Contribution à la navigation de robots mobiles : approche par modèle direct et commande prédictive

Nicolas Morette

UNIVERSITÉ D'ORLÉANS

ÉCOLE DOCTORALE [SCIENCES DE L'HOMME ET DE LA SOCIETÉ ou SCIENCES ET TECHNOLOGIES]

INSTITUT PRISME, Équipe Systèmes Robotiques Interactifs

THÈSE présentée par :

Nicolas MORETTE

soutenue le : **18 Décembre 2009**

pour obtenir le grade de : **Docteur de l'université d'Orléans**

Discipline : Robotique

Contribution à la Navigation de robots mobiles : approche par modèle direct et commande prédictive

THÈSE dirigée par :

Pierre VIEYRES	Professeur, IUT de Bourges

RAPPORTEURS :

Philippe HOPPENOT	Professeur, IUT GEII Evry
Philippe MARTINET	Professeur, IFMA Clermont-Ferrand

JURY :

Dominique MEIZEL	Professeur, ENSI Limoges (président de jury)
Philippe HOPPENOT	Professeur, IUT GEII Evry
Philippe MARTINET	Professeur, IFMA Clermont-Ferrand
Pierre VIEYRES	Professeur, IUT de Bourges
Cyril NOVALES	Maître de conférences, IUT de Bourges
Laurence JOSSERAND	Maître de conférences, IUT de Bourges

Remerciements

Tout d'abord, je remercie Youssoufi Touré, Professeur à l'Université d'Orléans et directeur de l'Institut Pluridisciplinaire de Recherche en Ingénierie des Systèmes, Mécanique et Energétique (PRISME), ainsi que Gérard Poisson, chef de l'équipe Systèmes Robotiques Interactifs (SRI), de m'avoir accueilli au sein de l'équipe de recherche en robotique berruyère.

Je suis très reconnaissant envers Messieurs Philippe Hoppenot et Philippe Martinet, respectivement Professeur à l'Université d'Evry Val d'Essonne et Professeur à l'Université Blaise Pascal de Clermont-Ferrand, pour avoir accepté de juger mon travail. Merci également à Monsieur Dominique Meizel, Professeur de l'Université de Limoges, de m'avoir fait l'honneur de présider le jury de cette thèse. Je les remercie tous également de s'être déplacés pour assister à la soutenance malgré les conditions météorologiques difficiles.

Un très grand merci à Cyril Novales, Maitre de Conférence à l'université d'Orléans, pour son encadrement, sa droiture et toute l'aide qu'il m'a apporté au cours de cette thèse. Merci aussi de m'avoir fait confiance et m'avoir épaulé pour les enseignements à l'IUT et au CFBS.

Je remercie ensuite Pierre Vieyres, Professeur à l'université d'Orléans et directeur de cette thèse, pour son soutien au fil de ces 3 années.

Je suis également reconnaissant envers les membres de l'équipe robotique qui m'ont aidé durant ce travail, et tout particulièrement Laurence Josserand pour ses précieux conseils. Je pense également à Gilles Mourioux, cette thèse s'appuie en partie sur ses travaux et je le remercie d'avoir fait le déplacement pour la soutenance.

Mes pensées vont également vers les personnes qui ont contribué au développement du projet CyCab, entre autres Sylvain Lefrançois, Denis Blanc, Ant'thy Abdul Bari, Jérémy Depardieu, Noura Ayadi et aujourd'hui Frédéric Jacquet qui reprend le flambeau.

J'ai été heureux de travailler en compagnie des doctorants du laboratoire, je pense particulièrement à Laurent Arcèse, Fabrice Babet, Arnaud Capri, Tahar Slama, Gregory Abraham, Loïc Youinou, Christophe Boulnois, et mes collègues de toujours Gwenaël Charron et Antoine Belconde.

Je pense également à toutes les personnes qui m'ont aidé de différentes manières au cours de ces 3 ans, notamment Laure Spina qui a toujours été disponible en cas de besoin. Merci également à Jean Pierre Martin pour avoir accepté d'être mon tuteur pédagogique pour le monitorat, et à Jacques Scharf pour son soutien dans les enseignements. Pour terminer mes pensées vont envers mes principaux soutiens, à savoir ma famille.

TABLE DES MATIERES

Table des figures

Préface

Ce manuscrit est le fruit de 3 années de travail menées dans l'équipe Systèmes Robotiques Interactifs de l'institut PRISME de l'Université d'Orléans. Ce document présente une étude sur la navigation utilisant la modélisation directe d'un robot mobile avec une approche par commande prédictive.

Le premier chapitre est une introduction générale à la robotique mobile à roues, il présente les aspects de perception et contrôle des robots autonomes, ainsi que les thématiques de recherche associées. La problématique plus spécifique de la navigation est exposée dans le second chapitre, et un état de l'art des approches existantes est alors dressé. Une réflexion sur ces différentes méthodes nous conduit vers une alternative par projection de trajectoires qui sera l'objet du troisième chapitre. Nous y étudions notamment l'une de ces méthodes : la navigation par ligne de fuite. A l'issue de cette étude et de simulations soulignant les limites de cette approche, nous proposons alors d'utiliser la commande prédictive pour améliorer la projection de trajectoires. Ce navigateur par commande prédictive, adapté à la robotique mobile à roues, constitue le cœur du quatrième chapitre et de notre travail. Enfin le dernier chapitre présente les résultats de simulations de cette méthode appliquée au robot mobile CyCab. Nous proposons alors des solutions sur le choix des familles de trajectoires et celui des algorithmes d'optimisation pour obtenir les meilleures trajectoires.

CHAPITRE I

INTRODUCTION A LA ROBOTIQUE MOBILE

L'objectif de ce chapitre, dans le contexte de la robotique mobile, est de sensibiliser le lecteur sur les travaux qu'il reste à mener pour aboutir à la réalisation d'un robot mobile totalement autonome. Après une présentation des différents types de robots mobiles, et des contraintes de terrain sur lequel ils sont conçus pour évoluer, nous aborderons les contraintes principales liées à leur cinématique, et les solutions développées pour y remédier. Enfin nous étudions les outils permettant aux robots de percevoir leur environnement et de s'y repérer, étape primordiale nécessaire à l'autonomie totale des robots mobiles.

Cet état de l'art offre une vision non exhaustive des thématiques de recherche associées au domaine de la robotique mobile, et présente l'ensemble des verrous scientifiques qu'il reste à lever pour aboutir au développement d'un robot autonome. Parmi ceux-ci, nous nous focalisons alors sur celui de la navigation d'un robot mobile.

I. Robot mobile autonome

Pour commencer, il nous faut expliciter la notion de robot mobile. En robotique, on distingue les robots en deux principaux types : les robots manipulateurs et les robots mobiles. Les robots manipulateurs ont une base fixe contrairement aux robots mobiles qui peuvent se déplacer. Ainsi pour étudier les déplacements de ces robots, nous pouvons soit utiliser un repère allocentrique (absolu), soit un repère égocentrique (fixé sur le robot).

Bien souvent, quand on parle de robotique mobile, on sous entend robots mobiles à roues. Ce sont en effet les systèmes les plus étudiés, parce qu'ils sont plus simples à réaliser que les autres types de robots mobiles, ce qui permet d'en venir plus rapidement à l'étude de leur navigation. Ce type de robots est notamment très souvent utilisé pour l'étude des systèmes autonomes. Vient ensuite la robotique mobile à pattes, avec notamment la robotique humanoïde, mais également des robots avec un nombre de pattes plus élevés qui offrent de bonnes propriétés pour la locomotion en milieu difficile (milieux forestiers et agricoles). La stabilité des mouvements de ce type de robots est en particulier un thème de recherche important [27]. Enfin il existe également de nombreux autres types de robots mobiles (robots marins [30], sous marins [21], drones volants, micro et nano

robots), généralement l'étude de ce type de robots se fait dans des thématiques spécifiques avec des problèmes particuliers à l'application visée.

Il existe 2 principaux modes de fonctionnement pour un robot mobile : télé-opéré et autonome. En mode télé-opéré, une personne pilote le robot à distance. Elle donne ses ordres via une interface de commande (joystick, clavier/souris…), et ceux-ci sont envoyés au robot via un lien de communication (internet, satellite …). D'ailleurs, suivant le niveau de téléopération, le terme « robotique » est plus ou moins justifié. Le robot doit donc obéir aux ordres de l'opérateur qui perçoit l'environnement autour du robot, par différents moyens (retour d'image, retour haptique…), de manière à donner des ordres adaptés au robot. Dans ce domaine, les efforts de recherche sont beaucoup portés sur les problèmes liés au réseau de télécommunication (retards dans le réseau de communication, problèmes de commande, pertes de données) et sur l'amélioration de la perception de l'environnement par l'opérateur (interfaces haptiques, retours d'efforts).

A l'inverse, en mode autonome le robot doit prendre ses propres décisions. Cela signifie qu'il doit être capable à la fois de percevoir correctement son environnement, mais également de savoir comment réagir en conséquence, suivant le niveau d'autonomie. C'est à lui de planifier son parcours et de déterminer avec quels mouvements il va atteindre son objectif. Les recherches dans ce domaine portent principalement d'une part sur la localisation du véhicule autonome et la cartographie de son environnement, d'autre part sur le contrôle de tels véhicules (structure de contrôle, stratégies de commande, planification).

Cette notion d'autonomie prise en exemple ci-dessus, que nous pourrions qualifier de décisionnelle, ne doit pas être confondue avec celle d'autonomie énergétique (capacité du robot à gérer efficacement son énergie, à la préserver, voire à se ravitailler), même si ces deux notions sont étroitement liées : idéalement une des préoccupations principales d'un robot mobile totalement autonome (du point de vue décisionnel), serait en effet de pouvoir gérer de lui-même ses réserves d'énergie.

Voyons maintenant les différents types d'environnement dans lesquels les robots mobiles sont amenés à se mouvoir.

II. Les différents types de terrain

Nous rencontrons principalement 3 types d'espaces de navigation : les terrains plats, les terrains accidentés et les espaces 3D. Les terrains plats sont généralement utilisés pour modéliser les milieux urbains et les intérieurs de bâtiments. Le robot évolue sur un plan 2D considéré sans pentes, et tout objet qui sort de cet espace 2D est considéré comme un obstacle (Figure 1- *Robucab [15]*). Cette représentation est la plus simple à étudier et la plus répandue pour les robots mobiles à roues. En première approche, elle permet de se concentrer sur les problèmes de contrôle et de navigation autonome du robot.

Figure 1- *Robucab [15]*

Les terrains accidentés (ou $2D\frac{1}{2}$) correspondent généralement aux milieux en extérieur, comme des forêts, des champs en robotique agricole, ou encore des terrains rocheux (Figure 2). La différence avec les terrains plats est la présence de pentes, de bosses et de creux sur le terrain d'évolution du robot. Cela interdit d'utiliser une métrique standard 2D et cela complique pour beaucoup la détection d'obstacles et la modélisation des déplacements du robot. De plus il devient également important de vérifier que le robot ne bascule pas quand il escalade une pente ou enjambe un obstacle. Le système de locomotion du robot doit dans ce cas être adapté à la topologie du terrain.

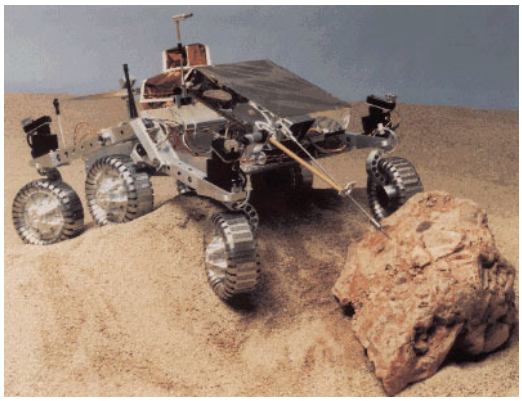

Figure 2- Le robot martien Sojouner [71]

Enfin les espaces d'évolution 3D sont par exemple utilisés pour modéliser la navigation des drones volants (Figure 4) et des robots sous-marins (Figure 3). Les problèmes rencontrés sont spécifiques à l'application visée.

Chaque type de terrain correspond à des problématiques bien spécifiques. Le type de robot étudié dans ce mémoire est destiné à circuler en environnement urbain, la modélisation terrain plat sera utilisée. Cela signifie que l'on considère que tous les mouvements sont contenus dans un plan de navigation, parallèle au sol.

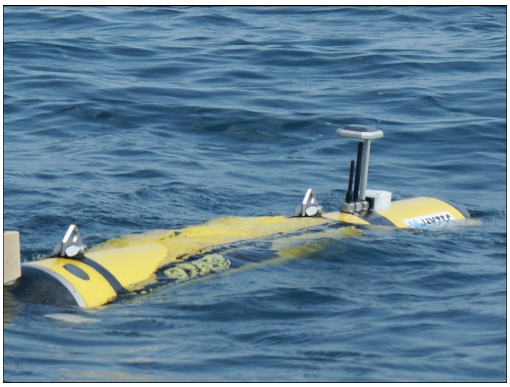

Figure 3- Le robot sous marin TAIPAN[21]

Figure 4- Drone de surveillance RESSAC de l'ONERA [25]

Pour la modélisation terrain plat, nous définissons un repère absolu (fixé dans l'environnement) $R = (0, \vec{x}, \vec{y}, \vec{z})$ donc l'axe $\vec{z}$ est perpendiculaire au sol. Nous définissons un repère mobile lié au robot $R' = (0', \vec{x}', \vec{y}', \vec{z}')$, dit égocentrique. Le point O' est le point de contrôle du robot. Généralement, sur un robot type voiture, le point de contrôle est fixé au centre de l'essieu non directeur. Ce repère égocentrique se déplace avec le robot. Pour réaliser une navigation, l'état du robot est totalement défini par le vecteur :

$$X = \begin{pmatrix} x \\ y \\ \theta \\ \dot{s} \end{pmatrix}, \qquad \text{Équation 1}$$

dans lequel θ désigne l'orientation du robot dans le plan $(0, \vec{x}, \vec{y})$ et $\dot{s}$ sa vitesse curviligne (Figure 5).

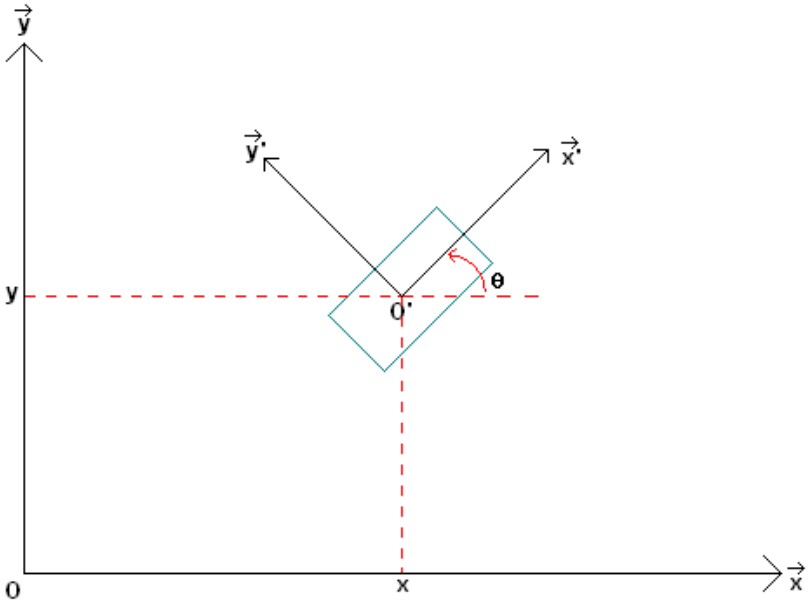

Figure 5- Modélisation du robot dans le repère absolu

La section suivante présente différents aspects de la modélisation sur sol plat : la classification des différents types de roue, les configurations holonomes et non holonomes, et la gestion des glissements dans le modèle.

III. Modélisation

1. Classification des types de roues

La mobilité d'un robot mobile dépend grandement du type de roues utilisées. Dans [12], Campion a présenté une classification des différents types de roues et configurations rencontrées en robotique mobile. Les différents types de roues que l'on rencontre sont :

- la roue fixe : cette roue n'autorise qu'un déplacement dans la direction de son plan médian, l'orientation n'est pas modifiable,
- la roue centrée orientable : elle possède un axe d'orientation en plus de l'axe de rotation, et cet axe d'orientation passe par le centre de la roue,
- la roue décentrée orientable ou roue folle : son axe d'orientation ne passe pas par le centre de la roue (c'est le cas par exemple des roues des chaises de bureau)

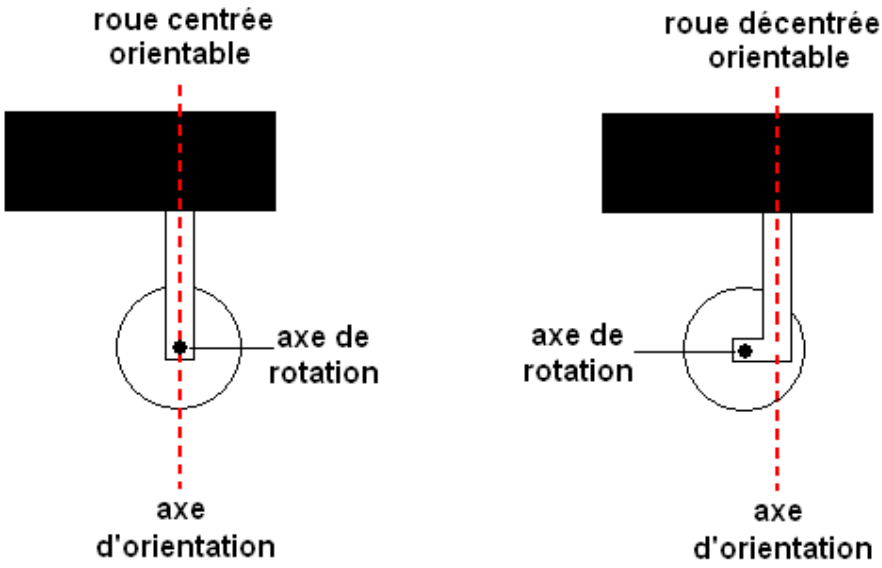

Figure 6- Roue centrée et roue décentrée

En plus de ces roues classiques (Figure 6), d'autres roues ont été développées pour accroitre la mobilité du robot. Elles permettent d'augmenter les capacités de déplacement dans toutes les directions du plan (Figure 7). Cependant, elles ne sont commandables que dans certaines de ces directions. Dans cette catégorie nous trouvons notamment :

- les roues suédoises : ces roues autorisent les glissements latéraux grâce à un système de galets remplaçant la bande de roulement classique, montés en inclinaison par rapport au plan de la roue. La combinaison de la rotation de la roue avec la rotation libre du galet en contact avec le sol permet le déplacement sans glissement sur le sol dans toutes les directions. Cependant le couple moteur que l'on peut transmettre à ces roues est très limité, ce qui réduit son utilisation en pratique.

- les roues tronco-sphériques (ou orthogonal wheels) [56] : cette structure utilisant deux roues libres en quadrature présente l'avantage de pouvoir transmettre un couple intéressant par rapport aux roues suédoises, mais souffre de petits problèmes de sauts au moment de la transition d'une roue support à l'autre.

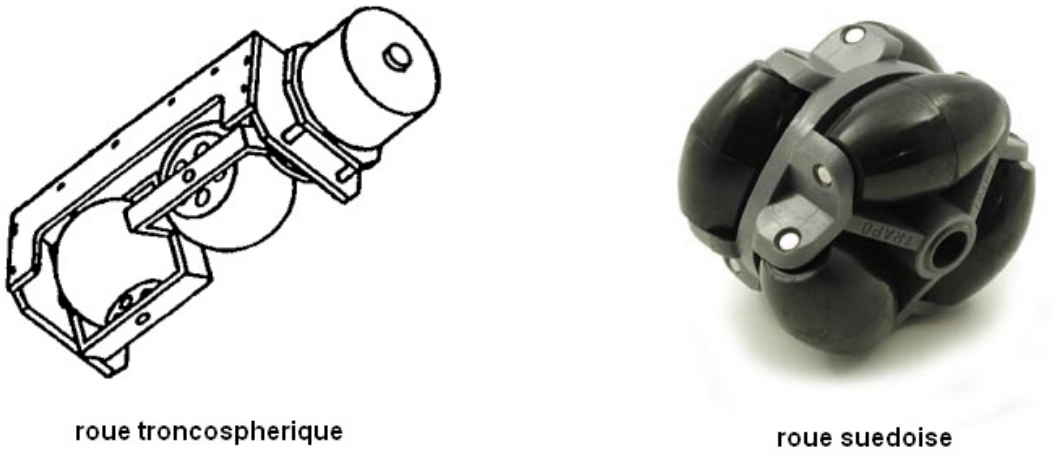

Figure 7- Roue troncosphérique et roue suédoise

2. Principales structures cinématiques

En associant les différents types de roues selon une structure mécanique donnée, le robot mobile disposera de plus ou moins de mobilité. Le nombre, le type et la disposition des roues engendrera ou non la contrainte de non holonomie du robot. Si on néglige les phénomènes dynamiques tels que l'inertie, un robot holonome est un robot capable à chaque instant de se déplacer dans n'importe quelle direction du plan, sans avoir à effectuer une reconfiguration de ses roues.

Tout système évoluant dans un plan 2D possède 3 degrés de liberté : une translation selon l'axe x, une translation selon l'axe y et une rotation autour d'un axe z normal à $(\vec{x}, \vec{y})$. Cependant une roue classique ne possède que 2 degrés de mobilité : elle ne peut que faire une translation (avancer ou reculer), ou une rotation sur elle-même. Elle ne peut pas déraper transversalement pour effectuer un mouvement de translation horizontal.

Cette contrainte empêche la plupart des véhicules « traditionnels » d'effectuer un déplacement instantané transversal (parallèlement à l'axe de rotation de la roue). Une voiture ne peut pas effectuer de créneau pour se garer, sans faire un certain nombre de manœuvres. C'est une contrainte que l'on retrouve sur tous les robots mobiles de type voiture ou à roues différentielles. De tels véhicules, possédant un nombre de degrés de mobilité inférieur au nombre de degrés de liberté, sont dits non-holonomes, et cette contrainte touche principalement les robots mobiles à roues.

Les configurations non-holonomes les plus courantes sont :

- le tricycle / la voiture : ces deux structures sont constituées d'un axe fixe (généralement à l'arrière) et d'un axe directeur (Figure 8). Dans le cas du tricycle, seule une roue est présente sur l'axe directeur, contrairement à la voiture qui en possède deux. La théorie d'Ackerman-Jeantaud

donne les conditions théoriques de non glissement et non dérapage pour les configurations de type voiture. Notamment les axes de rotation des 4 roues doivent s'intersecter en un point unique, le Centre Instantané de Rotation. Pour cela, la vitesse de la roue extérieure doit être légèrement supérieure à celle de la roue intérieure. La structure de type voiture peut être modélisée par une structure équivalente à 3 roues, ce qui revient au modèle du tricycle.

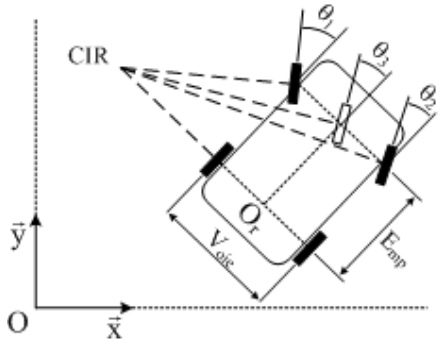

Figure 8- Equivalence modèle voiture et tricycle

- les roues différentielles (Figure 9) : cette structure également classique est constituée de deux roues motrices placées sur le même axe, et d'au moins un appui supplémentaire (généralement une ou deux roues folles). L'avantage de cette structure est qu'elle permet au véhicule de tourner sur place, suivant si les vitesses de rotation des deux roues motrices sont de signe opposées ou pas. Ainsi le robot peut pivoter rapidement, ce qui donne des capacités de déplacement intéressantes. Cependant, le déplacement latéral n'étant pas directement réalisable, cette structure n'est pas non plus holonome.

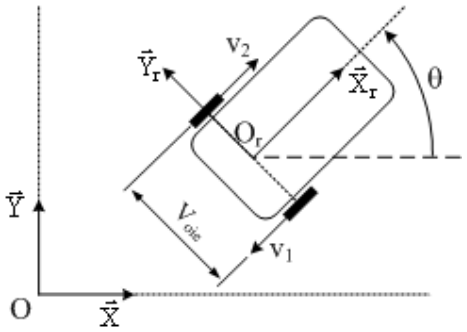

Figure 9- Modèle roues différentielles

En utilisant des roues telles que les roues suédoises ou tronco-sphériques sur des plateformes mobiles, des robots ayant la capacité de se mouvoir dans toutes les directions ont été créés. Ces robots, à 3 degrés de mobilité dits omnidirectionnels, permettent de s'affranchir de la contrainte de non holonomie. Leurs structures spécifiques leur permettent de se déplacer instantanément (à la dynamique près) dans toutes les directions en ayant n'importe quelle orientation, rendant possible le suivi de trajectoires de forme quelconque (Figure 10).

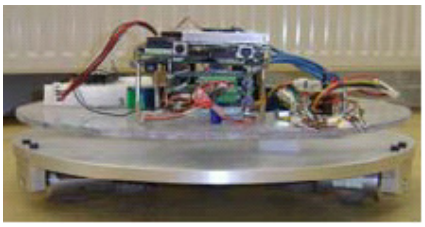

Figure 10- Le robot omnidirectionnel ROMNI [52]

3. Roulement avec ou sans glissement

La locomotion se fait grâce au frottement entre la roue du véhicule et le sol, et l'efficacité du mouvement dépend notamment du type de sol. Pour que l'hypothèse du roulement sans glissement soit validée, il faudrait théoriquement que le contact sol/roue ne se fasse qu'en un point, que le sol soit parfaitement plat, et que le rayon de la roue soit parfaitement constant sur toute sa périphérie.

Soient C le point de contact entre la roue et le sol, G le centre de rotation de la roue et $\dot{\alpha}$ sa vitesse de rotation (Figure 11).

