actionner ces roues. u_1 et u_2 sont représentés comme suit :

$$u_1(k) = \frac{r}{2}\left[\dot{\varphi}_d(k-1), \dot{\varphi}_g(k-1)\right]^T \tag{3.23}$$

$$u_2(k) = \frac{r}{L}\left[\dot{\varphi}_d(k-1), \dot{\varphi}_g(k-1)\right]^T \tag{3.24}$$

Les commandes initiales $\dot{\varphi}_d(0)$ et $\dot{\varphi}_g(0)$ sont calculées en résolvant le système des deux équations cinématiques (expliqué dans la prochaine partie).

Les matrices de covariance P_1 et P_2 sont calculées à l'aide des équations suivantes :

$$P_1(k) = \lambda^{-1}P_1(k-1) - \lambda^{-1}G_1(k){u_1}^T(k)P_1(k-1) \quad (3.25)$$

$$P_2(k) = \lambda^{-1}P_2(k-1) - \lambda^{-1}G_2(k){u_2}^T(k)P_2(k-1) \quad (3.26)$$

P_1 et P_2 sont initialisées comme suit :

$$P_1(0) = P_2(0) = \delta^{-1}I$$

I représente la matrice identité et δ le paramètre de régularisation qui est une constante positive très faible. Après plusieurs essais, δ sera fixé à 0.01.

3.6.3 Calcul des commandes $\dot{\varphi}_d$ et $\dot{\varphi}_g$

La dernière étape du processus de contrôle est de calculer les commandes $\dot{\varphi}_d$ et $\dot{\varphi}_g$. Une fois les paramètres cinématiques $\{\hat{\alpha}, \hat{\beta}, \hat{\gamma}, \hat{\psi}\}$ sont estimés et les vitesses linéaire et angulaire V et $\dot{\theta}$ sont calculées (section 3.3), $\dot{\varphi}_d$ et $\dot{\varphi}_g$ sont calculées en utilisant la cinématique inverse qui consiste à résoudre le système des équations cinématiques présenté ci-dessous.

$$\begin{cases} V(k) = \dfrac{r}{2}\left(\hat{\alpha}(k)\dot{\varphi}_d(k) + \hat{\beta}(k)\dot{\varphi}_g(k)\right) & (3.27) \\ \dot{\theta}(k) = \dfrac{r}{L}\left(\hat{\gamma}(k)\dot{\varphi}_d(k) - \hat{\psi}(k)\dot{\varphi}_g(k)\right) & (3.28) \end{cases}$$

3.7 Conclusion

La méthode de contrôle adaptatif du mouvement d'un robot à entraînement différentiel élaborée dans ce chapitre peut être résumée avec le schéma fonctionnel et l'algorithme suivants :

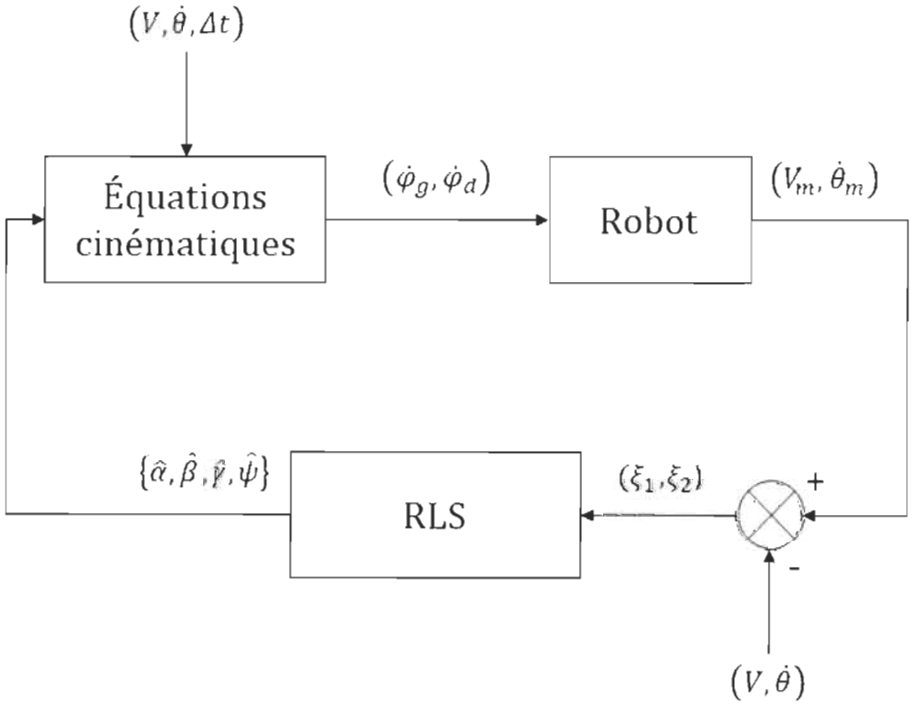


Figure 3.3 Schéma fonctionnel du processus de contrôle

initialisation :

$$\hat{w}_1(0) = [\alpha(0), \beta(0)]$$
$$\hat{w}_2(0) = [\gamma(0), \psi(0)]$$
$$P_1(0) = P_2(0) = \delta^{-1} I$$

calcul de $\dot{\varphi}_d(0)$ *et* $\dot{\varphi}_g(0)$ *en résolvant*

$$\begin{cases} V(0) = \frac{r}{2}\left(\alpha(0)\dot{\varphi}_d(0) + \beta(0)\dot{\varphi}_g(0)\right) \\ \dot{\theta}(0) = \frac{r}{L}\left(\gamma(0)\dot{\varphi}_d(0) - \psi(0)\dot{\varphi}_g(0)\right) \end{cases}$$

Pour $k = 1,2,\dots,N$ *calculer* :

$$u_1(k) = \frac{r}{2}\left[\dot{\varphi}_d(k-1), \dot{\varphi}_g(k-1)\right]^T$$
$$u_2(k) = \frac{r}{L}\left[\dot{\varphi}_d(k-1), \dot{\varphi}_g(k-1)\right]^T$$

$$G_1(k) = \frac{P_1(k-1)u_1(k)}{\lambda + {u_1}^T(k)P_1(k-1)u_1(k)}$$
$$G_2(k) = \frac{P_2(k-1)u_2(k)}{\lambda + {u_2}^T(k)P_2(k-1)u_2(k)}$$

$$\xi_1(k) = V_m(k) - \hat{w}_1(k-1)u_1(k)$$
$$\xi_2(k) = \dot{\theta}_m(k) - \hat{w}_2(k-1)u_2(k)$$

$$\hat{w}_1(k) = \hat{w}_1(k-1) + G_1(k)\xi_1(k)$$
$$\hat{w}_2(k) = \hat{w}_2(k-1) + G_2(k)\xi_2(k)$$

$$P_1(k) = \lambda^{-1}P_1(k-1) - \lambda^{-1}G_1(k){u_1}^T(k)P_1(k-1)$$
$$P_2(k) = \lambda^{-1}P_2(k-1) - \lambda^{-1}G_2(k){u_2}^T(k)P_2(k-1)$$

calcul de $\dot{\varphi}_g(k)$ *et* $\dot{\varphi}_d(k)$ *en résolvant*

$$\begin{cases} V(k) = \frac{r}{2}\left(\hat{\alpha}(k)\dot{\varphi}_d(k) + \hat{\beta}(k)\dot{\varphi}_g(k)\right) \\ \dot{\theta}(k) = \frac{r}{L}\left(\hat{\gamma}(k)\dot{\varphi}_d(k) - \hat{\psi}(k)\dot{\varphi}_g(k)\right) \end{cases}$$

Fin Pour

Fin Algorithme

Dans le chapitre suivant, cet algorithme sera traduit à un langage de programmation et appliqué sur le robot pour valider expérimentalement la méthode de contrôle conçue.

Chapitre 4 - Validation expérimentale

La méthode de contrôle adaptatif proposée dans ce mémoire a été validée expérimentalement. Cette méthode est testée sur trois types de surfaces différentes (tapis, ciment, contreplaqué). Elle est aussi comparée à la méthode de UMBMark. Ce chapitre présente alors une description du banc d'essai, le plan expérimental et les résultats obtenus.

4.1 Banc d'essai expérimental

4.1.1 Le Robot "Dr Robot X80Pro"

Figure 4.1 Robot "Dr Robot X80Pro"

Dr Robot X80Pro [42] est un robot mobile à entraînement différentiel qui est conçu pour les chercheurs afin de développer des applications robotiques avancées comme le contrôle de mouvement et la navigation autonome. Il possède deux roues motrices de rayon 8.25 cm et un entraxe de longueur 28 cm. Chaque roue est actionnée par un moteur électrique de 12V DC ce qui permet d'atteindre une vitesse de 0.75m/s. Ce robot possède une roue libre à l'arrière et une autre ajoutée en avant pour augmenter la stabilité de la plateforme.

4.1.2 Le GPS "Marvelmind"

Figure 4.2 Indoor GPS "Marvelmind"

Le GPS Marvelmind [43] est un système de positionnement en intérieur conçu pour fournir des données de localisation précises (±2cm) aux robots autonomes, aux véhicules (AGV) et aux drones. Il est également utilisé pour suivre d'autres objets sur lesquels une balise mobile est installée, par exemple, les systèmes de réalité virtuelle (VR), les casques pour les travailleurs de la construction ou des mines, etc.

Le système de positionnement est basé sur cinq balises à ultrasons qui sont réunies par un interface radio. Une balise mobile est installée sur le robot. Sa position et sa vitesse sont calculées en se basant sur les délais de propagation des signaux ultrasonores (Time-Of-Flight ou TOF) des balises fixes et utilisant la méthode de trilatération.

4.1.3 Le Compas "HMC5883L"

Figure 4.3 Le compas "HMC5883L"

Le compas "HMC5883L" [44] est conçu pour la détection des champs magnétiques faibles en utilisant une interface numérique. Il fournir le cap (l'orientation) et la vitesse angulaire d'un système sur lequel il est embarqué. Ce compas est très efficace, car il fournit une orientation de précision ±2 degrés.

4.1.4 Surfaces de travail

Dans ce travail de recherche les trois types de surfaces utilisés pour valider la méthode de contrôle sont les suivants :

Figure 4.4 Surfaces expérimentales

4.1.5 Logiciel de commande

Un logiciel validant la méthode de contrôle conçue est écrit avec le langage de programmation orienté objet C#. Il possède une interface graphique qui facilite la commande du robot et la communication avec l'opérateur. Le code comporte :

- Une déclaration de toutes les bibliothèques utilisées.
- Une fonction pour établir la connexion avec le robot en utilisant le WIFI
- Les fonctions pour l'envoi des commandes et la réception des données du GPS et du Compas
- Une fonction pour l'enregistrement des données dans un fichier Excel
- Une déclaration de toutes les variables utilisées dans le code
- Toutes les fonctions nécessaires pour le calcul vectoriel et matriciel
- Une fonction pour le calcul des commandes
- Une fonction pour commander le robot sans la méthode de contrôle
- Une fonction pour commander le robot avec la méthode de contrôle
- Une fonction pour l'estimation avec le RLS
- Une fonction pour l'arrêt du robot en cas d'urgence
- Une fonction pour l'affichage des données
- Une fonction pour le traçage des trajectoires suivies par le robot

Le code en entier est présenté et bien commenté dans l'annexe C. L'interface aussi est présentée dans la même annexe.

4.1.6 Chemin de référence

Un chemin de référence va nous permettre de mesurer les erreurs de la performance de la méthode de contrôle adoptée. Ce chemin est maintenu dans toutes les expériences. Il est représenté par une ligne noire de longueur 7 m et constitué par 15 points de navigation représentés par des cercles bleus. La distance entre deux points consécutifs est de 0.5 m. Les deux flèches en rouge représentent les orientations initiale et finale du robot. Le robot est supposé suivre ce chemin en passant par les points de navigation, en partant de la pose de départ ((0,0),0) et en arrivant à la pose finale ((0,3),180°). À chaque point de navigation, les poses sont mesurées à l'aide du GPS et le compas et les erreurs sont calculées. La figure suivante illustre le chemin de référence choisi :

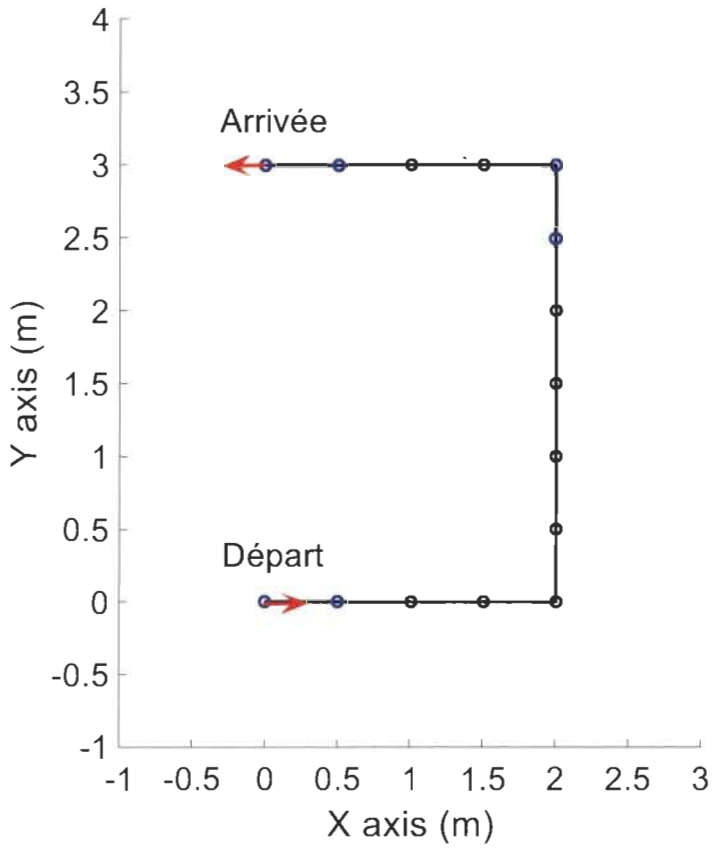


Figure 4.5 Chemin expérimental de référence

4.1.7 Paramètres expérimentaux

Pour réaliser les expériences, il est important de préciser les paramètres géométriques du robot, ainsi que les paramètres de l'algorithme d'estimation. Tous ces paramètres sont représentés dans le tableau suivant.

