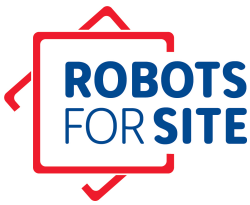

Sorbonne Université Sciences

Master Informatique, parcours Systèmes et Applications Répartis

Rapport de stage

Modélisation et pilotage des bases mobiles

Rédiger par:

Radia ACHAIBOU

Tuteurs de stage:

Dans l'organisme d'accueil : Quentin OLIVIER

L'enseignant référent : Pierre SENS

17 octobre 2022

Remerciements

Je tiens en premier lieu à exprimer ma profonde reconnaissance à Quentin Olivier, ancien ingénieur robotique et innovation à Robot For Sites qui m'a encadré et encouragé.

Je remercie Francis KALMBACHER mon maître de stage, qui occupe la fonction de directeur recherche et développement au sein de la société vinci construction France, et qui m'a accueilli comme stagiaire dans son service.

Je tiens à remercier mon enseignant référent, Pierre Sens, professeur à Sorbonne université Paris 6 pour l'aide qu'il a bien voulu m'accorder tout le long de la rédaction de mon rapport et d'avoir voulu examiner et évaluer mon travail.

Enfin, je tiens à remercier l'ensemble des personnes qui me sont proches : par leur compréhension pour le temps que j'ai consacré à mon stage, par leurs encouragements sans cesse renouvelés, ce travail leur doit aussi beaucoup.

Résumé

Mots clés : Robot mobile a roues, Modélisation des systèmes robotisés, ROS

Les robots mobiles évoluent de manière autonome dans l'espace, travaillent en groupe et offrent une flexibilité absolue à l'industrie. La dynamique du déplacement et le positionnement, des thèmes de recherche sur les robots mobiles qui mènent constamment à des développements surprenants. La commande des robots mobiles à roues est un sujet d'actives recherches. Le début de ce type d'études remonte au milieu des années 1980.
Le domaine est loin, aujourd'hui, d'avoir été complètement traité. Il continue donc de faire l'objet de très nombreuses publication.

Dans ce contexte, et afin d'augmenter sa productivité, l'entreprise Robots For Sites propose des solutions robotiques innovantes spécifiques au bâtiment.

Mon rôle au sein de cette entreprise a été, de travailler sur les lois de commande des robots mobiles à roues et la cohésion du modèle cinématique universel paramétrable pour en déduire un schéma de comportement qui nous facilite la gestion des bases mobiles. En particulier, en développant un noeud ROS (Robot Operating System) qui permet de récupérer les positions et orientations des bases mobiles en prenant en considération le type de roue de chaque base.

Ce rapport regroupe les différentes pistes de réflexions suivies et solutions retenues pour réaliser cette tâche. L'état de l'art a mis en évidence que plusieurs solutions étaient envisageables, que ce soit l'utilisation de plusieurs modélisations cinématiques selon le type de roue de chaque base mobile utilisée ou bien l'utilisation d'un seul modèle universel paramétrable.

Pour réaliser ce travail, et pour être efficace dans la conception de notre système robotique, nous avons utilisé ROS. Nous avons utilisé plusieurs bibliothèques intégrés à ROS qui nous ont permis de récupérer une estimation de la position et de la vitesse d'un robot dans l'espace libre.

Abstract

Keywords : Wheeled mobile robots, Modeling of robotic systems, ROS

Mobile robots move autonomously through space, working in groups and offer absolute flexibility to the industry. The dynamics of displacement and positioning are research themes on mobile robots that constantly lead to surprising developments.

The control of wheeled mobile robots is a subject of active research. The beginning of this type of study dates back to the mid-1980s. The field is far from having been completely covered today. It continues to be the subject of numerous publications.

In this context, and in order to increase its productivity, Robots For Sites offers innovative building-specific robotic solutions.

My role within this company has been to work on the control laws of mobile wheeled robots and the cohesion of the configurable universal kinematic model to deduce a behavior pattern that facilitates the management of mobile bases. In particular, by developing a ROS (Robot Operating System) node that allows to recover the positions and orientations of the mobile bases taking into consideration the type of wheel of each base.

This report brings together the various lines of thought followed and solutions chosen to carry out this task. The state of the art has shown that several solutions are possible, whether it be the use of several kinematic models according to the type of wheel of each mobile base used or the use of a single universal parameterizable model.

To do this work, and to be effective in designing our robotic system, we used ROS . We used several libraries built into ROS that allowed us to retrieve an estimate of the position and speed of a robot in free space.

Table des matières

Table des figures **vi**

1 Introduction **1**
1.1 Contexte 1
1.2 Les projets au sein de RFS 2
1.3 Stratégie et démarche utilisées 3

2 Le modèle **4**
2.1 Hypothèses de modélisation 4
2.1.1 Hypothèse : Description générale des robots mobiles et de leur environnement 5
2.1.2 Hypothèse : Roulement pur sans glissement 5
2.2 Description d'un robot mobile 6
2.2.1 Coordonnées généralisées décrivant le châssis d'un robot 6
2.3 Modèle cinématique 8
2.3.1 Modèles Cinématique Direct et Inverse 8
2.4 Le suivie d'une trajectoire 10
2.4.1 Approche Lyapunov 11
2.4.2 Approche linéarisation exacte 11
2.5 Conclusion 11

3 Projet **13**
3.1 Démarche et choix des solutions testées 13
3.1.1 Bases mobiles à roues non holonome 13
3.1.2 Bases mobiles à roues holonome 16
3.2 logiciel et structures utilisés 17
3.2.1 les structures et paquets Ros utilisés 18
3.3 Description des réalisations 19
3.3.1 Contrôler les actionneurs 21
3.3.2 Publication de la commande de mouvement pour le robot 21

3.3.3 Conversion de la commande de mouvement en signal d'entraînement du moteur . . . 21
3.3.4 Détermination de la position du robot . . . 22
3.4 Expérimentation . . . 25
3.4.1 Robot de ponçage . . . 25
3.4.2 Robot de percement . . . 25
3.4.3 Visualiseur ROS (RViz) . . . 27
3.4.4 Simulateurs ROS . . . 27
3.4.5 Librairies Principales de ROS utilisées . . . 28
3.4.6 Simulation . . . 29

4 Conclusions et perspectives **30**
4.1 Conclusion sur le projet et perspectives . . . 30
4.2 Apport personnel de ce stage . . . 31

5 Annexes **v**
5.1 Présentation de l'organisme d'accueil . . . v
5.1.1 Le Groupe Vinci . . . v
5.1.2 Vinci Energies . . . vi
5.1.3 Le réseau Actemium . . . vi

Bibliographie **x**

Table des figures

1.1 Projets RFS 2

2.1 Repères [O,$\vec{I}$,$\vec{J}$] et [P,$\vec{i}$, $\vec{j}$] 7
2.2 Robot mobile à roues différentielles[17] 8
2.3 Les différents Mouvements possible des bases mobiles holonome [28] 9
2.4 Conception de roue Mecanum 9
2.5 Poursuite de trajectoires admissibles 11

3.1 Démarche suivi pour le projet 13
3.2 Robot mobile de type voiture transformer en robot à roues différentielles . 14
3.3 Diagramme de répartition des noeuds ROS[3] 20
3.4 Roue différentiel du robot de ponçage 25
3.5 Le robot de ponçage 25
3.6 Le robot de percement 26
3.7 Roue mécanum du robot de percement 26
3.8 Environnement Rviz 27
3.9 Environnement Gazebo 28
3.10 Déplacement d'une base mobile à roues différentielles sous Rviz 29

5.1 Logo Vinci v
5.2 Logo Vinci Energies vi
5.3 La branche Vinci Energies vi

Chapitre 1

Introduction

1.1 Contexte

Dans le cadre de mon Master Informatique, parcours Systèmes et Applications Répartis à Sorbonne université sciences, j'ai eu l'oppotunité de réaliser mon projet de fin d'étude au sein de l'entreprise Robots For Site (RFS).

Robots For Sites (RFS), entreprise du groupe Vinci, résultant de la volonté commune des groupes Eurovia, Vinci Construction et Vinci Energies d'investir dans le développement de solutions innovantes dans le domaine de la robotique liée au bâtiment.

A travers ses projets l'entreprise RFS a pour objectif de développer des solutions de transformation technologique de chantiers 4.0, en répondant aux problématiques d'optimisation de ressources, des coûts et des processus.

Les systèmes robotiques utilisés par l'entreprise font appel à de nombreuses compétences telles que la mécanique, l'électrotechnique, le contrôle et la vision par ordinateur. Son système robotique peut donc être complexe et faire intervenir de nombreuses personnes. Afin que l'entreprise puisse être efficace dans la conception de son système robotique, réutilise au maximum des outils existants et favorise la collaboration entre les équipes, elle utilise plusieurs petits logiciels (des nœuds) exécutant une tâche (Ex : piloter des moteurs/actionneurs). Les nœuds sont disponibles via des packages ROS[1] regroupant différentes fonctions et librairies libres.

1.2 Les projets au sein de RFS

Actuellement différents projets sont à l'étude ou ont vu le jour comme les robots de désamiantage, de pose de carrelages ou bordures, de ponçage ou perçage. La stratégie de fonctionnement de l'entreprise est de créer des noeuds ROS spécifique à chaque tache afin que d'un robot à l'autre, seules les nouvelles spécificité ont besoin d'être implémenter, le reste provient de noeuds d'anciens projets, adaptés selon les besoins.

La mission des stagiaires au sein de cette entreprise est alors de concevoir un noeud ROS spécifique à une tâche de son robot. Pour ma part j'ai travaillé sur la partie "modélisation et pilotage" des bases mobiles avec des roues différentes.

Le cahier des charges prévoit le pilotage des bases mobiles grâce aux lois de commande et la cohésion entre le modèle géométrique et cinématique des différentes bases mobiles de RFS.