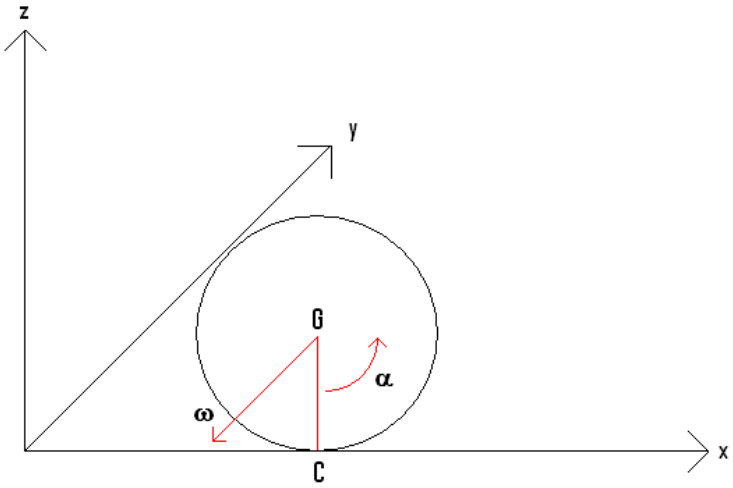

Figure 11- Roulement sur le sol

Mathématiquement, cette contrainte revient à dire que la vitesse de C par rapport au sol est nulle, ce qui peut s'écrire grâce à la loi de composition des vitesses :

$$\begin{aligned} &\overrightarrow{V_G} + \overrightarrow{CG} \wedge \vec{\omega} = \vec{0} \\ &\dot{x}\vec{x} + R\vec{z} \wedge (-\dot{\alpha}\,\vec{y}) = \vec{0} \\ &\dot{x}\vec{x} + \dot{\alpha}R\vec{x} = \vec{0} \\ &\dot{x} = -\dot{\alpha}R \end{aligned} \quad , \qquad \text{Équation 2}$$

On en déduit que, pour vérifier la contrainte de roulement sans glissement, le mouvement doit rester dans le plan de la roue avec la vitesse curviligne $\dot{\alpha}R$.

En réalité, le contact sur le sol se fait sur une surface avec le pneu de la roue. Les glissements sur le sol sont une source d'erreur importante pour certaines méthodes de localisation. Cela est notamment le cas pour l'odométrie classique, qui s'appuie sur cette hypothèse pour déterminer la position relative d'un robot par rapport à son point de départ, à partir de la mesure du nombre de tours parcourus par chaque roue. On sait que dans ce cas l'intégration des dérives dues aux glissements, entraine sur un parcours complet d'importantes erreurs de localisation.

Dans certaines applications robotiques, la précision du déplacement est un facteur important, il est alors nécessaire de prendre en compte les glissements dans la modélisation du robot. Il existe deux principales approches pour les intégrer : soit on passe à un modèle dynamique du robot, soit on reste sur un modèle cinématique classique (roulement sans glissement) dans lequel on introduit les effets du glissement sous forme de paramètres supplémentaires. La modélisation dynamique des phénomènes de glissements est plus complète mais elle requiert d'une part la mesure ou l'estimation, en temps réel, d'un grand nombre de paramètres tels que les coefficients de frictions, et d'autre part de paramètres supplémentaires inhérents à l'état des pneumatiques [23][57]. Ces nombreux paramètres ne sont pas triviaux à obtenir en ligne, ce qui limite l'utilisation de tels modèles en pratique. Les modèles cinématiques, modifiés pour intégrer les glissements, s'avèrent plus simple à mettre en œuvre de part le nombre réduit de paramètres à estimer (on ne modélise plus les phénomènes complets de glissement, mais juste leur effet sur la cinématique du véhicule), et ils permettent tout de même de prendre en compte avec une très bonne précision les phénomènes de glissement. Par exemple, dans [39], Lenain développe un modèle cinématique étendu qui lui permet d'améliorer la précision du suivi de trajectoire par un véhicule agricole sur un terrain particulièrement glissant.

IV. Détection d'obstacles et localisation

La perception de son environnement d'évolution est la base de tout système autonome. Sans une bonne perception et interprétation de ce qui l'entoure, un robot ne peut pas prendre de décision correcte. Cette partie vise à décrire les différents moyens mis à disposition au robot pour localiser les obstacles qui l'entourent. Ensuite les différentes méthodes de localisation du robot lui-même sont abordées. L'idée est de permettre au final de créer un modèle, plus ou moins simplifié, des interactions entre le robot et son environnement. Cette étape est nécessaire et primordiale pour la navigation d'un robot mobile autonome.

Pour cela, un robot est équipé de capteurs proprioceptifs qui fournissent des informations sur le robot lui-même, et extéroceptifs qui fournissent des informations sur ce qu'il y a autour de lui (son environnement).

1. Détection d'obstacles et cartographie

Les capteurs permettant de fournir des informations sur l'environnement extérieur peuvent être classés en deux catégories, passifs et actifs [61]. Dans le premier cas, on se contente de recueillir et d'analyser une énergie fournie par l'environnement, typiquement la lumière. Dans le second cas, c'est au capteur de générer une énergie, et de récupérer cette énergie après interaction sur le milieu extérieur. C'est le principe de base des télémètres (capteurs de mesures de

distances), qui sont largement utilisés pour tracer des cartes en ligne de l'environnement dans lequel évolue le robot. Les télémètres laser à balayage sont fréquemment utilisés pour la navigation de robots avec de très bonnes performances notamment en intérieur.

Le principe de ces télémètres repose sur le calcul du temps aller-retour mis par une impulsion lumineuse pour revenir sur le capteur. Une onde infrarouge de faible puissance est émise par la diode laser, et au même moment un chronomètre informatique est lancé. L'onde se réfléchit sur le premier objet rencontré en chemin, et revient sur le détecteur du capteur. Le temps mis par l'onde pour faire l'aller-retour permet de déterminer la distance de l'objet. Un miroir tournant motorisé permet de balayer toute une gamme d'angles devant le télémètre, dans le plan de balayage qui est parallèle au sol. La précision de ces appareils et leur robustesse aux variations de température en font des outils très intéressants pour les applications en robotique mobile de faible/moyenne vitesse.

Les capteurs ultrasonores utilisent des ondes sonores de fréquence non perceptible par l'oreille humaine, généralement dans la fourchette 20-200 khz. De la même manière que les télémètres laser, ils sont basés sur le principe de la mesure du temps aller-retour lors de la réflexion sur un obstacle. C'est la méthode employée par certains animaux pour percevoir leur environnement, comme les chauves-souris ou les dauphins : l'écholocalisation. Un avantage de ces capteurs est que contrairement aux télémètres, l'onde qu'ils émettent n'étant pas focalisée, ils perçoivent beaucoup plus facilement des éléments filiformes comme des pieds de chaises ou des grillages. Par contre leur portée est faible, et ils sont moins adaptés aux milieux de propagation non isotropes comme l'air.

Un des inconvénients des capteurs ultrasonores par rapport aux télémètres lasers est la divergence importante du faisceau ultrasonore, qui s'apparente plus à un cône qu'à un faisceau. Généralement l'ouverture de l'angle est de plusieurs dizaines de degrés, ce qui rend la localisation des obstacles imprécise. Ces capteurs sont donc plutôt utilisés pour des mesures à courte distance (de quelques centimètres à quelques mètres). Ils sont relativement sensibles aux variations de température, et la fréquence de mesure dépend de la distance maximale de détection (plus cette distance est grande, moins la fréquence d'acquisition des mesures est élevée). L'avantage de ces capteurs est qu'ils sont moins onéreux qu'un télémètre laser, et ils sont souvent utilisés dans des applications en intérieur avec des espaces de navigation assez restreints.

Les capteurs passifs se servent directement de l'énergie émise par l'environnement. C'est typiquement le cas des systèmes de vision par caméra en stéréo vision. La reconnaissance de primitives entre deux images permet d'évaluer la position/orientation d'un objet, et ainsi d'évaluer la profondeur. L'utilisation simultanée de deux caméras est cependant nécessaire pour y parvenir. Plus de deux caméras peuvent également être utilisées, de manière à améliorer la robustesse de la méthode.

La vision omnidirectionnelle s'avère également très intéressante dans le cadre d'applications en robotique mobile, dans le sens où elle permet de surveiller en même temps tout ce qui se passe autour du robot. La caméra est placée face à un miroir parabolique ou hyperbolique. L'image est complètement distordue ce qui complique la mesure de distances, mais l'aspect vision panoramique offre des avantages pour l'évitement d'obstacles dynamiques en environnement structuré. Les verticales deviennent des radiales, et les horizontales des arcs de cercle. L'utilisation de systèmes à base de vision est fortement développée, et pas seulement dans

le domaine de la robotique. Mais globalement, ce type de système reste fortement tributaire de la qualité de l'énergie recueillie : influence de la luminosité ou encore du contraste.

2. Fusion de données multi capteurs et cartographie

La localisation d'un robot mobile s'effectue par la mise en correspondance de différentes sources extéroceptives et proprioceptives. Généralement il s'agira de confronter les mesures de déplacements prises par odométrie avec une méthode de localisation absolue : soit reconnaissance et calcul de distance par rapport à des balises de position connue, soit mise en correspondance avec une carte construite en ligne et/ou présente dans une base de données, soit encore localisation externe du robot par des capteurs dans l'environnement (GPS). Le principe le plus simple pour effectuer cette mise en correspondance, consiste à utiliser les mesures de localisation absolue pour recaler périodiquement l'état du robot, obtenu par intégration des déplacements mesurés par l'odomètre. Cette méthode, bien que simple à utiliser, présente le problème de ne pas utiliser conjointement les différents moyens de mesure, mais successivement. Ainsi on ne tient pas compte des incertitudes liées tant à l'odométrie qu'à la méthode de localisation absolue.

Cependant nous savons que quelque soit la technologie utilisée pour la prise d'informations, aucune mesure n'est parfaite, il existe toujours une part d'incertitude sur celle-ci. Ces incertitudes peuvent provenir soit du principe de mesure lui même, soit des imperfections technologiques. Typiquement pour une mesure de distance avec un télémètre laser, on trouve des erreurs systématiques d'une quinzaine de millimètres en moyenne (erreur constante intrinsèque au télémètre utilisé), et une erreur statistique de 5 mm environ. Lorsque plusieurs méthodes de mesure sont utilisées conjointement, le principe utilisé pour mettre en concordance les informations consiste à effectuer une moyenne pondérée des différentes mesures par la confiance que l'on accorde à chacune (inversement à leur variance donc). Soit deux mesures z_1 et z_2 d'une même variable x, obtenues par des capteurs différents, avec des variances associées σ_1 et σ_2, alors la loi de Bayes nous donne la valeur estimée de x:

$$\hat{x} = \frac{{\sigma_2}^2}{{\sigma_1}^2 + {\sigma_2}^2}.z_1 + \frac{{\sigma_1}^2}{{\sigma_1}^2 + {\sigma_2}^2}.z_2 , \qquad \text{Équation 3}$$

et la variance associée à l'estimée :

$$\sigma = \frac{1}{\frac{1}{\sigma_1} + \frac{1}{\sigma_2}} , \qquad \text{Équation 4}$$

La variance associée à cette estimation est plus faible que les variances de chacune des mesures prises séparément (ce qui est logique, cela traduit simplement le fait que plus on recoupe d'informations provenant de sources différentes, plus on diminue les incertitudes). Cette méthode est intéressante si l'on dispose de ressources limitées pour le calcul, mais l'estimation peut être largement améliorée en utilisant les informations sur les mesures passées, et en filtrant avec un filtre de Kalman.

Pour des obstacles statiques, nous obtenons des mesures récurrentes selon une certaine fréquence d'acquisition. Pour des mesures récursives, la précision peut être améliorée en utilisant

un filtre de Kalman. Ce type de filtrage, très utilisé notamment en automatique, est un filtre statistique qui permet de réduire les incertitudes au fur et à mesure de l'acquisition de nouvelles mesures. Cette méthode est particulièrement utilisée en robotique pour la localisation du robot relativement aux obstacles [32]. L'algorithme utilise les connaissances sur la dynamique du robot et du système de mesure et sur les incertitudes associées à chaque mesure. Le calcul s'effectue en deux phases : une phase de prédiction de la mesure et de sa variance, suivie d'une phase de mise à jour de celle-ci par l'acquisition de nouvelles mesures. L'historique des mesures n'a pas besoin d'être gardé en mémoire, et le processus est récursif.

Pour pouvoir planifier les déplacements du robot, il est nécessaire d'établir une modélisation de l'environnement à partir des mesures des positions relatives des obstacles par rapport au robot. Concrètement il s'agit d'établir une cartographie locale des espaces où le robot pourra circuler, ou non, en considérant la position absolue du robot comme connue. Il existe 2 grands types de représentation pour l'environnement local : les cartes géométriques et les grilles d'occupation. Les premières peuvent être obtenues par traitement des données issues de mesures télémétriques, et en effectuant une reconnaissance des formes simples (murs, coins). A partir d'une connaissance des déplacements du robot et en comparant la carte courante avec des cartes mises en mémoire au fur et à mesure que le robot se déplace, la robustesse de la carte peut être améliorée [13]. Les grilles d'occupation sont des cartes discrétisées, généralement sous forme de grille d'un certain nombre de lignes et de colonnes. A chaque case de la grille est soit associée une valeur booléenne pour dire si la case est accessible par le robot ou non (occupée par un obstacle ou libre), soit une probabilité d'occupation (loi de Bayes).

3. Localisation

Les outils permettant la localisation d'un robot dans son environnement peuvent être classés en deux catégories : ceux par localisation à l'estime et ceux par localisation absolue [61]. Le principe de la première catégorie consiste à intégrer des informations sur les vitesses ou les accélérations fournies par des capteurs proprioceptifs (odomètres, centrales inertielles). L'avantage de ces méthodes est qu'elles sont indépendantes de l'environnement, par contre leur souci est leur manque de précision dû à la dérive temporelle. En effet les erreurs s'intégrant elles aussi au fur et à mesure du temps, il est nécessaire d'apporter régulièrement des recalages (Figure 12).

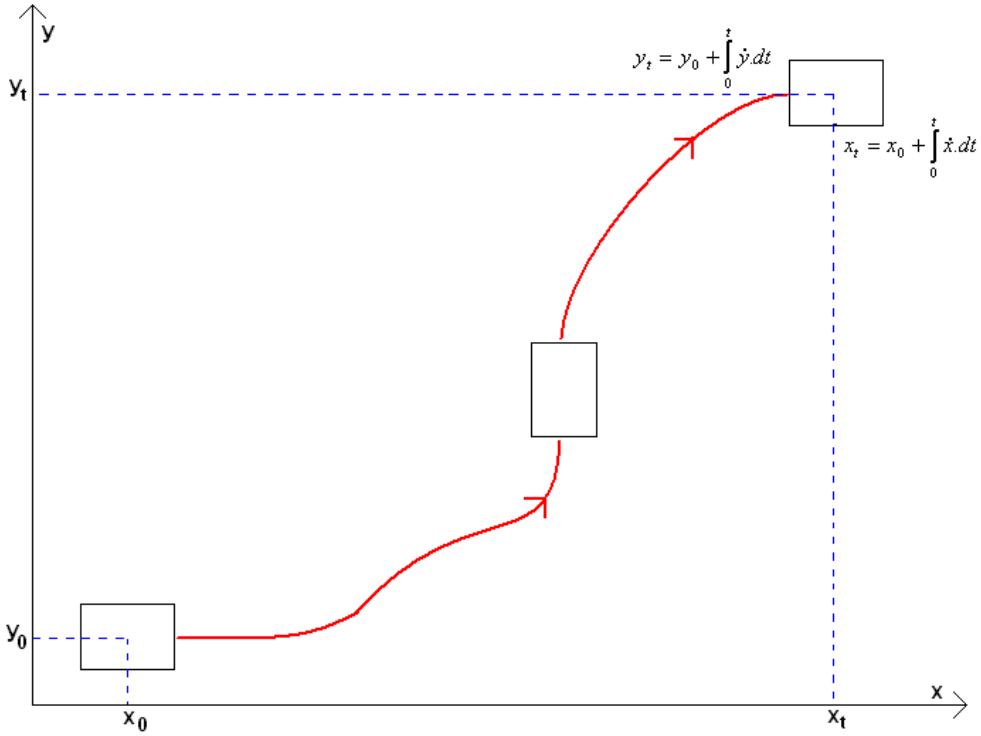

Figure 12- Localisation à l'estime

Parmi les méthodes de localisation à l'estime, le système le plus simple et le plus couramment utilisé pour la mesure de déplacement du robot est l'odométrie. L'hypothèse de roulement sans glissement que nous avons vu précédemment, nous permet de relier directement les déplacements du robot à la vitesse de rotation des roues. Par intégration des déplacements à chaque instant, on en déduit la position relative du robot par rapport à son point de départ. L'odométrie est une méthode de localisation très courante, simple, mais également très rapidement imprécise. En effet à cause du glissement des roues sur le sol, les erreurs s'accumulent au fur et à mesure que le robot avance, ce qui implique d'importantes erreurs sur les longs parcours s'il n'y a pas de recalage régulier. Cette méthode est de ce fait fortement tributaire de la qualité du sol sur lequel le robot se déplace.

Les incertitudes sur le diamètre exact des roues, sur les paramètres géométriques du robot, sur la résolution des codeurs, génèrent des erreurs de type systématique, qui vont s'accumuler très rapidement en odométrie. Cependant, ces erreurs peuvent être identifiées et évaluées pour faire un recalibrage du système et ainsi améliorer sa précision. Les erreurs non systématiques comme les glissements ou les irrégularités du sol, génèrent moins rapidement des erreurs, mais ne peuvent par contre pas être recalibrées puisqu'on ne peut pas les prévoir.

Pour l'exploration martienne, où le terrain est fortement accidenté, l'utilisation de système d'odométrie classique est impossible. Pour cette application, Cheng a proposé une technique d'odométrie alternative, dite odométrie visuelle, basée sur la reconnaissance de points singuliers dans l'image video du sol fournie par une caméra montée sur le robot [71] (les points de Harris). Connaissant le positionnement de la caméra par rapport au robot, le déplacement de ces points dans l'image permet d'évaluer les vitesses de déplacement du robot, et par intégration, de retrouver sa localisation relative par rapport à sa position initiale.

Le second type de méthode pour la localisation est la localisation absolue. Ces méthodes utilisent des éléments repérables par le robot dans l'environnement de navigation, de position connue, pour permettre au robot de se repérer relativement à ceux-ci. Ces éléments sont appelés des balises ou amers et sont dits soit réels, s'ils ont été placés spécialement pour permettre la localisation, soit virtuels s'il s'agit d'éléments présents naturellement.

Les balises réelles sont dites passives si elles ont pour but de réfléchir un signal émis par un appareil de mesure du robot (laser ou infrarouge). Il existe deux méthodes pour utiliser ces balises pour la localisation du robot : la méthode télémétrique (calcul de la distance robot/balise), qui nécessite la présence de deux balises pour calculer la position du robot dans le plan ; et la méthode par triangulation, qui consiste à mesurer les angles entre chaque balise et le robot, et qui elle nécessite l'utilisation de 3 balises.

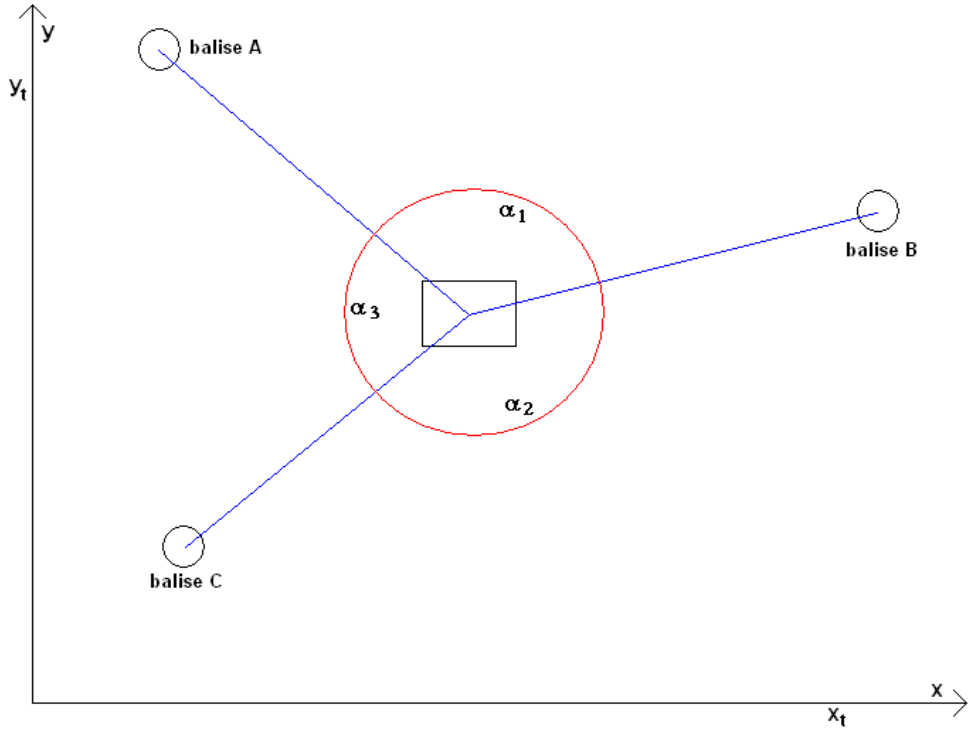

Figure 13- Localisation absolue (méthode par triangulation)

Les balises réelles sont dites actives si elles émettent un signal captable par le robot. En milieu extérieur, le système GPS (global positionning system) peut être utilisé pour obtenir des positions d'une précision de l'ordre du mètre. A la base développé par l'armée américaine dans les années 80 (lancement du premier satellite GPS en 1978), il fut ouvert aux civils en 1995. Jusqu'en 2000 les mesures étaient volontairement entachées d'une erreur d'une centaine de mètres, l'armée américaine craignant que ce système soit un avantage pour leurs ennemis. Malgré le retrait de cette erreur volontaire, la précision du système restait de l'ordre du mètre, à cause des incertitudes sur l'orbite et l'horloge des satellites, ainsi que les retards engendrés par la traversée des couches atmosphériques.

Pour améliorer cette précision, nous pouvons utiliser les GPS différentiels : avec un second récepteur GPS sur une base fixe et de position connue, il devient possible de mesurer l'erreur et d'en déduire la correction à apporter pour la zone environnante. Pour que ce système fonctionne,

il faut que la base mobile reste à une certaine portée de la base fixe. Cette distance varie suivant la gamme de fréquence utilisée pour l'envoi des corrections, et peut atteindre quelques dizaines de kilomètres pour les besoins de la navigation maritime.

Des satellites géostationnaires permettent également de corriger certaines erreurs de position. Ils envoient des corrections sur les orbites et les horloges des satellites GPS. Ainsi pour l'Europe c'est le système EGNOS (European Geostationary Navigation Overlay System) qui se charge d'envoyer ces corrections. La précision atteinte est de l'ordre de 3m. Le système de GPS RTK (real time kinematics) permet d'améliorer la précision à quelques centimètres, en utilisant la différence de phase de l'onde porteuse du signal, sa longueur d'onde étant d'une vingtaine de centimètres. En contrepartie la portée de la station de référence avec cette méthode n'est plus que de quelques kilomètres. Pour augmenter cette portée on a recours au RTK réseau, qui va utiliser la redondance d'informations pour communiquer les corrections à l'appareil, via un serveur de calcul à distance.

Les systèmes de localisation GPS sont très intéressants en rase campagne ou en banlieue. Cependant ils s'avèrent beaucoup moins efficaces en pleine ville ou en forêt. En effet pour fonctionner correctement un GPS a besoin de recevoir les informations de 4 satellites au moins, or la présence d'obstacles tels que les ponts et grands bâtiments en ville empêche parfois cette réception. Ainsi il n'est pas rare de perdre la localisation GPS pendant quelques minutes. C'est pour cela que beaucoup de recherches dans le domaine de la localisation sont aujourd'hui portées sur les méthodes alternatives au GPS en milieu urbain.

La vision peut être un moyen pour compléter la localisation par GPS. Ainsi Cappelle [16] utilise un modèle 3D de la place Stanislas à Nancy qui est confronté aux images obtenues par les caméras embarquées sur le robot pour déterminer précisément sa position. Yang propose d'utiliser les caméras du robot pour mettre au point un système d'odométrie visuelle [71].

4. Localisation et cartographie simultanées

En robotique mobile, le SLAM (simultaneous localization and mapping) consiste, pour un robot évoluant en milieu inconnu, à tracer une carte de l'environnement et localiser simultanément le robot dans celle-ci. La carte est construite de manière incrémentale au fur et à mesure que le robot évolue dans le terrain. En croisant les données perçues avec les informations géographiques dont il dispose en mémoire, le véhicule est capable de se localiser par rapport à des cartes préexistantes. L'idée du SLAM est donc de traiter conjointement les problèmes connexes que sont la navigation et la localisation d'un robot autonome.

La méthode alternative, moins gourmande en temps de calcul, consiste à utiliser la « mémoire visuelle » du véhicule, en repérant des éléments caractéristiques, pour se localiser par rapport à ceux-ci. Ces éléments sont extraits sous forme de primitives visuelles, que le robot cherchera à retrouver dans les images qu'il perçoit pour suivre un chemin précis [18][59].