Nom du paramètre	**Notation**	**Valeur**
Rayon des deux roues	r	8.25 cm
Longueur de l'entraxe	L	28 cm
Vitesse linéaire	V	0.5 m/s
Vitesse angulaire	$\dot{\theta}$	π rad/s
Paramètres cinématiques	$\{\alpha(0),\beta(0),\gamma(0),\psi(0)\}$	{1,1,1,1}
Paramètre de régularisation	δ	0.01
Facteur d'oubli	λ	0.99

Tableau 4.1 Paramètres expérimentaux

4.2 Résultats et discussions

4.2.1 Résultats

Dans cette section, les résultats obtenus lors des expériences sont présentés sous forme de figures et de tableaux comparatifs. Les figures représentent les mesures prises et les erreurs calculées dans chaque point de navigation et les tableaux représentent les erreurs moyennes calculées et les convergences en pourcentage vers la pose finale.

L'erreur moyenne et la convergence vers la pose finale sont calculées à l'aide des équations suivantes :

$$E_m = \frac{\sum_{i=1}^{l} E_i}{l} \tag{4.1}$$

$$\%C = \frac{E_f(sans\ contrôle) - E_f(avec\ contrôle)}{E_f(sans\ contrôle)} \times 100 \tag{4.2}$$

E_m représente l'erreur moyenne calculée, E_i l'erreur mesurée à la $i^{ème}$ point de navigation et l le nombre des points de navigation. $\%C$ représente le taux de convergence en pourcentage vers une position ou une orientation finale et E_f l'erreur mesurée au point de navigation final.

Il faut noter que les erreurs E_m, E_i et E_f peuvent être des erreurs en position ou en orientation tout dépend de la mesure sélectionné (position/orientation).

La méthode de contrôle proposée sera comparée aux autres méthodes (Sans contrôle et UMBMark) pour chaque type de surface.

Surface de tapis :

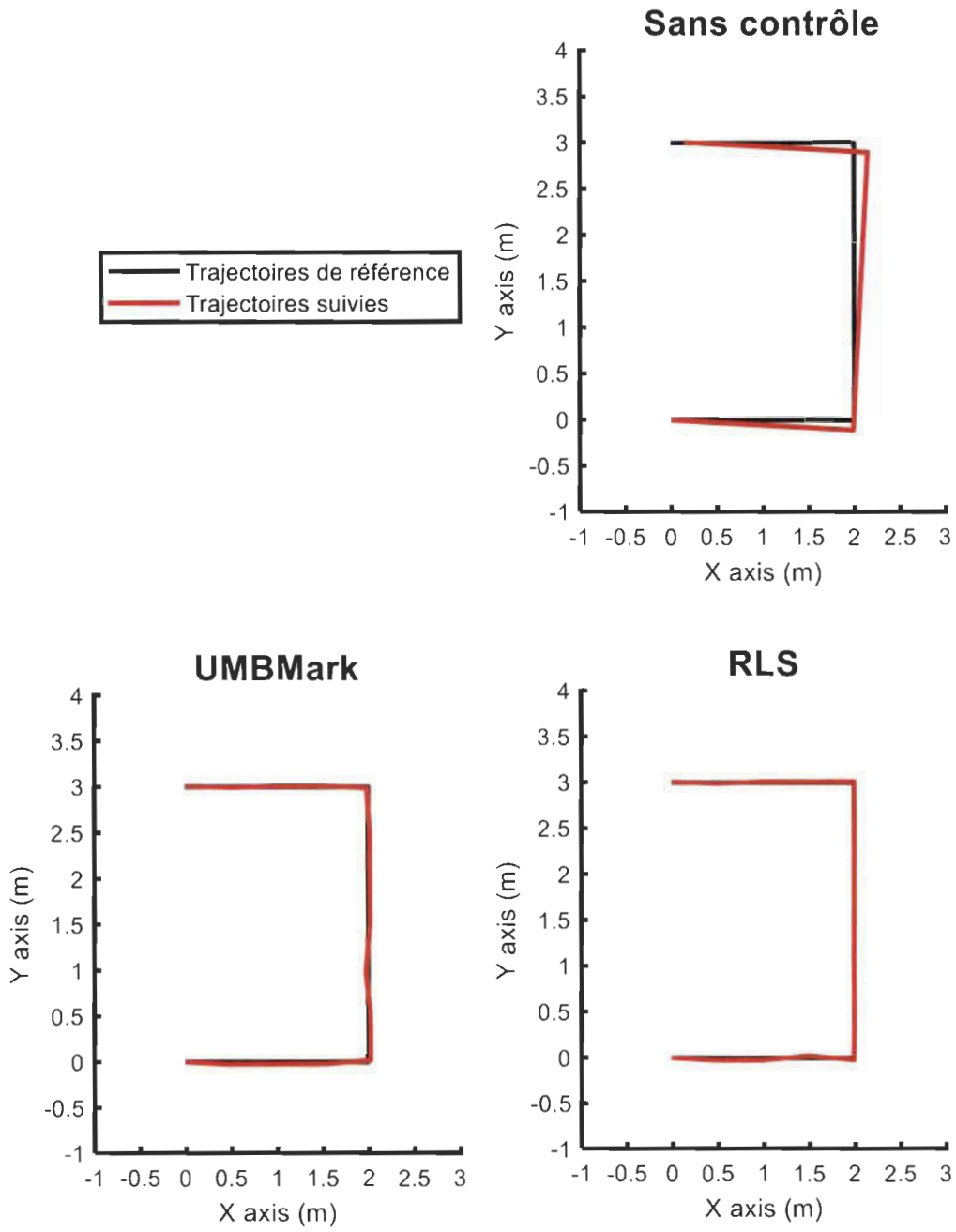


Figure 4.6 Mesure des positions (Tapis)

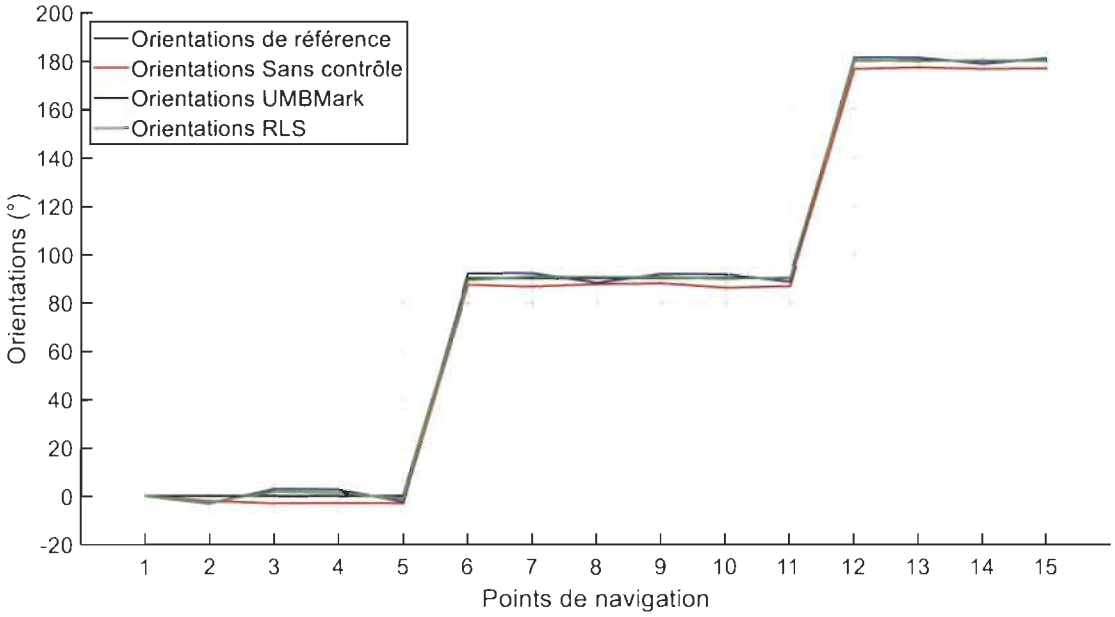


Figure 4.7 Mesure des orientations (Tapis)

	UMBMark	RLS
Position	72.84 %	84,92 %
Orientation	63.08 %	74,19 %

Tableau 4.2 Convergences vers la pose finale en pourcentage (Tapis)

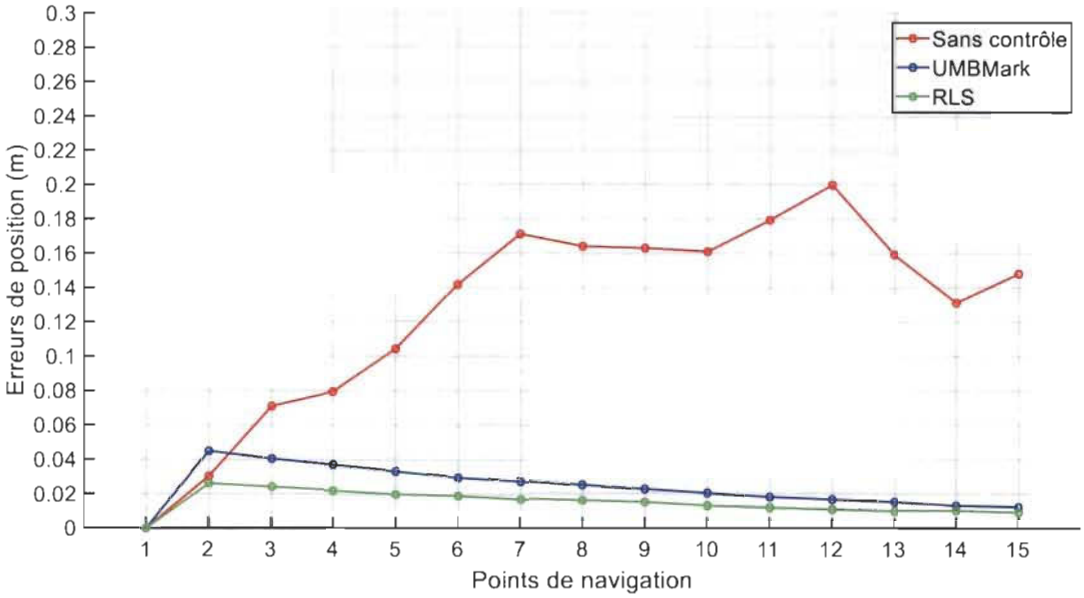


Figure 4.8 Mesure des erreurs de position (Tapis)

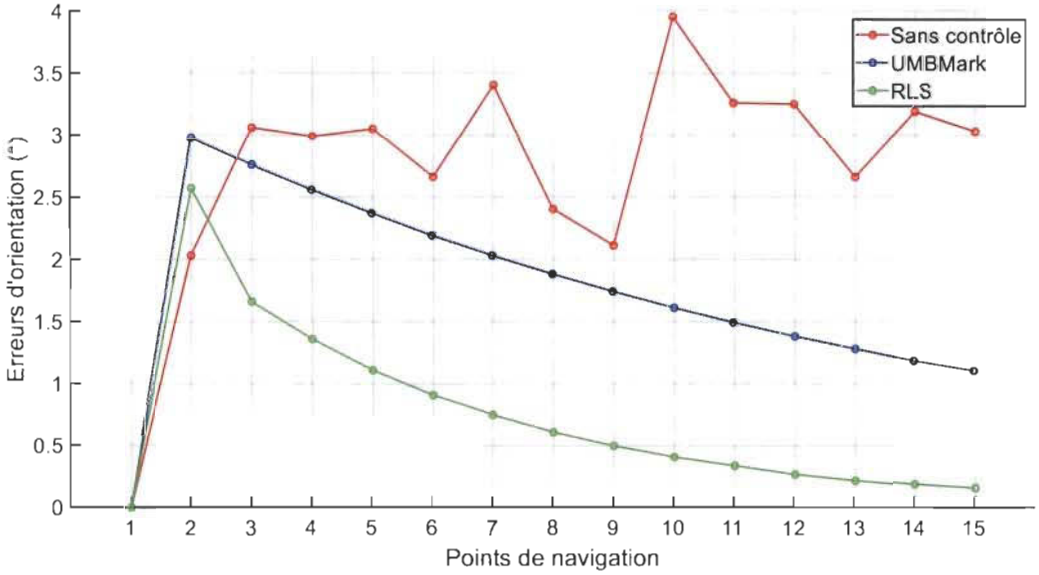


Figure 4.9 Mesure des erreurs d'orientation (Tapis)

	Sans contrôle	UMBMark	RLS
Position	0.119 m	0.023 m	0.016 m
Orientation	2,43 °	1.77 °	0,47 °

Tableau 4.3 Erreurs moyennes (Tapis)

➢ Surface de ciment :