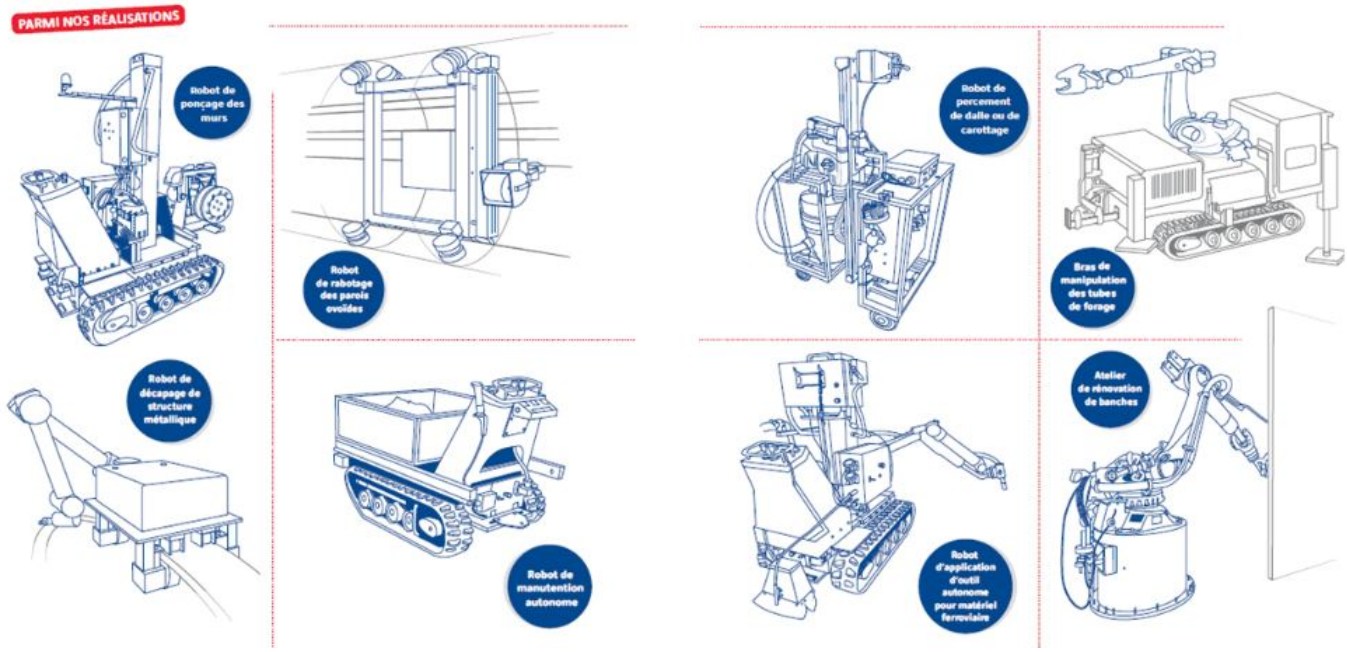

FIGURE 1.1 – Projets RFS

Mission confiée

La mission qui m'a été confié au sein de ce projet a été de développer un programme et une structure de donnée qui permet la mise en place des briques technologique du modèle cinématique direct et inverse des bases mobiles selon le type de roues (holonome ou différentiel) ; et l'utilisation d'une série de lois de commande permettant de récupérer

la position et l'orientation de la base mobile. Par ailleurs, afin que le module puisse être réutilisé sur d'autres robots, le code doit être robuste afin de pouvoir être repris et modifié facilement ultérieurement.

1.3 Stratégie et démarche utilisées

Pour parvenir à cet objectif, j'ai dans un premier temps fait des recherches dans la littérature sur la robotique mobile, les différentes roues des bases mobiles, ainsi que l'exploitation des données(vitesse angulaire, accélération...) par les actionneurs. L'exploitation de ces données permet de pouvoir déterminer l'orientation et la position de la base mobile.

Suite aux résultats de l'état de l'art et aux connaissances que j'ai acquises durant celui-ci, j'ai recensé les différentes options qui allaient permettre d'obtenir des informations sur la position des bases mobiles dans l'espace. Nous avons ensuite retenu les solutions les plus pertinentes pour piloter les robots. En cas d'échec de la validation nous retournions à l'état de l'art pour tester les autres solutions retenues, nous avons alors compris l'importance de l'état de l'art et le temps qu'il peut faire gagner au sein d'un projet, notamment si celui est amené à être repris par un tier.

Dans ce travail, nous ne nous sommes pas focalisés sur une application robotique précise ou sur un système mobile particulier. Nous avons proposé plusieurs lois de commande pour différents robots-type, représentatifs de la plupart des systèmes mobiles à roues.

La commande des robots mobiles à roues étant un sujet de recherche très riche et très concurrentiel, il apparaît difficile, dans le cadre de cette introduction, de présenter un état de l'art complet sur ce domaine. Aussi, nous avons préféré détailler, dans le chapitre 2, les résultats existant dans la littérature sur le point qui va être abordé.

Chapitre 2

Le modèle

Ce chapitre, est consacré à la modélisation des robots mobiles à roues, dans le but de développer par la suite les lois de commande.

Rechercher un modèle qui décrit avec exactitude le comportement d'un robot mobile à roues est une tâche ardue. Ces systèmes sont en effet un assemblage complexe d'une multitude d'éléments mécaniques, mais aussi électriques, élastiques (les pneumatiques ou les suspensions par exemple) ou encore thermodynamiques (dans le cas d'une propulsion par un moteur à explosion). Les constructeurs et les équipementiers automobiles sont parvenus malgré tout à écrire des modèles très fidèles. Mais comme attendu, ceux-ci sont de très grande dimension et ne peuvent donc être utilisés qu'hors-ligne, pour conduire par exemple des simulations destinées à vérifier la pertinence de tel ou tel élément pour la mobilité et/ou la sécurité du robot ou du véhicule étudié. Il serait illusoire de prétendre les utiliser pour calculer en-ligne une loi de commande, comme nous souhaitons le faire dans ce rapport.

Par conséquent, nous avons fait plusieurs hypothèses simplificatrices sur la structure des robots mobiles et sur leur environnement. Ces hypothèses sont présentées et discutées dans ce chapitre. Un niveau de simplification a été retenu. Les modèles ont été construits sous l'hypothèse que les roues roulent sans glisser sur le sol. C'est la description du contact roues-sol la plus fréquemment rencontrée dans la très vaste littérature sur la commande des robots mobiles.

2.1 Hypothèses de modélisation

Dans cette section, nous explicitons et discutons nos hypothèses de modélisation. Nous avons classé celles-ci en 2 groupes. Les Hypothèses 2.1.1 portent sur la structure générale des robots mobiles et sur leur environnement. Nous les supposerons toujours satisfaites.

Les Hypothèses 2.1.2 portent sur le contact roues-sol. Ce sont les hypothèses dites de roulement pur sans glissement. Elles seront supposées satisfaites ce qui nous conduira à un ensemble de modèles.

2.1.1 Hypothèse : Description générale des robots mobiles et de leur environnement

— **2.1.1.a.** Les robots mobiles sont supposés mono-corps (i.e. pas de remorque).

— **2.1.1.b.** le châssis des robots, de même que les pièces reliant les roues au châssis sont supposés rigides (ce qui exclut en particulier tout système de suspension). Dans ces conditions, les robots mobiles ne présentent :

  — ni roulis (i.e. rotation du châssis du robot autour d'un axe parallèle à la direction d'avancement).

  — ni tangage (i.e. rotation du châssis du robot autour d'un axe perpendiculaire à la direction d'avancement.

  — ni carrossage (i.e. les plans contenant chacune des roues restent à tout moment. verticaux).

— **2.1.1.c.** La dynamique des différents moteurs commandant les roues en rotation et en orientation est supposée négligeable. Dans ces conditions, les couples de rotation et d'orientation appliqués sur les roues peuvent être considérés comme étant les variables de commande, à spécifier par l'utilisateur.

— **2.1.1.d.** La surface d'évolution des robots est supposée horizontale et parfaitement plane.

— **2.1.1.e.** Il est supposé qu'aucune force aérodynamique n'agit sur les robots mobiles.

2.1.2 Hypothèse : Roulement pur sans glissement

— **2.1.2.a.** Chaque roue est supposée indéformable et la zone de contact roue-sol est supposée ponctuelle.

— **2.1.2.b.** La vitesse linéaire du point de contact d'une roue avec le sol est nulle

Discussions

Par les hypothèse 2.1.1.a, nous laissons de côté l'ensemble des robots mobiles multi-corps. Ces systèmes ne présentent cependant pas nécessairement de difficulté supplémentaire par rapport aux robots mono-corps.

Les Hypothèses 2.1.1.b à 2.1.1.e sont quant à elles, des hypothèses très classiques dans la littérature, dès lors que le modèle du robot mobile est recherché dans le but de construire par la suite d'une loi de commande. Elles simplifient en effet significativement le modèle du robot.

Les hypothèses 2.1.2 de roulement pur sans glissement sont également très classiques dans la littérature. Les modèles de robots obtenus sous ces hypothèses sont très compacts et très simples

2.2 Description d'un robot mobile

Le premier point dans la construction d'un modèle consiste à sélectionner un ensemble de variables permettant de repérer la configuration du système. Ces variables sont appelées coordonnées généralisées :

Définition 2.1 Coordonnées généralisées

On appelle coordonnées généralisées associées à un système mécanique S un ensemble de variables dont la donnée, combinée avec les constantes dérivant de la géométrie de S permet de connaître sans ambiguïté la position de chacun des points de S. Un modèle S se présentera donc sous la forme d'un système d'équations reliant les coordonnées généralisées, leurs dérivées et les commandes de S.

Schématiquement, un robot mobile à roues satisfaisant aux hypothèses 2.1.1 & 2.1.2 se présente sous la forme d'un châssis rigide équipé avec n roues, où $n >= 3$ afin que l'équilibre statique du système soit assuré. Nous décrivons ci-dessous, le groupe de coordonnées généralisées repérant tous les points du châssis.

2.2.1 Coordonnées généralisées décrivant le châssis d'un robot

Soit (voir Figure 2.1)

O : Un point immobile dans le domaine d'évolution du robot,
P : un point fixe du châssis,
[O,$\vec{I}$,$\vec{J}$] : un repère immobile dans le plan d'évolution du robot,
[P,$\vec{i}$, $\vec{j}$] : un repère attaché au châssis,

Nous noterons
(x, y) : les coordonnées de P dans [o,$\vec{I}$,$\vec{J}$],
θ : l'orientation du repère [P,$\vec{i}$, $\vec{j}$] par rapport au repère [O,$\vec{I}$,$\vec{J}$], i.e. $\theta = \widehat{\vec{I},\vec{i}}$

Ces 3 variables seront dans la suite regroupées dans un vecteur ζ :

$$\zeta = (x, y, \theta) \tag{2.1}$$

La donnée du vecteur ζ permet de caractériser sans ambiguïté la configuration du châssis d'un robot. Le châssis étant suposé mono-corps et rigide (hypothèse 2.1.1 & 2.1.2), tous les points qui le constituent sont fixes dans le repère [P,$\vec{i}$, $\vec{j}$]. Par conséquent, la donnée du vecteur ζ qui décrit ce repère par rapport au repère immobile [O,$\vec{I}$,$\vec{J}$], permet aussi de localiser sans ambiguïté tous les points du châssis du robot.