V. Les verrous scientifiques

Il existe de nombreuses thématiques de recherche dans ce milieu de la robotique mobile autonome, ce qui montre qu'aujourd'hui encore le problème spécifique des robots mobiles

autonomes est entier. Le Groupement De Recherche en Robotique, qui regroupe la communauté des chercheurs français dans le domaine de la robotique, a dégagé 4 grands axes de travail autour desquels s'articulent les colloques du Groupe de Travail:

- techniques de localisation et cartographie : cet axe regroupe tous les développements autour de la perception et de la localisation du robot. On y retrouve notamment les méthodes SLAM (Localisation et Cartographie Simultanées). Plus récemment l'utilisation de bases de données sous forme de cartes 2D ou 3D, mais également sous forme SIG (Système d'Informations Géographiques) a ouvert de nouvelles perspectives dans ce domaine. De manière générale la fusion de données est également un thème important, tant la nécessité de coupler diverses sources de mesures apparaît nécessaire pour améliorer la précision et garantir l'intégrité des informations,

- contrôle et commande des véhicules : cet axe regroupe les thématiques liées à la planification de chemin, la génération de trajectoires, et la commande des robots de manière générale. Une prise en compte de plus en plus poussée des contraintes et de la dynamique des robots est nécessaire, pour adapter au mieux les robots à leur environnement. La bonne gestion des obstacles et la prise en compte des incertitudes de mesures sont également des points clés de cette thématique,

- la communication inter-véhicules : on retrouve ici tous les travaux liés à la coopération entre robots, et le contrôle de flottilles de véhicules,

- l'interprétation de scènes : les recherches dans ce domaine visent à pousser plus loin la perception de son environnement par le robot, que la simple reconnaissance des objets. En effet dans certaines applications il est nécessaire que le robot appréhende plus finement son environnement que par une simple détection et localisation des obstacles. Les travaux concernent notamment la perception multi-capteurs et la représentation dynamique des scènes.

La perception d'une part et la commande au sens large d'autre part sont donc les deux thèmes majeurs de recherche pour obtenir un robot mobile parfaitement autonome. Parmi les problématiques liées à la commande, celle de la navigation tient un rôle important : elle consiste à déterminer les trajectoires que le robot sera capable de suivre pour lui permettre d'évoluer correctement au milieu d'obstacles, en considérant qu'il dispose d'une méthode de cartographie.

CHAPITRE II
NAVIGATION DES ROBOTS MOBILES

Dans le cadre de la navigation d'un robot mobile autonome, deux grandes familles de méthodes permettent à un robot d'atteindre une position souhaitée. Les méthodes sans trajectoire explicite (champs de potentiels, réseaux de neurones et logique floue) cherchent plutôt à contrôler le mouvement global du robot de manière à le guider vers son but. L'autre famille, celle des méthodes de suivi de trajectoire, telles que les fonctions transverses et les sorties plates, permet au robot de suivre « du mieux possible » une trajectoire de référence donnée, connaissant les contraintes cinématiques. Cependant, la synthèse de ces différentes approches nous amène à considérer une troisième famille alternative : les méthodes de navigation par modèle direct et par génération de trajectoires qui nous permet d'aboutir, dans le chapitre III, à la formalisation d'une méthode de navigation générale s'adaptant à n'importe quel type de robot et de mission.

Le lien fort entre le type de méthode de navigation et l'architecture de contrôle global nous conduit à une réflexion qui nous amène à redéfinir la problématique de navigation dans le cadre d'une architecture de contrôle.

I. Le problème de navigation

Par abus de langage, un navigateur désigne généralement un module chargé de faire se mouvoir le robot mobile dans son milieu d'évolution. Si on fait l'analogie avec le pilotage d'une voiture, se déplacer d'un endroit A à un endroit B nécessite au moins 3 tâches bien distinctes :

- réfléchir à un itinéraire pour se rendre au point B : prendre telle autoroute en direction de telle ville, prendre la sortie numéro tant, tourner à droite au niveau du premier rond point etc… C'est ce que nous appelons la planification de chemin. Il est préférable de déterminer son

itinéraire à l'avance, et pour y parvenir quand on effectue le trajet pour la première fois, il est nécessaire d'utiliser des cartes, ou une représentation préétablie, pour trouver le meilleur itinéraire possible,

- sur la route, il faut suivre du mieux possible l'itinéraire, en adaptant sa vitesse à la route, la circulation, aux limitations de vitesses, aux conditions météorologiques etc… Il faut prendre correctement les ronds points, les virages, se frayer un passage dans la circulation. Il faut déterminer des trajectoires qui vont permettre de suivre du mieux possible l'itinéraire prévu, tout en s'adaptant aux conditions de circulation. Ces trajectoires seront dépendantes du véhicule que l'on conduit (voiture, moto etc…), chaque véhicule ayant des capacités propres. Ce travail d'adaptation, c'est le travail du navigateur,

- enfin, pour pouvoir suivre une trajectoire donnée, il faut appuyer sur l'accélérateur, tourner le volant, de manière générale il faut agir sur le véhicule. C'est le travail du pilote, qui doit agir sur les actionneurs du système pour appliquer la trajectoire souhaitée.

Dans notre étude, le navigateur a donc pour mission de générer des trajectoires pour le robot. La notion de trajectoire est différente de celle de chemin dans le sens où elle intègre des informations temporelles le long du parcours du robot, c'est-à-dire des vitesses et parfois des accélérations (dérivées premières et secondes). Concrètement, un navigateur se différencie d'un planificateur de chemin par les éléments suivants :

- il travaille sur une carte de l'environnement plus localisée autour du robot,
- il travaille à plus court terme que le planificateur, il prévoit moins loin dans le temps,
- il intègre les contraintes liées au robot, il doit s'assurer que le robot est capable physiquement de réaliser les trajectoires demandées

Dans l'architecture de contrôle, le navigateur reçoit un chemin de la part du planificateur, qui peut être sous différentes formes plus ou moins complètes : série de points de passage (x,y) pouvant contenir des informations supplémentaires comme l'angle d'orientation du robot, des vitesses indicatives etc… Concrètement, le rôle du navigateur consiste à générer des trajectoires qui soient réalisables par le robot (c'est-à-dire qui respectent le modèle cinématique du robot, les contraintes de non holonomie ainsi que les différentes contraintes liées aux actionneurs), qui n'intersectent pas avec les obstacles dans l'environnement local de navigation, et qui suivent au mieux le parcours proposé par le planificateur.

En fonction des méthodes utilisées, nous allons nous apercevoir que les délimitations entre pilotage/navigation/planification sont plus ou moins définies, il peut y avoir un recouvrement des tâches entre ces 3 grandes fonctions. Par exemple le rôle de navigateur peut-être tenu en partie ou complètement par le planificateur de chemin, si celui-ci intègre les possibilités de mouvement du robot dans son calcul de parcours. Mais le navigateur est un point clé pour un robot mobile autonome, puisque c'est lui qui fait le lien entre les capacités de mouvement du robot et l'environnement.

II. Méthodes sans trajectoire

Le premier type d'approche pour résoudre le problème de navigation d'un robot mobile autonome est de rendre la cible du robot attractive pour celui-ci, de manière à ce qu'il cherche à

s'en rapprocher. Et inversement rendre les obstacles répulsifs, pour qu'il s'en écarte et reste à bonne distance. Dans cette approche, on ne cherche pas à anticiper la trajectoire du robot, on le laisse librement décider à chaque instant dans quelle direction il doit se rendre, à partir des informations recueillies concernant son environnement local. On fait l'hypothèse que contrôler son comportement général plutôt que de chercher à calculer une trajectoire, permettra tout de même au robot d'atteindre son but au final.

Nous allons voir 3 méthodes qui reposent sur cette philosophie : les réseaux de neurones, la logique floue et les champs de potentiels artificiels.

1. Champs de potentiels artificiels

Les méthodes de type APF (Artificial Potential Field) furent les premières employées dans les années 80 (Figure 14). Le principe des méthodes APF est de construire un champ de potentiels sur l'environnement de navigation du robot. La valeur de ce champ est minimale sur le point que le robot doit atteindre, et croit continûment au fur et à mesure que l'on s'écarte de ce point. Les obstacles génèrent un champ de potentiel répulsif pour le robot, de valeur supérieure à n'importe quel autre point ne correspondant pas à un obstacle du champ de potentiel. Et Le champ de répulsion s'étend autour des obstacles avec une intensité inversement proportionnelle à la distance par rapport à l'obstacle.

L'idée est alors de demander au robot de se déplacer dans la direction du gradient de potentiel négatif le plus fort, sur le champ de potentiel global obtenu.

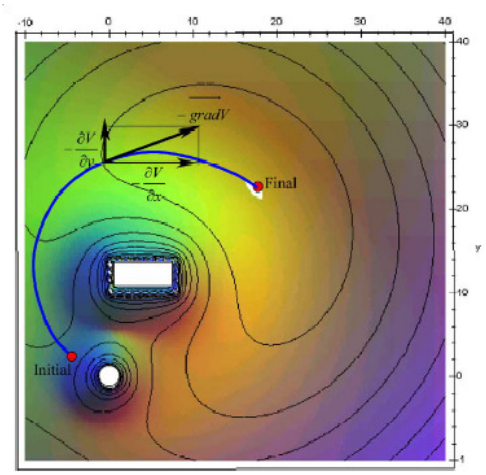

Figure 14- Carte des champs de potentiels autour d'un obstacle

Khatib dans [33], ainsi que Andrews et Hogan dans [6], furent les premiers à imaginer l'idée de forces imaginaires agissant sur le robot. Ces méthodes présentent l'avantage d'être simples à mettre en œuvre, et elles furent les premières à être implantées physiquement sur de vrais robots en 1985 par Brooks [11] et en 1989 par Arkin [7]. Malgré le matériel informatique encore limité à cette époque, les calculs de trajectoires/commande étant très rapides avec ce type d'approche, cela permit de premières expérimentations sur des robots relativement lents. Plus récemment dans [1], Agirrebeitia propose une extension du principe des méthodes APF pour la navigation de robots dans un espace 3D.

L'expérimentation a révélé certains problèmes récurrents liés au principe même de ces méthodes :

- minima locaux entrainant des situations où le robot est piégé (classiquement le piège en forme de U)
- pas de passage détecté entre des obstacles assez proches
- oscillations en présence d'obstacles et dans les passages étroits

Dans [37], Koren et Borenstein ont démontré mathématiquement les problèmes d'instabilité de ces méthodes (entrainant des oscillations), problèmes qui apparaissent fortement dès lors que ces méthodes sont implémentées sur des systèmes « rapides ».

2. Réseaux de neurones

Le neurone est l'entité de base du système de réaction animal, et les neurologues Mc Culloch et Pitts furent les premiers à étudier leur fonctionnement chez la grenouille [41]. Les réseaux de neurones désignent à la fois l'étude de ces systèmes biologiques et leur modélisation informatique, plus ou moins simplifiée, à différentes applications, comme la reconnaissance de caractères, son premier terrain d'application historique, ou la navigation des robots mobiles autonomes. Dans ce cadre a été définie la notion de réseaux de neurones artificiels.

Dans un réseau de neurones artificiel (Figure 15), chaque neurone possède plusieurs entrées et une sortie. Chacune de ses entrées se voit affecter un poids (dit poids synaptique) différent, et si la somme ainsi pondérée des signaux des différentes entrées dépasse un seuil, la sortie prend une valeur positive (le neurone déclenche).

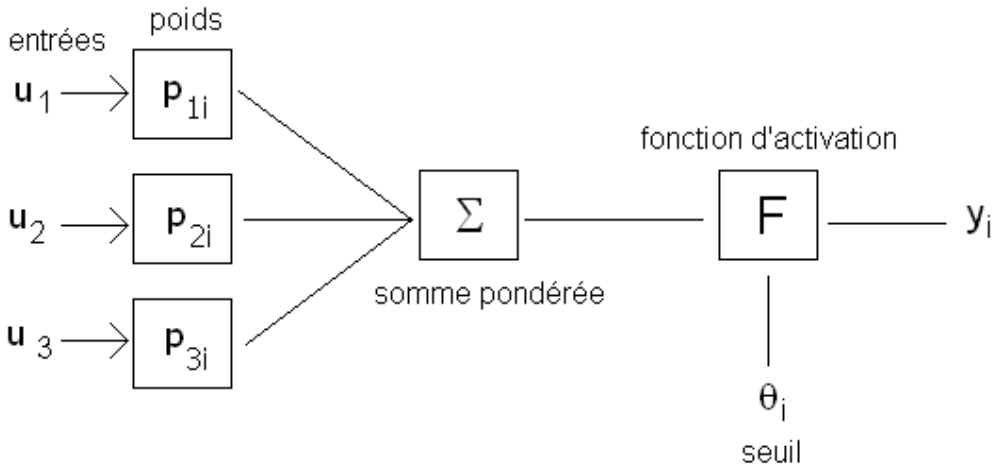

Figure 15- Modélisation d'un réseau de neurones artificiel

Un réseau complet est constitué de différentes couches de neurones. Les premières couches correspondent à des couches de détection, en robotique les neurones correspondants sont directement liés aux informations délivrées par les capteurs du robot. Les couches supérieures correspondent plutôt à des couches d'interprétation, et pour un robot elles seront plutôt liées aux effecteurs. De plus il y a généralement un certain nombre de couches intermédiaires entre les couches d'entrée et de sortie.

Un apprentissage du réseau est réalisable en adaptant les poids des entrées synaptiques. Pour cela, le principe est de donner au réseau un grand nombre d'exemples (set d'entrées donné), et de rétro propager l'erreur obtenue entre la sortie du réseau et la sortie attendue. L'algorithme d'apprentissage aura alors pour tâche de régler les différents poids synaptiques pour minimiser cette erreur.

De nombreux contrôleurs pour robots mobiles ont été construits sur ces principes [38][9]. Les dernières générations utilisent un codage en fréquence pour la transition de l'information, comme le font les neurones biologiques. Sur une fenêtre temporelle donnée, le nombre de pics émis quantifie par exemple la proximité d'un obstacle. Wang a développé un contrôleur basé sur ce principe, pour un robot unicycle équipé de capteurs ultrasonores [68].

3. Logique floue

La logique binaire présente l'avantage de la simplicité, mais est assez éloignée de la façon humaine de raisonner. Si on prend l'exemple de la qualification de la proximité d'un obstacle, la logique floue permet de faire intervenir des notions telles que « assez près » ou « très loin », au lieu de se cantonner à une définition binaire « obstacle ou pas obstacle ». Celle-ci a été formalisée par Zadeh en 1965.

Le principe d'un contrôleur basé sur la logique floue se décline en 3 phases : une phase de fuzzification, qui va transformer les variables d'entrée en variables floues ; une phase faisant appel à une table des règles de comportement, règles logiques du type « si (condition 1) et/ou (condition 2) alors (action sur les sorties) » ; et une dernière phase dite de defuzzification, permettant de traduire l'action déterminée par les règles de comportement en commande à envoyer aux actionneurs.

L'étape de fuzzification fait appel à des intervalles flous, qui vont délimiter l'espace des variables d'entrée en un certain nombre de sous ensembles flous (par exemple pour une proximité, on pourra avoir très proche (contact), assez proche, distance moyenne, assez loin et très loin) ; des fonctions d'appartenance sont alors utilisées pour définir le degré de vérité (probabilité d'appartenance) de la variable floue en fonction de la grandeur d'entrée. Ces fonctions peuvent être de type triangle, gaussienne etc... Ainsi pour une mesure de distance donnée, la règle d'appartenance nous dira « il y a 95% de chances que l'obstacle soit assez proche, 5% de chance que l'on soit en contact ». Cette notion est basée sur le fait qu'il y a toujours des incertitudes concernant les mesures des capteurs et les informations dont on dispose en général.

La seconde étape consiste en l'élaboration de règles de comportement pour le robot, suivant la combinaison des variables floues en entrée. La table des règles de comportement est construite manuellement, et est tributaire de l'expérience de la personne qui va régler le contrôleur. Pour un robot mobile, une règle pourra être : «s'il y a un obstacle assez proche sur la droite, il faut tourner à gauche et diminuer la vitesse du robot ».

La dernière étape, appelée defuzzification, consiste à transformer le comportement obtenu par la table des règles, en commande pour le robot. Une méthode qui peut être utilisée pour faire cela est celle des centres de gravité, qui va consister à faire la moyenne pondérée des commandes à appliquer. La pondération étant liée aux probabilités d'appartenance de chaque variable d'entrée.

Un contrôleur flou pour permettre la navigation d'un robot mobile de type voiture à double braquage, un Robucar, a été étudié dans [55]. Ce contrôleur permet au robot d'atteindre sa position finale tout en respectant les contraintes cinématiques du robot, mais ne tient pour le moment pas compte des obstacles. Des travaux sont en cours pour intégrer la prise en compte de l'orientation finale du véhicule.

Dans [17], R. Chatterjee et F. Matsuno utilisent la symétrie droite/gauche des règles de logique de comportement du robot pour simplifier celles-ci et ainsi diminuer le temps de calcul.

Le problème de ces méthodes par logique floue est généralement le même que celui des méthodes APF, c'est à dire le problème des minima locaux qui se traduit par le fait que le robot peut rester piéger dans des culs de sac par exemple. Une technique pour se sortir de ces pièges a été proposée par W.L. Xu [70]. Cette technique utilise des cibles virtuelles pour sortir le robot du piège dans lequel il est tombé, à partir du moment où il reconnaît être tombé dans un piège (Figure 16).

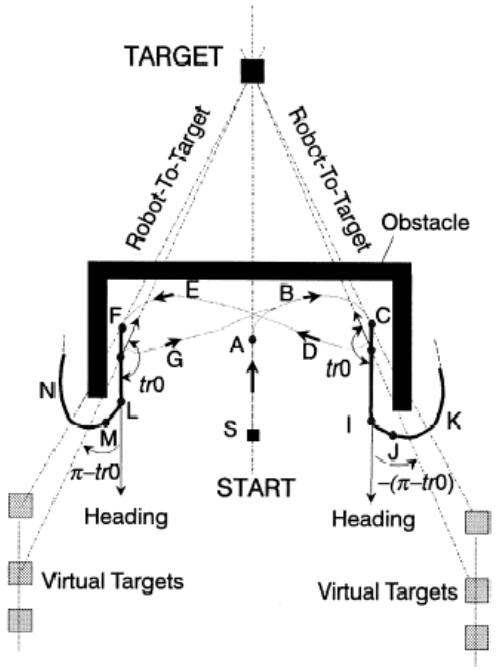

Figure 16- Le principe des virtual target pour sortir d'un minimum local[70]

Un autre problème inhérent aux méthodes par logique floue est qu'elles sont trop spécialisées pour un type d'environnement donné, et donc elles souffrent de problèmes d'adaptation à des environnements différents. Cependant il existe des méthodes dites d'apprentissage qui permettent au robot de modifier lui même ses règles de comportement au fur et à mesure qu'il explore un nouvel environnement. Le problème de ces méthodes est qu'elles demandent un temps d'apprentissage plus ou moins long avant que le robot ne puisse naviguer efficacement [24].

III. Méthodes de suivi de trajectoire

Le second type d'approche consiste à contrôler le robot pour lui faire suivre une trajectoire précise. Ces méthodes (Figure 17) nécessitent généralement de déterminer un modèle inverse du robot, c'est-à-dire un modèle qui permette de calculer les commandes à envoyer au robot (espace articulaire) connaissant la trajectoire que l'on souhaite lui faire suivre (dans l'espace cartésien).

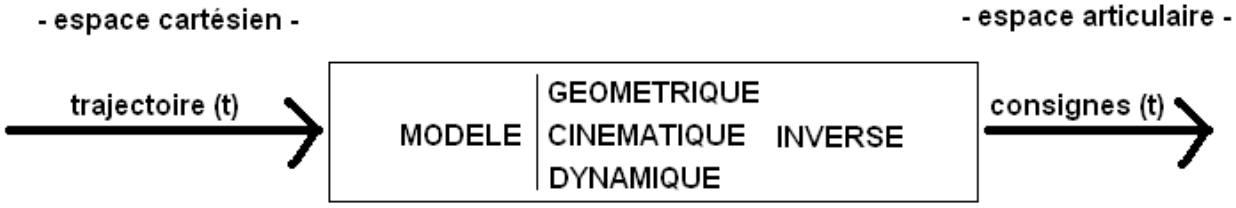

Figure 17- Approche par modèle inverse pour effectuer un suivi de trajectoire

La détermination d'un tel modèle est souvent le point central de ce type de méthode, et le but est d'obtenir la meilleure convergence possible du robot vers la trajectoire de référence.

Nous allons voir ici 2 méthodes qui ont obtenu de très bons résultats dans ce domaine : les sorties plates et les fonctions transverses. Dans la dernière partie nous évoquerons d'autres pistes qui sont étudiées dans ce domaine.

1. Sorties plates

La théorie des systèmes plats fut introduite en 1995 par Fliess [26]. Un système plat est un système $\dot{x} = ax + bu$ tel que:

$$\begin{aligned} x &= f(y, \dot{y}, \ddot{y}...y^{(k)}) \\ u &= g(y, \dot{y}, \ddot{y}...y^{(l)}) \\ y &= h(x, u, \dot{u}....u^{m)}) \end{aligned} \quad , \qquad \text{Équation 5}$$

Les différentes composantes du vecteur y sont les sorties plates du système. Concrètement cela revient à dire que l'on peut exprimer l'ensemble du système en fonction des sorties plates, et d'un nombre fini de ses dérivées. L'avantage d'écrire un système sous cette forme, dans le cas du suivi de trajectoire, est que cela permet de démontrer la convergence du système vers la trajectoire souhaitée. L'inconvénient est qu'il n'existe pas de méthode systématique pour déterminer la forme plate d'un système.

Une application pour le suivi de trajectoire des robots mobiles est donnée par Fraisse [28] et elle est utilisée pour la navigation d'une flottille de robots. Chaque robot suiveur a pour but de suivre un robot leader, lui-même contrôlé à distance par téléopération (Figure 18).

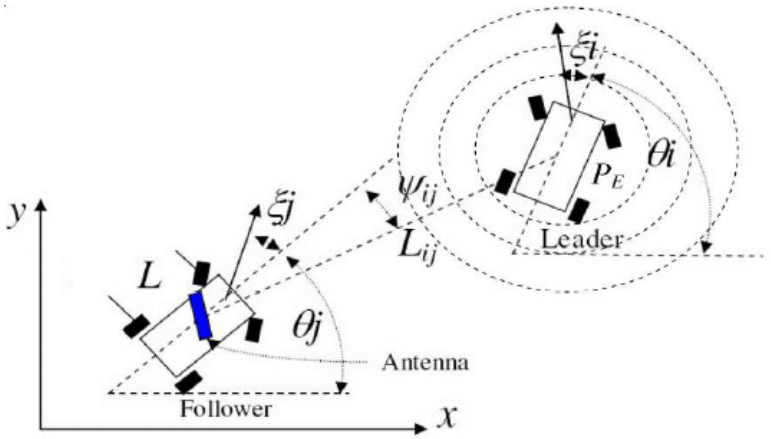

Figure 18- Modélisation d'une flotille de robot avec un robot leader et un ou plusieurs robots suiveurs

Dans un premier temps les variables d'état du robot (vitesses linéaire, angulaire…) sont exprimées en fonction des sorties plates (positions x et y et leur dérivées). Ces sorties plates sont ensuite paramétrées en fonction du temps. Le problème devient alors un problème d'optimisation par rapport au temps des sorties plates, pour que le robot atteigne l'état désiré en respectant un ensemble de contraintes. Le fait que le système soit plat permet de démontrer la convergence des trajectoires vers les trajectoires désirées.

Les sorties plates sont donc un outil intéressant dans le cadre du suivi de trajectoire en téléopération par exemple (trajectoire fournie par l'utilisateur) ou dans le cas d'une flottille de robots (trajectoire déterminée en fonction du leader). Elles garantissent de très bons résultats dès lors que l'on arrive à déterminer un modèle plat du robot à commander.

2. Fonctions transverses

La synthèse de lois de commande par retour d'état stabilisant est la problématique traitée par l'approche par fonctions transverses. Cette problématique, pour les systèmes non holonomes, est particulièrement ardue dès lors que la trajectoire de référence présente des points fixes. En effet, le système linéarisé n'est plus commandable au voisinage des points fixes. L'objectif de stabilisation asymptotique en un point, qui est impossible à atteindre, est alors délaissé pour une stabilisation pratique, qui consiste à stabiliser le système en un voisinage arbitrairement petit de ce point.

L'approche par fonctions transverses [49] offre une solution à ce problème en permettant une stabilisation pratique autour de n'importe quelle trajectoire de référence, réalisable ou non, pour un système commandable, et s'étend aux systèmes présentant certaines propriétés de symétrie comme la voiture [49]. Cette approche est basée sur l'utilisation de fréquences variables comme entrées de commande supplémentaire, et repose sur l'existence d'une fonction f vérifiant les conditions de transversalité [29]. Cette fonction permet de définir un repère compagnon, dans le voisinage proche du véhicule, et c'est ce repère qui est asservi sur la trajectoire de référence.

Cette méthode permet ainsi à un véhicule non holonome tel qu'une voiture ou un unicycle, de suivre une trajectoire non admissible, comportant par exemple des déplacements transversaux, sous une certaine erreur. Le robot effectuera alors des manœuvres autour de la trajectoire de référence. Cette erreur maximale est garantie à une valeur qui est fixée librement. Le compromis

sur cette erreur étant que plus elle est fixée petite, plus le robot fera un grand nombre de manœuvres pour suivre sa trajectoire (Figure 19).

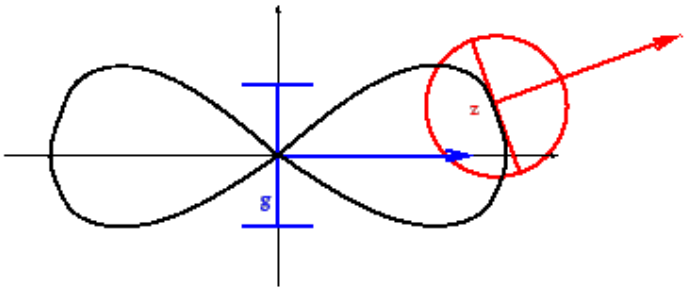

Figure 19- Base mobile s et sa transformée omnidirectionnelle z, avec le repère compagnon en rouge [29]

La restriction de cette méthode est liée à la difficulté de trouver les fonctions transverses pour certains systèmes, puisqu'il n'existe aucune méthode systématique pour les déterminer. De plus, la question de la robustesse de cette méthode, par exemple aux erreurs de mesures ou aux problèmes de glissement, reste un sujet ouvert.