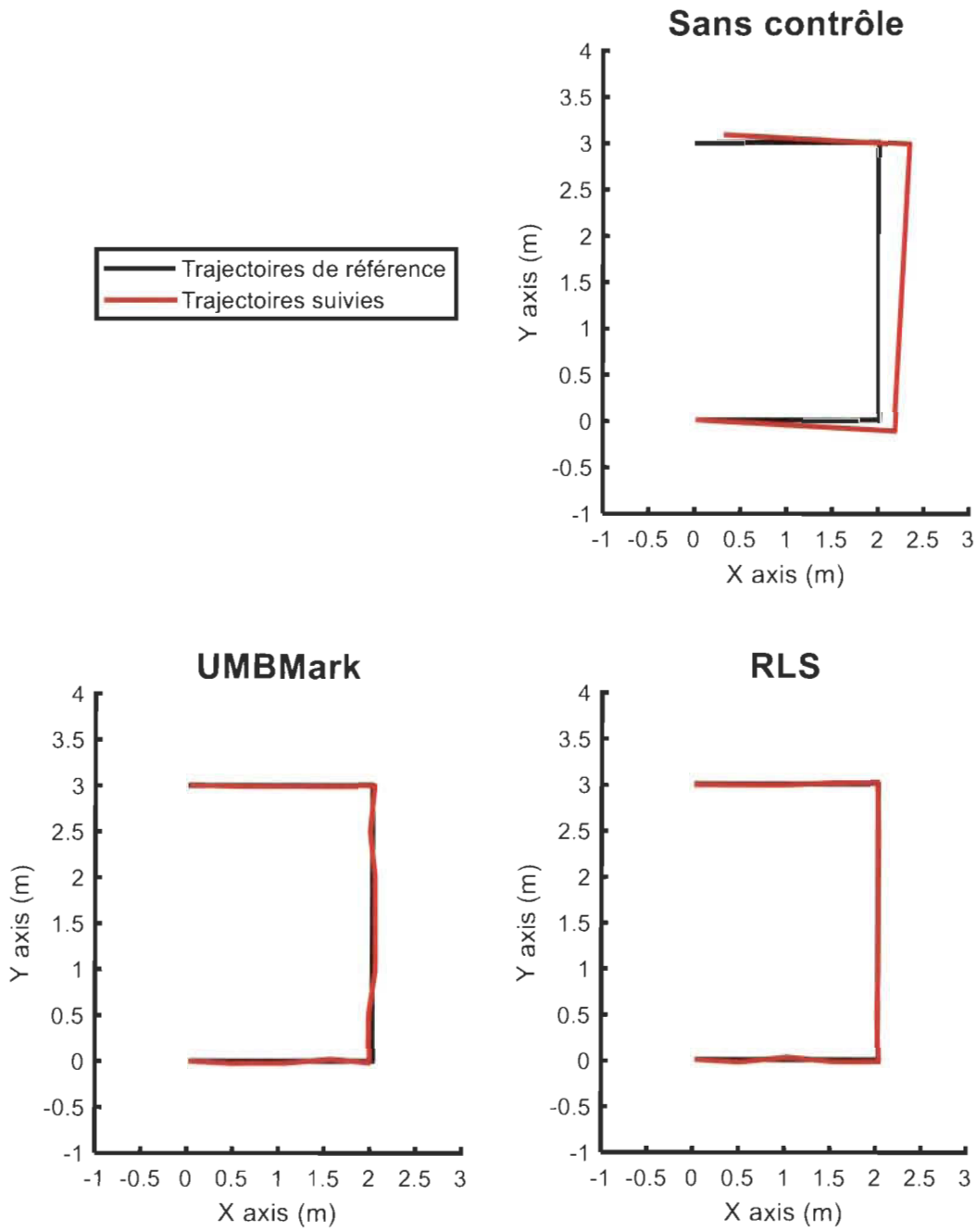


Figure 4.10 Mesure des positions (Ciment)

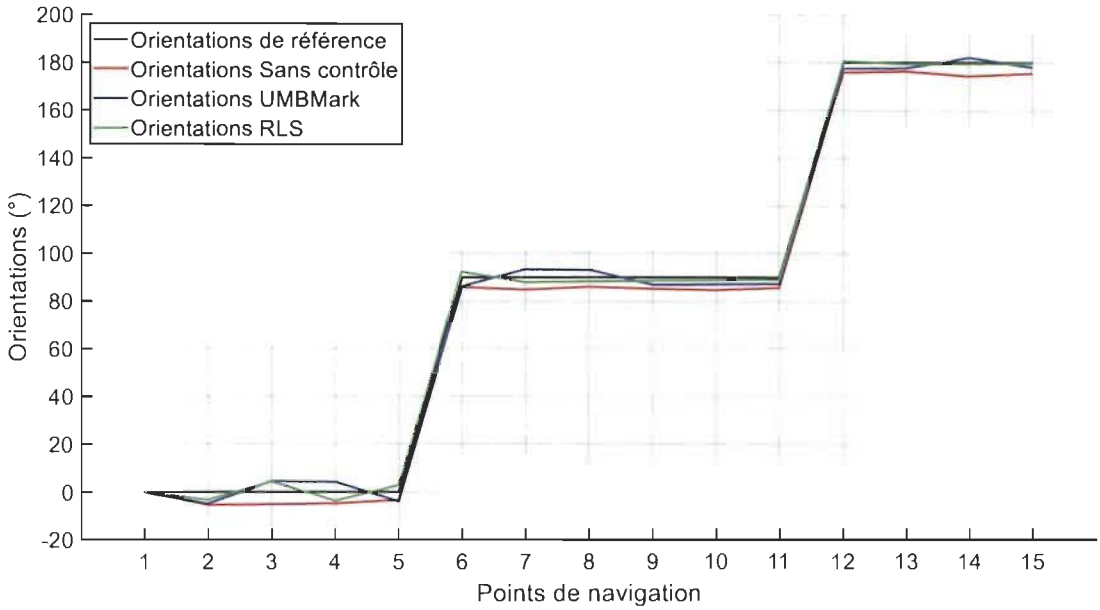


Figure 4.11 Mesure des orientations (Ciment)

	UMBMark	RLS
Position	69.28 %	80,18 %
Orientation	60.44 %	72,85 %

Tableau 4.4 Convergences vers la pose finale en pourcentage (Ciment)

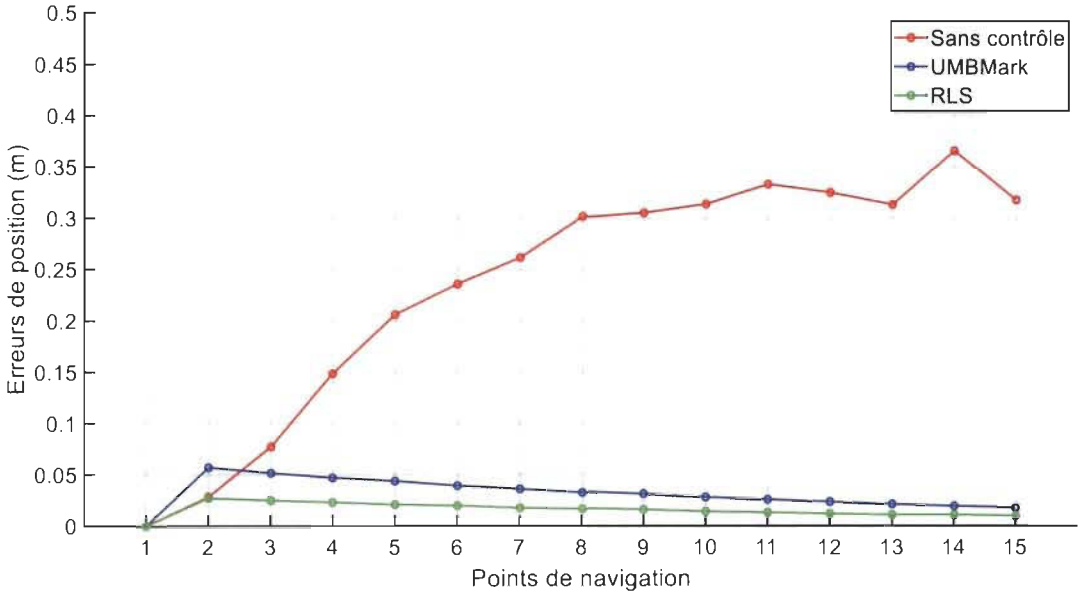


Figure 4.12 Mesure des erreurs de position (Ciment)

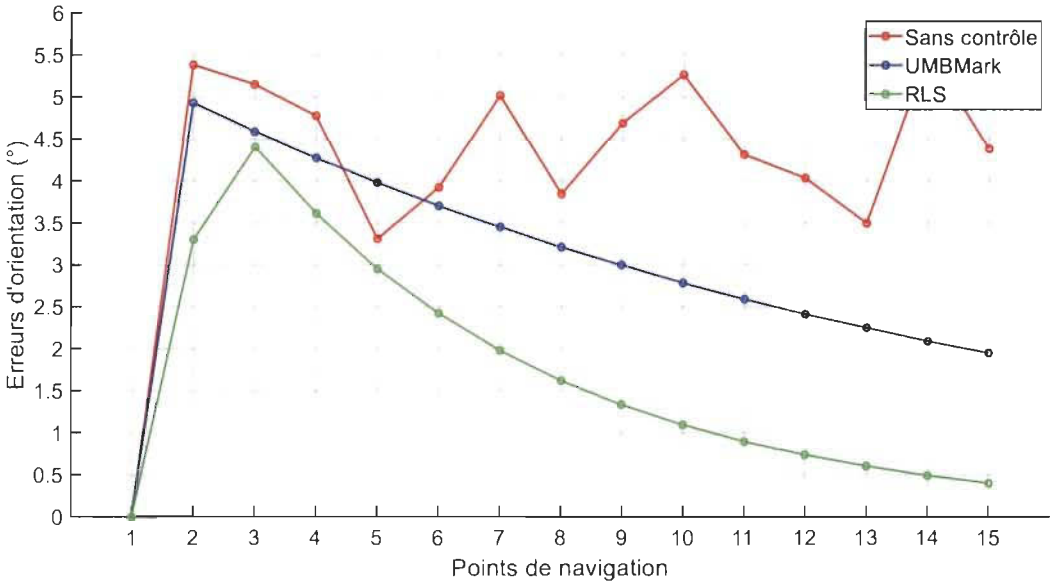


Figure 4.13 Mesure des erreurs d'orientation (Ciment)

	Sans contrôle	UMBMark	RLS
Position	0.221 m	0.031 m	0.023 m
Orientation	4,81 °	3.01 °	2,33 °

Tableau 4.5 Erreurs moyennes (Ciment)

➢ Surface de contreplaqué :

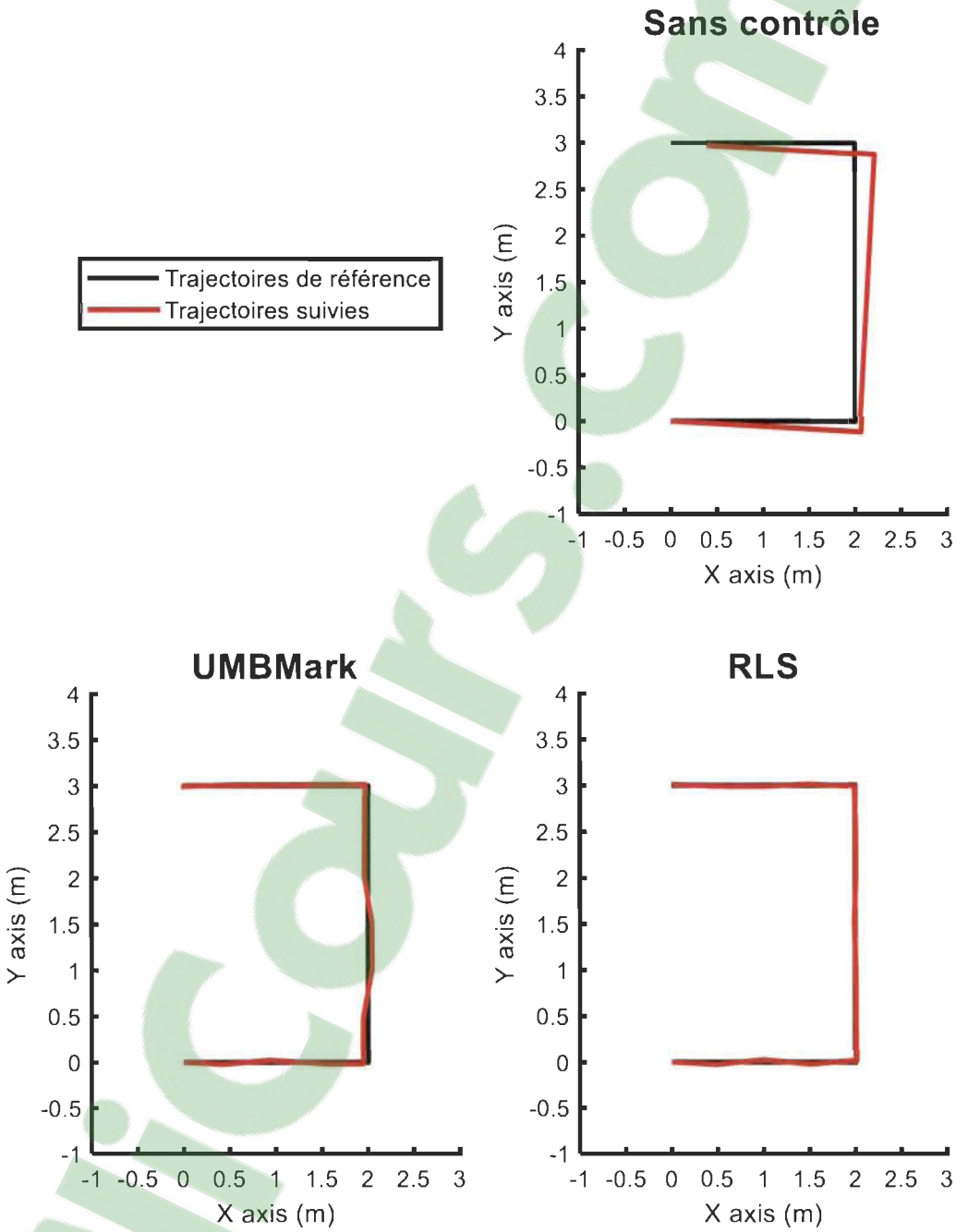


Figure 4.14 Mesure des positions (Contreplaqué)