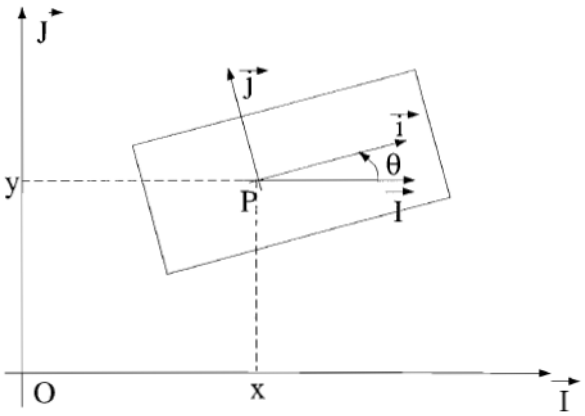

FIGURE 2.1 – Repères [O,$\vec{I}$,$\vec{J}$] et [P,$\vec{i}$, $\vec{j}$]

2.3 Modèle cinématique

C'est un outil de modélisation fondamental des systèmes mécaniques, qui rend compte exclusivement des mouvements possibles entre les différents sous-ensembles qui le constituent. Il permet donc d'analyser un mécanisme en vue de :

— Son étude géométrique et cinématique,

— Son étude statique ou dynamique.

2.3.1 Modèles Cinématique Direct et Inverse

Roues différentielles

Le modèle cinématique des bases mobile à roues différentielles est la relation mathématique qui fait correspondre le mouvement indépendant des roues au mouvement global du châssis du robot. Ce sujet fondamental est le fondement de tout contrôle de robot mobile, en ce sens qu'il est principalement responsable de la mobilité prévisible du robot. Dans ce qui suit, le modèle cinématique[17].

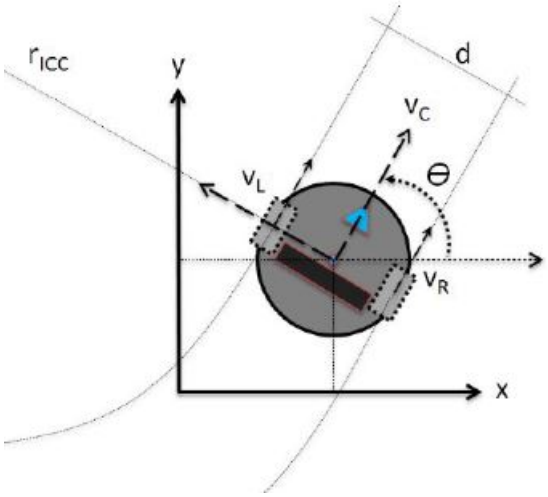

FIGURE 2.2 – Robot mobile à roues différentielles[17]

Le système de contrôle :

$$\begin{pmatrix} \dot{X} \\ \dot{Y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} v * cos\theta \\ v * sin\theta \\ r * (w_r - w_l)/2R_w \end{pmatrix} = \begin{pmatrix} (w_l + w_r) * cos\theta \\ (w_l + w_r) * sin\theta \\ r * (w_r - w_l)/2R_w \end{pmatrix}$$

Les entrées : w_l : vitesse de la roue gauche. w_r : vitesse de la roue droite.

Les bases mobiles à roues différentiel sont entraînées par un ensemble de deux roues coaxiales. Ces roues sont actionnées à l'aide de moteurs à courant continu hautes performances. Pour déterminer la relation entre le mouvement indépendant des deux roues et le mouvement du robot global.

Notons que w_r , w_l et v sont tous définis le long du même axe, qui se trouve dans la direction avant/arrière du châssis. Étant donné la vitesse angulaire de la roue, w_r et w_l, la vitesse du robot v, et la distance r, nous pouvons relier le mouvement des roues au mouvement du robot en utilisant le modèle cinématique direct pour l'entraînement différentiel du système.

Roues holonomes

Les roues Mecanum permettent un mouvement holonomique[28]. Cela signifie que la transmission peut se déplacer dans n'importe quelle direction pendant la rotation : vers l'avant, vers l'arrière, d'un côté à l'autre, en translation pendant la rotation, etc.

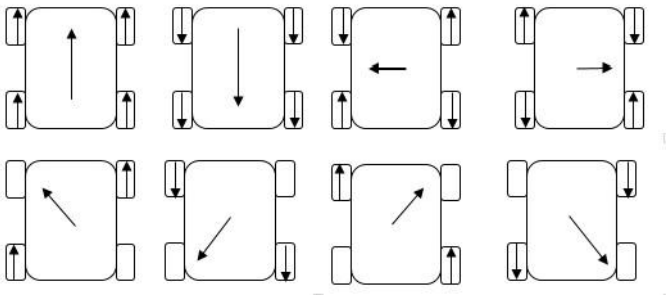

FIGURE 2.3 – Les différents Mouvements possible des bases mobiles holonome [28]

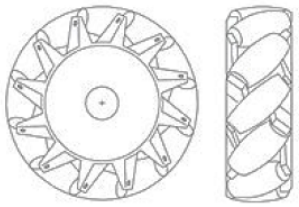

FIGURE 2.4 – Conception de roue Mecanum

Les roues Mecanum ont des rouleaux à un angle de 45° par rapport au reste de la roue, c'est ce qu'il les différencie des roues Omnidirectionnel. Comme ceux-ci sont en contact avec le sol au lieu de quelque chose de solide, au lieu que la roue crée une force parallèle à l'orientation de la roue, elle en crée une à 45° par rapport au parallèle. Selon la façon dont les roues sont entraînées, les composantes X ou Y des vecteurs de force peuvent s'annuler, ce qui permet un mouvement dans n'importe quelle direction. Le système de contrôle :

$$v_x = v * sin\theta$$

$$v_y = v * cos\theta$$

$$w_1 = v_x + v_y$$

$$w_2 = v_x - v_y$$

$$w_3 = v_x + v_y$$

$$w_4 = v_x - v_y$$

Nous obtenons :
La vitesse longitudinale :

$$v_x = \frac{r}{4}(w_1 + w_2 + w_3 + w_4)$$

vitesse transversale

$$v_y = \frac{r}{4}(-w_1 + w_2 + w_3 - w_4)$$

vitesse angulaire

$$w_z = \frac{r}{4(l_x + l_y)}(-w_1 + w_2 - w_3 + w_4)$$

Avec l_x, l_y, la moitié de la distance entre les roues avant et a moitié de la distance entre la roue avant et les roues arrière. Et r, distance du centre de la roue au centre du rouleau.

2.4 Le suivie d'une trajectoire

Une base mobile doit être capable de se déplacer d'un point A à un point B. Le problème de suivi d'une trajectoire de référence pour un robot mobile est apparu comme un problème de premier ordre pour la communauté roboticienne dans ces dernières années. En effet, la forte utilisation des robots mobiles dans les domaines où l'être humain ne peut pas être présent, nécessite la mise en œuvre de lois de commande autonomes et performantes pour assurer les tâches assignées aux robots. Plusieurs travaux concernant la poursuite de trajectoire ont été développés dans ce contexte[6]. Après une rapide formalisation de ce problème, une liste non exhaustive des différentes commandes disponibles dans la littérature est présentée.

2.4.1 Approche Lyapunov

Plusieurs fonctions de Lyapunov ont été proposées pour réaliser la poursuite d'une trajectoire mobile. Nous retenons la lois de poursuite cherchant à faire converger l'état du système $\zeta(t) = (x(t), y(t), \theta(t))$ vers une trajectoire de référence $\zeta_{ref}(t) = (x_{ref}(t), y_{ref}(t), \theta_{ref}(t))$. Etant donnée une trajectoire de référence admissible ζ, l'objectif de la poursuite est de trouver la commande u en fonction de ζ, ζ_{ref}, et du temps t telle que l'erreur de poursuite $\zeta_e = \zeta - \zeta_{ref}$ converge asymptotiquement vers zéro. Le principe de la poursuite de trajectoire est illustré par la figure 2.5.

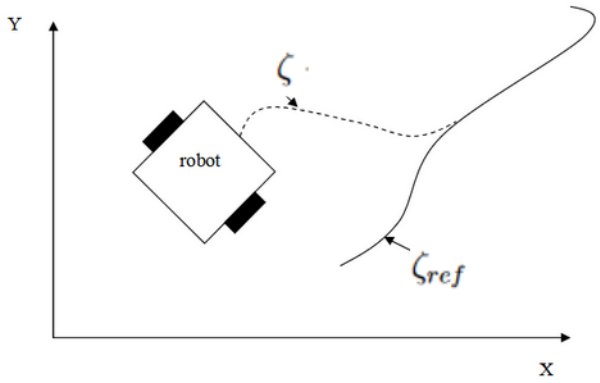

FIGURE 2.5 – Poursuite de trajectoires admissibles

2.4.2 Approche linéarisation exacte

On dispose en automatique linéaire d'un grand nombre d'outils, souvent simples d'utilisation, permettant d'analyser et de commander un système. Aussi, une démarque naturelle au moment d'aborder l'étude et la commande d'un système non-linéaire est de rechercher s'il existe un jeu de coordonnées généralisées avec lequel ce système se ré-écrive sous une forme linéaire [bibliographie].

2.5 Conclusion

Ce travail a permis de mettre en lumière les différentes hypothèses nécessaires pour la réalisation du projet. Les différents modèles existants pour représenter les robots mobiles à roues différentiel et à roues holonomes ont été présentés. La seconde partie du travail a consisté à rechercher dans la littérature les différentes méthodes et algorithmes qui pouvaient permettre à une base mobile de suivre une trajectoire.

Ce travail d'état de l'art s'inscrit comme une étape initiale de mon projet d'étude. Celui-ci consistera à développer des lois de commandes qui permettent aux bases mobiles de suivre une trajectoire. Suite à cet état de l'art, deux approches possible ont été dégagé :

— La première approche consiste à développer le modèle cinématique direct et inverse des bases mobiles ainsi qu'une loi de commande permettant de récupérer la position et l'orientation d'une base mobile holonome ou non-holonome à chaque instant t ainsi que l'état du système. Cette méthode est la plus rapide à mettre en place.