3. Autres recherches sur le suivi de trajectoire

D'autres méthodes pour faire du suivi de trajectoires sont basées sur la reconnaissance d'images le long du parcours. Ces images peuvent être prises avec une caméra, ou avec des télémètres laser. Le principe est de créer une base de données d'images à reconnaître, et pour cela le plus simple est de piloter le robot en manuel dans une phase dite d'enregistrement, en lui faisant suivre le parcours de référence. Une fois la base de données créée, on peut procéder à la phase de suivi de trajectoire en autonome. A chaque instant d'échantillonnage, l'image vue par le robot est comparée à l'image qui correspond dans la base de données, et la corrélation entre les deux images permet de calculer les commandes à envoyer au robot, pour le recaler sur la trajectoire de référence.

Ce type de méthode a été appliqué à la navigation de véhicule en milieu urbain en utilisant des camera video [19] [18], et en intérieur de bâtiments en utilisant des télémètres lasers [1].

La reconnaissance d'amers est un principe proche de la reconnaissance d'images, sauf que cette fois ci on repère des éléments remarquables fixes dans l'environnement de navigation, pour guider le robot le long du parcours. Cette méthode a été appliquée sur un véhicule CyCab, évoluant en milieu extérieur grâce à des balises installées dans l'environnement de navigation [58].

IV. Discussion

1. Synthèse des méthodes de navigation

Le problème de navigation en robotique mobile n'est pas un problème parfaitement arrêté. La frontière entre planification de chemin, navigation, et pilotage est plus ou moins définie, il existe souvent des recouvrements entre ces différents domaines connexes. Ces différences d'appréhension du problème se ressentent notamment au niveau des structures de contrôle des

robots, qui sont en quelque sorte le reflet de la manière dont chaque équipe perçoit et découpe les différentes tâches nécessaires à la réalisation d'un système robotique autonome. Il existe ainsi beaucoup de définitions différentes du navigateur, d'où certaines différences dans la manière de répondre à la problématique.

Par exemple, beaucoup de travaux ne font pas la distinction entre chemin, trajectoire et même pilotage, et tous ces problèmes sont ainsi résolus par un unique module. C'est le cas d'un grand nombre d'approches à base de réseaux de neurones et de logique floue, qui cherchent à résoudre tous ces problèmes en même temps en contrôlant un comportement global du robot. Ce genre d'approche peut donner de bons résultats dans une situation donnée. Mais il n'y pas de planification à long terme de ce que le robot doit faire, le robot n'est donc pas capable de s'adapter à une situation inconnue. Il s'agit généralement d'approcher de la meilleure manière possible un comportement souhaité dans un type de situation donné, ce qui ne correspond pas à la définition d'un système autonome qui serait capable de s'adapter à différentes situations non prévues.

Une autre approche pour la navigation porte sur la détermination de méthodes pour effectuer un suivi de trajectoire ou de chemin. La trajectoire servant de référence peut être fournie par un module de planification, ou encore provenir d'un robot leader qu'il faut suivre à une certaine distance. En considérant un robot classique de type voiture, le problème à traiter provient notamment des contraintes de non holonomie des robots, qui réduisent sa mobilité et l'empêchent donc effectuer certains mouvements pour rattraper la trajectoire de référence. Dans ce domaine, les approches par fonctions transverses ou celles par les sorties plates ont montré d'excellents résultats. La difficulté de ces méthodes est qu'elles nécessitent de déterminer un modèle bien spécifique du robot. Les méthodes pour déterminer ces modèles ne sont pas systématiques, et de plus ces modèles n'existent pas pour tous les robots.

2. Nomenclature et lien avec une architecture de commande pour l'autonomie

L'étude des méthodes de navigation pour un robot autonome a mis en évidence les recouvrements entre les différentes fonctions nécessaires au commandement d'un robot mobile autonome. Il en découle une nécessité de définir plus précisément les différentes notions que nous développerons dans la suite de ce manuscrit, de manière à bien se mettre d'accord sur ce dont on parle. Cette nomenclature nous amènera alors à présenter l'architecture de commande sur laquelle elle s'appuie et que nous utiliserons dans les chapitres suivants.

Ainsi, pour la suite de ce document, nous définissons les notions de chemin et trajectoire de la manière suivante :

- un chemin est une succession continue ou discrète de positions et orientations du robot ; un chemin s'inscrit dans une planification globale du parcours que doit effectuer le robot. Typiquement un chemin sera constitué de différents points de passages successifs que le robot doit atteindre, et en outre le chemin est décorrèlé de la notion de temps. Au mieux le chemin prendra en compte les paramètres géométriques du robot (circonférence du robot), mais pas ses capacités cinématiques. Il en résulte que le chemin détermine par où le robot devra passer, mais pas au bout de combien de temps il y passera.
- une trajectoire contient les paramètres cinématiques du robot le long de son parcours, elle est donc directement reliée à la notion de temps contrairement au chemin. Il en découle qu'une

trajectoire prendra en compte les capacités cinématiques et dynamiques du robot. Une trajectoire sera donc définie sur un horizon temporel donné, et prendra en compte des contraintes de nature plus locale que le chemin

Cette manière de décomposer la notion de chemin et de trajectoire s'inscrit dans l'architecture de commande multi-niveaux présentée par Gilles Mourioux dans [52], et utilisée par le laboratoire et sur laquelle nous nous appuyons pour le développement de nos robots. Il s'agit d'une structure à plusieurs niveaux, qui contient plusieurs boucles action/perception imbriquées les unes dans les autres. Les couches supérieures sont les plus lentes et les plus « intelligentes », au sens où elles intègrent des informations provenant des couches inférieures, et elles travaillent à plus long terme et sur des espaces plus grands. Plus on descend dans les niveaux, et plus on a des boucles rapides et réactives.

Chaque bloc de la partie décision (Figure 20) a pour entrée les ordres d'un bloc analogue de niveau supérieur, et les informations fournies par un bloc de la partie « perception ». Ces informations lui permettent de prendre ses décisions, qui sont les sorties du bloc. Ces ordres joueront alors le rôle de consigne pour le bloc décisionnel inférieur.

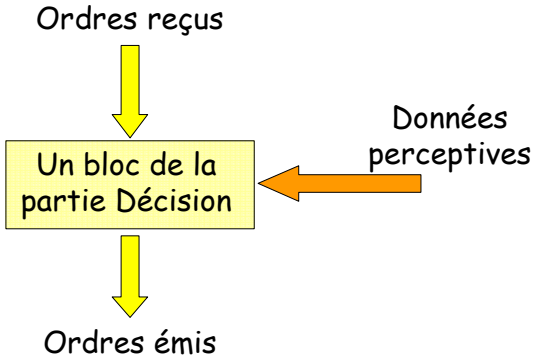

Figure 20- Bloc de la partie decision [51]

Les blocs de la partie perception (Figure 21) reçoivent des informations émanant des capteurs. L'ensemble ou une partie des données reçues sont transmises vers les blocs de la partie décision, et également vers les blocs des niveaux supérieurs de la partie perception. Ces données peuvent avoir subi au préalable d'un traitement dans les blocs de la partie décision. Ainsi, niveau après niveau, les données brutes sont enrichies par les traitements successifs et la confrontation avec d'anciennes et nouvelles informations gardées en mémoire. Cela permet notamment d'améliorer la robustesse des informations et de réaliser des cartes.

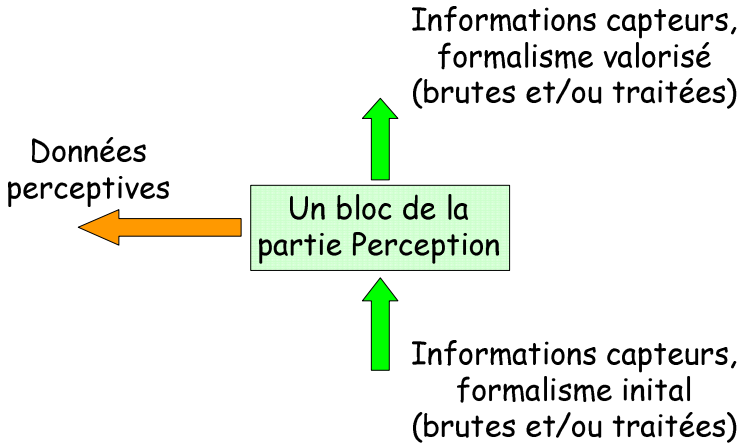

Figure 21- Bloc de la partie perception [51]

L'architecture générale (Figure 22) est ainsi composée de différentes boucles action/perception sur différentes niveaux, placés au dessus du système mécatronique articulé. Les boucles de bas niveau sont rapides et travaillent à court terme, elles correspondent à ma partie réactive de l'architecture. Les boucles de haut niveau travaillent sur des horizons temporels et spatiaux plus importants, sont plus lentes, et forment la partie délibérative de cette architecture.

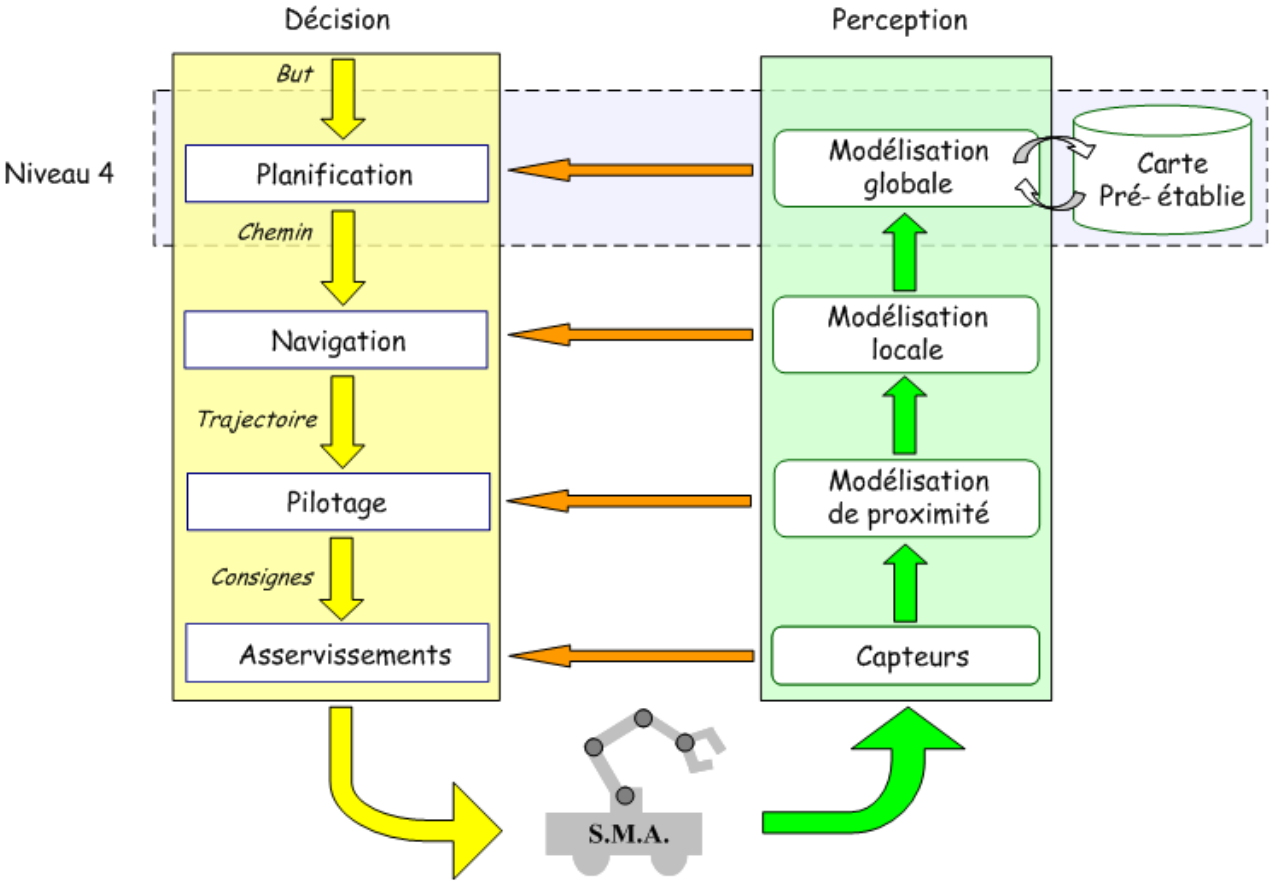

Figure 22- Architecture de contrôle (thèse de Gilles Mourioux chapitre 3)

Le niveau 0 représente la partie physique du robot, à savoir le système mécanique articulé et les actionneurs pour faire bouger le robot. Cet élément constitue la base sur laquelle toute l'architecture est construite.

Le niveau 1 est constitué des boucles d'asservissement bas niveau du robot. La partie perception contient les capteurs, et les informations qu'ils délivrent sont utilisées par le bloc de la partie décision en vue de l'asservissement de chaque moteur.

Le second niveau est celui du pilote. Il reçoit une trajectoire à suivre de la part du navigateur, et utilise les informations fournies par le bloc perception pour déterminer les consignes à envoyer aux asservissements de chaque moteur. La partie perception effectue un traitement des données brutes, par exemple sous forme de segmentation ou de détermination de l'espace libre, qu'elle fournit au pilote. En plus de sa mission principale consistant à transformer une trajectoire fournie par le navigateur en consignes pour les moteurs, le pilote peut également avoir la capacité de prendre certaines décisions comme des manœuvres d'évitement d'obstacle en urgence.

Le niveau 3 est celui du navigateur. Cette couche utilise un modèle de l'environnement local du robot, généralement sous forme de carte, fabriqué par la partie perception de ce niveau. Cette carte est réactualisée en permanence par les nouvelles informations des capteurs externes qui sont remontées et traitées par les blocs perception inférieurs. Le navigateur se sert de cette carte locale pour générer une trajectoire pour le robot, lui permettant de se déplacer entre les obstacles tout en cherchant à suivre du mieux possible un parcours. Ce parcours lui est fourni par le niveau supérieur, le planificateur. La trajectoire générée tient ainsi compte des obstacles, mais elle tient également compte des contraintes cinématiques voire dynamiques du robot. Le navigateur est donc le bloc charnière entre ce que l'on souhaite que le robot fasse (le chemin), les contraintes liées aux capacités de mouvement du robot, et les contraintes locales que sont les obstacles. Une fois la trajectoire déterminée, le navigateur l'envoie au pilote, qui se chargera de la faire appliquer par les actionneurs.

Le quatrième et dernier niveau représenté est celui du planificateur. La partie décision de ce bloc se voit fournir un but à atteindre, qui peut être une destination finale ou une série de lieux à visiter. Sa mission consiste alors à déterminer le chemin optimal pour atteindre ce but, chemin qui sera envoyé comme consigne au bloc navigateur du 3^{e} niveau. Pour cela, il utilise un modèle global de son environnement fourni par la partie perception du niveau 4. Pour fabriquer cette carte globale, la partie perception utilisera généralement des cartes préétablies présentes dans sa base de données, qu'elle pourra fusionner avec des informations courantes comme la carte locale de manière à actualiser la modélisation de l'environnement.

Dans l'architecture de contrôle proposée par Gilles Mourioux, il existe un niveau supérieur nommé générateur de mission, qui permet de donner un niveau d'autonomie supplémentaire au système. Ce niveau détermine un but ou une mission utilisable comme entrée pour le planificateur de chemin. Enfin, cette architecture permet également d'intégrer la téléopération par l'intermédiaire d'un troisième bloc, en surplus des blocs décision et perception. Ce bloc téléopération est subdivisé en couches de la même manière que les deux autres blocs, ce qui permet de faire intervenir un mode téléopéré à n'importe quel niveau.

Cette synthèse des deux grandes familles de navigation (sans et avec trajectoires) a montré les limites de leur adaptabilité à n'importe quelle cinématique de robots mobiles à roues, ou bien celles d'assurer un comportement global homogène du robot tout au long de sa mission. Ces méthodes de navigation sont incluses dans une architecture de commande du robot, et doivent « cohabiter » avec d'autres fonctionnalités comme le pilotage et la planification de chemin. L'architecture de contrôle présentée servira donc de cadre aux approches de navigation alternatives basées sur des modèles directs et sur la génération de trajectoires que nous développerons par la suite.

CHAPITRE III
NAVIGATION PAR MODELE DIRECT

Une famille alternative de méthodes de navigation que nous considérons est celle par modèle direct qui présente l'avantage de se baser sur un élément qui est toujours défini pour un robot mobile : son modèle cinématique direct. L'ensemble des trajectoires issues de ce modèle direct peut être projeté dans une représentation de l'environnement d'évolution du robot. Parmi cet ensemble, il est possible de choisir la trajectoire optimale, suivant des critères définis, qui permet au robot de rallier une position donnée tout en évitant les obstacles.

Une méthode directe, par lignes de fuite, est appliquée à un robot de type unicycle. L'analyse des résultats de simulation met en évidence la simplicité de mise en œuvre pour la détermination de trajectoires et évitement d'obstacles ;mais elle pose un problème de formalisation puisque l'on se place dans un espace discret pour l'obtention des lignes de fuite. Nous nous orientons alors vers un outil de l'automatique, la commande prédictive, pour formaliser une nouvelle méthode qui sera décrite dans le chapitre IV.

I. L'approche par modèle direct

La philosophie de cette approche est d'utiliser un modèle direct du robot (espace articulaire vers cartésien) pour prévoir ses trajectoires suivant la commande qu'on lui envoie. L'ensemble des trajectoires réalisables par le robot est projeté dans une carte de l'environnement local de manière à déterminer la meilleure trajectoire pour rejoindre la cible, en évitant les obstacles. L'idée est donc de regarder quels sont les mouvements que le robot est capable d'effectuer, puis choisir parmi ceux-ci, celui qui est le plus adapté à la situation présente. On raisonne donc à l'inverse des approches de type suivi de trajectoire.

Le modèle cinématique direct détermine les déplacements du robot, dans le repère cartésien, en fonction des dérivées des coordonnées articulaires. La matrice permettant de faire ce calcul est la matrice jacobienne J:

$$\dot{X} = J.\dot{q}\,, \qquad \text{Équation 6}$$

où $\dot{X} = (\dot{x}, \dot{y}, \dot{\theta})^T$ est le vecteur des déplacements du robot dans l'espace cartésien, et $\dot{q} = (\dot{q}_1, \dot{q}_2, ...\dot{q}_n)^T$ est celui des déplacements dans l'espace articulaire. Cependant en robotique mobile à roue, la connaissance de ce modèle n'est pas toujours utile pour commander le robot, et on utilise plutôt un modèle cinématique simplifié, dit modèle cinématique en posture, qui permet de relier les vitesses cartésiennes du robot directement en fonction des commandes appliquées :

$$\dot{X} = C(q).u\,, \qquad \text{Équation 7}$$

Dans le cas d'un robot de type voiture, $u = (v, \xi)^T$ (respectivement la vitesse longitudinale du robot et l'angle de braquage moyen des roues) et le modèle cinématique est de la forme :

$$\dot{X} = \begin{cases} v.\cos\theta \\ v.\sin\theta \\ \dfrac{v}{L}.\tan\xi \end{cases}, \qquad \text{Équation 8}$$

Avec θ l'orientation du robot dans l'espace cartésien et L l'empattement du véhicule (distance entre l'essieu avant directeur et l'essieu arrière moteur).

Le principe des méthodes par modèle direct, est d'utiliser ce type de modèle, pour calculer les trajectoires réalisables par le robot, dans son environnement d'évolution. Les trajectoires finales dépendent des commandes envoyées, mais également de la configuration courante du robot (position, orientation), de son état courant (cinématique), et de l'environnement lui-même dans lequel il évolue. L'intégration de tous ces paramètres permet de déterminer un modèle capable de prédire le mouvement du robot, sur un horizon de temps donné.

Une fois le modèle déterminé, on est donc capable de prévoir les trajectoires réalisables par le robot, sur un horizon temporel donné. Les trajectoires réalisables sont alors projetées dans un modèle de l'environnement d'évolution du robot, et à sélectionner parmi celles-ci celle qui semble la plus appropriée pour remplir l'objectif assigné au robot. Concrètement, cela nécessite de déterminer un critère pour sélectionner la meilleure trajectoire au sens de ce critère.

Figure 23- Principe général des méthodes par projection de trajectoire

La trajectoire choisie ne doit pas faire entrer le robot en collision avec les obstacles. La trajectoire du robot étant calculée par rapport à un point particulier du véhicule (typiquement le centre de l'essieu arrière pour un véhicule de type voiture), il est nécessaire de tenir compte de la taille du robot pour définir les conditions de non collision.

Les trajectoires qui sont sûres doivent être départagées au moyen d'un critère de sélection. Suivant la mission assignée au robot, ce critère peut être de minimiser la distance parcourue, minimiser la consommation, se rapprocher le plus possible d'une trajectoire de référence etc… Suivant la méthode de navigation, la gestion des obstacles peut être incluse dans le critère de sélection et traitée de manière globale, ou bien à part, en effectuant par exemple un filtrage préalable des trajectoires à risque (Figure 23).

De manière générale, la généralisation de ce type d'approche à n'importe quel système de robot en interaction avec son environnement, est un point fort de ces méthodes par projection. Quelque soit le robot mobile étudié, un modèle cinématique direct du robot peut être déterminé et une fonction d'entrée donnée provoquera une seule trajectoire du robot. C'est une différence importante par rapport aux modèles inverses, qui n'existent pas forcément pour une trajectoire donnée. Autre point important, la possibilité de prendre en compte les interactions robot/environnement dans la modélisation. La présence de pentes, la structure du sol (sol boueux, glissant etc…) peuvent être intégrés dans le modèle, pour une prédiction approfondie des mouvements du robot dans un environnement spécifique.

La méthode Motion Generation développée par Bonnafous en 2001, utilise ce principe de projection de trajectoires réalisables par le robot dans un modèle de l'environnement local du robot [10]. Un set de trajectoires prédéfinies, sous forme d'arcs de cercle que le robot LAMA (robot à 6 roues motrices) est capable de suivre, est projeté dans un espace modélisant son milieu de navigation (Figure 24). A chaque trajectoire est associé un critère de risque pour le robot (collision avec les obstacles, renversement etc...), et un critère d'intérêt par rapport à la mission assignée. La trajectoire qui présente le meilleur rapport entre ces deux critères est sélectionnée. Cette méthode n'utilise pas directement de modèle direct du robot, les trajectoires sont tirées d'une base de donnée contenant des trajectoires que l'on sait réalisables par le robot. Ainsi l'adaptabilité à tous types de robots n'est pas directe, ce qui est une restriction de cette méthode. Cependant le principe de projection de trajectoires réalisables par le robot correspond totalement aux méthodes de navigation par modèle direct.

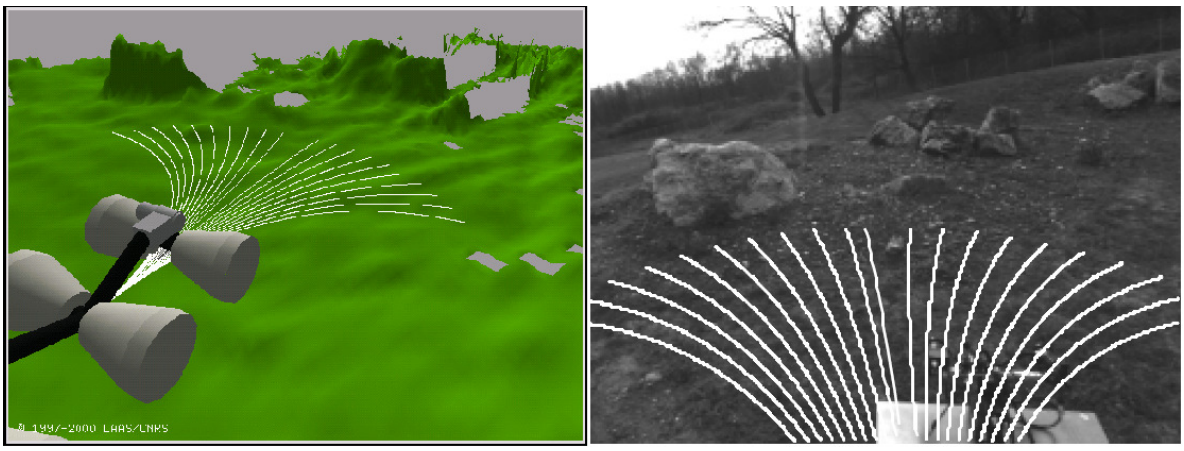

Figure 24 - La méthode Motion Generation appliquée sur le robot LAMA du LAAS

Dans la méthode Neural Network Hybrid développée par Belker et Schulz [9] détermine l'espace des commandes admissibles par le robot (respect des contraintes des actionneurs). Cet espace subit alors une discrétisation, et les commandes discrètes ainsi obtenues sont injectées dans le modèle cinématique direct du robot. Les trajectoires générées sont projetées dans une

carte de l'environnement du robot sous forme de grille (type grille d'occupation [69], Figure 25). Une fonction de rétribution, basée sur le principe des chaînes de Markov, associe à chaque case de la grille une valeur de rétribution qui quantifie l'intérêt pour le robot de se situer sur cette case (proximité du but à atteindre, distance par rapport aux obstacles). La trajectoire qui conduit le robot sur la case avec la meilleure valeur associée est sélectionnée, et les commandes correspondantes sont envoyées comme consignes aux actionneurs du robot. Les difficultés liées à cette méthode étaient dues principalement aux temps de calculs élevés (reconstruction de la grille et des valeurs de rétribution en temps réel), qui conduisaient à imposer des compromis contraignants notamment sur la résolution de la grille, le nombre de trajectoire générées lors de la discrétisation et la dynamique de réaction du robot.