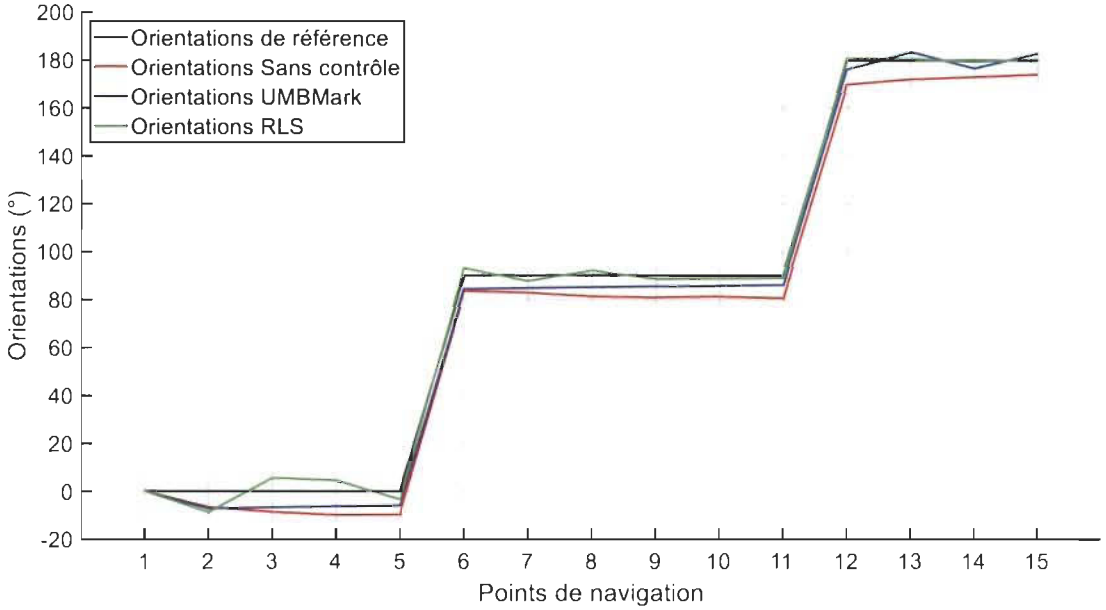


Figure 4.15 Mesure des orientations (Contreplaqué)

	UMBMark	RLS
Position	66.79 %	77,04 %
Orientation	58.03 %	70,94 %

Tableau 4.6 Convergences vers la pose finale en pourcentage (Contreplaqué)

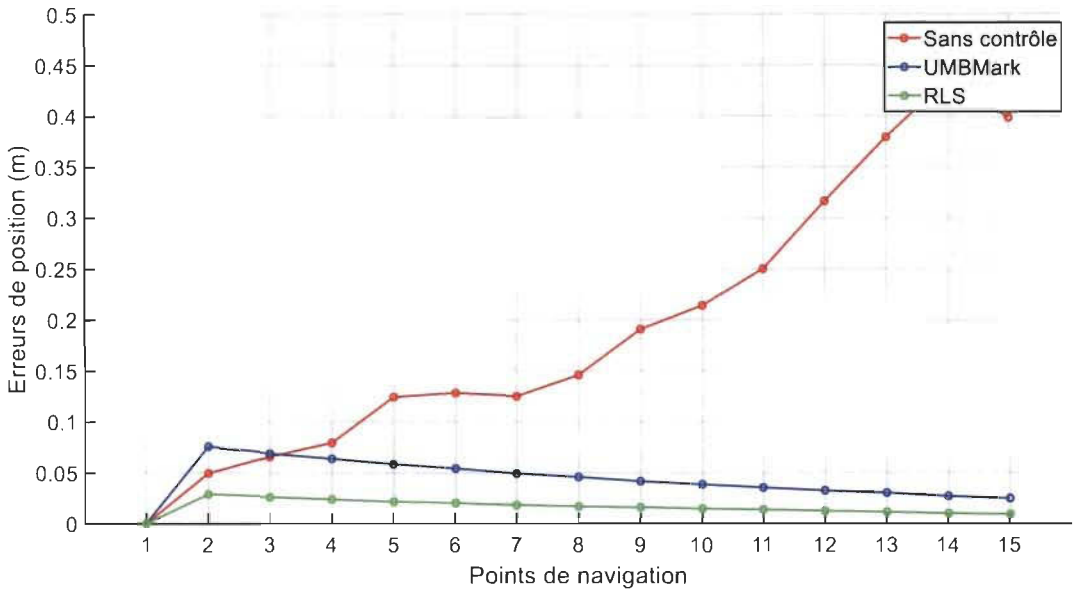


Figure 4.16 Mesure des erreurs de position (Contreplaqué)

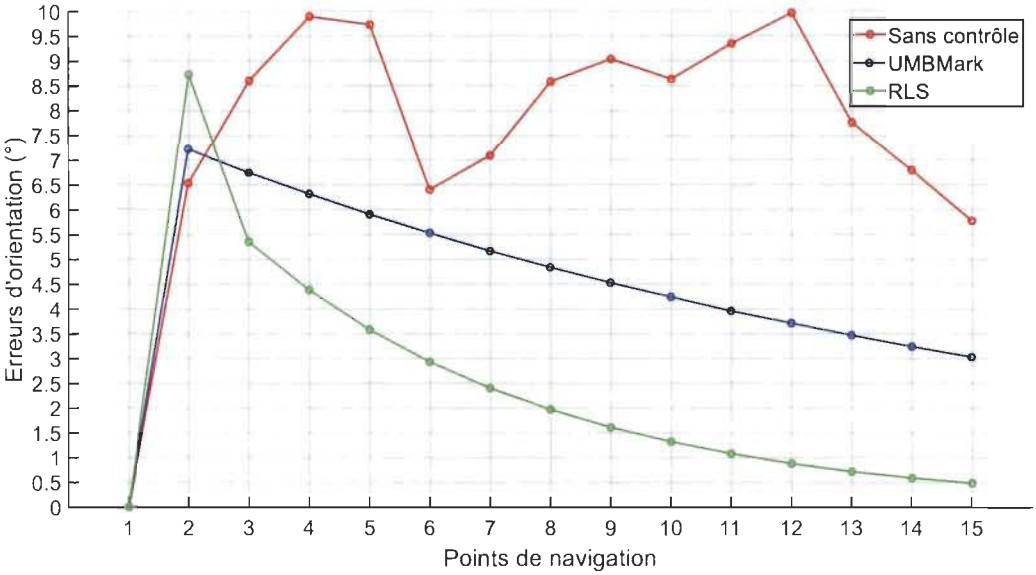


Figure 4.17 Mesure des erreurs d'orientation (Contreplaqué)

	Sans contrôle	UMBMark	RLS
Position	0.297 m	0.043 m	0,038 m
Orientation	6,65 °	4.52 °	3,32 °

Tableau 4.7 Erreurs moyennes (Contreplaqué)

4.2.2 Discussions

L'analyse et la discussion des résultats peuvent être généralisées pour les trois types de surface puisque les résultats obtenus pour chaque surface se rapprochent.

Les courbes montrent clairement que sans une méthode de contrôle le robot subit des déviations remarquables lors de son mouvement d'un point de navigation à un autre. Les erreurs en position et en orientation et les erreurs moyennes qui représentent l'aptitude du robot à suivre le chemin de référence lors de son voyage sont énormes par rapport aux méthodes de contrôle. Il est clair aussi que le robot ne converge pas vers la pose finale sans l'utilisation d'une méthode de contrôle.

Les résultats indiquent que l'application des deux méthodes de contrôle (UMBmark et RLS) diminuent les déviations du robot et optimisent son suivi des trajectoires planifiées. Cependant, les figures et les tableaux montrent que la méthode de contrôle adaptatif en utilisant l'algorithme des moindres carrés récursifs est plus efficace que la méthode de contrôle UMBMark. D'après les figures, la méthode RLS offre une convergence plus rapide vers la pose finale que UMBMark. Les tableaux Tableau 4.2, Tableau 4.4 et Tableau 4.6 montrent que RLS offre une convergence de 10% de plus que UMBMark. Avec la méthode de contrôle élaborée, le robot peut atteindre une convergence de 84% en position et 74% en orientation. Les tableaux Tableau 4.3, Tableau 4.5 et Tableau 4.7 montrent que les erreurs moyennes en position et en orientation pour la méthode RLS sont plus petites que la méthode UMBMark, ce qui prouve aussi que la méthode de contrôle proposée permet au robot de converger rapidement vers les trajectoires de référence et éviter le plus possible les déviations au cours de son mouvement.

Chapitre 5 - Conclusion générale

5.1 Conclusion

La modélisation des équations cinématiques améliorées a permis de décrire le mouvement du robot dans des conditions réelles en tenant compte des erreurs du mouvement comme les glissements qui sont dues à la nature de contact entre le sol et les roues. L'estimation des paramètres cinématiques en utilisant l'algorithme des moindres carrés récursifs RLS a permis de calculer les bonnes commandes pour actionner les roues et par la suite réajuster et optimiser le déplacement le robot afin de suivre son chemin planifié avec plus de précision ce qui nous mène à aboutir à l'objectif de ce projet de recherche, le contrôle du mouvement d'un robot mobile à entraînement différentiel.

Les résultats obtenus dans la validation expérimentale prouvent que l'application de la méthode de contrôle proposée pour plusieurs types de surfaces optimise le suivi des trajectoires du robot et réduit énormément ses erreurs de mouvement en position et en orientation. Les résultats montrent aussi que la méthode proposée est plus robuste que la méthode UMBMark en offrant un bon réajustement du mouvement et une meilleure convergence vers la pose finale. Pour conclure, les résultats satisfaisants obtenus dans ce projet de recherche prouvent que l'application de la méthode de contrôle du robot en utilisant l'algorithme des moindres carrés récursifs du mouvement est efficace et performante.

5.2 Perspectives

Ce projet de recherche nous mène à proposer deux futures études :

- Tester la méthode de contrôle du mouvement sur des surfaces déformables et/ou glissantes comme du sable ou de la glace.
- Tester la performance de la même méthode en exploitant plusieurs techniques de localisation du robot simultanément.

Références

[1] BostonDynamics. Available: https://www.bostondynamics.com/

[2] RobotShop. Available: https://www.robotshop.com/blog/ja/how-to-make-a-robot-lesson-3-actuators-2-3703

[3] Starship. Available: https://www.starship.xyz/

[4] Tesla. Available: https://www.tesla.com

[5] P. P.-D. m. robot. Available: https://www.generationrobots.com/en/402395-robot-mobile-pioneer-3-dx.html

[6] Robotix. Available: https://2019.robotix.in/tutorial/mechanical/drivemechtut/

[7] Blickle. Available: http://catalog.blickle.ca/?startpage=344&_ga=2.36881248.1583658158.1559697589-475904697.1559697589

[8] C. Samson and K. Ait-Abderrahim, "Feedback control of a nonholonomic wheeled cart in cartesian space," in *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, 1991, pp. 1136-1141: IEEE.

[9] B. d'Andréa-Novel, G. Campion, and G. Bastin, "Control of nonholonomic wheeled mobile robots by state feedback linearization," *The International journal of robotics research,* vol. 14, no. 6, pp. 543-559, 1995.

[10] A. De Luca and M. D. Di Benedetto, "Control of nonholonomic systems via dynamic compensation," *Kybernetika,* vol. 29, no. 6, pp. 593-608, 1993.

[11] G. Klancar, D. Matko, and S. Blazic, "Mobile robot control on a reference path," in *Proceedings of the 2005 IEEE International Symposium on, Mediterrean Conference on Control and Automation Intelligent Control, 2005.*, 2005, pp. 1343-1348: IEEE.

[12] S. He, "Feedback control design of differential-drive wheeled mobile robots," in *ICAR'05. Proceedings., 12th International Conference on Advanced Robotics, 2005.*, 2005, pp. 135-140: IEEE.

[13] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010.

[14] A. Ollero and O. Amidi, "Predictive path tracking of mobile robots. Application to the CMU Navlab," in *Proceedings of 5th International Conference on Advanced Robotics, Robots in Unstructured Environments, ICAR*, 1991, vol. 91, pp. 1081-1086.

[15] J. E. Normey-Rico, J. Gómez-Ortega, and E. F. Camacho, "A Smith-predictor-based generalised predictive controller for mobile robot path-tracking," *Control Engineering Practice,* vol. 7, no. 6, pp. 729-740, 1999.

[16] G. Klančar and I. Škrjanc, "Tracking-error model-based predictive control for mobile robots in real time," *Robotics and autonomous systems,* vol. 55, no. 6, pp. 460-469, 2007.

[17] Y. Kanayama, Y. Kimura, F. Miyazaki, and T. Noguchi, "A stable tracking control method for an autonomous mobile robot," in *Proceedings., IEEE International Conference on Robotics and Automation*, 1990, pp. 384-389: IEEE.

[18] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to autonomous mobile robots*. MIT press, 2011.

[19] P. K. Padhy, T. Sasaki, S. Nakamura, and H. Hashimoto, "Modeling and position control of mobile robot," in *2010 11th IEEE International Workshop on Advanced Motion Control (AMC)*, 2010, pp. 100-105.

[20] A. D. Araujo, P. J. Alsina, and S. M. Dias, "Nonholonomic wheeled mobile robot positioning controller using decoupling and variable structure model reference adaptive control," in *2006 American Control Conference*, 2006, p. 6 pp.: IEEE.

[21] F. C. Vieira, A. A. Medeiros, P. J. Alsina, and A. P. A. Jr, "Position and orientation control of a two-wheeled differentially driven nonholonomic mobile robot," in *the International Conference on Informatics in Control*, Setúbal (Portugal), 2004, pp. 256-262.

[22] O. Trigui, "Méthode de gestion énergétique d'un véhicule électrique basée sur l'estimation en ligne de la masse et de coéfficient de résistance au roulement," Université du Québec à Trois-Rivières, 2017.

[23] N. Ouasli, R. B. Mehrez, and L. El Amraoui, "Parameter estimation of one wheel vehicle using nonlinear observer," in *2014 International Conference on Electrical Sciences and Technologies in Maghreb (CISTEM)*, 2014, pp. 1-8: IEEE.

[24] L. R. Ray, "Nonlinear tire force estimation and road friction identification: Simulation and experiments," *Automatica,* vol. 33, no. 10, pp. 1819-1833, 1997.

[25] J. Dakhlallah, S. Glaser, S. Mammar, and Y. Sebsadji, "Tire-road forces estimation using extended Kalman filter and sideslip angle evaluation," in *2008 American control conference*, 2008, pp. 4597-4602: IEEE.

[26] F. Gustafsson, "Slip-based tire-road friction estimation," *Automatica,* vol. 33, no. 6, pp. 1087-1099, 1997.

[27] S. Müller, M. Uchanski, and K. Hedrick, "Slip-based tire-road friction estimation during braking," in *Proceedings of 2001 ASME International Mechanical Engineering Congress and Exposition*, 2001, pp. 213-220.

[28] J. Wang, L. Alexander, and R. Rajamani, "Friction estimation on highway vehicles using longitudinal measurements," *Journal of dynamic systems, measurement, and control,* vol. 126, no. 2, pp. 265-275, 2004.