— La seconde solution serait d'investir dans un capteur intelligent (Lidar) de detection de mouvement/ emplacement des bases mobiles ainsi qu'une loi de commande en utilisant l'approche Lyapunov permettant à une base mobile holonome ou non-holonome de se déplacer d'un point A à un point B et d'avoir à chaque instant t l'état du système. Dans ce projet, nous n'avons pas la possibilité d'utiliser un capteur intelligent. .

— La dernière méthode consiste à construire un modèle cinématique universel pour tous types de robots mobiles quelconque et construire par la suite, des lois de linéarisation par bouclage d'état dynamique. Cette méthode est complexe et avancée. Toutefois c'est une méthode réutilisable dans d'autres projets de l'entreprise.

Le rapport de l'état de l'art effectué a également pour vocation de simplifier le travail des prochains employés qui travailleront sur, avec des modèles cinématiques ou suivie de trajectoire.

Chapitre 3

Projet

3.1 Démarche et choix des solutions testées

L'objectif final de mon projet était de réaliser un noeud ROS permettant de déterminer la position et l'orientation du robot en fonction des mesures de rotation des roues. Pour y parvenir, nous avons créer un modèle cinématique de robot pour chaque type de roues.

Pour cela nous avons suivi la démarche expliquée en introduction, celle-ci peut se résumer à la figure 3.1 ci dessous

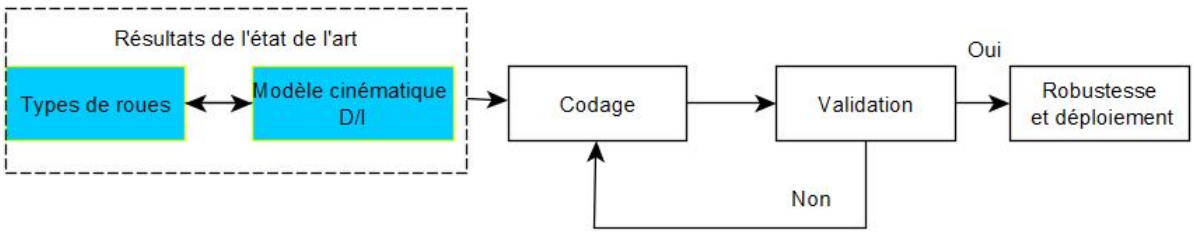

FIGURE 3.1 – Démarche suivi pour le projet

3.1.1 Bases mobiles à roues non holonome

Toutes les bases mobiles à quatres roues avec un entraînement séparé pour chaque roue, nous les avons traitées comme des bases à deux roues afin de simplifier le calcul cinématique. Deux roues virtuelles (marquées $W_L et W_R$ sur le schéma) auront un axe passant par le centre géométrique du robot. De cette façon, nous pouvons utiliser un modèle cinématique plus simple de robot à roues différentielles. Le nom "différentiel" vient du fait que le robot peut changer de direction en faisant varier la vitesse relative de rotation de ses roues et ne nécessite pas de mouvement de direction supplémentaire. Le schéma du robot est présenté ci-dessous 3.2

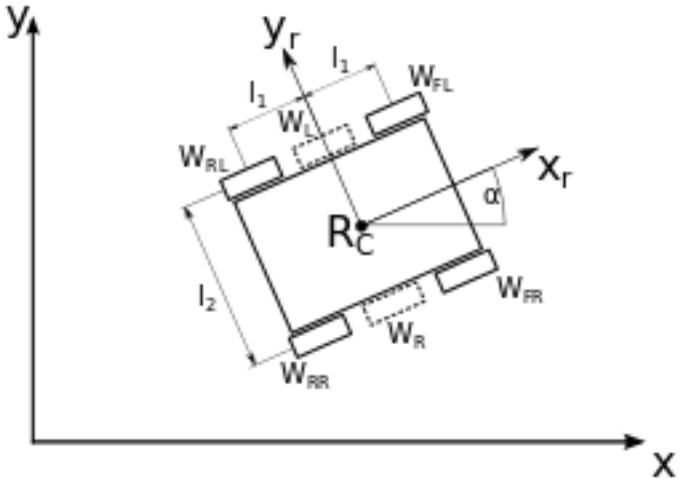

FIGURE 3.2 – Robot mobile de type voiture transformer en robot à roues différentielles

Avec :

- R_c - centre géométrique du robot.
- x_c - position x du centre géométrique du robot.
- y_c - position y du centre géométrique du robot.
- x_r - axe x local du robot qui détermine l'avant du robot.
- α - position angulaire du robot.
- W_{FL} - roue avant gauche.
- W_{FR} - roue avant droite.
- W_{RL} - roue arrière gauche.
- W_{RR} - roue arrière droite.
- W_L - roue gauche virtuelle.
- W_R - roue droite virtuelle.
- l_1 - distance entre le centre du robot et les roues avant/arrière.
- l_2 - distance entre les roues gauche et droite du robot.

Le robot peut se déplacer que dans le plan $x - y$ et il a 3 DOF (degrees of freedom). Cependant, tous les DOF ne sont pas contrôlables, ce qui signifie que le robot ne peut pas se déplacer dans toutes les directions de ses axes locaux (par exemple, il ne peut pas se déplacer latéralement). Un tel système d'entraînement est appelé non holonome. Lorsque la quantité de DOF contrôlables est égale au total des DOF, un robot peut être appelé holonome. Pour y parvenir, certains robots mobiles sont construits à l'aide de roues Omnidirectionnel ou Mecanum et grâce au mouvement de vectorisation, ils peuvent changer de position sans changer de cap (orientation).

Cinématique directe

La position du robot est déterminée par un tuple (x_c, y_c, α). La tâche cinématique directe consiste à trouver la nouvelle position du robot $(x_c, y_c, \alpha)'$ après un temps δt pour des paramètres de contrôle donnés :

— v_R - vitesse linéaire de la roue virtuelle droite.
— v_L - vitesse linéaire de la roue virtuelle gauche.

Dans notre cas la vitesse angulaire ω et la position angulaire ϕ de chaque roue virtuelle seront une moyenne de ses homologues réels :

$$\phi_{WL} = \frac{\phi_{WFL}+\phi_{WRL}}{2}$$
$$\phi_{WR} = \frac{\phi_{WFR}+\phi_{WRR}}{2}$$
$$\omega_{WL} = \frac{\omega_{WFL}+\omega_{WRL}}{2}$$
$$\omega_{WR} = \frac{\omega_{WFR}+\omega_{WRR}}{2}$$

Vitesse linéaire de chaque roue virtuelle :

$$v_R = \omega_{WR} * r$$
$$v_L = \omega_{WL} * r$$

où r - le rayon de la roue.

Nous pouvons déterminer la position angulaire et la vitesse du robot avec :

$$\alpha = (\phi_{WR} - \phi_{WL}) * \frac{r}{l_2}$$
$$\dot{\alpha} = \frac{d\alpha}{dt}$$

La vitesse du robot sur la composante x et la composante y :

$$\dot{x_c} = (v_L + \dot{\alpha}\frac{l_2}{2})\cos(\alpha)$$
$$\dot{y_c} = (v_L + \dot{\alpha}\frac{l_2}{2})\sin(\alpha)$$

Pour obtenir la position :

$$x_c = \int_0^t \dot{x_c} dt$$
$$y_c = \int_0^t \dot{y_c} dt$$

Nous supposons que la position de départ est $(0, 0)$.

3.1.2 Bases mobiles à roues holonome

Le calcule du modèle cinématique direct et inverse est fastidieux, nous préférons ajouter toutes les étapes de calcul en Annexe, dans ce qui suit les équations obtenues et que nous avons développé sous ROS.

Nous obtenons :
La vitesse longitudinale :

$$v_x = \frac{r}{4}(\omega_1 + \omega_2 + \omega_3 + \omega_4)$$

vitesse transversale

$$v_y = \frac{r}{4}(-\omega_1 + \omega_2 + \omega_3 - \omega_4)$$

vitesse angulaire

$$w_z = \frac{r}{4(l_1 + l_2)}(-\omega_1 + \omega_2 - \omega_3 + \omega_4)$$

Avec :

- $\omega_i[rad/s], i \in [1, 4]$ - vitesse angulaire des roues ;
- $v_x, v_y[m/s]$ - vitesse linéaire du robot ;
- $\omega_z[rad/s]$ - vitesse angulaire du robot ;
- r - désigne le rayon de la roue ;
- l_1 -la moitié de la distance entre les roues avant. ;
- l_2 - la moitié de la distance entre la roue avant et les roues arrière.

La vitesse résultante et sa direction dans l'axe des coordonnées fixes (x, y, z) peuvent être obtenues par les équations suivantes :

$$\alpha = \tan^{-1}(\frac{v_y}{v_x})$$
$$v_R = \sqrt{v_x^2 + v_y^2}$$

3.2 logiciel et structures utilisés

Le système d'exploitation robotique (ROS) est un intergiciel robotique (c-à-d. une collection de cadres logiciels pour le développement de logiciels robotisés). ROS est considéré comme un méta système d'exploitation open source, qui fournit des services conçus pour le système informatique hétérogène tels que l'abstraction matérielle, le contrôle de périphériques de bas niveau, la mise en œuvre de fonctionnalités couramment utilisées, le passage de messages entre les processus et la gestion des paquets. Les ensembles en cours d'exécution de processus basés sur ROS sont représentés dans une architecture graphique où le traitement a lieu dans les noeuds qui peuvent recevoir. Malgré l'importance de la réactivité et de la faible latence dans le contrôle des robots, ROS n'est pas en soi un système d'exploitation en temps réel (RTOS), bien qu'il soit possible d'intégrer ROS avec du code en temps réel. Le manque de soutien pour les systèmes en temps réel est abordé dans la création de ROS 2.0. Les logiciels de l'écosystème ROS peuvent être séparés en trois groupes :

— Outils indépendants de la langue et de la plateforme utilisés pour construire et distribuer des logiciels basés sur ROS ;

— Implémentations de bibliothèques clients ROS telles que roscpp, rospy et roslisp ;

— Paquets contenant du code lié à l'application qui utilise une ou plusieurs bibliothèques clientes ROS.