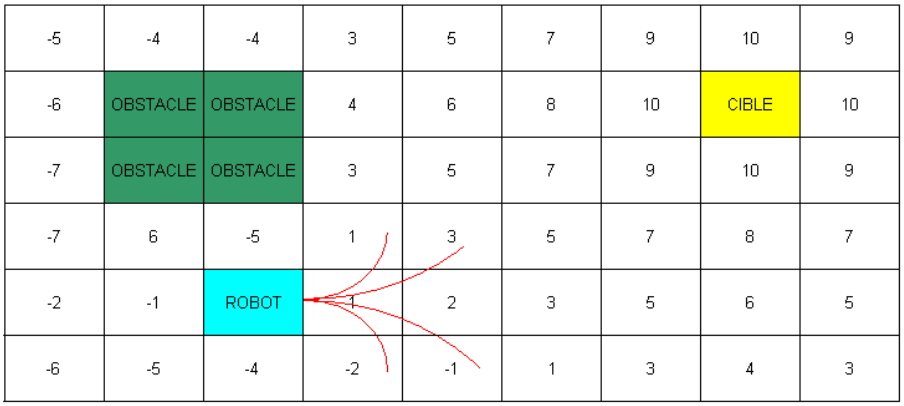

Figure 25 - la méthode Neural Network hybride : projection d'un set de trajectoires réalisables par le robot dans une carte de l'environnement sous forme de grille

La méthode des lignes de fuite mise au point par Novales [53], permet de déterminer une trajectoire qui respecte les différentes contraintes du robot, respecte son modèle cinématique et lui permette de suivre du mieux possible une courbe de rattrapage (sorte de chemin de référence) fourni par un module de navigation globale. Tout d'abord un ensemble de trajectoires admissibles par le robot (trajectoires qui respectent ses différentes contraintes cinématiques, dynamiques...) est obtenu en effectuant une discrétisation des commandes admissibles par le robot. Ces commandes sont injectées dans le modèle cinématique direct du robot et sur un horizon temporel τ, et les trajectoires obtenues sont appelées lignes de fuite du véhicule. Ces lignes de fuite sont projetées dans une carte locale de l'environnement, et les trajectoires qui entrent en collision avec les obstacles sont éliminées (Figure 26). Les trajectoires restantes sont appelées lignes de fuite libres. Ces trajectoires sont alors comparées entre elles au moyen d'un critère de sélection, qui quantifie l'écart entre chaque trajectoire et la courbe de rattrapage fournie par le module de navigation globale. La trajectoire associée au meilleur critère de sélection est choisie, et les commandes correspondant à un partie réduite de cette trajectoire (sur un temps d'échantillonnage inférieur à l'horizon temporel τ) sont envoyées comme consigne au module de pilotage du robot. La capacité réactive de cette méthode a été validée [54], mais cette méthode n'avais jamais été implantée avec un navigateur global.

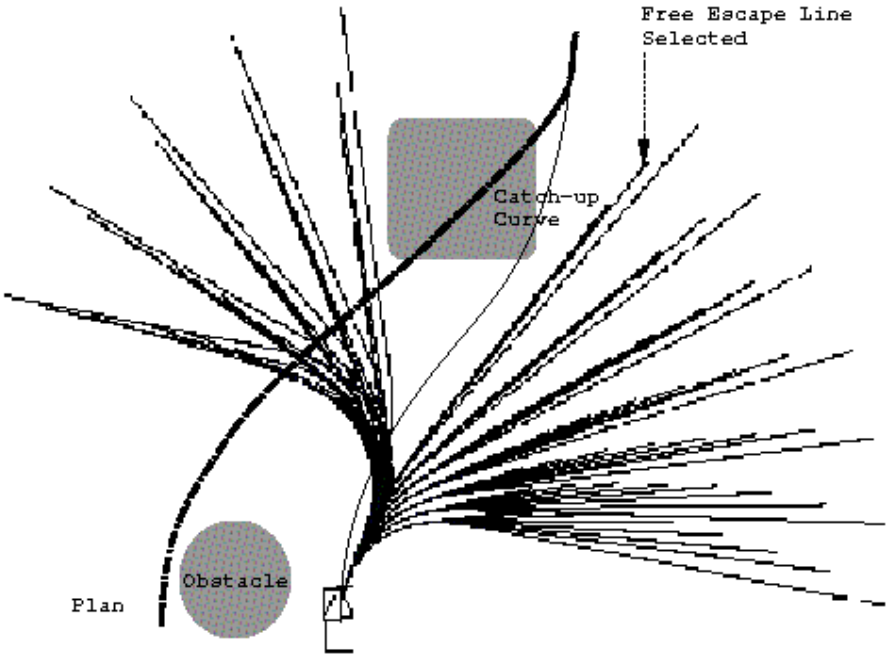

Figure 26 - Projection des lignes de fuite du robot dans un modèle local de l'environnement

Le principe général de ces approches par modèle direct et projection de trajectoires est donc de partir des trajectoires réalisables par le robot, et sélectionner parmi un ensemble représentatif de ces trajectoires celle qui permet de remplir du mieux possible la mission de navigation assignée au robot. Par l'utilisation du modèle cinématique direct du robot, et la prise en compte explicite des contraintes sur les commandes, ce type de méthode est facilement adaptable à n'importe quel type de robot mobile (du moment qu'on connait le MCD). En contrepartie, le problème de ces méthodes est qu'elles sont par nature discrètes (test d'un nombre fini de trajectoires), et le nombre de trajectoires considérées affecte directement le temps de calcul. Pour être exploitable en temps réel, il est donc nécessaire de trouver des compromis entre la discrétisation des trajectoires admissibles (et donc leur précision par rapport au but du robot), et les temps de calcul.

Comme base de développement de notre travail, nous avons choisi de repartir de la méthode de navigation par lignes de fuite pour deux principales raisons : d'une part, la prise en compte explicite du modèle cinématique direct du robot, permettant de faciliter l'adaptation à tous types de robots mobiles ; d'autre part, l'adaptation plus facile de cette méthode dans l'architecture de contrôle utilisée par le laboratoire (cf. chapitre II partie IV), avec des entrées qui sont premièrement une carte locale de l'environnement local du robot (en provenance du bloc perception de niveau 3), et deuxièmement une courbe de rattrapage qui pourra être lié au chemin fourni par le module de planification de chemin de l'architecture (module de décision de niveau 4). Dans la section suivante, nous présentons la nouvelle formalisation de cette méthode et comment nous l'adaptons à notre architecture de contrôle, en l'implémentant notament avec un planificateur de chemin lui fournissant des points de passages successifs à atteindre. Des simulations sur différents robots sont ensuite effectuées, de manière à tester l'application de cette méthode à différentes cinématiques de robots. Nous tenterons de mettre en évidence les points positifs liés à ce type d'approche, et nous discuterons des limites.

II. La méthode des lignes de fuite

Nous considérons que le robot se trouve dans l'état X_0 à l'instant t_0, et cherche à atteindre une position cible tout en évitant les obstacles sur son chemin. Les différentes étapes de la méthode des lignes de fuite sont :

- générer toutes les trajectoires Γ admissibles par le robot, c'est-à-dire respectant ses différentes contraintes (cinématiques, dynamiques etc…) sur un horizon temporel τ ; ces trajectoires sont également appelées lignes de fuite,
- éliminer les lignes de fuite non-libres (qui intersectent ou passent trop près des obstacles), en projetant celles-ci dans une carte locale mise à jour régulièrement pendant la navigation,
- choisir parmi les lignes de fuite libres celle que l'on proposera au pilote, à l'aide d'un critère de sélection qui dépend de la mission assignée au robot.

Cette méthode utilise uniquement le modèle direct du robot, pour déterminer les trajectoires réalisables par le robot et les projeter dans une carte locale. Ainsi la matrice jacobienne inverse n'a pas besoin d'être définie. L'opération entière est réitérée à une période T_e inférieure à τ (Figure 27).

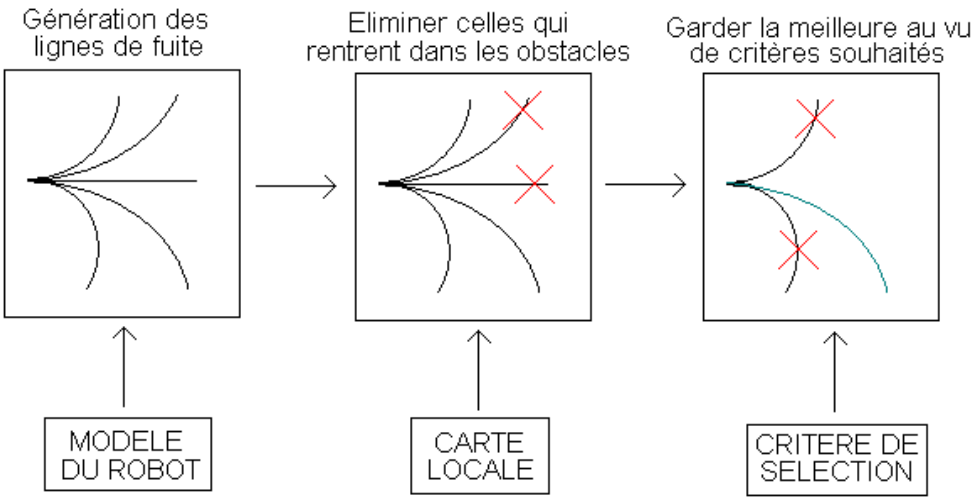

Figure 27- Principe des lignes de fuite

Uniquement une partie de la trajectoire sélectionnée est réellement appliquée par le robot, le processus complet est ensuite réitéré à l'instant d'échantillonnage suivant $t_0 + T_e$, avec les nouvelles informations collectées au cours de la progression du robot. Cette réitération du processus toutes les T_e secondes joue le rôle de bouclage, en mesurant la position réelle du robot et des obstacles.

Dans la section suivante, nous développons le formalisme de la méthode des lignes de fuite, en reprenant les 3 principales phases du processus. Ce formalisme s'appuie notamment sur le formalisme de Sontag [65] (Figure 28). Le système automatisé est défini comme un ensemble d'espaces et de fonctions définies sur ces espaces :

$$\Sigma = (\sigma, \chi, \mu, \phi, \Psi, h), \qquad \text{Équation 9}$$

σ est l'espace des temps,

μ est l'espace d'entrée du système ; les vecteurs de cet espace sont notés u,

χ est l'espace d'état du système ; les vecteurs de cet espace sont notés X,

Ψ est l'espace de sortie du système ; les vecteurs de cet espace sont notés Y.

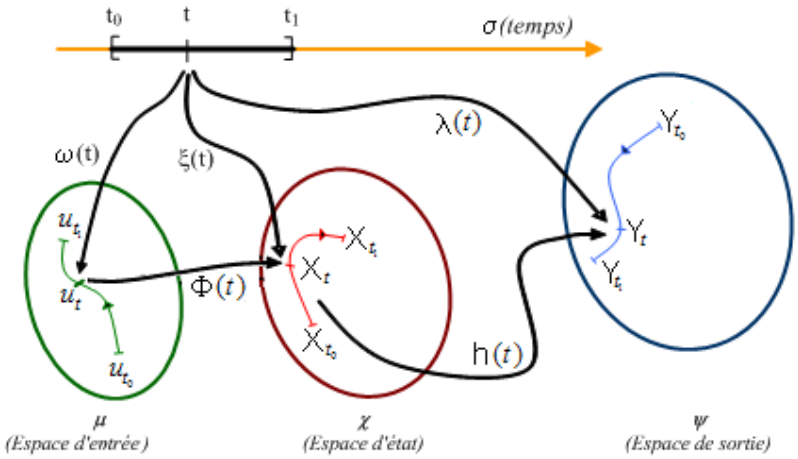

Figure 28- Formalisme de Sontag

ω(t) est la fonction d'entrée, elle permet de connaître l'entrée u du système (la commande) à un instant t donné : $u_t = \omega(t)$.

ξ(t) est la fonction d'état simplifiée, elle permet de connaître l'état X du système à un instant t donné : $X_t = \xi(t)$.

λ(t) est la fonction de sortie simplifiée, elle permet de connaître la sortie Y du système à un instant t donné : $Y_t = \lambda(t)$.

ϕ est la fonction d'état définie telle que l'image de ω par ϕ dans l'espace d'état soit ξ : $\xi(t) = \phi \circ \omega(t)$.

h est la fonction de sortie définie telle que l'image de ξ par h dans l'espace de sortie soit λ : $\lambda(t) = h \circ \xi(t)$ ou encore : $\lambda(t) = h \circ \phi \circ \omega(t)$.

1. Trajectoires admissibles par le robot et lignes de fuite

La première partie de la méthode consiste à générer les trajectoires admissibles par le robot. Pour cela, des fonctions d'entrée u doivent être définies, de manière à générer des trajectoires correspondantes par le biais du modèle cinématique direct du robot.

Une fonction d'entrée u est admissible par le robot sur un intervalle $[t_0, t_0+\tau]$ si elle respecte les contraintes suivantes :

- contraintes cinématiques : contraintes correspondant au modèle cinématique direct du robot, c'est à dire ses possibilités de mouvements,

- contraintes de saturation des actionneurs (e.g. : vitesses de rotation maximales des moteurs en charge en marche avant et en marche arrière, butées pour la direction des roues orientées),

- contraintes sur la dynamique des actionneurs (e.g. : accélérations et décélérations maximales des différents actionneurs).

Nous appelons $\Omega_{[t0,\ t0+\tau]}$ l'ensemble des commandes admissibles par le robot sur l'intervalle $[t_0,\ t_0+\tau]$.

$$\Omega[t_0, t_0+\tau] = \{u[t_0, t_0+\tau]\} \text{u admissible par le robot}\}, \qquad \text{Équation 10}$$

Nous appelons $\Lambda_{[t0,\ t0+\tau]}$ l'ensemble des trajectoires admissibles par le robot sur l'intervalle $[t_0,\ t_0+\tau]$. Il correspond à l'ensemble des trajectoires dont la fonction d'entrée $u \in \Omega_{[t0,\ t0+\tau]}$. On appelle également cet ensemble $\Lambda_{[t0,\ t0+\tau]}$ les lignes de fuite du robot à l'instant t_0 et sur l'horizon temporel τ.

$$\Lambda_{[t0,\ t0+\tau]} = \{\ \Gamma = (\xi,\omega) \,/\, \omega \in \Omega_{[t0,\ t0+\tau]}\}, \qquad \text{Équation 11}$$

2. Élimination des lignes de fuite non-libres

L'étape suivante consiste à comparer l'image de chaque $\Gamma \in \Lambda$ dans l'espace de sortie Ψ, avec la carte des obstacles que nous appelons Θ. Cette carte peut se présenter sous plusieurs formes, mais doit impérativement permettre de connaître la position des obstacles par rapport à celle du robot. A l'aide du critère d'élimination C_{el}, l'ensemble des lignes de fuite libres Λ_L est déterminé.

$$\Lambda_L = \{\ \Gamma = (\xi,u) \,/\, C_{el}\,(\lambda_\Gamma, \Theta) = 0, \Gamma\ \varepsilon\ \Lambda_{[t0,\ t0+\tau]}\}, \qquad \text{Équation 12}$$

En fait, les lignes de fuite que l'on considère comme dangereuses sont éliminées, parce qu'elles intersectent les obstacles ou parce qu'elles passent trop près de ceux-ci, au sens du critère d'élimination. L'ensemble des lignes de fuite résultantes constitue Λ_L.

Par association, nous pouvons alors définir l'ensemble des commandes admissibles libres Ω_L comme l'ensemble des fonctions d'entrées u qui correspondent aux lignes de fuite libres Λ_L.

$$\Omega_L = \{\ u \in \Omega \,/\, \Gamma = (\xi,u) \in \Lambda_L\}, \qquad \text{Équation 13}$$

A ce niveau de notre méthode, le robot dispose, sur l'horizon temporel $[t_0,\ t_0+\tau]$, de toutes les trajectoires qu'il va pouvoir effectuer réellement sans rencontrer aucun obstacle.

3. Sélection de la meilleure ligne de fuite libre

La dernière étape consiste à déterminer la meilleure ligne de fuite parmi Λ_L pour atteindre un chemin ε fourni par le niveau de contrôle supérieur (le planificateur). Ce chemin peut se présenter sous la forme d'un point de passage à atteindre, ou encore d'un parcours continu ou discontinu.

Pour sélectionner la meilleure ligne de fuite libre, nous appliquons un critère, C_{choix} qui permet de quantifier l'intérêt de chaque ligne de fuite libre pour atteindre cet objectif. La ligne de fuite choisie est celle à laquelle est associée la valeur du critère optimale. Cette ligne de fuite est appelée $\Gamma_{choisie}$. La fonction d'entrée qui lui correspond est appelée $\omega_{choisie}$, et elle est envoyée en consigne au niveau de contrôle inférieur (le pilote), qui est chargé de faire en sorte qu'elle soit appliquée par le robot.

$$\Gamma_{choisie} = \{ \Gamma = (\xi,u) / C_{choix} (\Gamma, \varepsilon) \text{ optimal}, \Gamma \in \Lambda_L \} \quad \text{Équation 14}$$

$$u_{choisie} = \{ u \in \Omega_L / \Gamma = (\xi,u) = \Gamma_{choisie} \} \quad \text{Équation 15}$$

Le choix des critères C_{choix} et C_{el} dépend de l'application à effectuer par le robot et de la mission. Par exemple pour une mission d'exploration, si le robot a pour but de cartographier la zone environnante, on pourra établir un critère d'élimination C_{el} assez sévère pour s'assurer que le robot passe à bonne distance des obstacles (car nous ne connaissons pas leur forme dans l'espace et que nous voulons prendre un minimum de risques). On pourra aussi chercher à établir un critère de choix C_{choix} qui favorise les trajectoires les moins coûteuses en énergie (pour les missions d'exploration par exemple).

Par contre si nous prenons une mission de type industrielle, nous allons plutôt chercher à favoriser des notions comme la rapidité du robot, et donc le critère C_{choix} devra favoriser les trajectoires les plus rapides (aux dépens de la consommation d'énergie). Donc nous devons adapter ces critères suivant la situation (robot, mission, application, milieu dans lequel évolue le robot…)

Un autre point à prendre en compte est le fait que seul le début des commandes correspondant à la trajectoire choisie ($u_{choisie}$) sera en fait appliqué par le robot. En effet l'opération complète de navigation par lignes de fuite (de la génération des lignes de fuite du robot à la sélection de $\Gamma_{choisie}$ et $u_{choisie}$), est recommencée de manière périodique, et avec une période inférieure à τ (environ dix fois inférieure). Donc au final, seule une courte portion de la trajectoire $\Gamma_{choisie}$ sera effectivement suivie par le robot.

4. Application à divers robots

Dans [47], nous avons appliqué cette méthode des lignes de fuite à la navigation du robot RAOUL (Robot Autonome d'Observation d'Univers Locaux). Il s'agit d'un robot à roues différentielles, construit à partir d'une plate-forme de Robosoft nommée Robuter. Il fonctionne sous RT Linux, et est équipé de 2 télémètres laser Sick, un à l'avant et un à l'arrière scannant l'environnement sur 180° chacun, et un goniomètre laser pouvant calculer la position/orientation du robot en se référant à 3 balises postées dans l'environnement (Figure 29). Les travaux de Canou [13] ont permis de développer un module de cartographie locale en ligne, qui met à jour ses informations grâce aux mesures effectuées par les télémètres laser toutes les 25ms.

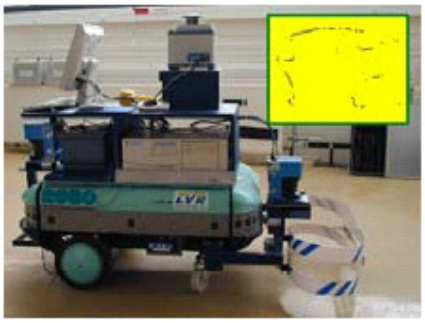

Figure 29- Le robot RAOUL

Ce type de véhicule à roues différentielles a été souvent utilisé pour ses propriétés de mouvement (capacité à tourner sur lui-même sans manœuvrer). Il était couramment employé en tant que base mobile pour des robots mobiles équipés de bras manipulateurs, même si aujourd'hui il tend à se faire remplacer par les robots omnidirectionnels qui offrent plus de souplesse dans la faisabilité des trajectoires de placement. Le modèle du robot à roues différentielles est le suivant :

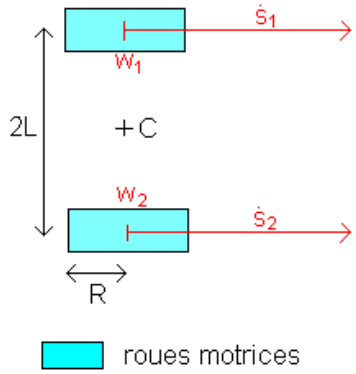

Figure 30- Modèle des roues différentielles

En tenant compte du sens trigonométrique pour ω_1 et ω_2, les vitesses de rotation des deux roues motrices, nous obtenons les équations de mouvement de chaque roue: $\dot{s}_1 = -\omega_1.R$ et $\dot{s}_2 = -\omega_2.R$ (Figure 30). Les équations de mouvement du robot sont:

$$\begin{cases} \dot{s} = \dfrac{\dot{s}_1 + \dot{s}_2}{2} = \dfrac{\omega_2.R - \omega_1.R}{2} \\ \dot{\theta} = \dfrac{\dot{s}_1 - \dot{s}_2}{2L} = \dfrac{\omega_2.R + \omega_1.R}{2L} \end{cases}, \qquad \text{Équation 16}$$

Dans une même famille de trajectoires, il existe une infinité de fonctions d'entrée, et une infinité de trajectoires correspondantes. Dans le cas du robot à roues différentielles, il existe une infinité de couples (ω_1, ω_2) admissibles qui vont générer une infinité de trajectoires à tester, ce qui n'est pas réalisable en pratique. Ainsi, nous sommes contraints de tester un nombre fini de trajectoires. Pour ce faire, une discrétisation de l'espace de commande (ω_1, ω_2) est réalisée. Dans les simulations, nous avons discrétisé cet espace en 5 par 5, cela signifie que nous générons 5*5 = 25 trajectoires à chaque itération de la méthode.

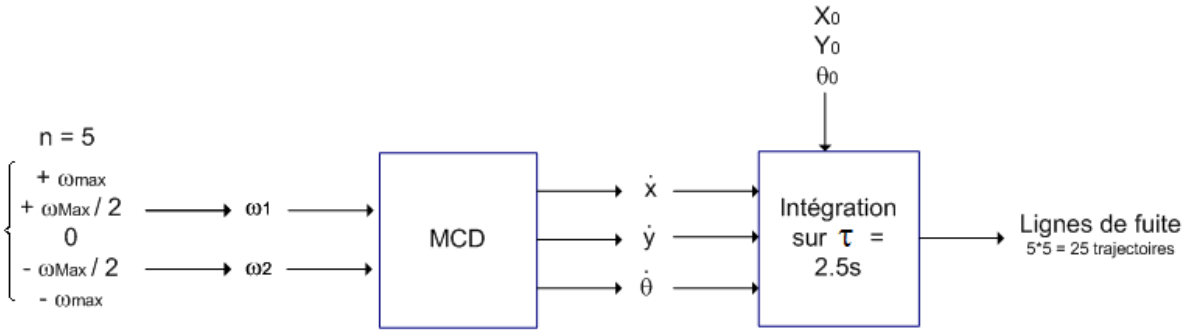

Figure 31 - Discrétisation des entrées du robot et génération des lignes de fuite

En utilisant des fonctions d'entrée de type linéaire (variations linéaires de la vitesse des deux roues entre deux consignes de vitesse, en respectant les différentes saturations des actionneurs), et après passage à travers le modèle cinématique décrit ci-dessus, nous obtenons le type de traces de trajectoires représentées sur la Figure 32.

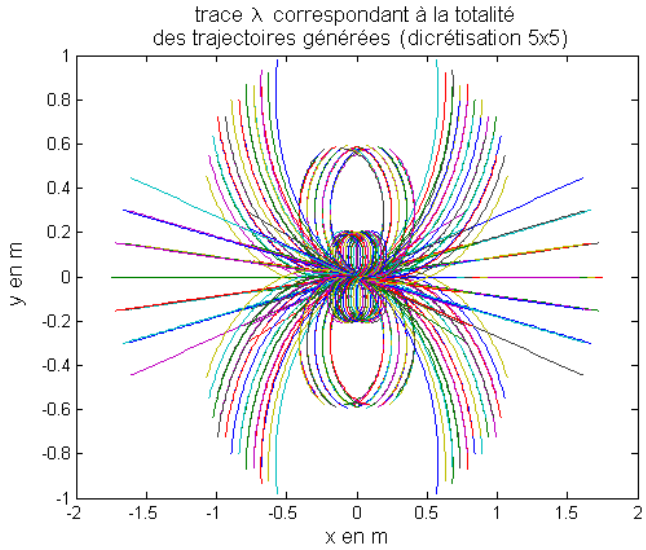

Figure 32- Panel de trajectoires réalisables par le robot à roues différentielles

Ces lignes de fuite sont projetées à partir de l'état courant du robot, et confrontées à une carte de l'environnement du robot (Figure 33). Les trajectoires qui entrent en collision avec les obstacles sont alors éliminées, en accord avec un critère d'élimination C_{el}:

$$C_{el} : dist\left\{\lambda_i(t) / sgmt_j\right\} - d_s > 0 \;\forall\; j \qquad \text{Équation 17}$$

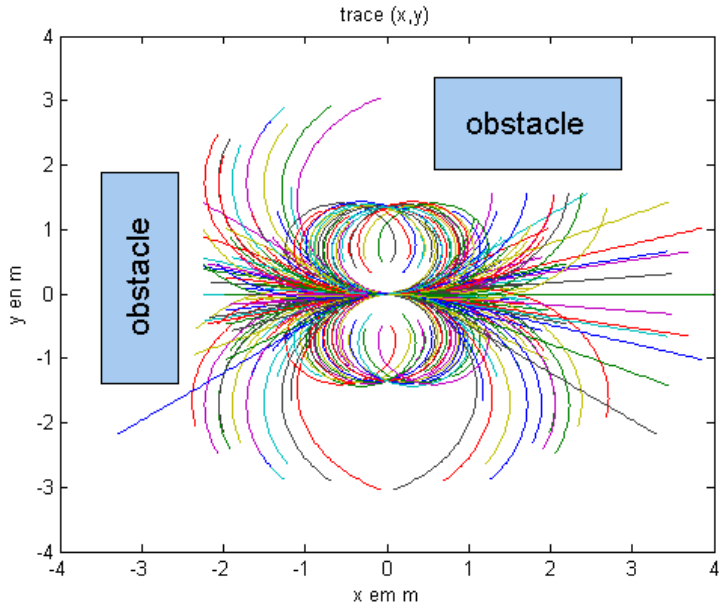

Figure 33 - Projection des lignes de fuite dans une carte locale, et élimination des trajectoires non libres

Les lignes de fuite restantes sont appelées lignes de fuite libre. A chacune d'entre elle est alors associé une valeur par le biais d'un critère de sélection C_{choix}. Ce critère permet de quantifier l'intérêt de la trajectoire dans le but d'atteindre un point de passage donné. Le critère que nous avons développé tient compte de la distance cartésienne entre le point final de la trajectoire et le point de passage, ainsi que de la différence entre l'angle d'orientation du robot et l'angle où se situe le point de passage.

$$C_{choix}(\Gamma_i, \varepsilon) = \sqrt{(x_i(t_0 + T_p) - x_\varepsilon)^2 + (y_i(t_0 + T_p) - y_\varepsilon)^2} \times fact \qquad \text{Équation 18}$$

$$fact = 1 + k_\theta \times \left| \theta_i(t_0 + \tau) - \theta_{cible/robot} \right| \qquad \text{Équation 19}$$

La trajectoire à laquelle est associée la plus faible valeur de critère est sélectionnée, et les commandes correspondant à cette trajectoire sont envoyées en consigne pour le module de pilotage. Ces consignes seront suivies pendant un temps d'échantillonnage inférieur à l'horizon temporel sur lequel sont calculées les lignes de fuite. Le processus complet est alors réitéré à l'instant d'échantillonnage suivant.