[29] R. Rajamani, G. Phanomchoeng, D. Piyabongkarn, and J. Y. Lew, "Algorithms for real-time estimation of individual wheel tire-road friction coefficients," *IEEE/ASME Transactions on Mechatronics,* vol. 17, no. 6, pp. 1183-1195, 2012.

[30] B. Li, H. Du, and W. Li, "A novel cost effective method for vehicle tire-road friction coefficient estimation," in *2013 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 2013, pp. 1528-1533: IEEE.

[31] A. De Luca, G. Oriolo, and M. Vendittelli, "Control of wheeled mobile robots: An experimental overview," in *Ramsete*: Springer, 2001, pp. 181-226.

[32] G. Oriolo, A. De Luca, and M. Vendittelli, "WMR control via dynamic feedback linearization: design, implementation, and experimental validation," *IEEE Transactions on control systems technology,* vol. 10, no. 6, pp. 835-852, 2002.

[33] R. Fierro and F. L. Lewis, "Control of a nonholomic mobile robot: Backstepping kinematics into dynamics," *Journal of robotic systems,* vol. 14, no. 3, pp. 149-163, 1997.

[34] S. Nandy, S. N. Shome, G. Chakraborty, and C. S. Kumar, "A modular approach to detailed dynamic formulation and control of wheeled mobile robot," in *2011 IEEE International Conference on Mechatronics and Automation*, 2011, pp. 1471-1478: IEEE.

[35] D. Aneesh, "Tracking controller of mobile robot," in *2012 International Conference on Computing, Electronics and Electrical Technologies (ICCEET)*, 2012, pp. 343-349: IEEE.

[36] J. Borenstein and L. Feng, "Measurement and correction of systematic odometry errors in mobile robots," *IEEE Transactions on robotics and automation,* vol. 12, no. 6, pp. 869-880, 1996.

[37] Z. Kachour, S. Kelouwani, Y. Dube, and K. Agbossou, "Low Speed Electric Vehicle Trajectory Following with Variable Rolling Resistance," in *2017 IEEE Vehicle Power and Propulsion Conference (VPPC)*, 2017, pp. 1-5: IEEE.

[38] S. S. Haykin, "Adaptive filter theory," Pearson Education India, 2005, pp. 436-465.

[39] S. S. Haykin, "Adaptive filter theory," Pearson Education India, 2005, pp. 231-319.

[40] J. J. Gray, "Carl Friedrich Gauss," in *Encyclopædia Britannica*, ed: Encyclopædia Britannica, inc., 2019.

[41] R. L. Plackett, "Some Theorems in Least Squares," *Biometrika,* vol. 37, no. 1/2, pp. 149-157, 1950.

[42] *Dr Robot*. Available: http://www.drrobot.com/products_item.asp?itemNumber=X80Pro

[43] *Marvelmind Robotics*. Available: https://marvelmind.com/product/starter-set-hw-v4-9-plastic-housing/

[44] *Parallax Inc.* Available: https://www.parallax.com/product/29133

Publications

Low speed electric vehicle trajectory following with variable rolling resistance

Zayd Kachour, Sousso Kelouwani, Yves Dubé, Kodjo Agbossou
Hydrogen Research Institute (IRH)
Université du Québec à Trois-Rivières QC G9A 5H7
Email: {Zayd.Kachour, Sousso.Kelouwani, Yves.Dube, Kodjo.Agbossou}@uqtr.ca

***Abstract*— This paper investigates the trajectory following of a low-speed electric vehicle under varying tire/road rolling resistance. In particular, when an electric mobile platform is moving in a warehouse, the rolling surface characteristics may vary due to the different type of floors and material used in the warehouse. Hence, the conventional kinematic model (with fixed parameters) is not valid. Through an adaptive estimation of the kinematic parameters, a new approach that allows the low-speed electric vehicle to follow a trajectory whilst minimizing the kinematic energy, despite the complexity of the rolling surface characteristic, is discussed. Preliminary validations in simulating suggest that this approach is effective whilst reducing in the same time among of motion energy.**

Index Terms —Adaptive kinematic control; Path planning; Electric autonomous vehicles; Rolling resistance; Vehicle dynamics

I. INTRODUCTION

The next generation of industry (Industry 4.0) will require efficient and effective intra-warehouse mobility solutions. Autonomous mobility and secure motion are the two key performances that are being tackled to achieve this ambitious goal. Fortunately, these two goals are complementary and the same sensing/interaction infrastructure can be used. To improve the energy efficiency, several manufacturers are using low-speed electric vehicle (electric forklift, electric pallet truck, etc.) [1]. However, it is well known that at low-speed, one of the most important energy loss is due to the rolling resistance. Modern warehouses and manufacturing buildings involve different type of rolling surfaces, each with its own characteristics. These rolling surfaces affect directly the vehicle kinetic energy. In particular, on slippery surfaces, an intelligent motion control should be used to allow the vehicle to follow the desired trajectory whilst minimizing the kinetic energy. To control the vehicle motion, two steps are minimally involved: (i) estimation in real-time of the vehicle position and (ii) selection of a feedback control strategy that permits the vehicle to follow the desired trajectory [2].

The most basic and commonly used approach for estimate the vehicle position during a motion is the odometry estimation. Odometry is a basic technic for estimating the position of a moving vehicle using the onboard proprioceptive sensors that measure the movement of the vehicle wheels. The physical principles used for position estimation are based on fixed values of motion parameters (wheels physical and geometrical characteristics, wheelbase distance, position of the center of gravity, etc.). However, these parameter values are not known precisely. Even if they were known, the interaction between the wheels and the floor may induce different and unpredictable motion behavior. Therefore, it is important and challenging to develop a new kinematic control approach.

In the literature, some authors have been specialized in the research of corrective models to control the movement of autonomous vehicles. Studies on the calibration of odometry have been carried out for several years. Hence, the UMBmark method [3] is a useful calibration technique for a two-wheel differential low-speed vehicle. The wheel diameter error and the wheelbase error are estimated by measuring the errors of the final position after performing a motion in the navigation environment. During the test, the low-speed vehicle is driven along a square of 4 meters in length clockwise (CW) and counterclockwise (CCW). The UMBmark method [3] assumes that the diameter error and the wheelbase error are independent. In practice, this assumption is not valid because both errors occur simultaneously. In addition, an approximation with the small angle hypothesis has been adopted during the estimation process in order to simplify them.

Lee [4] proposed an odometry calibration scheme for a differential drive low-speed vehicle that considers the interaction of wheel diameter errors and wheelbase. Therefore, the consideration of the coupled effect of two sources of error improves the accuracy of the calibration.

Jung [5, 6] proposed a calibration scheme for a robot with a differential drive which consists of measuring the final orientation error after the test. Using these orientation error measurements, approximation errors are eliminated and the calibration accuracy is improved.

Most of the reported works did not tackle the challenging problem of position estimation under stochastics rolling resistance condition due to unpredictable wheel/road interaction on different rolling resistances. In this paper, the methods in [7, 8, 9] is used and extended in order to handle all type errors, including stochastics phenomena which affect the motion.

As a first step, the kinematic equations are rewritten and the linearity between the unknown parameters and the observations is shown. This approach naturally leads to an online linear identification problem that allows application of the online least-squares (RLS) method.

The rest of the paper is organized as follows. Section II presents the problem formulation and the different models. In section III, an online and real-time kinematic parameter identification approach is proposed. In section IV, we show some preliminary results, whereas the conclusion is presented in section V.

II. MODELLING

A differential-drive mobile platform as shown in Fig.1 is considered. This small-scale robotic platform allows us to easy test our methods for low speed electric vehicle.

Fig. 1. Low-speed electric vehicle using a robotic mobile platform [10].

The linear and angular velocity of the robot V and ω are calculated, respectively, by the two following formulas:

$$V = \frac{r_R\dot{\phi}_R + r_L\dot{\phi}_L}{2} \quad (1)$$

$$\omega = \frac{r_R\dot{\phi}_R - r_L\dot{\phi}_L}{b} \quad (2)$$

where
$\dot{\phi}_R$: the angular velocity of the right wheel
$\dot{\phi}_L$: the angular velocity of the left wheel
r_R : the radius of the right wheel
r_L : the radius of the left wheel
b : the distance between the two wheels

Consequently, the kinematic equations of the platform can be written as follows by:

$$\dot{x} = V\cos\theta \quad [5]$$
$$\dot{y} = V\sin\theta \quad (4)$$
$$\omega = \dot{\theta} \quad (5)$$

where $\dot{x}, \dot{y}$ are derivative coordinates of the robot and θ is the heading angle.

The robot pose can be implemented by discrete-time integration of [5].

$$x_k = x_{k-1} + \Delta t V_{k-1}\cos(\theta_{k-1} + \Delta t\omega_{k-1}/2) \quad (6)$$
$$y_k = y_{k-1} + \Delta t V_{k-1}\sin(\theta_{k-1} + \Delta t\omega_{k-1}/2) \quad (7)$$
$$\theta_k = \theta_{k-1} + \Delta t\omega_{k-1} \quad (8)$$

where the index k represent the kth time sample and Δt denote the sampling period.

III. PROPOSED MODEL FOR POSE ESTIMATION WITH PARAMETER IDENTIFICATION

To estimate the kinematics parameters of the vehicle, an Online Recursive Least Squares is used. As proposed in [9], let us rewrite and replace V_k and ω_k in (6), (7) and (8).

$$\Delta x_k = \Delta t\frac{r_R\dot{\phi}_{R,k-1} + r_L\dot{\phi}_{L,k-1}}{2}\cos(\theta_{k-1} + \Delta\theta_k/2) \quad (9)$$

$$\Delta y_k = \Delta t\frac{r_R\dot{\phi}_{R,k-1} + r_L\dot{\phi}_{L,k-1}}{2}\sin(\theta_{k-1} + \Delta\theta_k/2) \quad (10)$$

$$\Delta\theta_k = \Delta t\frac{r_R\dot{\phi}_{R,k-1} - r_L\dot{\phi}_{L,k-1}}{b} \quad (11)$$

where

$$\Delta x_k = x_k - x_{k-1},\ \Delta y_k = y_k - y_{k-1},\ \Delta\theta_k = \theta_k - \theta_{k-1}$$

(9), (10) and (11) can be rewritten as follows:

$$\Delta x_k = \frac{r_R}{2}\Delta t\dot{\phi}_{R,k-1}\cos(\theta_{k-1} + \Delta\theta_k/2) + \frac{r_L}{2}\Delta t\dot{\phi}_{L,k-1}\cos(\theta_{k-1} + \Delta\theta_k/2) \quad (12)$$

$$\Delta y_k = \frac{r_R}{2}\Delta t\dot{\phi}_{R,k-1}\sin(\theta_{k-1} + \Delta\theta_k/2) + \frac{r_L}{2}\Delta t\dot{\phi}_{L,k-1}\sin(\theta_{k-1} + \Delta\theta_k/2) \quad (13)$$

$$\Delta\theta_k = \frac{r_R}{b}\Delta t\dot{\phi}_{R,k-1} - \frac{r_L}{b}\Delta t\dot{\phi}_{L,k-1} \quad (14)$$

Let us define the vectors $C_1, C_2, \Phi_{x,k}, \Phi_{y,k}$ and $\Phi_{\theta,k}$:

$$C_1 = \begin{bmatrix} \frac{r_R}{2} \\ \frac{r_L}{2} \end{bmatrix}, \quad C_2 = \begin{bmatrix} \frac{r_R}{b} \\ -\frac{r_L}{b} \end{bmatrix}$$

$$\Phi_{x,k} = \begin{bmatrix} \Delta t\dot{\phi}_{R,k-1} cos(\theta_{k-1} + \Delta\theta_k/2) & \Delta t\dot{\phi}_{L,k-1} cos(\theta_{k-1} + \Delta\theta_k/2) \end{bmatrix}^T$$

$$\Phi_{y,k} = \begin{bmatrix} \Delta t\dot{\phi}_{R,k-1} \sin(\theta_{k-1} + \Delta\theta_k/2) & \Delta t\dot{\phi}_{L,k-1} \sin(\theta_{k-1} + \Delta\theta_k/2) \end{bmatrix}^T$$

$$\Phi_{\theta,k} = \begin{bmatrix} \Delta t\dot{\phi}_{R,k-1} & \Delta t\dot{\phi}_{L,k-1} \end{bmatrix}^T$$

(12), (13) and (14) can be rewritten in a compact form:

$$\Delta x_k = \Phi_x C_1 \quad (15)$$
$$\Delta y_k = \Phi_y C_1 \quad (16)$$
$$\Delta\theta_k = \Phi_\theta C_2 \quad (17)$$

where C_1 and C_2 contain the parameters to estimate later using the RLS method and Φ_x, Φ_y and Φ_θ are the regressors of (12), (13) and (14).

Let us consider a path that contains n+1 navigation points. The motion controller must identify the kinematic parameters (C_1 and C_2) in order to let the platform follow these n+1 navigation points. Therefore, a desired path with these n+1 navigation points $(x_d(k), y_d(k)), k = 0,1,2, \ldots, n$ must be followed.

First of all C_1, C_2 and the inverse of correlation matrix P_x and P_θ are initialized.

Let us mention too that the indexes d, m and avg m denotes desired, measured and average, respectively.

Then a series of equation in the online RLS algorithm is used to estimate the parameters C_1 and C_2 and to control the movement of the vehicle over time.