Deux outils indépendants de la langue et les principales bibliothèques clientes (C++, Python et Lisp) sont publiés sous les termes de la licence BSD, et en tant que tel est un logiciel libre et gratuit pour un usage commercial et de recherche. La majorité des autres paquets sont sous licence sous une variété de licences open source. Ces autres progiciels mettent en œuvre des fonctionnalités et des applications couramment utilisées telles que les pilotes matériels, les modèles de robots, les types de données, la planification, la perception, la localisation et la cartographie simultanées, les outils de simulation et d'autres algorithmes.

Les principales bibliothèques clientes de ROS (C++, Python et Lisp) sont orientées vers un système de type Unix, principalement en raison de leur dépendance à de grandes collections de dépendances logicielles open-source. Pour ces bibliothèques clientes, Ubuntu Linux est listé comme "Supporté" tandis que d'autres variantes telles que Fedora Linux, macOS, et Microsoft Windows sont désignées comme "Experimental" et sont supportées par la communauté.

3.2.1 les structures et paquets Ros utilisés

Cette sous section fournit les connaissances de base nécessaires et établit les conditions préalables à la réalisation de notre projet.
Les concepts importants à étudier et à mettre en œuvre sont les suivants :

— **Nodes** est l'un des concepts les plus importants dans ROS. Nous pouvons penser à un noeud comme une fonction. Il prend une certaine entrée, effectue une opération sur elle et donne une sortie. Un noeud peut s'abonner ou publier sur un sujet. Les noeuds sont fondamentaux dans ROS pour la communication avec le robot et pour effectuer une tâche. Les noeuds sont écrits en langage C++ pour ce projet.

— **Topic** peut être considéré comme le fil entre deux nœuds qui transporte les données. Les Topics sont le processus de transmission de données entre les nœuds. Certains noeuds sont responsables de la publication de certaines données sur un sujet spécifique où les autres nœuds (abonnés) pourront demander ces données (messages) du Topic.

— **Messages** sont des structures de données qui décrivent les données, les nœuds ROS publient ou reçoivent. les noeuds communiquent, envoient et reçoivent des messages à travers les Topics.

— **Services** est une autre façon de transmettre les données entre les nœuds. C'est une opération synchrone où le robot doit arrêter le traitement en attendant pour une réponse le service.

L'actif principal de ROS est ses bibliothèques et paquets. Le paquet ROS est défini comme suit :

Un paquet dans ROS est un répertoire de travail qui contient tous les fichiers ROS nécessaires comme les fichiers cpp exécutables, les fichiers de configuration, de compilation et les fichiers de lancement.

Il est important de comprendre la structure du paquet ROS. Il contient les dossiers suivants :

- **src** : dossier source qui contient tous les fichiers exécutables.
- **launch** dossier de lancement qui contient tous les fichiers de lancement.
- **package.xml** fichier qui contient d'autres dépendances de paquets et chemins url. Dans le cas où l'on veut être en mesure d'appeler d'autres packages à partir de ce fichier.
- **CMakeLists.txt** fichier qui contient des commandes de compilations et des cpp .

Un autre concept important dans ROS est le système de construction. La construction dans ROS utilise le système **Catkin** qui est un macros système et infrastructure de compilation de bas niveau pour ROS. Il fournit un Système de compilation où l'on peut coder dans des fichiers C++ et ROS s'occupera automatiquement de l'implémentation sur le matériel. Il est important de noter que Catkin fournit également un worksapce et a de nombreux paquets liés à la mise en œuvre des tâches spécifiques. Le worksapce où nous travaillons sur la construction est **catkin_ws**. C'est le répertoire où tous les les paquets sont installés.

3.3 Description des réalisations

La répartition des noeuds (Figure 3.3) met en avant le travail effectué et illustre les différents échanges entre le noeud développé et les autres noeuds ROS de RFS. Nous avons développé le noeud "kinematic model" contenant les packages du modèle cinématique direct et inverse des bases mobiles selon deux type de roues : non holonome et holonome. Rappelons que le but de ce projet est de piloter une base mobile et déterminer sa position et son orientation à un instant t.

Pour envoyer des commandes de mouvement au robot, on utilise le module *geometry_ msgs /Twist* qui est intégré à la plateforme de ROS. Une fois un objet de classe *geometry_ msgs/Twist* initialisé, nous obtenons un objet avec deux variables de type *Vector3*.

L'utilité du module "*geometry_ msgs/Twist*" dans le cadre de ce projet

Pour que la base mobile puisse tourner et se déplacer dans une direction, nous devons spécifier la quantité de mouvement de rotation (angulaire) et la quantité de mouvement linéaire. Une fois les paramètres requis dans les variables de *geometry_ msgs/Twist* définies et publiés, la base mobile se déplace en conséquence.

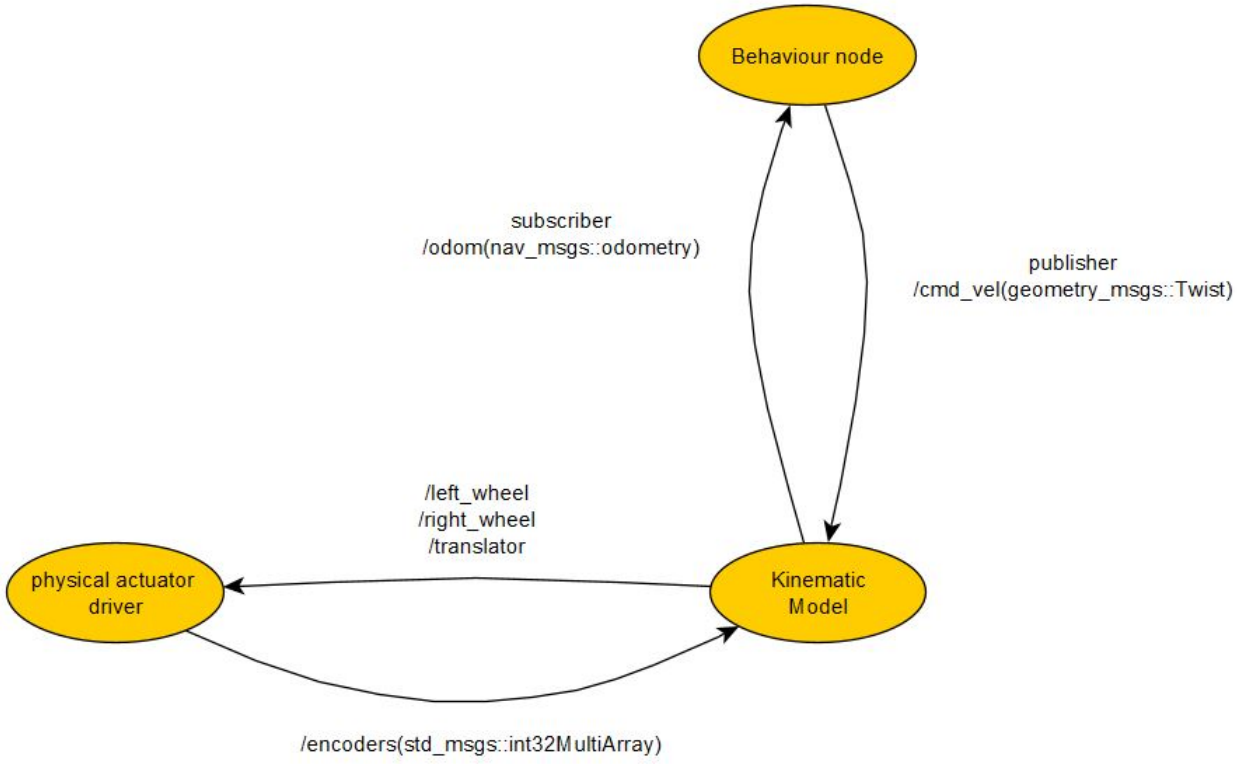

FIGURE 3.3 – Diagramme de répartition des noeuds ROS[3]

Le noeud "kinematic model" s'abonne à *Odometry* ROS message */odom* qui permet d'estimer la position du robot mobile, ainsi que sa vitesse actuelle.

L'utilité du module "*/odom*" dans le cadre de ce projet

L'odométrie décrit l'état "interne" du robot, c'est-à-dire la position intégrée à l'aide d'encodeurs de roue (une estimation de ce que le robot fait en ce moment, position et vitesse). Compte tenue d'un objectif, déplacer le robot mobile vers cet objectif, "kinematic model node" envoie des messages *Twist* à un contrôleur "behaviour node", qui après d'autres calcul, entraîne le virage des roues du robot.

L'utilité du module "*/encoders*" dans le cadre de ce projet

On peut obtenir l'odométrie des encodeurs à partir des roues, le "kinematic model node" s'abonne au noeud "physical actuator driver" pour récupérer les tiques. Il reçoit les tiques des encodeurs (environs 2626 tiques pour une rotation complète des roues) et donc par un simple calcul nous récupérons les vitesses des roues.

3.3.1 Contrôler les actionneurs

Le moyen le plus courant d'envoyer des commandes de mouvement au robot consiste à utiliser le type de message **geometry_msgs/Twist**. Ensuite, le nœud de pilote de moteur doit utiliser les données qui y sont stockées pour contrôler le moteur.

Le message **geometry_msgs/Twist** exprime la vitesse dans l'espace libre et se compose de deux champs :

- Vector3 linear - représente la partie linéaire de la vitesse [m/s].
- Vector3 angulaire - représente la partie angulaire de la vitesse [rad/s].

On contrôle le mouvement de la base mobile dans le plan $x - y$ en manipulant la composante x du vecteur de vitesse linéaire et la composante z du vecteur de vitesse angulaire.

3.3.2 Publication de la commande de mouvement pour le robot

On utilise le clavier pour contrôler le mouvement de notre base mobile. Pour obtenir les événements clés et les convertir en messages **geometry_msgs/Twist**, on utilise le noeud **teleop_twist_keyboard.py** du package **teleop_twist_keyboard**.

3.3.3 Conversion de la commande de mouvement en signal d'entraînement du moteur

Nous avons créé un nœud pour l'interface des moteurs. le nœud s'abonne (subscribe) au sujet (topic) avec des messages **geometry_msgs/Twist**, pilote les moteurs, lit les encodeurs et publie (publish) leur état dans le sujet approprié.