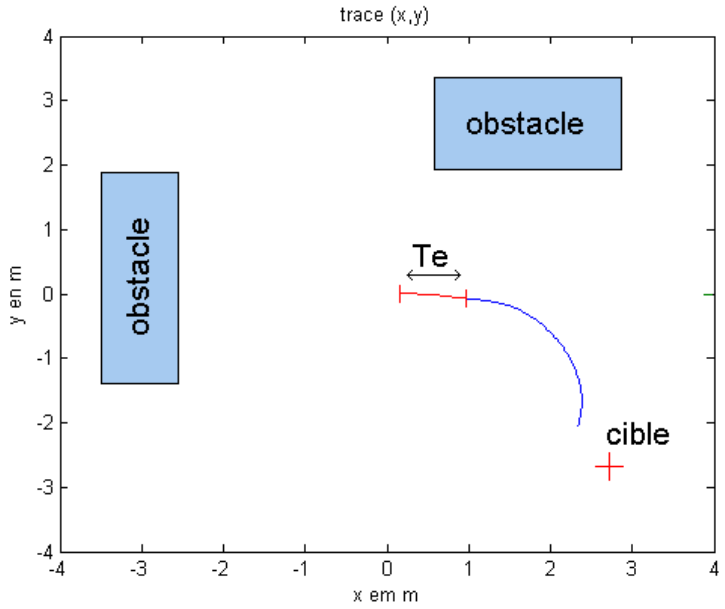

Figure 34 - Sélection de la meilleure ligne de fuite et application des commandes correspondantes sur un échantillon temporel

En utilisant ce type de trajectoires, nous avons cherché à valider notre navigateur local par lignes de fuite en simulation. Nous considérons que le robot dispose d'une carte locale de son environnement, et qu'il dispose d'un ensemble de points de passage à franchir successivement. Sa mission consiste alors à suivre au mieux le chemin prédéfini tout en s'adaptant aux contraintes locales (obstacles et capacités de mouvement du robot). Un exemple de résultat obtenu est présenté sur la Figure 35. Dans cette représentation, le robot est assimilé à un point (son point de contrôle). La trace des déplacements accomplis par le centre du robot est représentée en bleu, et chaque nouvelle itération du programme de navigation est symbolisée par un petit trait perpendiculaire à la trace. Les obstacles sont assimilés à des segments, et ne sont pas directement représentés sur la carte. Ce que l'on représente sur la carte est en fait un élargissement de ces segments, d'une distance égale à la demi-longueur maximale entre le centre du robot et sa périphérie, à laquelle on ajoute une marge de sécurité. Chaque segment d'obstacle élargi donne alors une sorte d'ellipse, dans laquelle le point de contrôle du robot ne doit pas pénétrer sous risque de collision entre le robot et l'obstacle.

Dans cette simulation, le robot part arrêté du point de coordonnées (-8, -8), et il doit se rendre au point (-6, 4), via un certain nombre de points de passage représentés par les croix en rouge. Les obstacles segments sont élargis de 60 cm pour simuler la taille du robot.

Les simulations ont permis de valider l'efficacité de ce type d'approche, en trouvant des trajectoires permettant de concilier les contraintes locales du problème de navigation avec les objectifs globaux (chemin à suivre). De plus, nous avons observé que cette méthode laissait une certaine autonomie au système, avec par exemple des marches arrières improvisées par le robot pour se sortir efficacement de certains pièges.

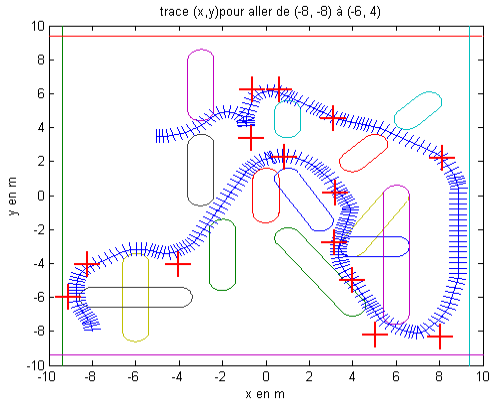

Figure 35- Navigateur par lignes de fuite appliqué à un robot à roues différentielles

Cette méthode a également été appliquée à un robot mobile à double essieu directeur, le CyCab dans [48], c'est-à-dire un véhicule de type voiture avec à la fois les roues avant et les roues arrière directrices. L'avantage de ce type de véhicule est sa manœuvrabilité améliorée du fait de la diminution du rayon de braquage, ce qui en fait un véhicule intéressant pour les milieux relativement encombrés. Le modèle du véhicule étudié est le suivant :

$$\dot{\theta}_M = v.\frac{\sin(\xi + k\xi)}{L.\cos(\xi)}$$

$$\dot{x}_M = \frac{1}{2}v\left(\cos(\theta + k\xi) + \cos(\theta + \xi)\right), \qquad \text{Équation 20}$$

$$\dot{y}_M = \frac{1}{2}v\left(\sin(\theta + k\xi) + \sin(\theta + \xi)\right)$$

Les lignes de fuite générées à partir de ce modèle sont représentées sur la Figure 36 (seuls des mouvements en marche avant ont été générés) :

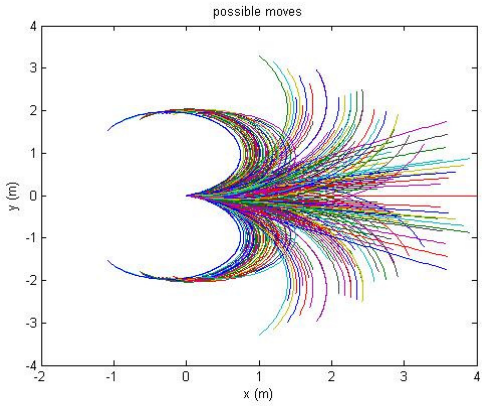

Figure 36- Panel de trajectoires réalisables par le robot à double braquage

La Figure 37 montre les résultats d'une simulation de navigation par ligne de fuite appliquée au véhicule CyCab dans un environnement de 40x40 m². La trace des déplacements effectués par le point de contrôle C du robot sur le sol est représentée en bleue, et les lignes bleues perpendiculaires représentent la fréquence d'échantillonnage de la méthode. Les croix rouges sont les points de passage fournis au navigateur, que le robot doit atteindre successivement.

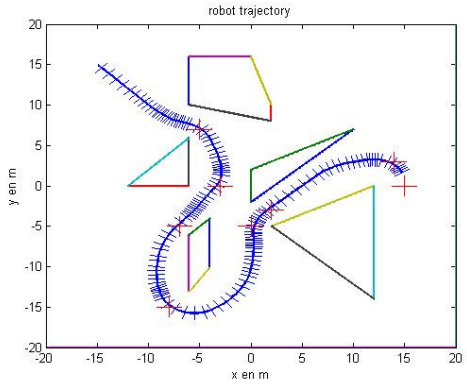

Figure 37- Navigateur par lignes de fuite appliqué à un véhicule à double braquage CyCab

Contrairement à la simulation précédente, cette fois-ci les obstacles, sous forme de polygônes, sont directement représentés sur la figure. La distance entre la périphérie du robot et l'obstacle le plus proche est représentée sur la Figure 38, ce qui permet de vérifier que le robot n'entre pas en collision avec les obstacles. En effet cette distance reste toujours supérieure à la largeur du robot qui est fixée à 1.44m dans la simulation.

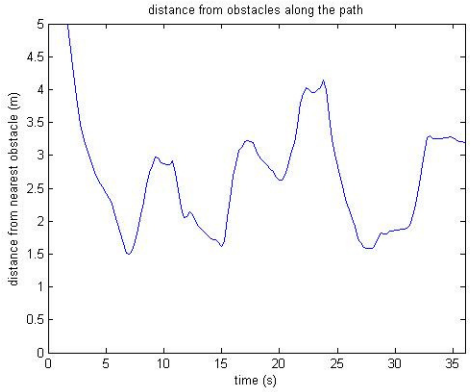

Figure 38- Distance robot/obstacles le long du parcours

Cette simulation a permis de valider encore une fois l'intérêt de cette méthode dans le cadre de la navigation d'un robot de type voiture, ou a double braquage.

5. Limites de l'approche par ligne de fuite

Ses principaux atouts sont le respect de la cinématique du robot, et le fait qu'un modèle inverse du véhicule ne soit pas nécessaire (modèle qui n'existe pas toujours ou qui peut souffrir de points de singularités). Cette méthode est donc facilement adaptable à n'importe quel type de robot mobile dont on connait le modèle cinématique direct.

Elle nécessite cependant de travailler en coopération avec un planificateur de chemin, qui lui fournit soit un chemin complet soit des points de passage. En effet, bien qu'elle repousse le

problème des minima locaux en testant différentes trajectoires sur un horizon de prédiction plus ou moins élevé (ce qui permet de trouver des trajectoires de contournement dans certains cas), elle reste tout de même soumise aux problèmes de minima locaux qui peuvent intervenir par exemple lorsque le robot est confronté à un obstacle en forme de U. Ainsi, l'efficacité de la méthode reste tributaire de la qualité des points de passage fournis par le planificateur de chemin. Cela est illustré par la simulation suivante sur la Figure 39.

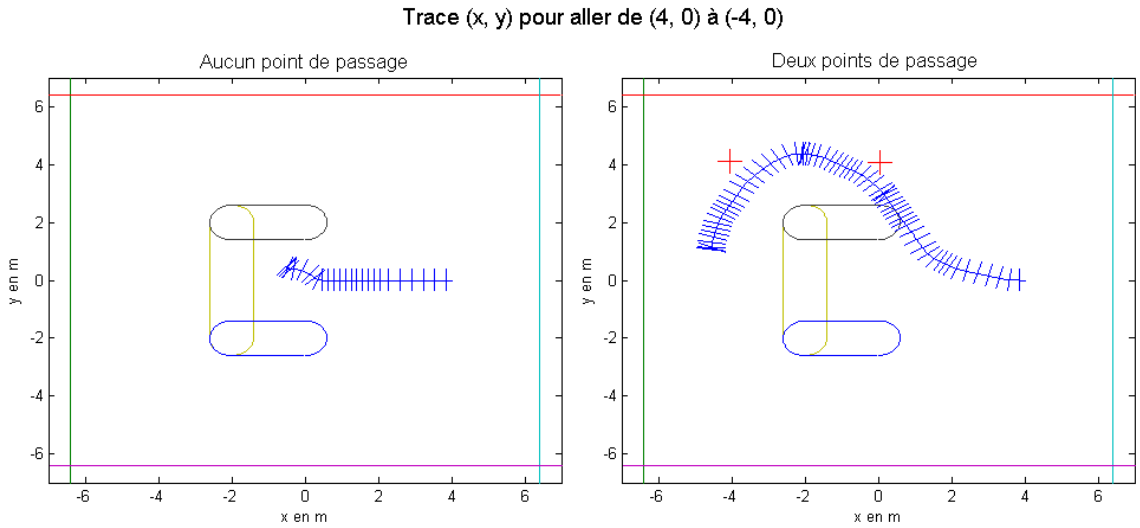

Figure 39- Piège en U et influence du planificateur

Ce problème de minima locaux est directement lié à la longueur de l'horizon de prédiction, au type de famille de trajectoires et au nombre de trajectoires testées, qui permettent ou non de trouver des trajectoires de contournement. Ce problème est également lié au temps de calcul disponible, qui autorisera à tester un certain nombre de trajectoires en ligne.

Une autre limitation de la méthode est liée à la discrétisation des fonctions d'entrée, qui va découler en un nombre fini de trajectoires testées. La précision des trajectoires pour atteindre un point donné sera directement liée à ce nombre de trajectoires testées et donc à la taille de la discrétisation. Cette limitation fait qu'il ne sera pas possible d'atteindre certains points sous une certaine précision, ce qui fait que la méthode peut être mise en défaut si on lui demande de positionner précisément le robot. Cette observation est illustrée sur la Figure 40, où le robot, qui ne parvient pas à se positionner assez près du point de coordonnées (5, -5), va tourner autour de celui-ci sans jamais trouver de trajectoire satisfaisante.

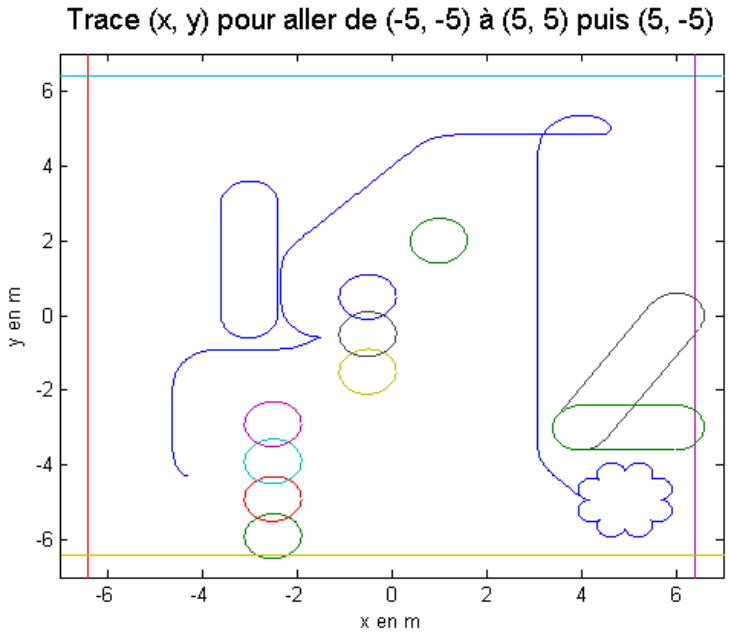

Figure 40- Tolérance trop faible sur la position à atteindre

Ce problème est directement lié au fait que l'on teste un nombre fini de trajectoires. Pour pouvoir trouver une trajectoire permettant de rallier un point précis, sous réserve que la trajectoire existe, il faut pouvoir déterminer une fonction d'entrée optimale.

III. Des lignes de fuite à la commande prédictive

Fort de ces résultats, nous avons cherché à améliorer notre approche en modernisant celle-ci par l'utilisation d'outils tels que les algorithmes d'optimisation pour affiner nos trajectoires, et en cherchant des moyens d'être plus robuste aux problèmes de minima locaux ; tout ceci en conservant ce qui fait le point fort de la méthode, à savoir l'utilisation de modèle cinématique direct pour chercher des solutions de navigation à partir des trajectoires réalisables par le robot.

C'est ainsi que nous nous sommes intéressés au formalisme de la commande prédictive. Cette méthode avancée de commande, utilisée en automatique pour permettre à un système donné de suivre une trajectoire de référence, possède en effet un certain nombre de points communs avec notre méthode de navigation par ligne de fuite : l'utilisation d'un modèle du système à commander pour prévoir son évolution sur un horizon temporel glissant, le bouclage implicite de la méthode par une mise à jour périodique des informations sur l'évolution réelle du système et de son environnement. Nous avons ainsi cherché à voir si nous ne pouvions pas trouver certains outils, comme les méthodes d'optimisation de trajectoires, pour les adapter à notre problème de navigation.

Dans la section suivante, nous présentons la commande prédictive telle qu'elle est définie en automatique. Après une courte présentation, nous voyons plus en détail son principe en s'attachant à montrer le rôle des différents éléments sur lesquels repose cette méthode.

1. Présentation de la commande prédictive

La commande prédictive (également appelée commande optimale sur horizon glissant ou based model control en anglais) est une méthode de commande avancée qui utilise un modèle du système à commander pour prédire son comportement dans le futur. A chaque période d'échantillonnage, la séquence de commandes optimales est calculée, et seul le début de cette séquence est appliqué au système. Propoi [60] dans les années 60 fut le premier à penser à introduire le modèle du système à l'intérieur de la boucle de commande, et de s'en servir pour anticiper les comportements de celui-ci.

Cette méthode a été développée à la base dans l'industrie pétrolière (Shell), puis étendue progressivement dans différents domaines (chimie, automobile, aérospatiale etc…). Richalet en 1976 fut le premier à l'utiliser dans une application industrielle, sous le nom Identification et Commande (IDCOM) [64]. Initialement limitée aux systèmes relativement lents, elle est aujourd'hui utilisée dans des applications plus rapides grâce notamment aux progrès de l'informatique. Ainsi elle commence à être développée dans des applications robotiques, comme par exemple pour effectuer du suivi de trajectoire [36].

Pour pouvoir appliquer la commande prédictive, il faut avoir un modèle du système, qui permette de prévoir ses évolutions sur un horizon temporel donné T_p (dit horizon de prédiction) [4] [5]. Il faut également une trajectoire de référence pour servir de modèle pour atteindre le but fixé (Figure 41). Le problème est mis sous la forme d'un problème d'optimisation sous contraintes : le critère à optimiser est l'erreur entre la trajectoire de référence et la trajectoire attendue du système, suivant la commande envoyée, sur l'intervalle $[t_0, t_0+ T_p]$. Cette commande doit respecter un certain nombre de contraintes sur la commande elle-même ou sur la trajectoire qu'elle engendre au système. Un algorithme d'optimisation est alors nécessaire pour déterminer la commande qui minimise ce critère.

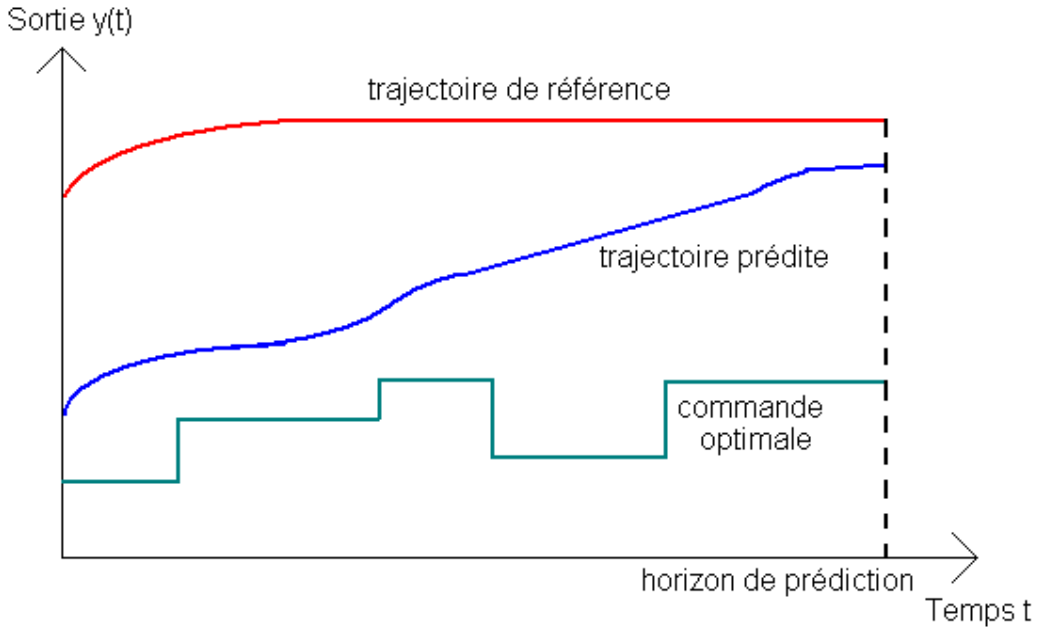

Figure 41- Principe de la commande prédictive

Une fois la commande choisie, de la même manière que pour les lignes de fuite, seule les premiers échantillons de celle-ci sera réellement appliquée au système, avant de reprendre le processus complet.

Le souci de cette méthode est qu'elle peut s'avérer gourmande en temps de calcul, c'est pour cela qu'à la base elle n'était appliquée que sur le contrôle de processus lents, notamment dans les processus chimiques où elle était particulièrement employée [44] [62].

Dans [40], Lenain développe une commande prédictive pour une meilleure prise en compte des glissements qui apparaissent inévitablement lors du suivi d'une trajectoire de référence par des véhicules agricoles. Une précision de l'ordre de 5cm en latéral est demandée pour ce suivi. Tant que le véhicule ne glisse pas, des lois de commande basées sur l'utilisation d'un GPS RTK comme capteur de position ont prouvé leur efficacité, mais ce n'est évidement pas le cas en terrain agricole. L'intégration des glissements dans la modélisation cinématique du robot [39] ont permis de garantir une précision satisfaisante, sauf lors des transitions de la courbe de référence (changements de courbure). Pour pallier à ce problème, l'utilisation d'une commande prédictive a permis d'anticiper le comportement du véhicule lors des modifications de courbure et ainsi d'améliorer sensiblement la précision du suivi de trajectoire même lors des phases de transition.

Le problème de la navigation d'un robot mobile peut être abordé sous la forme d'un problème d'optimisation sous contraintes ; considérant les obstacles et les contraintes cinématiques du robot comme les contraintes du problème d'optimisation, le modèle cinématique direct du robot est utilisé pour prévoir l'évolution du robot suivant la commande envoyée à ses actionneurs. En 2007 Klancar a implanté cette méthode sur un petit robot à roues différentielles, et à démontré son efficacité pour faire du suivi de trajectoires : la trajectoire du robot est souple et converge efficacement vers la trajectoire de référence [36].

2. Principe

Le principe de fonctionnement de la commande prédictive est très proche de la manière dont procéderait une personne pour atteindre un objectif donné. Pour des facilités de compréhension, nous l'illustrons avec un schéma qui s'apparente à un problème de planification de chemin.

Au début du processus, nous sommes dans une situation donnée, et le problème consiste à atteindre une situation désirée. Entre les deux, se dressent un certain nombre de contraintes, qui nous empêchent de faire certaines actions. Pour résoudre ce type de problème, l'idée de base est de commencer par regarder les différentes manières possibles d'atteindre notre objectif. Si plusieurs manières permettent d'atteindre l'objectif, nous les comparons, suivant un critère préétabli, pour déterminer celle qui nous parait la plus appropriée. En fait, nous cherchons à déterminer la stratégie optimale pour résoudre notre problème (Figure 42).

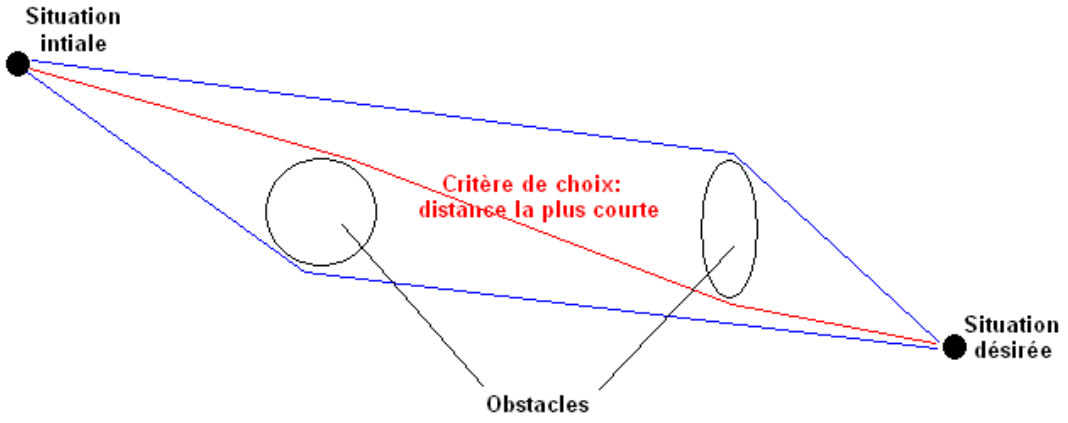

Figure 42- Choix d'un comportement optimal

Une fois cette stratégie optimale déterminée, nous commençons à l'appliquer. Nous avançons d'un premier pas selon ce qui était convenu, puis nous nous arrêtons pour faire le point. En effet, tout ne s'est pas forcément passé comme il était prévu, la stratégie convenue peut avoir été plus ou moins bien respectée. Les contraintes peuvent également avoir changé, ou alors elles peuvent avoir été mal évaluées. Globalement, le problème est remis à jour avec toutes les nouvelles informations à disposition (Figure 43).