The following algorithm is used:

Initialization:

$$\text{Desired Path} = \begin{pmatrix} x_d(0) & \cdots & x_d(n) \\ y_d(0) & \cdots & y_d(n) \end{pmatrix}$$

$$\hat{C}1(0) = \begin{bmatrix} \frac{r_{R,avg}}{2} \\ \frac{r_{L,avg}}{2} \end{bmatrix}, \quad \hat{C}2(0) = \begin{bmatrix} \frac{r_{R,avg}}{b_{avg}} \\ -\frac{r_{L,avg}}{b_{avg}} \end{bmatrix},$$

$$P_x(0) = P_\theta(0) = \delta^{-1} I$$

$$x_m(0) = x_d(0)$$
$$y_m(0) = y_d(0)$$

For each instant of time, k = 1, 2, …n, compute

$$\Delta x_d(k) = x_d(k) - x_m(k-1)$$
$$\Delta y_d(k) = y_d(k) - y_m(k-1)$$

$$(\rho_k, \theta_k) = (\sqrt{\Delta x_d(k)^2 + \Delta y_d(k)^2}, tan^{-1}\frac{\Delta y_d(k)}{\Delta x_d(k)})$$

$$\dot{\phi}_R(k-1) = \frac{\rho_k}{2\Delta t\hat{C}1_1(k-1)}$$

$$\dot{\phi}_L(k-1) = \frac{\rho_k}{2\Delta t\hat{C}1_2(k-1)}$$

$$\Phi_x(k) = \begin{bmatrix} \Delta t\dot{\phi}_R(k-1) cos\,\theta_k & \Delta t\dot{\phi}_L(k-1) cos\,\theta_k \end{bmatrix}^T$$

$$\Phi_y(k) = \begin{bmatrix} \Delta t\dot{\phi}_R(k-1) sin\,\theta_k & \Delta t\dot{\phi}_L(k-1) sin\,\theta_k \end{bmatrix}^T$$

$$\Phi_\theta(k) = \begin{bmatrix} \Delta t\dot{\phi}_R(k-1) & \Delta t\dot{\phi}_L(k-1) \end{bmatrix}^T$$

In practice, there are some uncertainties on the platform pose. Without losing generality, let assume the following representation of these uncertainties.

$$A_{x,k} = random([\frac{-\rho_k}{10}; \frac{\rho_k}{10}])$$

$$A_{y,k} = random([\frac{-\rho_k}{15}; \frac{\rho_k}{15}])$$

$$A_{\theta,k} = random([\frac{-\rho_k\pi}{20}; \frac{\rho_k\pi}{20}])$$

Where $random([a\,;b])$ is a random number from a uniform distribution density function. Therefore the error for each quantity is:

$$Err_x(k) = A_{x,k}$$
$$Err_y(k) = A_{y,k}$$
$$Err_\theta(k) = A_{\theta,k}$$

$$\Delta x_m(k) = \Delta x_d(k) + Err_x(k)e^{-\frac{k}{\alpha}}$$
$$\Delta y_m(k) = \Delta y_d(k) + Err_y(k)e^{-\frac{k}{\alpha}}$$
$$\Delta\theta_m(k) = Err_\theta(k)e^{-\frac{k}{\alpha}}$$

$$\pi_x(k) = P_x(k-1)\Phi_x(k)$$
$$\pi_\theta(k) = P_\theta(k-1)\Phi_\theta(k)$$

$$G_x(k) = \frac{\pi_x(k)}{\lambda + \Phi_x(k)^T \pi_x(k)}$$

$$G_\theta(k) = \frac{\pi_\theta(k)}{\lambda + \Phi_\theta(k)^T \pi_\theta(k)}$$

$$\xi_x(k) = \Delta x_m(k) - \hat{C}1(k-1)\Phi_x(k)$$
$$\xi_\theta(k) = \Delta\theta_m(k) - \hat{C}2(k-1)\Phi_\theta(k)$$

$$\hat{C}1(k) = \hat{C}1(k-1) + G_x(k)\xi_x(k)$$
$$\hat{C}2(k) = \hat{C}2(k-1) + G_\theta(k)\xi_\theta(k)$$

$$P_x(k) = \lambda^{-1}P_x(k-1) - \lambda^{-1}G_x(k)\Phi_x(k)^T P_x(k-1)$$
$$P_\theta(k) = \lambda^{-1}P_\theta(k-1) - \lambda^{-1}G_\theta(k)\Phi_\theta(k)^T P_\theta(k-1)$$

$$x_m(k) = x_m(k-1) + \Delta x_m(k)$$
$$y_m(k) = y_m(k-1) + \Delta y_m(k)$$

Knowing the platform pose $(x_m(k), y_m(k))$, given the next desired point $(x_d(k+1), y_d(k+1))$, the kinematic controller compute the values of corresponding linear and angular velocities $V(k)$ and $\omega(k)$, respectively, that will permit de platform to reach the next point from the current estimated pose.

IV. Validation in Simulation

The goal of the simulation is to validate the provided online trajectory following method in presence of varying rolling resistances. These rolling resistances impacting the platform kinematic are represented by random values added to the reference trajectory points. Fig. 1 shows a reference frame including the desired trajectory with reference via points (points situated directly on the desired trajectory). Starting from the initial point at (0,0) in the reference frame of Fig. 1, the controller must let the platform to move closely to the desired trajectory, despite the impact of stochastic phenomena. The last point of the trajectory is (0,5) in the reference frame.

The RLS and the control system are simulated using selected parameters in TABLE I.

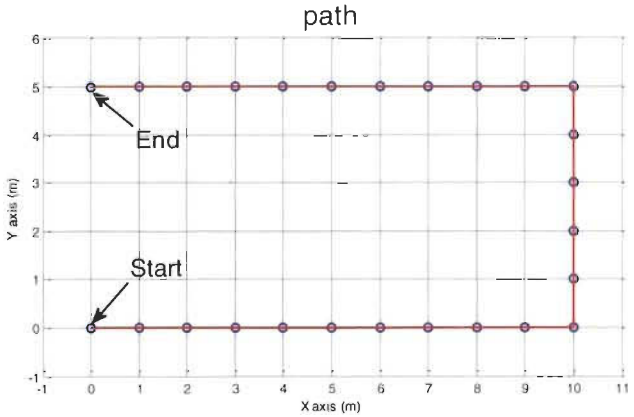


Fig. 1. Reference Path.

TABLE I. RLS AND INITIAL KINEMATICS PARAMETERS

Parameters	Values
$r_{R,avg}$	0.0825 m
$r_{L,avg}$	0.0825 m
b_{avg}	0.287 m
Δt	1s
δ	1
λ	1

Fig. 2 shows the results when the conventional kinematic controller is used (no adaptive control and no parameter identification). On can observe that the platform is unable to follow the desired trajectory (see Fig. 3).

If the proposed online parameter is used with the kinematic controller, the result is much better as shown in Fig. 4. We can observe that at the begin, the error on each axis is high and as the platform is moving the identification parameter process helps the controller to adapt its command in order to achieve a move with a path close to the desired one. The evaluation in detail of the dynamic of the involves errors indicates that they are converging toward an equilibrium point of 0 (see Fig. 5).

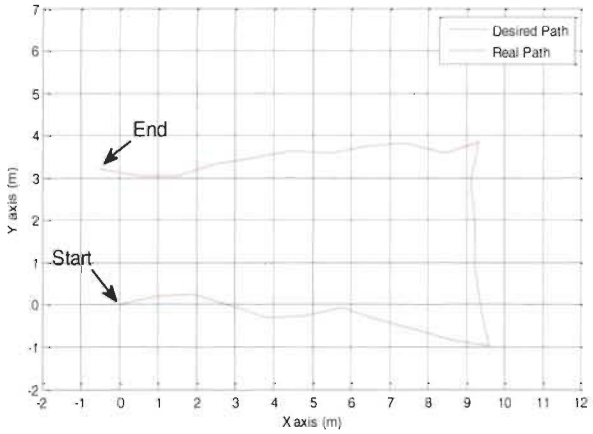


Fig. 2: Comparison between desired path and the path without the adaptive controller (real path).

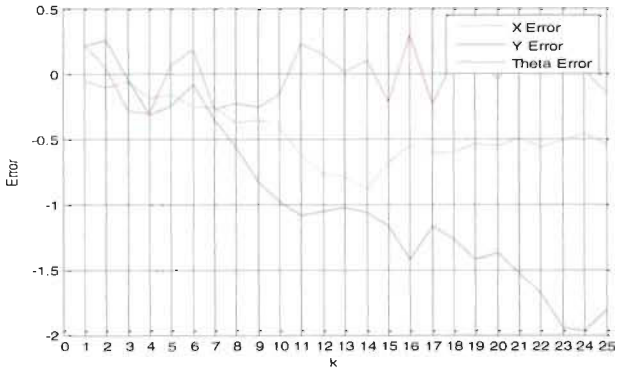


Fig. 3: Comparison of different errors when no online parameter identification is used.

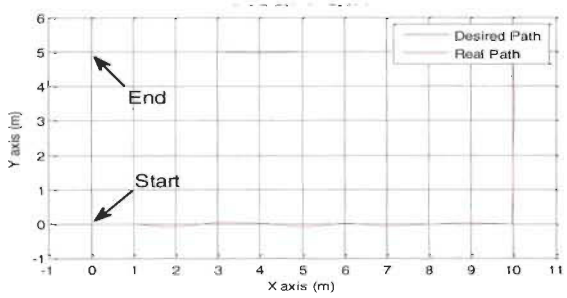


Fig. 4. Comparison between desired path and the path with the adaptive controller (real path).

In addition to providing the capability of moving the platform despite the influence of stochastic tire/road rolling resistance, the proposed approach can improve significantly the energy used. Indeed, Table II presents the comparison of the estimated kinetic energy for the platform show in Fig. 1. Note that for the computation of the kinetic energy, a platform mass of 3.5kg is used.

Table II shows that the kinetic energy with adaptation if very close to the kinetic energy of the desired trajectory. Furthermore, the adaptive controller achieves more than 10% of kinetic energy reduction, compared to the conventional kinematic controller without adaptation.

TABLE II. KINETIC ENERGY COMPARISON USING A PLATFORM MASS OF 3.5kg

Kinetic Energy (J)	
Desired trajectory	1.75
Trajectory without adaptive controller	2.0828
Trajectory with adaptive controller	1.7805

V. CONCLUCION

Due to the interaction between the tire and the navigation surface, the motion of a low-speed vehicle is impacted by different phenomena, which may exhibit some stochastic behaviors. Therefore, the kinematic controller must take into account these perturbations in order to achieve good outcomes. The proposed approach can help the kinematic controller to do so and it is based on the well-known RSL algorithm. The preliminary simulation results indicate that this approach can provide satisfactory path following results while minimizing at the same time the platform kinetic energy. In a future work, this method will be extended to include measurement noises and an experimental validation will be also performed.

ACKNOWLEDGMENT

This work was supported by the Natural Science and Engineering Research Council of Canada and FRQNT.

REFERENCES

[1] François Martel, Sousso Kelouwani, Yves Dubé, Kodjo Agbossou, "Optimal economy-based battery degradation management dynamics for fuel-cell plug-in hybrid electric vehicles", Journal of Power Sources, Volume 274, 15 January 2015, Pages 367-381

[2] Zalzal, Vincent and Gava, Raphael and Kelouwani, Sousso and Cohen, Paul, "Acropolis: A fast prototyping robotic application", International Journal of Advanced Robotic Systems, vol 6 (1), pp. 8, 2009.

[3]. Borenstein, J. and Feng, L., "Measurement and Correction of Systematic Odometry Errors in Mobile Robots," IEEE Transactions on Robotics and Automation, Vol. 12, No. 6, pp. 869-880, 1996.

[4]. Lee, K., Jung, C., and Chung, W., "Accurate Calibration of Kinematic Parameters for Two Wheel Differential Mobile Robots," Journal of Mechanical Science and Technology, Vol. 25, No. 6, pp. 1603-1611, 2011.

[5]. Jung, C. and Chung, W., "Calibration of Kinematic Parameters for Two Wheel Differential Mobile Robots by using Experimental Heading Errors," International Journal of Advanced Robotic Systems, Vol. 8, No. 5, pp. 134-142, 2011.

[6]. Jung, C. and Chung, W., "Accurate Calibration of Two Wheel Differential Mobile Robots by using Experimental Heading Errors," Proc. of IEEE International Conference on Robotics and Automation (ICRA), pp. 4533-4538, 2012.

[7] G. Antonelli, S. Chiaverini, and G. Fusco, "Exciting trajectories for mobile robot odometry calibration," in Preprints 7th IFAC Symp. Robot Control, Wroclaw, Poland, Sep. 2003, pp. 429–434.

[8] , G. Antonelli, S. Chiaverini and G. Fusco, "An odometry calibration method for mobile robots based on the least-squares technique," Proceedings of the 2003 American Control Conference, 2003., 2003, pp. 3429-3434 vol.4.

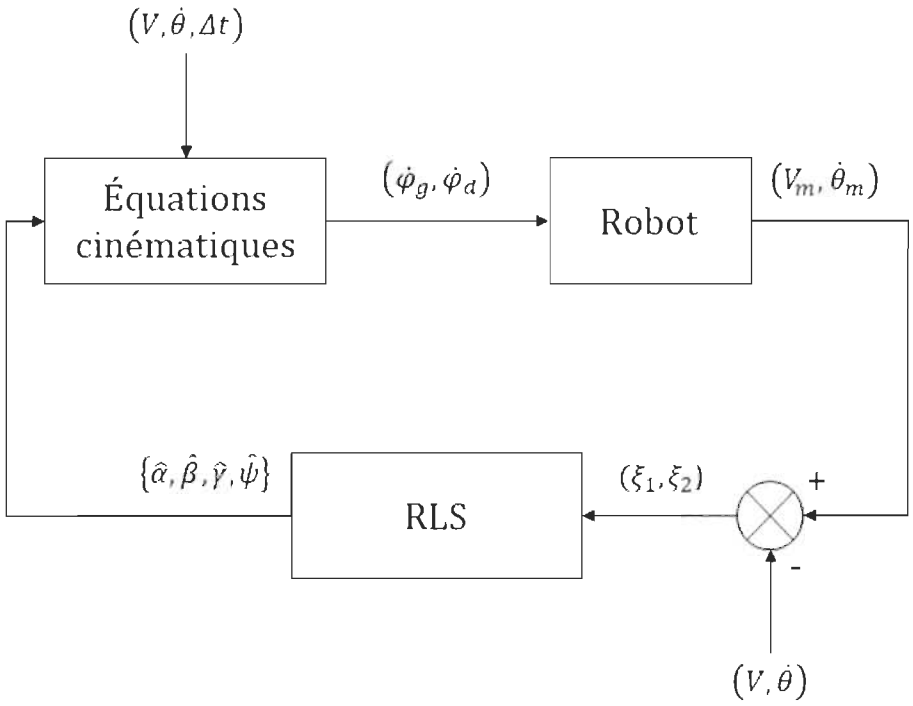
[9] G. Antonelli, S. Chiaverini and G. Fusco, "A calibration method for odometry of mobile robots based on the least-squares technique: theory and experimental validation," in *IEEE Transactions on Robotics*, vol. 21, no. 5, pp. 994-1004, Oct. 2005.