Après avoir défini le type de capteur de distance utilisé ainsi que le type d'IMU (International Mathematical Union) utilisé, nous définissons la fonction qui permet de gérer les messages entrants :

```
void twistCallback(const geometry_msgs::Twist &twist)
        {
         rosbot.setSpeed(twist.linear.x, twist.angular.z);
        }
```

La fonction qui permet d'initialiser l'abonné à la commande de vitesse :

```
void initCmdVelSubscriber()
        {
         ros::Subscriber<geometry_msgs::Twist> *cmd_sub =
         new ros::Subscriber<geometry_msgs::Twist>
         ("/cmd_vel", &twistCallback);
         nh.subscribe(*cmd_sub);
        }
```

Fonction de gestion des demandes entrantes de réinitialisation l'odométrie du robot (la position et la vitesse) :

```
void resetCallback(const std_msgs::Bool &msg)
        {
                if (msg.data == true)
                {
                        rosbot.reset_odometry();
                }
        }
```

Fonction d'initialisation des demandes de remise à zéro de l'odométrie des abonnés :

```
void initResetOdomSubscriber()
{
  ros::Subscriber<std_msgs::Bool> *odom_reset_sub =
  new ros::Subscriber<std_msgs::Bool>("/reset_odom", &resetCallback);
  nh.subscribe(*odom_reset_sub);
}
```

3.3.4 Détermination de la position du robot

Afin de déterminer la position du robot, nous avons besoin du modèle cinématique direct. Donc, nous allons utiliser les encodeurs attachés à chaque moteur et traiter leurs mesures avec les équations indiquées dans la section 3.1.1.

On rajoute les fichiers (headers) :

```
#include "geometry_msgs/PoseStamped.h"
#include "tf/tf.h"
```

Définir le type de message et le publisher pour la position du robot :

```
 geometry_msgs::PoseStamped pose;
        ros::Publisher *pose_pub;
```

Création d'une structure de données :

```
std::vector<float> rosbot_pose;
```

Fonction de publication de la position du robot :

```
void initPosePublisher()
{
 pose.header.frame_id = "odom";
 pose.pose.orientation = tf::createQuaternionFromYaw(0);
 pose_pub = new ros::Publisher("/pose", &pose);
 nh.advertise(*pose_pub);
}
```

Dans la fonction main pour l'initialisation de PosePublisher :

```
initPosePublisher();
```

Mise en place d'un noeud avec C++

Pour pouvoir être utilisé facilement au sein de l'entreprise, le code a été adopté en C++ sous la forme d'une classe correspondant à un noeud sous ROS. Dans un soucis constant d'uniformité et de réutilisation de code par une tierce personne, les codes ont été réalisé en suivant un manuel de bonne conduite rédigé par les ingénieurs RFS à destination des employés.

Le choix a été fait d'avoir un noeud commune à la partie modèle cinématique direct et inverse. La structure du code suit alors l'arborescence suivante :

kinematic_robot :

- include
  - kinematic_fonctions.hpp

- launch

    - mon_launch.launch

- src

    - kinematic_fonction.cpp

- package.xml

- CmakeLists.txt

- readme.txt

Le fichier principal est le fichier "kinematic_fonction.cpp" qui contient les différentes fonctions utilisées et le code principal. Les autres fichiers permettent le bon fonctionnement de l'exécution de ce code principal.

Le détail de l'utilisation, et fonctionnement des différents fichier, est donné dans le readme.txt et est également présenté succinctement ci-dessous :

- Le fichier kinematic_fonctions.hpp contient les définitions de la classe, des fonctions et des variables ainsi que les packages et bibliothèques utilisées.

- Le fichier mon_launch.launch permet de lancer proprement le programme en tant que noeud ROS.

- Le fichier kinematic_fonctions.cpp est le corps du programme codé en C++, il contient le "Main" ainsi que les fonctions utilisées par le programme "Main".

- Le fichier package.xml permet de vérifier si l'utilisateur dispose des packages requis pour lancer le script. Le cas échéant, il lance l'installation des packages manquants.

- Le fichier CmakeLists.txt permet de gérer la compilation du fichier principal en C++ pour qu'il puisse être exécuté.

- Le fichier readme.txt donne le fonctionnement rapide du code et les instructions pour utiliser le programme.

3.4 Expérimentation

3.4.1 Robot de ponçage

Le Robot de ponçage est constitué d'une base roulante équipée de 2 roues motrices et une roue folle.
Un aspirateur est également installé sur l'embase pour absorber les poussières/gravats.

FIGURE 3.4 – Roue différentiel du robot de ponçage

FIGURE 3.5 – Le robot de ponçage

3.4.2 Robot de percement

Le Robot de Percement est constitué d'une base roulante équipée de 4 roues omnidirectionnelles (mecanum). Chacune de ces roues est motorisée par un moteur brushless.

Un bras Staubli est installé sur l'embase et permet d'atteindre les différents points à percer. L'extrémité du bras est équipée d'un perforateur Hilti pour réaliser les différents perçages.

Un aspirateur est également installé sur l'embase et relié à l'extrémité du bras pour absorber les poussières/gravats. L'embase embarque une batterie 48V qui alimente les roues lorsque le robot est conduit vers sa zone de travail. Les opérations de perçage sont pilotées via une tablette tactile. Le déplacement du robot vers sa zone de travail est réalisé via une commande filaire.

principe de fonctionnement

On considère deux modes de fonctionnement qui sont activés par un sélecteur de position via le coffret électrique.

— **Mode roulage** : Il consiste à déplacer le robot depuis sa zone de déchargement vers la zone de travail (et inversement). Il est piloté par l'opérateur via une commande filaire. Cette commande actionne uniquement les roues, en vitesse lente. Lors de cette phase, les lidars ne sont pas activés.

— **Mode travail** : L'opérateur trace des lignes et des points à l'aide d'une bombe de peinture sur le mur à percer. Une fois le robot amené à proximité de la zone de travail, il prend le contrôle du robot via la tablette selon deux modes :
  — **Mode semi_automatique** : Il permet à l'opérateur de piloter indépendamment chaque actionneur afin de bien positionner le robot sur le premier perçage.
  — **Mode automatique** : Il permet de lancer le cycle de perçage du robot en suivant de façon automatique le tracé préalablement réalisé par l'opérateur.

FIGURE 3.6 – Le robot de percement

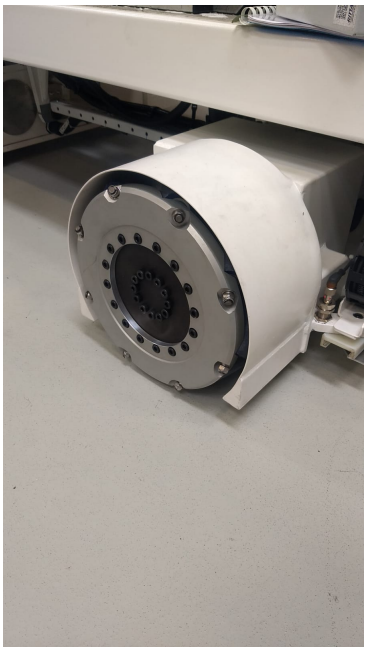

FIGURE 3.7 – Roue mécanum du robot de percement

L'entreprise était en plein déménagement/ reconstitution en mois d'août dernier, les tests n'ont pas pû être effectué sur les bases mobiles car :

Le robot de percement a été déplacé vers le site de Marolle et je n'ai pas pû avoir accès. En ce qui concerne le robot de ponçage, son contrôleur est défectueux, l'entreprise a commandé un autre pour le remplacer mais les délais d'approvisionnement sont longs.

Tous les tests ont été effectués en simulation sous Ros et Rviz, et validés par les ingénieurs.

3.4.3 Visualiseur ROS (RViz)

RViz est l'un des visualiseurs 3D disponibles dans ROS qui peut visualiser des valeurs 2D et 3D à partir de sujets et de paramètres ROS. RViz permet de visualiser des données telles que des modèles de robots, des données de transformation 3D (TF) de robots, des nuages de points, des données laser et d'image, ainsi qu'une variété de données de capteurs différentes :

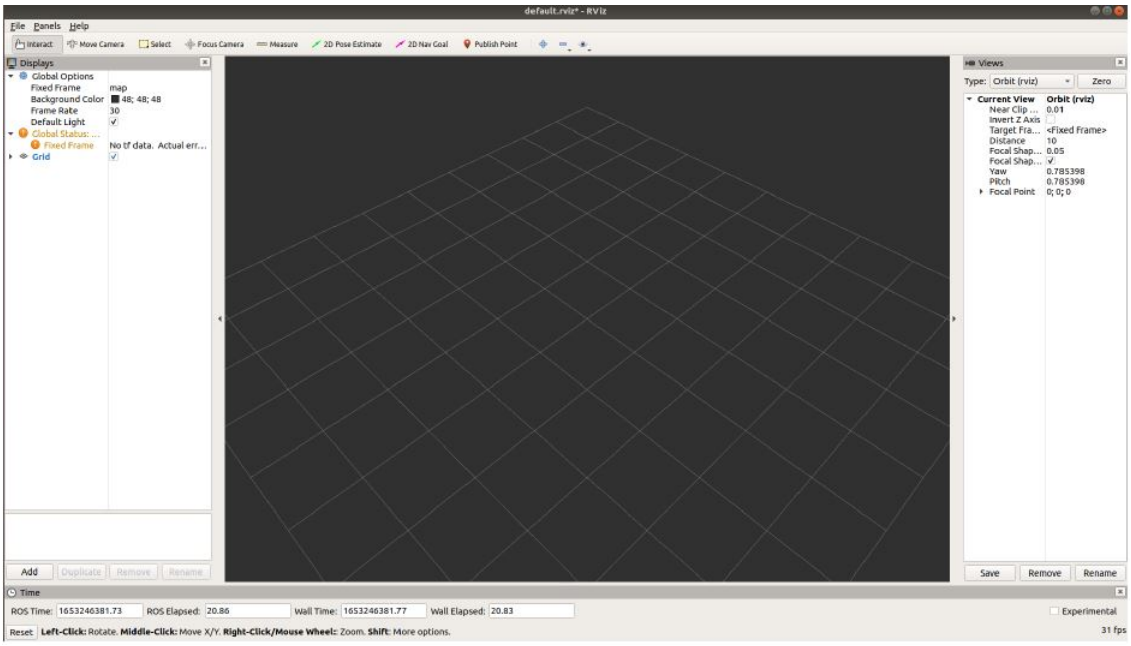

FIGURE 3.8 – Environnement Rviz

3.4.4 Simulateurs ROS

L'un des simulateurs robotiques open source étroitement intégré à ROS est Gazebo [2]. Gazebo est un simulateur robotique dynamique qui propose une grande variété de modèles de robots et une prise en charge étendue des capteurs. Les fonctionnalités de Gazebo peuvent être ajoutées via plugins. Les valeurs des capteurs sont accessibles par ROS en

utilisant les topics, des paramètres et des services. Gazebo peut être utilisé lorsque votre simulation nécessite une compatibilité totale avec ROS.