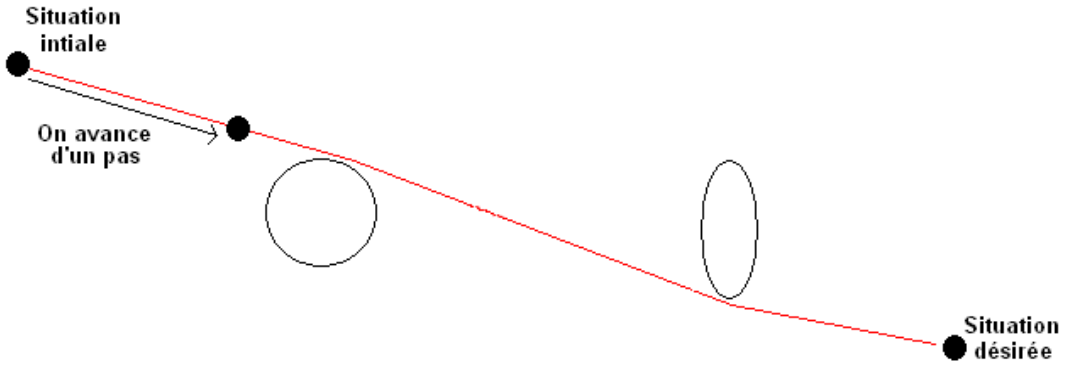

Figure 43- Application du comportement choisi sur un premier pas

Le problème ayant évolué, il est alors nécessaire de redéfinir une nouvelle stratégie optimale. De nouveau, les différentes options sont envisagées, et l'une d'elle est choisie selon le critère de choix (Figure 44).

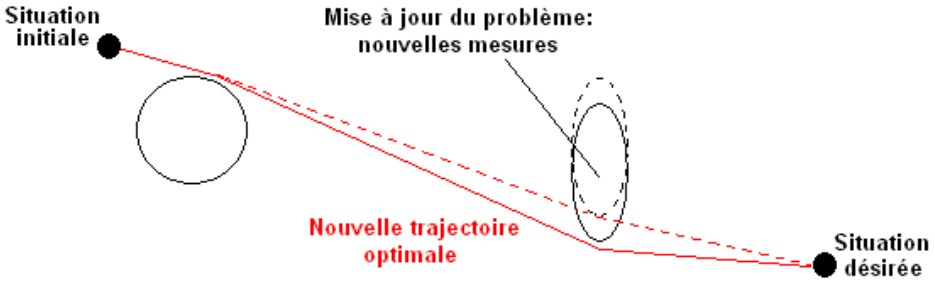

Figure 44- Mise à jour de la situation

A nouveau, nous commençons à appliquer la stratégie choisie, et une fois le premier pas effectué, nous réévaluons la situation (Figure 45).

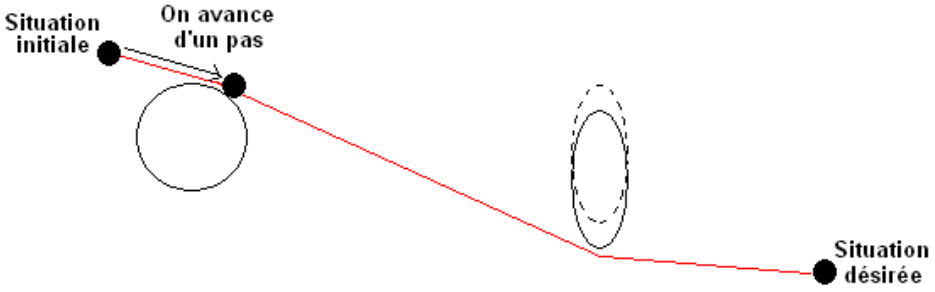

Figure 45- Re-application du comportement optimal

Ainsi, le principe de la commande prédictive est de modéliser un problème de suivi de trajectoire par le système à commander, à l'instant t_0, sous la forme d'un problème d'optimisation sous contraintes. Cette modélisation s'appuie sur un modèle capable de prédire les évolutions du système sur un horizon de temps défini, que l'on appelle horizon de prédiction T_p.

La commande prédictive peut donc être utilisée dès lors que l'on possède les éléments suivants :

- un modèle du système que l'on souhaite commander, qui permette de prédire les évolutions du système, sur l'horizon de prédiction T_p, suivant la séquence de commandes appliquée au système sur ce même horizon,
- une trajectoire de référence pour ce système : trajectoire au sens automatique du terme et pas robotique, c'est-à-dire qui n'est pas associée à des commandes (mais juste une succession continue ou discrète d'états du système),
- et enfin un algorithme d'optimisation sous contraintes, qui puisse fonctionner dans une application temps réel.

Pour transformer le problème de suivi de trajectoire en problème d'optimisation, on définit une fonction coût J, qui dépendra :

- de l'état initial du système X_0 à t_0,
- de la commande u définie sur l'intervalle $[t_0, t_0+T_p]$,
- du temps t.

Cette fonction coût est également liée à la trajectoire de référence. Le but de l'optimisation est donc de minimiser cette fonction coût, et la « variable » sur laquelle on joue pour cela est la commande u. Cela doit être fait en respectant généralement un certain nombre de contraintes sur la commande u, le temps t ou l'état X du système. C'est là qu'intervient l'algorithme d'optimisation, puisqu'il devra résoudre ce problème en temps réel.

La première partie de la commande u solution du problème d'optimisation est appliquée sur le système, et le processus complet est réitéré de manière périodique à une certaine fréquence d'échantillonnage f_e. Le bouclage du système est implicite, il résulte du fait que l'état initial du système considéré dans le problème d'optimisation, est obtenu par l'intermédiaire des différents capteurs permettant d'estimer l'état du système en temps réel.

3. Discussion

Nous retrouvons dans cette méthode de nombreux points communs avec notre méthode des lignes de fuite : une projection de l'évolution du système suivant la commande appliquée, via un modèle, et sur un horizon de prédiction T_p ; une recherche de la fonction de commande u permettant de minimiser un critère ; seulement une partie de la solution trouvée est appliquée au système réel, et le processus est réitéré à la période d'échantillonnage T_e.

Cependant cette méthode requiert une trajectoire de référence, que nous ne possédons pas dans le cas de notre problème de navigation. En effet nous possédons au mieux des points de passage fournis par un planificateur de chemin, avec éventuellement des orientations / vitesses de passage. De plus, les obstacles sont une contrainte forte dans le cas du navigateur, qui peut entrainer des problèmes de minima locaux, qui sont moins présents dans les problèmes classiques de commande prédictive.

Cette famille alternative de navigation par modèle direct – notamment par lignes de fuite – montre sa facilité de mise en œuvre en prenant en compte la cinématique et l'environnement du robot pour l'évitement d'obstacles. Mais les résultats des simulations montrent également les limitations de cette méthode, car elle se résout de manière numérique dans un espace discret. Pour travailler dans un espace continu, nous nous appuyons sur la commande prédictive utilisée en automatique de régulation de process. Cette formalisation permet la projection d'un ensemble de trajectoires sur un horizon temporel, duquel nous pouvons déterminer la trajectoire optimale au sens d'une fonction coût. Cette approche sera développée dans le cadre de la navigation pour une application de robotique mobile.

CHAPITRE IV
COMMANDE PREDICTIVE POUR UN NAVIGATEUR DE ROBOT MOBILE

Structurellement, la commande prédictive peut nous permettre de réaliser une navigation par modèle direct. Mais nous devons y intégrer les contraintes cinématiques et d'environnement du robot. Le problème de navigation se pose alors sous la forme d'un problème d'optimisation sous contraintes. Les éléments de la commande prédictive, à savoir la fonction coût, les commandes et les contraintes, doivent être définis et spécifiés dans un cadre robotique : l'utilisation de différentes familles de trajectoires dans les commandes permet d'adapter le comportement du robot aux différentes phases de déplacement ; l'intégration d'une carte locale de l'environnement du robot dans les contraintes permet de générer des trajectoires évitant les obstacles ; et la résolution de la fonction coût se fait au choix par des algorithmes déterministes ou stochastiques.

I. Introduction: rôle du navigateur et principe général

Le module de navigation joue un rôle de pivot dans l'architecture, il fait le lien entre la partie réactive et la partie délibérative de l'architecture. Il possède deux entrées (Figure 46). La première entrée est un chemin fourni par un module de planification sous la forme d'un parcours continu ou de points de passages successifs. La seconde entrée est une carte de l'environnement local du robot, qui peut être présentée sous différentes formes (nous considérerons le cas des grilles d'occupation et des cartes segmentées).

Le travail du navigateur consiste, à partir de ces deux entrées, à déterminer une trajectoire réalisable par le robot (*i.e.* qui respecte ses différentes contraintes cinématiques et dynamiques telles que la non holonomie du robot ou les saturations des actionneurs), qui n'intersecte pas les obstacles, et qui lui permette de suivre au mieux le chemin fourni par le planificateur de chemin.

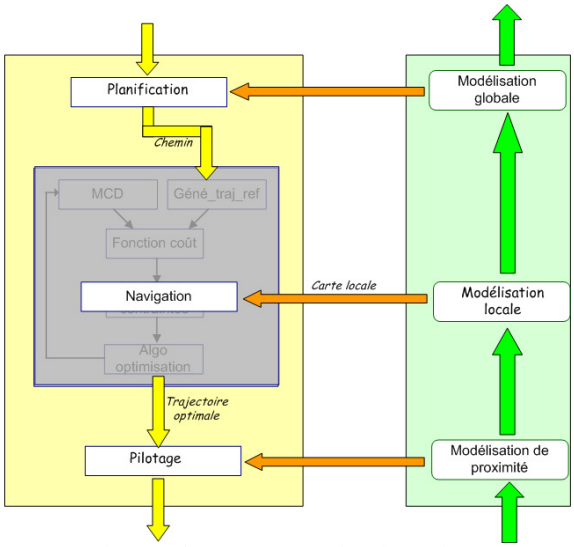

Figure 46: entrées et sorties du navigateur

La méthode de navigation que nous proposons se déroule suivant plusieurs étapes (Figure 47). Tout d'abord, une courbe de rattrapage, permettant au robot de rallier le chemin fourni par le planificateur, est calculée. En effet, à cause des erreurs de localisation, des contraintes locales dues au robot ou aux obstacles dynamiques, le robot ne se situera jamais réellement à l'endroit prévu par le planificateur. Une fois cette courbe déterminée, elle est paramétrée en fonction du temps de manière à obtenir une trajectoire de référence pour notre navigateur. En se basant sur la connaissance du modèle cinématique direct du robot et de ses contraintes, on peut déterminer les trajectoires réalisables par le robot sur un horizon temporel Tp donné. Ces trajectoires sont projetées dans la carte locale de l'environnement. Un algorithme d'optimisation sous contraintes se charge alors de déterminer la trajectoire réalisable par le robot la plus proche de la trajectoire de référence, au sens d'un critère que nous avons développé. Enfin, les commandes correspondant à cette trajectoire sont fournies aux actionneurs du robot, et appliquées sur un temps d'échantillonnage $T_e<T_p$ avant de reconsidérer l'ensemble du problème de navigation et réitérer le processus complet.

Nous développons maintenant ces différents points successifs.

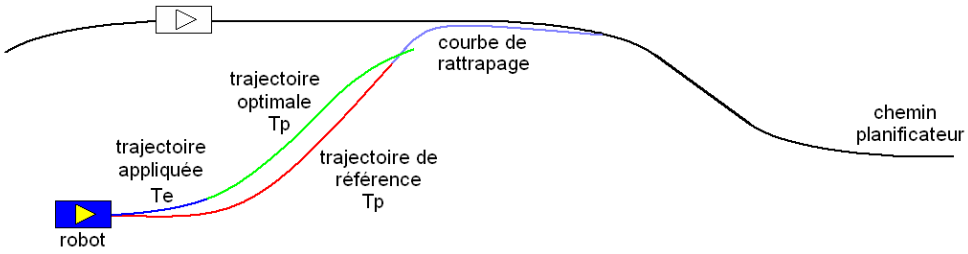

Figure 47: Fonctionnement général du navigateur par commande

II. Modélisation

Nous commençons ici par développer la modélisation du robot mobile d'une part, et le problème de navigation dans sa globalité d'autre part. A la fin du second chapitre, nous avons redéfini précisément le cadre de ce problème, et nous intégrons ici les différents éléments qui le composent, de manière à obtenir la forme d'un problème d'optimisation sous un certain nombre de contraintes.

1. Le robot

Nous commençons par définir un horizon de prédiction, noté T_p, qui est la période sur laquelle nous définissons l'état et la commande du robot. Soit T_e la période d'échantillonnage d'envoi des commandes au robot. La période d'échantillonnage et l'horizon de prédiction sont liés par la formule :

$$T_p = k.T_e, \text{ avec } k \in \text{IN} \qquad \text{Équation 21}$$

L'horizon de prédiction, qui est donc un multiple de la fréquence d'échantillonnage, est le temps sur lequel nous allons modéliser l'évolution de notre robot.

Nous considérons un robot mobile évoluant sur terrain plat. Nous faisons ainsi l'hypothèse que ses mouvements sont contenus dans un plan parallèle au sol. Nous nous plaçons dans le repère allocentrique $R = (0, \vec{x}, \vec{y}, \vec{z})$, repère fixe indépendant du robot, dans lequel l'axe z est perpendiculaire au plan d'évolution du véhicule robotisé. Nous définissons l'état du robot sur l'intervalle de temps $[t_0, t_0 + T_p]$ par le vecteur X:

$$X[t_0, t_0 + T_p] = \begin{pmatrix} x \\ y \\ \theta \\ \dot{s} \end{pmatrix} [t_0, t_0 + T_p] \qquad \text{Équation 22}$$

Dans ce vecteur d'état, x et y désignent respectivement l'abscisse et l'ordonnée du point de contrôle C du robot. θ désigne l'orientation du robot dans le plan $(0, \vec{x}, \vec{y})$ et $\dot{s}$ représente sa vitesse curviligne dans ce même plan (Figure 48).

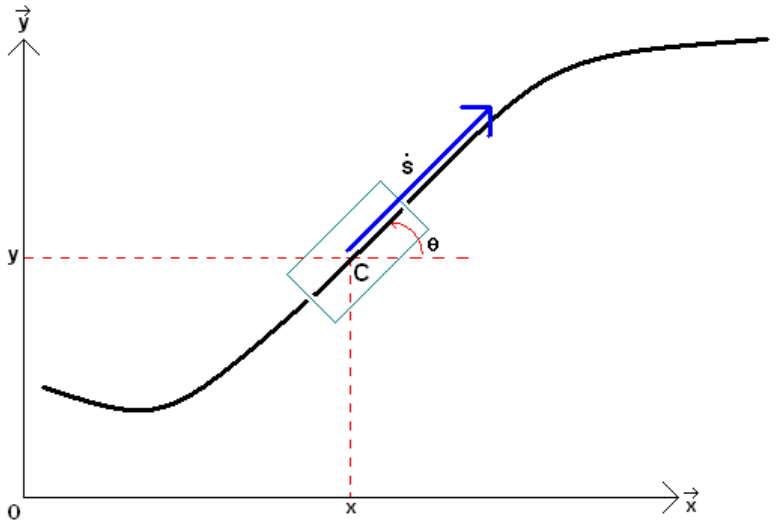

Figure 48- Modélisation du robot mobile

Nous définissons ensuite le vecteur de commande u, également sur l'intervalle temporel $[t_0, t_0 + T_p]$. C'est un vecteur de m lignes qui contient les m commandes articulaires qui sont appliquées au robot:

$$u[t_0, t_0 + T_p] = \begin{pmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dots \\ \dot{q}_m \end{pmatrix} [t_0, t_0 + T_p] \qquad \text{Équation 23}$$

En robotique mobile, le modèle cinématique du robot permet de déterminer les vitesses cartésiennes $\dot{x}$, $\dot{y}$ et $\dot{\theta}$ en fonction de ses m commandes articulaires $\dot{q}_1$ à $\dot{q}_m$. La matrice permettant de faire ce calcul est la matrice jacobienne, notée le plus souvent J. Le modèle cinématique nous permet donc de déterminer les déplacements du robot en fonction de la commande appliquée au véhicule:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = J. \begin{bmatrix} q_1 \\ \dots \\ q_m \end{bmatrix} \qquad \text{Équation 24}$$

Nous pouvons grâce à ce modèle déterminer l'ensemble de l'état X du robot à partir des commandes appliquées et de l'état initial du robot. En effet l'ensemble du vecteur d'état X peut être aisément déterminé à partir des vitesses cartésiennes si l'on connaît l'état initial du robot :

$$\begin{cases} x(t) = x_0 + \int_{t_0}^{t_0+T_p} \dot{x}.dt \\ y(t) = y_0 + \int_{t_0}^{t_0+T_p} \dot{y}.dt \\ \theta(t) = \theta_0 + \int_{t_0}^{t_0+T_p} \dot{\theta}.dt \\ \dot{s}(t) = \sqrt{(\dot{x}^2 + \dot{y}^2)} \end{cases} \qquad \text{Équation 25}$$

Nous déterminons ainsi un modèle global de l'évolution de l'état de notre robot en fonction du temps, des commandes appliquées sur ce temps et de l'état initial du robot. Ce modèle constitue l'élément clé de notre navigateur basé sur la commande prédictive, puisque c'est lui qui nous permet d'anticiper les mouvements du robot et ainsi prévoir la commande qui sera la plus adaptée à notre système.

La formalisation de l'état notre robot mobile dans son espace d'évolution et dans le temps nous conduit à modéliser le problème de navigation du robot dans son environnement que nous considérons comme partiellement connu, c'est-à-dire que le robot est capable de modéliser en temps réel son environnement proche et d'y localiser des obstacles éventuels.

2. Le problème de navigation

Dans le chapitre II, nous avons vu que le problème de navigation en robotique mobile revenait à déterminer une trajectoire réalisable par le robot, qui ne le fasse pas entrer en collision avec les obstacles, et qui lui permette de suivre au mieux un chemin fourni par un planificateur. Cela implique de faire un choix dans les trajectoires que le robot est capable d'effectuer, et donc de déterminer un critère pour faire ce choix.

Une trajectoire est réalisable par le robot si elle respecte ses capacités de mouvement. Les contraintes de non holonomie sont contenues dans le modèle cinématique du véhicule. Ce modèle étant utilisé pour prévoir les trajectoires que le robot peut effectuer sur un horizon temporel donné, il en résulte que les contraintes de non holonomie sont implicites dans la formulation du problème. Par contre, les contraintes cinématiques telles que les saturations des actionneurs ou les butées articulaires, doivent être clairement formulées pour que la trajectoire soit bien réalisable par le robot. Ce type de contrainte jouera sur les fonctions d'entrée de notre robot, sur sa commande u. L'ensemble des trajectoires qui vérifient ces contraintes est appelé ensemble des trajectoires admissibles.

Il faut également s'assurer que la trajectoire du robot ne heurte pas les obstacles. Cette condition peut se mettre sous la forme d'une contrainte liée à l'environnement, qui joue sur l'état du robot X. La trajectoire du robot étant calculée par rapport à son point de contrôle, il est nécessaire de tenir compte de la taille du robot dans la formulation de ce type de contrainte, c'est-à-dire en première approche la plus grande distance entre le point de contrôle et la périphérie du véhicule, quand on se projette dans le plan de mouvement. Généralement cette contrainte revient à vérifier que la distance entre le point de contrôle du véhicule et les obstacles reste supérieure à une distance égale à la taille du robot plus une marge de sécurité. Cette marge de sécurité tient compte des incertitudes sur la localisation des obstacles et du robot.

Enfin, pour pouvoir choisir la meilleure trajectoire pour suivre le chemin fourni par un planificateur, il faut établir un critère de choix. Le planificateur sera plus ou moins évolué suivant le robot étudié, mais dans tous les cas il fournira au minimum plusieurs points de passage consécutifs que le robot devra traverser pour atteindre son but final. A partir des informations fournies par le planificateur, une courbe de rattrapage, permettant de relier la position actuelle du robot et la position désirée, peut être déterminée. Cette courbe est une courbe purement géométrique, qui n'est pas reliée au temps (pas de vitesses le long du parcours), mais qui peut tenir compte de certaines contraintes géométriques liées au robot (taille, rayon de braquage). L'idée de base pour relier ces points de passage était d'utiliser des lignes droites et des arcs de cercle, mais cela introduit des discontinuités. Typiquement des courbes polynomiales sont utilisées pour obtenir une continuité, comme les courbes de Bézier ou les beta-splines.

A partir de la courbe de rattrapage, nous pouvons déterminer une trajectoire de référence pour le robot, en associant une vitesse de référence à la courbe de rattrapage (Figure 49). En multipliant cette vitesse de référence par l'horizon de prédiction du système, nous obtenons une distance de référence qui représente la distance que le robot peut raisonnablement parcourir le long de la courbe de rattrapage. Le choix de la valeur de la vitesse de référence doit être fonction du type de l'encombrement du terrain dans lequel évolue le robot. Si le planificateur fournit une vitesse indicative au point de passage, cette vitesse peut être utilisée comme vitesse de référence. En terrain dégagé, la vitesse maximale du robot peut être employée. Dans tous les cas, cette vitesse doit correspondre à une vitesse de déplacement réaliste pour le robot.

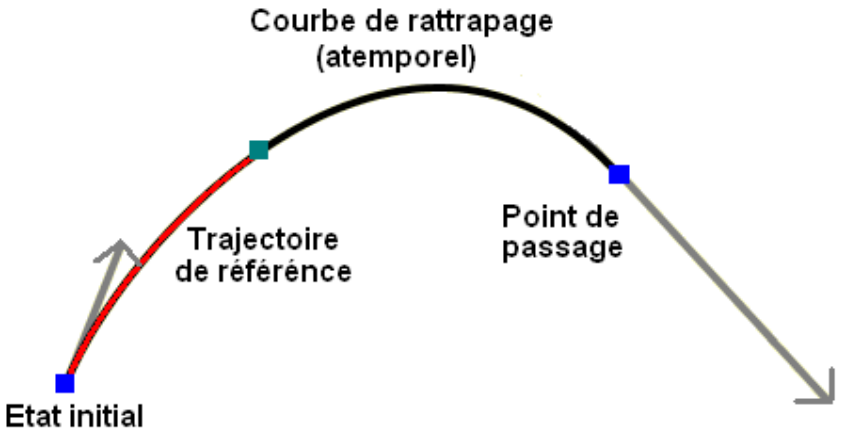

Figure 49- Courbe de rattrapage et trajectoire de référence

L'écart entre la trajectoire prévue du robot suivant la commande appliquée, et la courbe de rattrapage, est quantifié par l'intermédiaire d'une fonction de coût, Z. Ainsi, tout le problème de navigation peut s'écrire sous la forme d'un problème d'optimisation sous contraintes :

$$\begin{cases} \min Z(X_0, t_0, T_p, u) \quad sous\ contr \quad C(X_0, t_0, T_P, u) \le 0 \quad \forall t \in [t_0, t_0 + T_p] \\ u \in [t_0, t_0 + T_p] \end{cases}$$

Équation 26

les contraintes correspondent d'une part aux obstacles dans l'environnement du robot, et d'autre part aux conditions d'admissibilité des commandes pour le robot.

La première partie de la fonction u solution de ce problème d'optimisation, sur la période d'échantillonnage $[t_0, t_0 + T_e]$, est appliquée au robot. Et le problème complet d'optimisation est de nouveau résolu pour la période suivante $[t_0 + T_e, t_0 + T_e + T_p]$, avec intégration des nouvelles informations sur la positon des obstacles et l'état réel du robot.

L'un des problèmes de base est celui du temps réel, qui impose un temps d'exécution des calculs suffisamment court pour que le système puisse réagir à temps. Cela n'est pas un problème intrinsèque à la robotique, la commande prédictive a été étendue à des systèmes de plus en plus rapides avec l'amélioration des outils informatiques, mais il est particulièrement important dans les applications de robotique mobile.

Plus l'horizon de prédiction T_p est important, plus le modèle du robot est capable de prévoir les effets à long terme d'une commande donnée, et ainsi tendre plus rapidement vers le comportement souhaité. Cependant le nombre de calculs augmente linéairement avec l'augmentation de T_p.

Maintenant que nous avons vu comment transformer notre problème de navigation sous forme de problème d'optimisation, nous allons étudier plus en détail les différents éléments qui composent ce problème, à savoir la fonction de coût J, les fonctions de commande u, et les contraintes C, et ainsi aboutir à une formalisation du système complet.

III. La fonction de coût

1. Formulation générale

La fonction de coût est l'élément central du problème d'optimisation. C'est elle qui va faire le lien entre le problème de navigation (le problème réel, physique), et le problème mathématique d'optimisation qui va lui être associé. Cette fonction a pour but de quantifier l'intérêt d'un mouvement donné du robot, dans le cadre du problème de navigation que nous venons de décrire. Pour cela, la fonction coût associe à chaque fonction de commande u sur $[t_0,t_0+T_p]$ une valeur $Z(X_0, t_0, T_p, u)$. Elle s'écrit généralement sous la forme :

$$Z(X_0,t_0,T_p,u) = \psi(x(t_0+T_p)) + \int_{t_0}^{t_0+T_p} l(X(\tau),u(\tau),\tau)d\tau \qquad \text{Équation 27}$$

Elle se compose d'une fonction quantifiant le coût de parcours l et d'une fonction quantifiant le coût terminal ψ. Le coût de parcours évalue l'écart entre chaque point de la trajectoire prévue du robot, et le point de la trajectoire de référence qui lui correspond. Le coût terminal quantifie l'écart entre la l'état prévu du robot en t_0+T_p et le point final de la trajectoire de référence, et a un poids qui lui est propre. Généralement des fonctions de type erreur quadratique sont privilégiées, pour leurs propriétés de convexité qui sont utiles aux algorithmes d'optimisation comme nous le verrons par la suite.

Plus le comportement du système se rapproche du comportement désiré (trajectoire de référence), plus la fonction Z doit être faible. Ainsi le comportement optimal se situe là où la fonction Z est minimale (minimum global). Cependant, si la fonction Z n'est pas convexe, on peut trouver des minima locaux. Il est donc préférable d'avoir une fonction coût qui soit convexe, mais cela n'est pas toujours (rarement) possible en réalité.

L'obtention des informations nécessaires au calcul de la trajectoire de référence découle du fonctionnement de l'architecture de commande du robot étudié. En effet le navigateur du robot peut aussi bien se voir fournir un chemin détaillé du parcours qu'il doit effectuer, avec des vitesses et des orientations de référence, qu'un simple point à atteindre et éloigné de sa position actuelle. A partir des informations disponibles, une courbe de rattrapage est calculée de manière à satisfaire à ces informations.