[10] DrRobot, Mobile Robot Development Platform (2017), Retrieved from http://www.drrobot.com/products_item.asp?itemNumber=x80.

Annexe A - Boucle de contrôle

A.1 Schéma fonctionnel

Le schéma fonctionnel du processus de contrôle :



A.1 Algorithme

initialisation :

$$\widehat{w}_1(0) = [\alpha(0), \beta(0)]$$
$$\widehat{w}_2(0) = [\gamma(0), \psi(0)]$$
$$P_1(0) = P_2(0) = \delta^{-1} I$$

calcul de $\dot{\varphi}_d(0)$ *et* $\dot{\varphi}_g(0)$ *en résolvant*

$$\begin{cases} V(0) = \frac{r}{2}\left(\alpha(0)\dot{\varphi}_d(0) + \beta(0)\dot{\varphi}_g(0)\right) \\ \dot{\theta}(0) = \frac{r}{L}\left(\gamma(0)\dot{\varphi}_d(0) - \psi(0)\dot{\varphi}_g(0)\right) \end{cases}$$

Pour $k = 1,2,\dots,N$ *calculer* :

$$u_1(k) = \frac{r}{2}\left[\dot{\varphi}_d(k-1), \dot{\varphi}_g(k-1)\right]^T$$
$$u_2(k) = \frac{r}{L}\left[\dot{\varphi}_d(k-1), \dot{\varphi}_g(k-1)\right]^T$$

$$G_1(k) = \frac{P_1(k-1)u_1(k)}{\lambda + {u_1}^T(k)P_1(k-1)u_1(k)}$$
$$G_2(k) = \frac{P_2(k-1)u_2(k)}{\lambda + {u_2}^T(k)P_2(k-1)u_2(k)}$$

$$\xi_1(k) = V_m(k) - \widehat{w}_1(k-1)u_1(k)$$
$$\xi_2(k) = \dot{\theta}_m(k) - \widehat{w}_2(k-1)u_2(k)$$

$$\widehat{w}_1(k) = \widehat{w}_1(k-1) + G_1(k)\xi_1(k)$$
$$\widehat{w}_2(k) = \widehat{w}_2(k-1) + G_2(k)\xi_2(k)$$

$$P_1(k) = \lambda^{-1}P_1(k-1) - \lambda^{-1}G_1(k){u_1}^T(k)P_1(k-1)$$
$$P_2(k) = \lambda^{-1}P_2(k-1) - \lambda^{-1}G_2(k){u_2}^T(k)P_2(k-1)$$

calcul de $\dot{\varphi}_g(k)$ *et* $\dot{\varphi}_d(k)$ *en résolvant*

$$\begin{cases} V(k) = \frac{r}{2}\left(\hat{\alpha}(k)\dot{\varphi}_d(k) + \hat{\beta}(k)\dot{\varphi}_g(k)\right) \\ \dot{\theta}(k) = \frac{r}{L}\left(\hat{\gamma}(k)\dot{\varphi}_d(k) - \hat{\psi}(k)\dot{\varphi}_g(k)\right) \end{cases}$$

Fin Pour

Fin Algorithme

Annexe B - Fiches techniques

B.1 "Dr Robot X80Pro"

Introduction

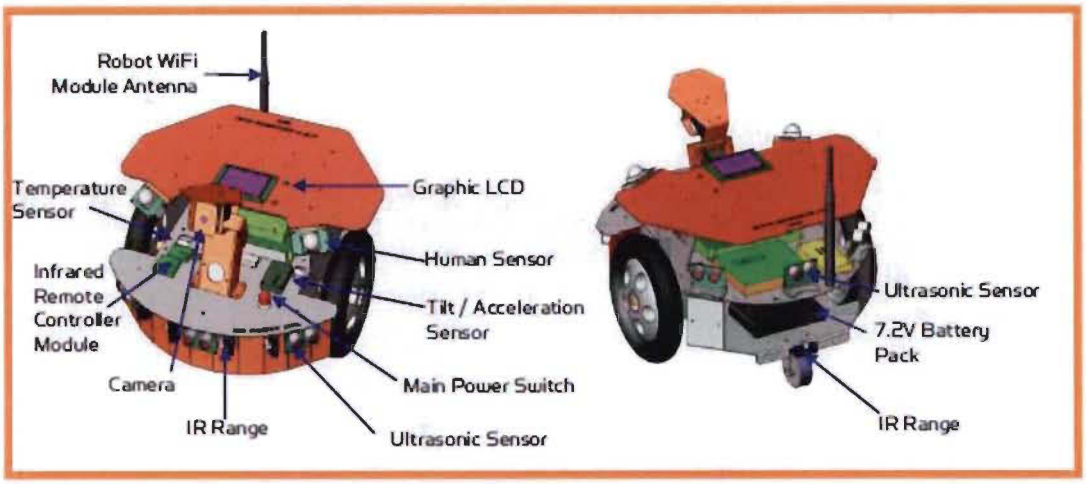
X80Pro is an upgraded version of X80, it has everything X80 has plus extras of 3 more Ultrasonic sensors (DUR5200), Tilting sensor, (DTA5102), Temperature sensor (DAT5280), IR remote control module (MIR5538), IR remote controller (DIR5538), the 128x64 mono graphic display (MGL5128) and two stronger (550oz-in, 40Kg.cm) motors.

Key Features

- *Two 12V motors with over 550oz.-inch(40kg.cm) torque each*
- *Fully wireless networked 802.11g*
- *OS independent application development tools*
- *Max speed of 75 cm/sec*
- *128x64 graphic LCD, Display image , message or sensor data*
- *Collision detection sensors include 6 Ultrasonic range sensors and 7 IR range sensors*
- *2 Pyroelectric human motion sensors*
- *Tilting sensor*
- *Temperature sensor*
- *IR remote control module and IR remote controller*
- *Comprehensive circuit protection*
- *Max payload 15 kg (optional 40 kg) with robot weight of 3.5 kg*
- *Dimension 38cm (L) x 35cm(W) x 28cm (H)*
- *Extended operating time. 3 hours nominal operation time for each recharging.*
- *Upgrade options:*
    - : *Vision-landmark base indoor localization (indoor GPS, position/orientation) sensor and the landmarks together provide precise position and direction information covering every inch of the floor.*
    - : *Laser scanner*
    - : *Power and battery systems for 6 hours operation time are available.*

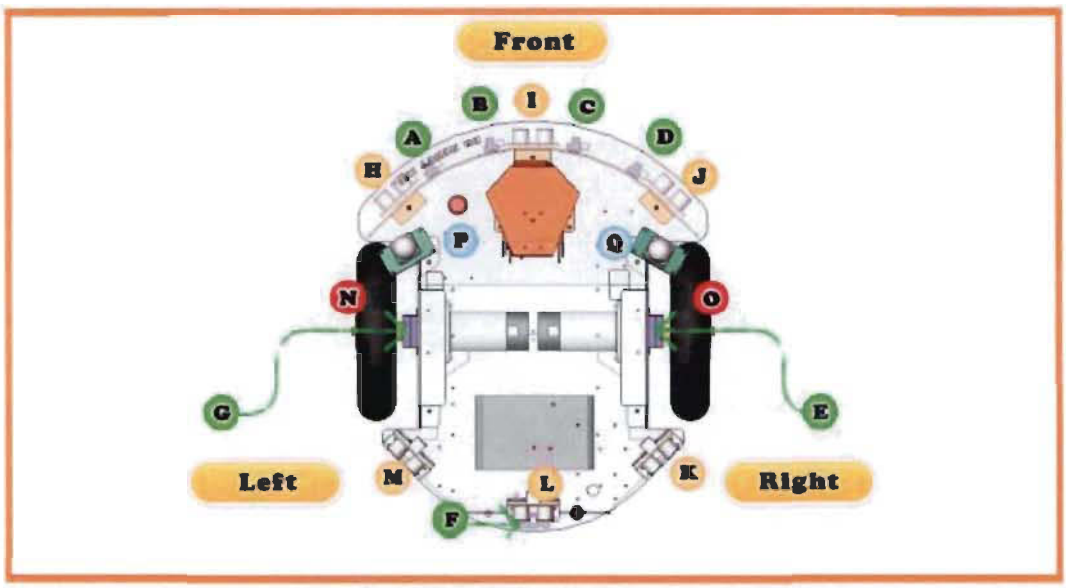
Sensors and External Components

The figure below illustrates the key functional components you will identify on the outside of X80Pro robot.



X80Pro Overview

The robot comes with 6 ultrasonic range sensors and 7 IR range sensors. These range sensors are for environment detection and collision avoidance.



X80Pro Sensor Module Location (Top View)

Sensor Module	Location
Ultrasonic #1	H – Left front
Ultrasonic #2	I – Middle front
Ultrasonic #3	J – Right front
Ultrasonic #4	K – Right middle
Ultrasonic #5	L – Rear
Ultrasonic #6	M – Left middle
Human Sensor #1	P – Left front
Human Sensor #2	Q – Right front
Infrared Range Sensor #1	A – Front left
Infrared Range Sensor #2	B – Front middle
Infrared Range Sensor #3	C – Front middle
Infrared Range Sensor #4	D – Front right
Infrared Range Sensor #5	E – Right
Infrared Range Sensor #6	F – Rear
Infrared Range Sensor #7	G – Left
Quadrature Encoder #1	N - Left , use channel 1
Quadrature Encoder #2	O - Right, use channel 2

B.2 GPS "Marvelmind"

2. Executive summary

Marvelmind Indoor Navigation System is an off-the-shelf indoor navigation system designed to provide precise (±2cm) location data to autonomous robots, vehicles (AGV), and copters. It also used to track other objects that the mobile beacon installed on, for example, virtual reality (VR) systems, helmets for construction workers or miners, etc.

The navigation system based on stationary ultrasonic beacons which united by radio interface in a license-free band. The location where a mobile beacon installed is calculated based on the propagation delay of an ultrasonic signal (Time-Of-Flight or TOF) to a set of stationary ultrasonic beacons using trilateration.

The stationary beacons form the map automatically. No manual entering of coordinates or distance measurement is required. If the stationary beacons not moved, the map built only once and the system is then ready to function after 7–10 seconds after the modem powered

Key requirements proper system functionality

- For 3D (X, Y, Z) tracking - An unobstructed sight by a mobile beacon of three or more stationary beacons simultaneously
- For 2D (X, Y) tracking - An unobstructed sight by a mobile beacon of two stationary or more stationary beacons simultaneously

Distance to the nearest 2 or 3 beacons – not more than 30 meters

Key capabilities:

Parameter	Technical Specifications
Distance between beacons	Reach up to 50 meters in lab conditions. Recommended distance is 30 meters (Transducer4 to Transducer4 looking straight at each other and other transducers are off)
Coverage area	- Reach up to 1000 m^2 with the Starter Set configurations - Coverage for larger territories is similar to cellular networks
Location precision	- Absolute: 1–3% of the distance to the beacons - Differential precision: ±2 cm
Location update rate	- 0.5–45Hz - Can be set manually - Depends on the distance between the mobile and stationary beacons (shorter distance—higher update rate) - Depends on the number of mobile beacons (update rate of 25Hz for 1 mobile beacon, 25Hz/2 for 2 mobile beacons, and 25Hz/3 for 3 mobile beacons) - Depends on the radio interface profile (500kbps vs. 38kbps) - Slightly depends on the number of stationary beacons—different than for mobile beacons
Power supply	Internal: LiPol battery 1000mAh - Battery lifetime depends on usage - Stationary beacon with 16Hz update rate => up to 72h (tested). - Stationary beacon with 1Hz update rate => ~72h*16 => 1 month - Mobile beacon with 8Hz update rate – 12h (tested)
	External: micro USB – recommended for permanent use
Weight	Mobile beacon from starter set: - 59 grams (including battery 1000mAh and housing and antenna 50mm) - 27 grams (bare board w/o battery)
Beacon size	Size: 55x55x33 mm (with 50mm antenna: 55x55x65mm)

B.3 Compas "HMC5883L"

Compass Module 3-Axis HMC5883L

▾ Overview

The Compass Module is designed for low-field magnetic sensing with a digital interface, to provide heading information for your microcontroller project.

This compact sensor fits into small projects such as UAVs and robot navigation systems.
The sensor converts any magnetic field to a differential voltage output on 3 axes. This voltage shift is the raw digital output value, which can then be used to calculate headings or sense magnetic fields coming from different directions. Example code in PBASIC, Spin, and C are provided below.