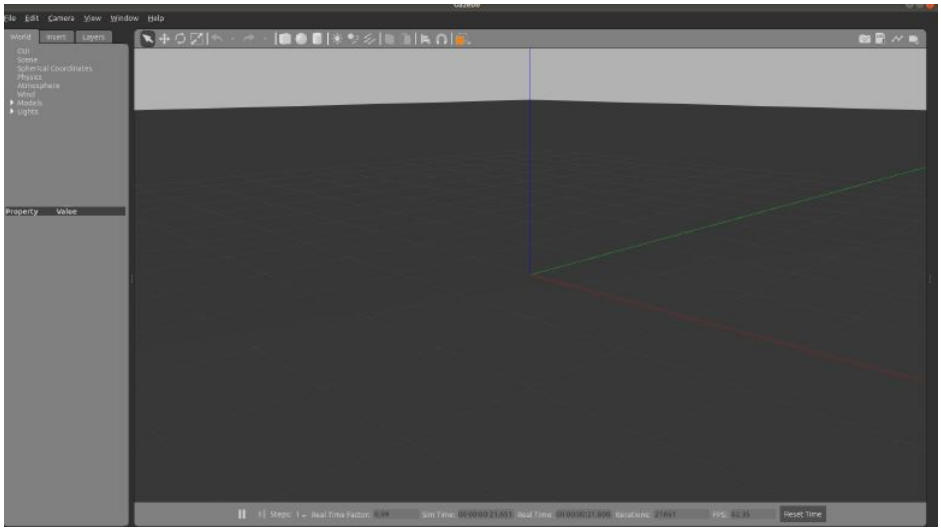

FIGURE 3.9 – Environnement Gazebo

3.4.5 Librairies Principales de ROS utilisées

Les bibliothèques ROS sont utilisées pour développer des noeuds ROS. Ainsi, nous pouvons simplement l'utiliser sans tout implémenter à partir de zéro. Les principales bibliothèques ROS sont en C++ et Python. Voici une liste des bibliothèques utilisées :

— roscpp : est une implémentation C++ de ROS. Il fournit une bibliothèque client qui permet aux programmeurs C++ de s'interfacer rapidement avec les rubriques, services et paramètres ROS . roscpp est la bibliothèque client ROS la plus utilisée et conçue pour être la bibliothèque hautes performances pour ROS.

— std_msgs : package ROS qui fournit des types de base de message pour communiquer entre les différents noeuds.

— gazebo_ros : Fournit des plugins ROS qui offrent des éditeurs de messages et de services pour s'interfacer avec Gazebo.

— geometry_msgs : Fournit des messages pour les primitives géométriques courantes telles que les points, les vecteurs et les poses. Ces primitives sont conçues pour fournir un type de données commun et faciliter l'interopérabilité dans tout le système.

3.4.6 Simulation

Nous avons créé un package ROS open source configurable qui peut être utilisé pour effectuer une navigation autonome avec des systèmes à entraînement différentiel. Le robot est piloté à l'aide de teleop_twist_keyboard (progiciel ROS).

On utilise le clavier pour contrôler le mouvement de notre base mobile. Grâce au modèle cinématique direct développé dans le fichier "kinematic_fonction.cpp" , nous traitons les mesures obtenus de la part des encodeurs attachés à chaque moteur de la base mobile avec les équations indiquées dans la section.

L'objectif fixé à été atteint, la base mobile se déplace d'un point A à un point B. Nous avons alors réussi à déterminer la position dans l'espace d'une base mobile à chaque instant t ainsi que son orientation

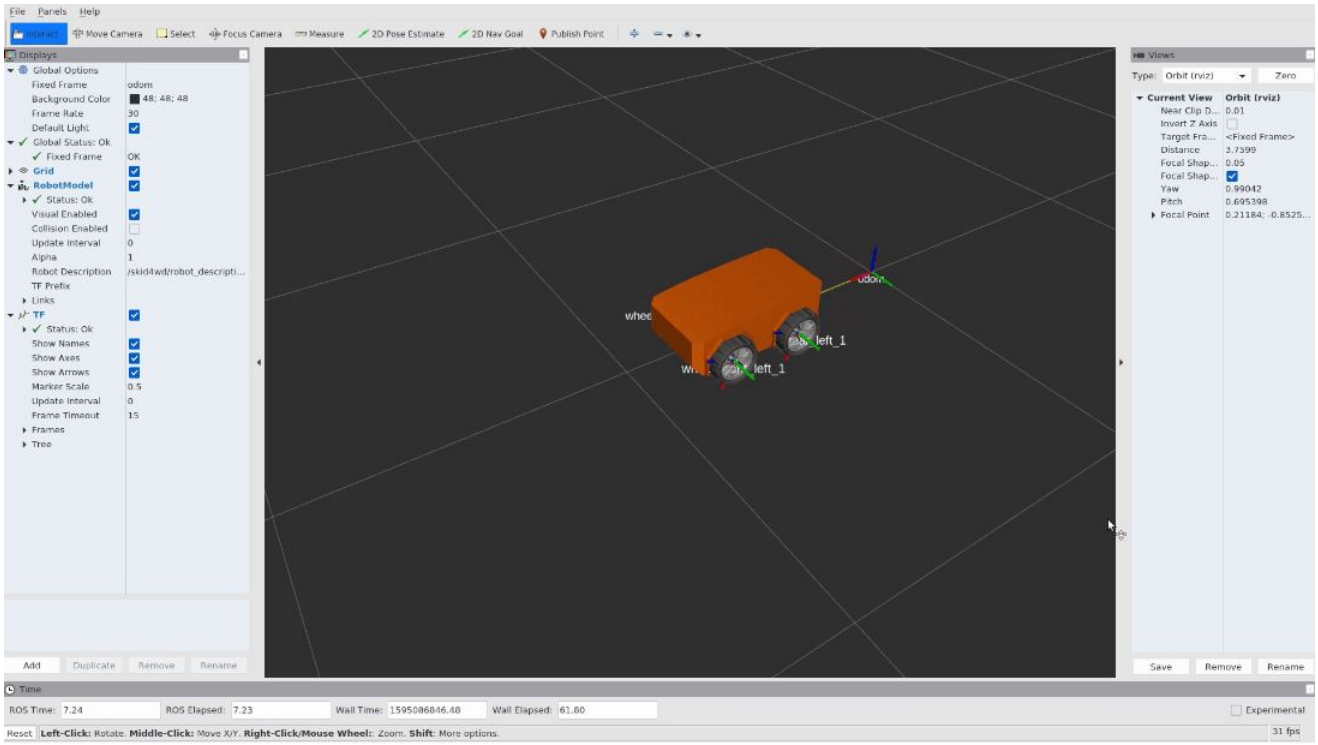

FIGURE 3.10 – Déplacement d'une base mobile à roues différentielles sous Rviz

Chapitre 4

Conclusions et perspectives

4.1 Conclusion sur le projet et perspectives

L'objectif du projet était de travailler sur la cohésion du modèle cinématique paramétrable et la réalisation d'une noeud ROS permettant l'obtention de la position et l'orientation des bases mobiles en utilisant les lois de commande.

Nous avons exploré différentes pistes, en particulier, la construction d'un modèle cinématique universel pour un robot mobile quelconque qui s'est avéré compliqué à mettre en œuvre car cette solution nécessite plusieurs phases de test et des approximations mathématiques. Cependant elle n'est pas pertinente dans une vision d'industrialisation de notre produit car cela ne permet pas la création de noeud ROS.

En outre, la mise en évidence de deux classes de robots mobiles utilisées chez RFS et la construction des modèles cinématiques s'est avérée intéressante dans la phase de test. Nous avons alors opté pour la création d'un noeud ROS à partir d'un programme codé en C++ afin d'être compatible avec l'ensemble des programmes déjà développés au sein de RFS. Nous avons alors réussi à déterminer la position dans l'espace d'une base mobile ainsi que son orientation.

Il manque dans la phase de validation, les tests sur les bases mobiles de RFS. Ce qui permettrait d'affiner le code et de fixer un maximum de paramètres actuellement variables et réglables par l'utilisateur. Des pistes d'améliorations sur des points particuliers de code ont également été communiquées à l'entreprise pour une reprise plus facile du projet lors de l'industrialisation.

4.2 Apport personnel de ce stage

D'un point de vue des compétences techniques, ce stage m'a permis de découvrir la mécanique industrielle, la robotique, la modélisation analytique qui consiste à prendre un système et à le modéliser à l'aide d'un système d'équations, d'apprendre à coder en C++ en suivant une syntaxe commune au sein du groupe. J'ai également appris à utiliser ROS, même si je n'ai pas acquis toutes les spécificités du fait que son utilisation dans mon projet est restée limitée.

Ce stage m'a également permis de développer des compétences transversales comme la prise de parole devant un public dans le but de défendre un projet, ou encore la réalisation d'une présentation synthétique.

J'ai également pu approfondir des compétences déjà acquises au cours d'expériences précédentes, telles que la gestion de mon emploi du temps ou les interactions entre les différents acteurs du projet. En effet, au sein de RFS, la capacité des stagiaires à être autonomes et maîtres de son projet est primordiale car c'est au stagiaire de solliciter les interlocuteurs pour des conseils, planification des réunions, présentation des résultats, etc.

Je remarque, à posteriori, qu'il m'est arrivé de me diriger sans le vouloir dans des impasses. Certaines solutions semblaient intéressantes à explorer mais étaient finalement peu pertinentes. J'aurai pu gagner du temps en prenant davantage de recul sur mon travail et mes objectifs finaux, toutefois ces travaux m'ont beaucoup appris.

L'organisation et le mode de fonctionnement d'une start-up et du prototypage ont été instructifs et m'ont permis d'avoir une vision globale des activités d' une entité de recherches et développements.

J'ai eu du mal à avancer depuis le départ de mon tuteur de stage, nous pouvions solliciter les ingénieurs référents, mais ils étaient dans la majorité du temps en déplacement.

Chapitre 5

Annexes

5.1 Présentation de l'organisme d'accueil

5.1.1 Le Groupe Vinci

FIGURE 5.1 – Logo Vinci

Actemium fait partie du groupe **VINCI** qui compte 222 397 collaborateurs répartis dans plus de **100 pays** et qui réalise un chiffre d'affaires de plus de **40 milliards d'euros**. Avec près de **2200 entreprises**, le groupe VINCI intervient dans un vaste panel d'activités réparties dans deux pôles de plusieurs sous-groupes chacun.

- **VINCI Concessions** : premier opérateur européen de concessions d'infrastructures de transport. Il exerce son activité dans les domaines des auto- routes, des aéroports, des ouvrages de franchissement, du ferroviaire et des stades.
- **VINCI Contracting** : réunit, au sein de 3 100 entités, un très grand ensemble d'expertises dans les métiers des énergies et des technologies de l'information, des travaux routiers et ferroviaires, du bâtiment et des travaux publics. Les 192 315 collaborateurs interviennent dans une centaine de pays et sur 270 000 chantiers par an.

5.1.2 Vinci Energies

FIGURE 5.2 – Logo Vinci Energies

La branche VINCI Energies rassemble près de 1500 entreprises. Il s'agit d'un acteur européen majeur et d'un leader français dans les secteurs des technologies de l'information et des énergies. L'activité de la branche se répartit autour de 4 domaines d'activités :

- **Infrastructures : Citeos** (éclairage public et équipements urbains dynamiques) et Omexom (construction de réseaux électriques de haute et très haute tension).
- **Industrie : Actemium** (développement et modernisation des équipements pour l'industrie et l'optimisation des outils de production) .
- **Tertiaire : VINCI Facilites** (services liés aux bâtiments et à leurs occupants, tels que la performance) énergétique et la gestion documentaire.
- **ICT : Axians** (communication et systèmes d'information).

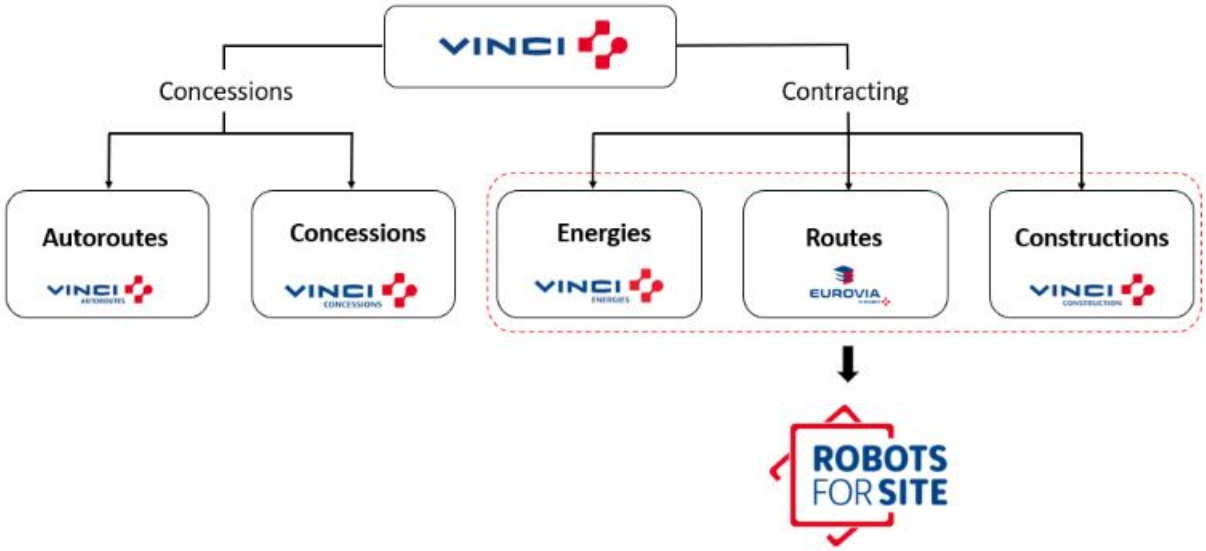

FIGURE 5.3 – La branche Vinci Energies

5.1.3 Le réseau Actemium

Actemium est la marque de **VINCI Energies** dédiée à l'industrie. Forte de 20.000 collaborateurs, son chiffre d'affaires annuel dépasse les 2 milliards d'euros. La vocation d'Actemium est d'améliorer l'avantage compétitif et la performance industrielle de ses

clients. La marque regroupe 400 entreprises réparties dans 41 pays, entièrement dédiées au développement et à la modernisation des processus industriels.

Actemium intervient sur les secteurs d'activité suivants :

- **Génie électrique** : fabrication d'armoires de contrôle-commande, conduite de chantiers (câblage, raccordements...), équipement de shelters. (construction de réseaux électriques de haute et très haute tension).
- **Génie mécanique** : production de machines personnalisées, études CAO, implantation de cellules robotisées, modification de la cinématique, conformité de l'installation.
- **Robotique Automatisme** : programmation d'automates, contrôle, robotique industrielle et mobile, programmation relais/automates de sécurité, programmation vision, programmation de supervisions, levées d'obsolescences.

Les robots mobiles évoluent de manière autonome dans l'espace, travaillent en groupe et offrent une flexibilité absolue à l'industrie. La dynamique du déplacement et le positionnement, des thèmes de recherche sur les robots mobiles qui mènent constamment à des développements surprenants.

A ce titre Robots For Site souhaite développer un module permettant de piloter leurs robots grâce à une série de lois de commande. Plusieurs études ont été menées sur les différentes bases mobiles par ses partenaires sans toutefois atteindre les objectifs souhaités.

Bibliographie

[1] Ros(robot operating system), https ://www.ros.org/.

[2] Simulateur gazebo, http ://gazebosim.org.

[3] yed - graph editor, https ://www.yworks.com/products/yed/download.

[4] Ahmed Alloum. *Modélisation et commande dynamique d'une automobile pour la sécurité de conduite*. PhD thesis, Compiègne, 1994.

[5] Bernard Bayle. Robotique mobile. *Ecole Nationale Superieure de Physique de StrasbourgUniversite Louis Pasteur*, 2007, 2008.

[6] Said Bentalba. *Methodes de synthese des lois de commande floue : Application a la robotique mobile*. PhD thesis, Amiens, 1999.

[7] Sašo Blažic, El-Hadi Guechi, Jimmy Lauber, Michel Dambrine, and Gregor Klancar. Path planning and path tracking of industrial mobile robots. In *Intelligent Industrial Systems : Modeling, Automation and Adaptive Behavior*, pages 84–124. IGI Global, 2010.

[8] Guy Campion, Georges Bastin, and Brigitte Dandrea-Novel. Structural properties and classification of kinematic and dynamic models of wheeled mobile robots. *IEEE transactions on robotics and automation*, 12(1) :47–62, 1996.

[9] Guy Campion, Brigitte d'Andrea Novel, and Georges Bastin. Modelling and state feedback control of nonholonomic mechanical systems. In *[1991] Proceedings of the 30th IEEE Conference on Decision and Control*, pages 1184–1189. IEEE, 1991.

[10] R Courant and D Hilbert. Methods of mathematical physics, vol. 2, interscience publishers, new york, london, 252–257. 1962.

[11] Brigitte d'Andrea Novel, Georges Bastin, and Guy Campion. Modelling and control of non-holonomic wheeled mobile robots. pages 1130–1131, 1991.

[12] Brigitte d'Andréa Novel, Guy Campion, and Georges Bastin. Control of nonholonomic wheeled mobile robots by state feedback linearization. *The International journal of robotics research*, 14(6) :543–559, 1995.

[13] Michael Defoort, Jorge Palos, Annemarie Kokosy, Thierry Floquet, Wilfrid Perruquetti, and David Boulinguez. Experimental motion planning and control for an autonomous nonholonomic mobile robot. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 2221–2226. IEEE, 2007.

[14] Ioan Doroftei, Victor Grosu, and Veaceslav Spinu. *Omnidirectional mobile robot-design and implementation*. INTECH Open Access Publisher London, UK, 2007.

[15] Gregory Dudek and Michael Jenkin. *Computational principles of mobile robotics*. Cambridge university press, 2010.

[16] Yawvi A Fiagbedzi and Allan E Pearson. A state observer for systems described by functional differential equations. *Automatica*, 26(2) :321–331, 1990.

[17] Zayd Kachour. *Contrôle du mouvement d'un robot mobile à entraînement différentiel*. PhD thesis, Université du Québec à Trois-Rivières, 2019.

[18] Frédéric Large. *Navigation autonome d'un robot mobile en environnement dynamique et incertain*. PhD thesis, Université de Savoie, 2003.

[19] Jean-Claude Latombe. Motion planning : A journey of robots, molecules, digital actors, and other artifacts. *The International Journal of Robotics Research*, 18(11) :1119–1128, 1999.

[20] Olivier Lefebvre. *Navigation autonome sans collision pour robots mobiles nonholonomes*. PhD thesis, Institut National Polytechnique de Toulouse-INPT, 2006.

[21] Alessandro De Luca, Giuseppe Oriolo, and Marilena Vendittelli. Control of wheeled mobile robots : An experimental overview. *Ramsete*, pages 181–226, 2001.

[22] Elie Maalouf, Maarouf Saad, and Hamadou Saliah. A higher level path tracking controller for a four-wheel differentially steered mobile robot. *Robotics and autonomous systems*, 54(1) :23–33, 2006.

[23] Pascal Morin and Claude Samson. Motion control of wheeled mobile robots. *Springer handbook of robotics*, 1 :799–826, 2008.

[24] Ju I Neımark and NA Fufaev. Dynamics of nonholonomic systems, volume 33 of translations of mathematical monographs. *American Mathematical Society, Providence, Rhode Island*, 1972.

[25] G. Bright J. Potgieter O. Diegel, A. Badve and S. Tlale. "improved mecanum wheel design for omni-directional robots,". page 27–29, November, 2013.

[26] Roland Siegwart and R Illah. Nourbakhsh, introduction to autonomous mobile robots. *hc alk. paper*, 1 :13–36, 2004.

[27] Olivier Stasse. *Robot Operating System*. english edition, 2016.

[28] Hamid Taheri, Bing Qiao, and Nurallah Ghaeminezhad. Kinematic model of a four mecanum wheeled mobile robot. *International journal of computer applications*, 113(3) :6–9, 2015.

[29] Rob PA van Haendel. Design of an omnidirectional universal mobile platform. 2005.

[30] UR Zimmer. Mobile robotics : Path planning and motion control. *Université Nationale d'Australie, Camberra, Australie*, 2001.