Key Features:

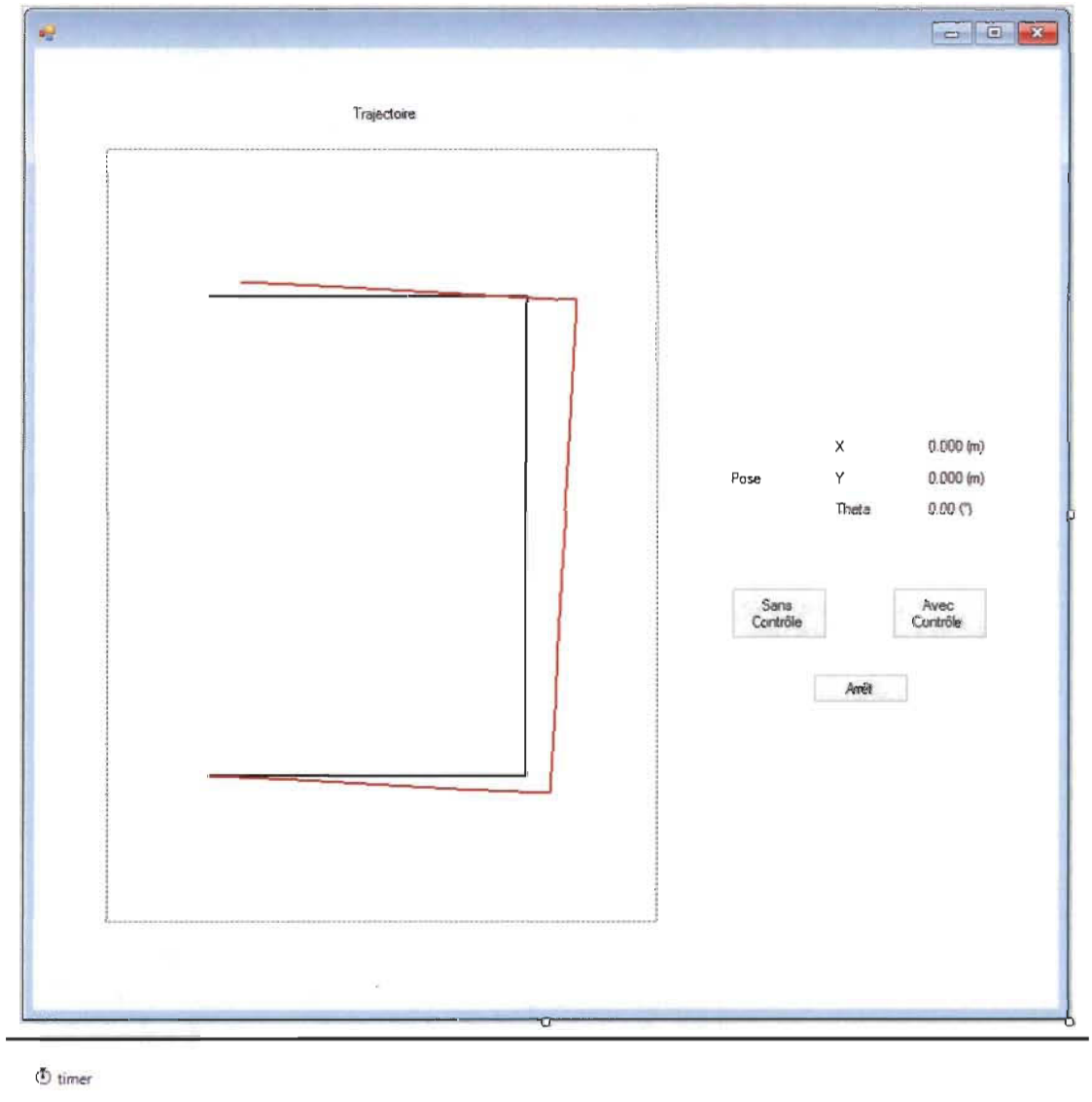
- Measures Earth's magnetic fields
- Precision in-axis sensitivity and linearity
- Designed for use with a large variety of microcontrollers with different voltage requirements

▾ Details

- Wide magnetic field range (+/-8 gauss)
- 1 to 2 degree compass heading accuracy
- Fast 160 Hz maximum output rate
- Power Requirements: 2.7 to 6.5 VDC
- Communication Interface: I2C (up to 400 kHz)
- Dimensions: 0.73 x 0.65 in (1.9 x 1.7 cm)
- Operating temperature: -22 to +185 °F (-30 to +85 °C)

Annexe C - Interface graphique et code C#

C.1 Interface graphique



C.2 Code C#

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Net;
using System.Net.Sockets;
using System.IO.Ports;
using System.IO;
using Excel = Microsoft.Office.Interop.Excel;

namespace Contrôle du Mouvement
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
// Fonctions mathématiques (Calcul des coordonnées polaires et calcul matriciel)
private double[] CoordonnéesPolaires(double Point1x, double Point1y, double Point2x, double Point2y)
private double[,] MultiplierScalaire(double scalaire, double[,] Vecteur)
private double[,] TransposéVecteur(double[,] Vecteur)
private double[,] SommeVecteurs(double[,] Vecteur1, double[,] Vecteur2)
private double MultiplierVecteurs(double[,] Vecteur1, double[,] Vecteur2)
private double[,] SoustraireMatrice(double[,] Matrice1, double[,] Matrice2)
private double[,] MultiplierMatrices(double[,] Matrice1, double[,] Matrice2)
// Fonctions pour l'acquisition des données du GPS et du Compas
            private void OuvrirPorts()
            private void GPS()
            private void Compas()
// Fonction pour le calcul des commandes du robot
private void CalculCommandes(double V, double ThetaV)
{
PhiVg = (2*W2[1, 2]*V + L*W1[1, 2]*ThetaV) / (r*(W1[1, 1]*W2[1, 2] + W1[1, 2]*W2[1, 1]));
PhiVd = (1 / W2[1, 2]) * (W2[1, 1] * PhiVg - ThetaV * L / r);
}
```

```
// Fonctions d'envoi des commandes et déplacement du robot
private double CalculRotation(double Point1x, double Point1y, double Point2x, double Point2y, double
orientation)
private void ArrêterRobot()
private async void RotationRobot(double PhiVg, double PhiVd, short DeltaT)
private async void TranslationRobot(double PhiVg, double PhiVd, short DeltaT)
private byte CalculateCRC(byte[] lpBuffer, int nSize)
// Fonction de la navigation sans contrôle
private void SansContrôle()
{
double Rotation, Translation;
short DeltaT;

        for (int k = 1; k < Tréf.GetLength(0); k++)
        {
             // Mouvement de Rotation
             // Calcul de l'angle de rotation
             Rotation = CalculRotation(Tréf[k-1, 0],Tréf[k-1, 1],Tréf[k, 0],Tréf[k, 1],Tréf[k-1,1]);
                   if (Rotation != 0)
                   {
                       // Calcul de ΔT
                       DeltaT = (short)(Math.Abs(Rotation) / ThetaVr);
                       // Calcul des Commandes (Vitesses angulaires des roues)
                       CalculCommandes(0, Math.Sign(Rotation) * ThetaVr);
                       // Envoi des commandes de la rotation
                       RotationRobot(PhiVg, PhiVd, DeltaT);
                   }
                   // Mouvement de Translation
                   // Calcul de la distance
              Translation = CoordonnéesPolaires(Tréf[k-1,0], Tréf[k-1,1], Tréf[k,0], Tréf[k,1])[0];
                   //
                   DeltaT = (short)(Translation / Vr);
                   CalculCommandes(Vr, 0);
                   // Envoi des commandes de la translation
                   TranslationRobot(PhiVg, PhiVd, DeltaT);
                   // Enregistrement des mesures
                   Données[k, 0] = xGPS;
                   Données[k, 1] = yGPS;
                   Données[k, 2] = Theta;
         }
}
```

```
// Fonction de l'estimation des paramètres cinématiques (Recursive Least Squares)
private void RLS(double PhiVg, double PhiVd, double Vm, double ThetaVm)
{
                // initialisation des paramètres et des variables
                double Sigma = 0.01;
                double[,] P1 = new double[2, 2] { { 1 / Sigma, 0 }, { 0, 1 / Sigma } };
                double[,] P2 = new double[2, 2] { { 1 / Sigma, 0 }, { 0, 1 / Sigma } };
                double[,] u1 = new double[2, 1];
                double[,] u2 = new double[2, 1];
                double Lambda = 0.99;
                double[,] G1 = new double[2, 1];
                double[,] G2 = new double[2, 1];
                double E1;
                double E2;
                // Calcul des vecteurs de régression u1 et u2
                u1[0, 1] = 0.5 * r * PhiVg;
                u1[1, 1] = 0.5 * r * PhiVd;
                u2[0, 1] = r / L * PhiVg;
                u2[1, 1] = r / L * PhiVd;
                // Calcul des gains G1 et G2
                G1 = MultiplierScalaire(1 / (Lambda +
MultiplierVecteurs(MultiplierMatrices(TransposéVecteur(u1), P1), u1)), MultiplierMatrices(P1, u1));
                G2 = MultiplierScalaire(1 / (Lambda +
MultiplierVecteurs(MultiplierMatrices(TransposéVecteur(u2), P2), u2)), MultiplierMatrices(P2, u2));
                // Calcul des erreurs E1 et E2
                E1 = Vm - MultiplierVecteurs(W1, u1);
                E2 = ThetaVm - MultiplierVecteurs(W2, u2);
                // Mise à jour des vecteurs de paramètre W1 et W2
                W1 = SommeVecteurs(W1, MultiplierScalaire(E1, G1));
                W2 = SommeVecteurs(W2, MultiplierScalaire(E2, G2));
                // Mise à jour des matrices de covariance P1 et P2
                P1 = MultiplierScalaire(1 / Lambda, SoustraireMatrice(P1,
MultiplierMatrices(MultiplierMatrices(G1, TransposéVecteur(u1)), P1)));
                P2 = MultiplierScalaire(1 / Lambda, SoustraireMatrice(P2,
MultiplierMatrices(MultiplierMatrices(G2, TransposéVecteur(u2)), P2)));
}
```

```
// Fonction de la navigation avec la méthode de contrôle
private void AvecContrôle()
{
                double Rotation, Translation, distM, Vm, ThetaVm, deltaThetaM;
                double a = 0;
                double b = 0;
                double c = 0;
                short DeltaT;

                for (int k = 1; k < Tréf.GetLength(0); k++)
                {
                    // Mouvement de Rotation
                    // Calcul de l'angle de rotation
                    Rotation = CalculRotation(xGPS, yGPS, Tréf[k, 0], Tréf[k, 1], Theta);
                    if (Rotation != 0)
                    {
                        // Calcul de ΔT
                        DeltaT = (short)(Math.Abs(Rotation) / ThetaVr);
                        // Calcul des Commandes (Vitesses angulaires des roues)
                        CalculCommandes(0, Math.Sign(Rotation) * ThetaVr);
                        // Envoi des commandes de la rotation
                        RotationRobot(PhiVg, PhiVd, DeltaT);
                        // Mesure des vitesses linéaire et angulaire
                        distM = CoordonnéesPolaires(Données[k-1,0],Données[k-1,1], xGPS, yGPS)[0];
                        deltaThetaM = Theta - Données[k - 1, 2];
                        Vm = distM / DeltaT;
                        ThetaVm = deltaThetaM / DeltaT;
                        a = xGPS;
                        b = yGPS;
                        c = Theta;
                        // Estimation des paramètres cinématiques
                        RLS(PhiVg, PhiVd, Vm, ThetaVm);
                    }
                    // Mouvement de Translation
                    // Calcul de la distance
                    Translation = CoordonnéesPolaires(xGPS, yGPS, Tréf[k, 0], Tréf[k, 1])[0];
                    //
                    DeltaT = (short)(Translation / Vr);
                    CalculCommandes(Vr, 0);
```

```
            // Envoi des commandes de la translation
            TranslationRobot(PhiVg, PhiVd, DeltaT);
            //
            distM = CoordonnéesPolaires(a, b, xGPS, yGPS)[0];
            deltaThetaM = Theta - c;
            Vm = distM / DeltaT;
            ThetaVm = deltaThetaM / DeltaT;

            RLS(PhiVg, PhiVd, Vm, ThetaVm);
            // Enregistrement des mesures
            Données[k, 0] = xGPS;
            Données[k, 1] = yGPS;
            Données[k, 2] = Theta;
        }
    }
    // Fonction de l'enregistrement des mesures de position et d'orientation
    private void EnregistrementDonnées()
    // Fonction de traçage des trajectoires (référence et réelle)
    private void TracerTrajectoire()

//_____________________________________________________________________________//

    // Déclaration des paramètres géometriques du robot
    double r = 0.0825;
    double L = 0.28;
    // Initialisation des paramètres cinématiques
    double[,] W1 = new double[1, 2] { { 1, 1 } };
    double[,] W2 = new double[1, 2] { { 1, 1 } };
    // Initialisation des vitesses linéaire et angulaire de reférence
    double Vr = 0.5;
    double ThetaVr = Math.PI;
    // Déclaration des variables des vitesses angulaires des roues
    double PhiVg;// Gauche
    double PhiVd;// Droite
    // Déclaration de la trajectoire de reférence
    static double[,] Tréf = new double[15, 3]
    { { 0, 0, 0 }, { 0.5, 0, 0 }, { 1, 0, 0 }, { 1.5, 0, 0 }, { 2, 0, 0 },
      { 2, 0.5, Math.PI / 2 }, { 2, 1, Math.PI / 2 }, { 2, 1.5, Math.PI / 2 },
      { 2, 2, Math.PI / 2 }, { 2, 2.5, Math.PI / 2 }, { 2, 3, Math.PI / 2 },
      { 1.5, 3, Math.PI }, { 1, 3, Math.PI }, { 0.5, 3, Math.PI },
      { 0, 3, Math.PI } };
```

```
        // Déclaration D'autres variables
        static double NombreIncRobot = 756; // nombre d'incrément
        static double Coeff = NombreIncRobot / (2 * Math.PI);
        short SommeIncGauche = 0;
        short SommeIncDroite = 0;
        double[,] Données = new double[15, 3];// tableau de stockage des mesures
// Établissement de la communication avec le robot et les instruments de mesures (GPS et compas)
        UdpClient client = new UdpClient();
        static IPAddress ip = IPAddress.Parse("192.168.0.201");
        static int port = 10001;
        byte[] paquet = new byte[23];
        SerialPort portGPS;
        SerialPort portCompas;
        double xGPS, yGPS, Theta;

        //_____________________________________________________________________//

        private void Form1_Load_1(object sender, EventArgs e)
        {
            // Démarrage de l'acquisition des données de la position et de l'orienrtation
            OuvrirPorts();
            var t1 = new Task(() => GPS());
            t1.Start();
            var t2 = new Task(() => Compas());
            t2.Start();
            timer.Start();
        }
        private void timer_Tick(object sender, EventArgs e)
        {
            // Affichage des données de la position et de l'orienrtation
            label28.Text = xGPS.ToString("0.000");
            label27.Text = yGPS.ToString("0.000");
            label26.Text = (Theta * 180 / Math.PI).ToString("0.00");
        }
        // Bouton Navigation sans contrôle
        private void SansContrôle_Click(object sender, EventArgs e)
        {
            SansContrôle();
            EnregistrementDonnées();
            TracerTrajectoire();
        }
```

```
    // Bouton Navigation avec contrôle
    private void AvecContrôle_Click(object sender, EventArgs e)
    {
        AvecContrôle();
        EnregistrementDonnées();
        TracerTrajectoire();
    }
    // Bouton d'Arrêt de mouvement
    private void Arrêter_Click(object sender, EventArgs e)
    {
        ArrêterRobot();
    }
}
```