2. Trajectoire de référence

Comme nous venons de le voir, pour déterminer une fonction coût, il est nécessaire de déterminer au préalable une trajectoire de référence. Ce dont on dispose dans notre problème de navigation, c'est d'un chemin fourni par un planificateur. Supposons ici que ce chemin est constitué d'un certain nombre de points de passages successifs à atteindre. Entre la position courante du robot et le prochain point de passage à atteindre, nous pouvons déterminer ce que l'on appelle une courbe de rattrapage. Une méthode pour tracer une telle courbe est d'utiliser les équations de Bézier, définie par 4 points de contrôles P1 à P4 [42]:

$$\begin{cases} x_r(l) = (1-l)^3 x_1 + 3l(1-l)^2 x_2 + 3l^2(1-l)x_3 + l^3 x_4 \\ y_r(l) = (1-l)^3 y_1 + 3l(1-l)^2 y_2 + 3l^2(1-l)y_3 + l^3 y_4 \end{cases} \text{ avec } 0 < l < 1 \qquad \text{Équation 28}$$

Cette courbe passe par les points P1(x1, y1) et P4(x4, y4), et les vecteurs $\overrightarrow{P1P2}$ et $\overrightarrow{P3P4}$ sont tangents à la courbe en P1 et P4 respectivement (Figure 50).

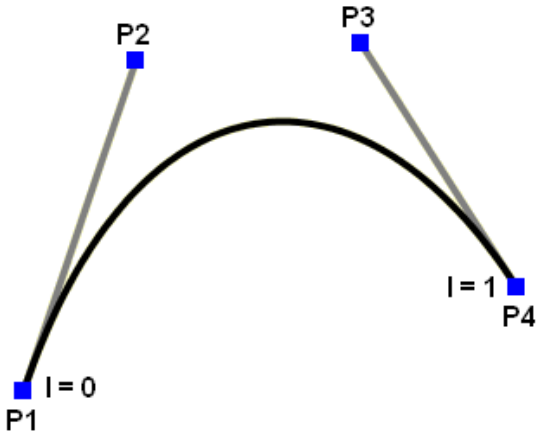

Figure 50- Courbe de Bezier

Nous supposons que le planificateur nous donne un point de passage à atteindre avec en plus une orientation et une vitesse. Nous appelons cet état à atteindre X_d. Comme nous voulons que la courbe de rattrapage fasse le lien entre l'état courant X_0 du robot et l'était désiré X_d et respecte les vitesses et orientations en ces points, nous posons :

$$P1:\begin{cases} x_1 = x_0 \\ y_1 = y_0 \end{cases},\ P2:\begin{cases} x_2 = x_1 + \frac{1}{3}.\dot{s}_0.\cos(\theta_0) \\ y_2 = y_1 + \frac{1}{3}.\dot{s}_0.\sin(\theta_0) \end{cases},\ P3:\begin{cases} x_3 = x_4 - \frac{1}{3}.\dot{s}_d.\cos(\theta_d) \\ y_3 = y_4 - \frac{1}{3}.\dot{s}_d.\sin(\theta_d) \end{cases} \text{ et}$$

$$P4:\begin{cases} x_4 = x_d \\ y_4 = y_d \end{cases} \qquad \text{Équation 29}$$

Cette courbe est purement géométrique, et ne contient aucune information concernant la cinématique du véhicule. Cette courbe de rattrapage ne peut donc pas jouer directement le rôle de trajectoire de référence puisqu'elle n'est pas liée au temps. Cependant, si nous supposons une distance de déplacement que le robot est capable de parcourir le long de cette courbe pendant l'horizon temporel T_p, nous pouvons alors calculer une trajectoire de référence le long de cette courbe de rattrapage, en effectuant une paramétrization dans le temps. Nous notons D la longueur de la courbe de Bezier. Si on considère que le robot est capable de parcourir $v\max \times T_p$ pendant le temps T_p, alors deux cas de figure se présentent :

1er cas : $v_{\max} \times T_p \le D$; physiquement cela signifie que le robot n'est pas capable de parcourir la totalité de la longueur de la courbe de rattrapage pendant T_p. Dans ce cas, la trajectoire de référence ne sera calquée que sur une partie réduite de la courbe de rattrapage (Figure 51). L'équation de notre trajectoire de référence, qui dépend du temps t, sera :

$$\begin{cases} x_r(t') = (1-t')^3 x_1 + 3t'(1-t')^2 x_2 + 3t'^2 (1-t') x_3 + t'^3 x_4 \\ y_r(t') = (1-t')^3 y_1 + 3t'(1-t')^2 y_2 + 3t'^2 (1-t') y_3 + t'^3 y_4 \end{cases} \text{ avec } t' = \frac{v_{\max}}{D}.t \text{ et}$$

$$0 < t < T_p \qquad \text{Équation 30}$$

L'équation complète de la trajectoire de référence sera alors :

$$X_{ref}(t) = \begin{pmatrix} x_{ref}(t) \\ y_{ref}(t) \\ \theta_{ref}(t) \\ s_{ref}(t) \end{pmatrix} = \begin{pmatrix} x_r(\frac{\dot{s}_d}{D}.t) \\ y_r(\frac{\dot{s}_d}{D}.t) \\ \arctan_2(\dot{x}_r, \dot{y}_r) \\ v_{\max} \end{pmatrix} \text{ avec } t \in [0, T_p] \text{ et } \frac{\dot{s}_d}{D} t \leq 1 \qquad \text{Équation 31}$$

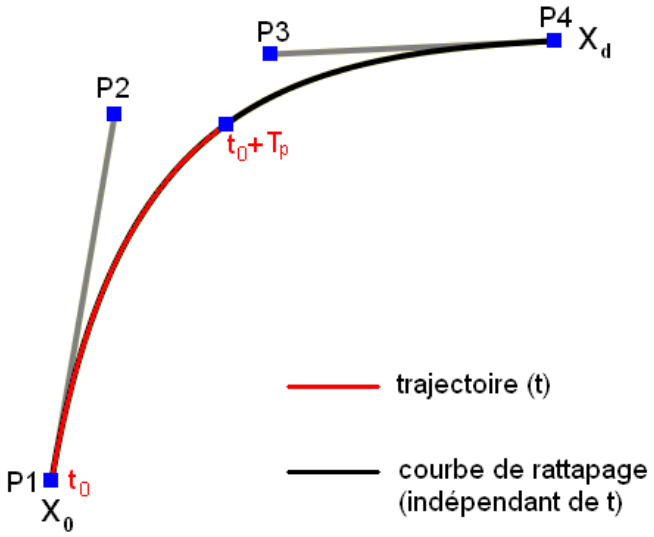

Figure 51- 1er cas : courbe de rattrapage trop longue

2e cas : $v_{\max} \times T_p > D$; ce cas correspond à la situation où le robot est assez proche du prochain point à atteindre pour être capable d'y parvenir sous l'horizon temporel T_p. Dans ce cas l'équation de courbe paramétrée Y restera valable pour $0 < t < \frac{D}{v_{\max}}$, et pour $t \geq \frac{D}{v_{\max}}$ de nouveau 2 cas sont possibles. Soit le point X_d est le point final du parcours du robot, le planificateur ne nous a pas fourni d'autres points à atteindre par la suite. Dans ce cas, le robot doit s'arrêter une fois ce point atteint, et donc l'équation de la trajectoire de référence sera :

$$X_{ref}(t) = \begin{pmatrix} x_{ref}(t) \\ y_{ref}(t) \\ \theta_{ref}(t) \\ s_{ref}(t) \end{pmatrix} = \begin{pmatrix} x_d \\ y_d \\ \theta_d \\ 0 \end{pmatrix} \text{ avec } t \in [\frac{D}{v_{\max}}, T_p] \qquad \text{Équation 32}$$

L'autre possibilité est que le point X_d n'est qu'un point intermédiaire avant un prochain point $X_{d'}$ fourni par le planificateur (Figure 52). Dans ce cas, il faut prolonger la courbe de rattrapage entre X_d et $X_{d'}$, cette fois ci en posant :

$$P1':\begin{cases} x_{1'} = x_d \\ y_{1'} = y_d \end{cases},\ P2':\begin{cases} x_{2'} = x_{1'} + \frac{1}{3}.\dot{s}_d.\cos(\theta_d) \\ y_{2'} = y_{1'} + \frac{1}{3}.\dot{s}_d.\sin(\theta_d) \end{cases},\ P3':\begin{cases} x_{3'} = x_{4'} - \frac{1}{3}.\dot{s}_{d'}.\cos(\theta_{d'}) \\ y_{3'} = y_{4'} - \frac{1}{3}.\dot{s}_{d'}.\sin(\theta_{d'}) \end{cases} \text{ et}$$

$$P4':\begin{cases} x_{4'} = x_{d'} \\ y_{4'} = y_{d'} \end{cases}$$

Équation 33

La continuité entre les deux morceaux de trajectoire de référence, entre X_0 et X_d d'une part, et entre X_d et $X_{d'}$ d'autre part, est vérifiée grâce aux conditions aux limites imposées en X_d sur chacune des deux courbes. L'équation de la trajectoire pour $t \geq \frac{D}{v_{max}}$ devient alors :

$$X_{ref}(t) = \begin{pmatrix} x_{ref}(t) \\ y_{ref}(t) \\ \theta_{ref}(t) \\ s_{ref}(t) \end{pmatrix} = \begin{pmatrix} x_{r'}(\frac{\dot{s}_d}{D}.t) \\ y_{r'}(\frac{\dot{s}_d}{D}.t) \\ \arctan_2(\dot{x}_{r'}, \dot{y}_{r'}) \\ v_{max} \end{pmatrix} \text{ avec } t \in [\frac{D}{v_{max}}, T_p]$$

Équation 34

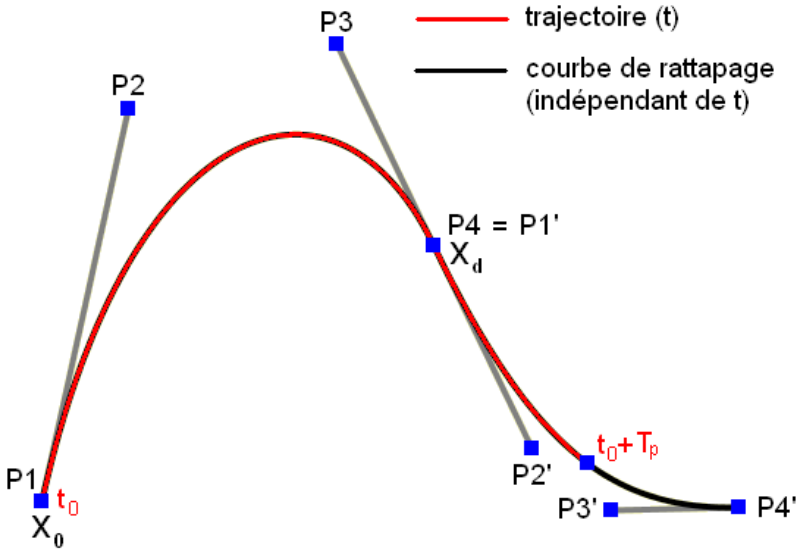

Figure 52- Second cas : courbe de rattrapage trop courte

A partir des informations sur l'état courant du robot et l'état désiré, nous avons déterminé une courbe de rattrapage qui est une courbe purement géométrique et indépendante du temps. En prenant en compte la vitesse maximale du robot, et donc la distance maximale qu'il est capable de parcourir pendant l'horizon de temps T_p, nous avons paramétré cette courbe dans le temps, permettant ainsi d'obtenir une trajectoire de référence.

Il est à noter qu'à aucun moment la notion d'obstacle n'intervient ici. Cette trajectoire de référence est donc à la fois décorrélée des contraintes locales de l'environnement du robot, mais aussi des contraintes de mouvement du robot. Toutes ces contraintes seront traitées spécifiquement par la suite. Cette non prise en compte des contraintes dans la détermination de la trajectoire de

référence est un point intrinsèque à la commande prédictive, et qui peut être discutable. Nous y reviendrons à la fin de cette partie..

3. Discussion

Pour passer d'une courbe de rattrapage atemporelle, à une trajectoire de référence (qui dépend donc du temps), nous avons effectué une paramétrisation en prenant la distance que le robot est capable de parcourir pendant T_p comme référence. Cette distance a été calculée par rapport à la vitesse maximale autorisée du robot. Si ce choix s'avère compréhensible dans le cadre d'une navigation dans un environnement bien connu et relativement dégagé, ou avec quelques obstacles à contourner, il peut être inadapté si on considère un environnement très chargé ou très peu connu (phases d'exploration initiales). L'adaptation de cette vitesse de référence au type de mission et/ou type de terrain pourrait s'avérer un choix judicieux de manière à obtenir des trajectoires de référence plus adaptées.

L'adaptation de cette vitesse de référence pourrait aussi être effectuée grâce à des informations fournies par le planificateur de mission. Lorsqu'un conducteur d'automobile planifie son chemin pour se rendre entre deux villes, il est capable d'évaluer le temps qu'il mettra pour effectuer son parcours, suivant le type de route empruntée (autoroute, nationale…) et donc la vitesse moyenne sur le parcours. Par analogie, on pourrait très bien envisager que le planificateur de mission, en plus de fournir des points de passages au navigateur, lui fournisse également une vitesse indicative, dépendant de l'encombrement du parcours, de l'état du terrain etc… et que le robot respectera ou pas suivant les contraintes locales qu'il rencontre.

IV. Commande et familles de trajectoires

Comme nous venons de le voir, le problème de navigation du robot mobile peut se mettre sous la forme d'un problème d'optimisation sous contraintes dans lequel nous cherchons à minimiser une fonction coût Z. La "variable" sur laquelle on joue pour minimiser ce critère est la commande en boucle ouverte appliquée au système u sur $[t_0,t_0+T_p]$. On suppose que le système robotique étudié possède m entrées correspondant aux m degrés de mobilité du véhicule. La commande u est totalement définie sur $\Re^m * \left[t_0 ; t_0 + T_p\right]$ par :

$$u = \left\{u(t)\right\}_{t\in\left[t_0;t_0+T_p\right]} \qquad \text{Équation 35}$$

Et la j^{eme} composante de u vaut :

$$u_j = \left\{u_j(t)\right\}_{t\in\left[t_0;t_0+T_p\right]} \text{ avec } j \in \{1,...,m\} \qquad \text{Équation 36}$$

Pour respecter les contraintes sur les entrées, cette fonction u doit appartenir à l'ensemble $\Omega_{[t0,\ t0+\tau]}$ des commandes admissibles pour le robot. Comme nous l'avons vu au chapitre III, cela signifie que cette fonction doit respecter un certain nombre de contraintes sur la cinématique et la dynamique des actionneurs, notamment les butées et saturations. Cette commande peut également respecter les contraintes dynamiques du robot comme par exemple ne pas le renverser.

L'espace des solutions u admissibles est de dimension infinie, ce qui rend la recherche de solution optimale très complexe. Pour pouvoir résoudre ce problème, une contrainte de conception est ajoutée au problème : cette contrainte est que la fonction u solution doit être issue d'un e

fonction paramétrée, avec un nombre fini de paramètres. Trouver la fonction de commande u(t) optimale reviendra alors à déterminer le jeu de paramètres optimaux.

1. L'approche classique : fonctions constantes par morceaux

Habituellement, les fonctions de commande u utilisées en automatique sont des fonctions constantes par morceaux. Ces fonctions définies sur l'intervalle $[t_0, t_0+T_p]$ sont constituées de N_p « morceaux » constants de largeur T_e tels que :

$$T_e = \frac{T_p}{N_p} = \frac{1}{f_e}$$ Équation 37

Sur ces N_p intervalles de prédiction, seuls les N_c premiers intervalles sont commandés, et les N_p - N_c intervalles restants sont fixes et prennent la même valeur que sur le dernier intervalle commandé (Figure 53).

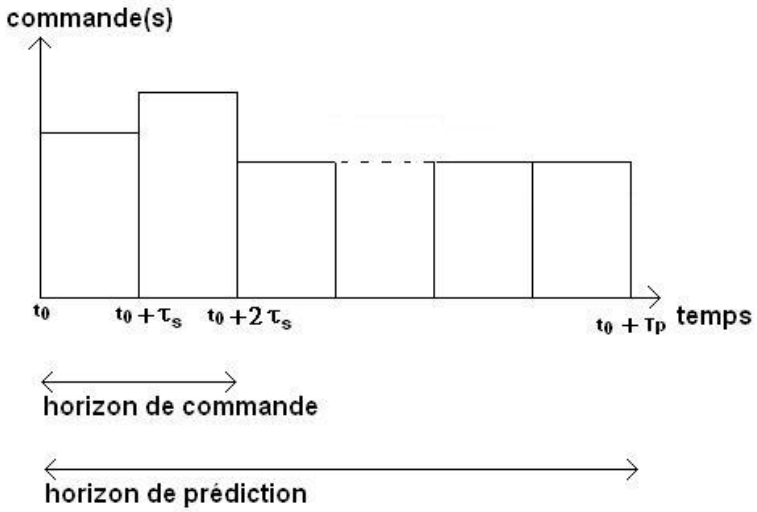

Figure 53- Discrétisation classique de l'espace des temps

Par analogie, on appelle T_c l'horizon de commande. Dans ce cas, le nombre de paramètres n_p à trouver dans notre problème d'optimisation est donc :

$$n_p = N_c.m$$ Équation 38

Avec m le nombre de commandes du système considéré. Le problème d'optimisation revient à trouver les n_p paramètres permettant de minimiser la fonction coût définie précédemment. Dans la pratique, l'horizon de commande N_c est généralement limité à 1 ou quelques intervalles, de manière à limiter le nombre de calcul pour l'optimisation ; ce nombre de calcul augmentant de manière exponentielle avec N_c. Cela n'empêche pas d'obtenir de très bons résultats pour les problèmes de suivi de trajectoire classiques. Mais pour une application en navigation de robot mobile, avec les fortes contraintes dues aux obstacles et à la non holonomie du robot qui lui empêchent de suivre facilement une trajectoire de référence, se pose la nécessité de travailler sur des commandes permettant des trajectoires plus complexes pour contourner des obstacles ou effectuer des manœuvres.

2. Notre approche : familles de trajectoires

A chaque type de véhicule correspond une manière de le piloter. Par exemple, le pilotage d'une voiture diffère de celui d'une moto. Cela se traduit en termes robotiques par des profils de commande spécifiques. Dans notre approche navigation pour la robotique mobile, plutôt que d'utiliser des commandes constantes par morceaux, et de se limiter au final à une détermination des quelques premiers morceaux, nous proposons de définir des familles de trajectoires paramétrées. Quelques exemples de fonction sont représentés sur la Figure 54. Ces familles de trajectoire vont correspondre à des familles de fonctions de commande u générant des mouvements du robot tels que des manœuvres de contournement d'obstacle ou des créneaux.

Le nombre de paramètres qui définissent une famille de trajectoire influe sur la richesse des mouvements que le robot sera capable d'anticiper. Par exemple, pour pouvoir effectuer des manœuvres de contournement d'obstacle (braquage dans un sens puis dans l'autre), avec un robot mobile de type voiture, au moins deux paramètres p_1 et p_2 sont nécessaires. Ne pas inclure au moins deux paramètres dans la composante du braquage de la fonction de commande u ne signifie pas que le robot ne sera pas capable d'effectuer un contournement d'obstacle au final (vu que la commande est recalculée à la période d'échantillonnage T_e) ; mais cela signifie que le robot ne pourra pas anticiper qu'il est capable de contourner l'obstacle. Et donc à l'instant t_0 où il tente de résoudre son problème de contournement d'obstacle, il se prive d'un certain nombre de solutions intéressantes.

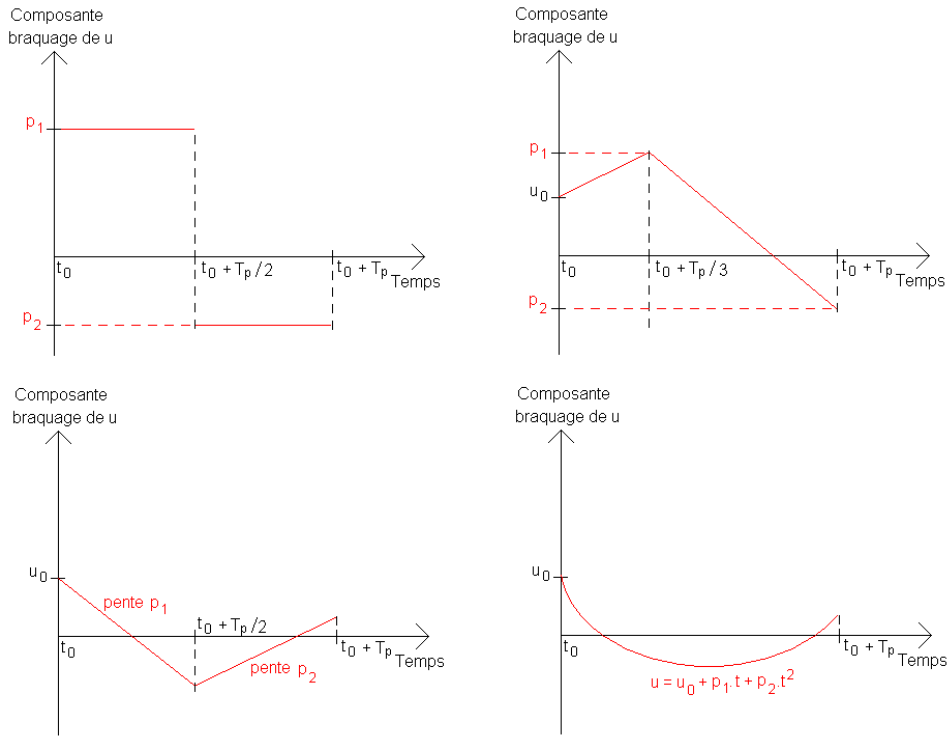

Figure 54- Exemples de familles de fonction u sur le braquage permettant d'effectuer un contournement d'obstacle

Par l'intermédiaire du modèle cinématique direct du robot, chaque famille de fonction de commande u va générer une famille de trajectoire spécifique. Par abus de langage, ce sont ces couples fonctions de commande / trajectoires que l'on appelle familles de trajectoires. Le choix d'une famille de trajectoire plutôt qu'une autre dépend d'un certain nombre de facteurs :

- la mobilité du robot : la cinématique du robot et notamment son nombre de degrés de mobilité par rapport au nombre de degrés de liberté est un facteur très important dans la détermination du nombre de paramètres caractérisant la famille de trajectoires. Dans le cas de véhicules évoluant sur terrain plat, nous nous retrouvons très souvent dans le cas de non holonomie classique avec 3 degrés de liberté pour 2 degrés de mobilité ce qui a pour conséquence d'empêcher le robot d'atteindre « en 1 coup » une position orientation donnée. Dans ce cas on préconisera

d'utiliser des familles de trajectoires avec au moins deux paramètres pour chaque composante de la commande. La Figure 55 montre l'exemple de la voiture qui doit se garer sur une place de parking permet d'illustrer cette constatation.

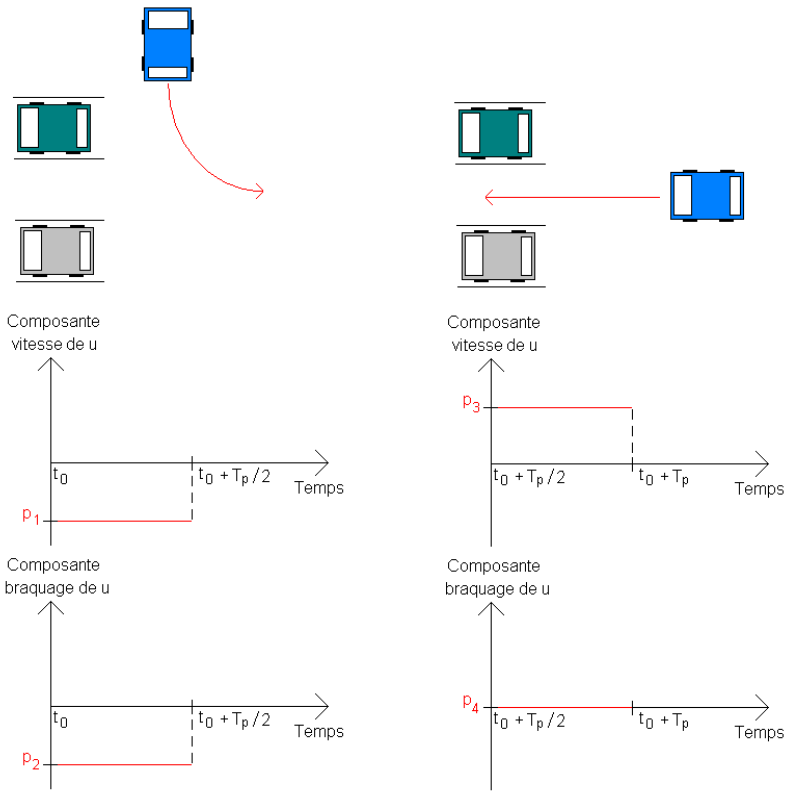

Figure 55- Nombre de paramètres nécessaires pour permettre à une voiture d'effectuer un mouvement pour se garer

- la dynamique du robot : la fonction de commande doit tenir compte de l'inertie du véhicule. Les trajectoires générées doivent être réalisables par le robot, il est donc préférable d'utiliser au maximum des fonctions continues. Les limites sur les butées articulaires, vitesses et accélérations maximales seront explicitement intégrées dans les contraintes du problème d'optimisation, comme nous le verrons par la suite.

Les n_p paramètres définissant notre famille de trajectoires seront les n_p variables à optimiser pour minimiser notre fonction coût. Plus ce nombre de paramètres sera important, plus les mouvements que le robot sera capable de prévoir pour poursuivre sa trajectoire de référence et atteindre son objectif seront riches. En contrepartie, plus ce nombre de paramètres sera important et plus le nombre de calculs pour l'optimisation du problème le seront également.

En projetant une famille de commandes paramétrées à travers le modèle cinématique du robot, nous obtenons une infinité de trajectoires possibles, qui va correspondre à une famille de trajectoires (Figure 56). Ainsi nous sortons du formalisme discret inhérent aux méthodes de projection de trajectoires habituelles.

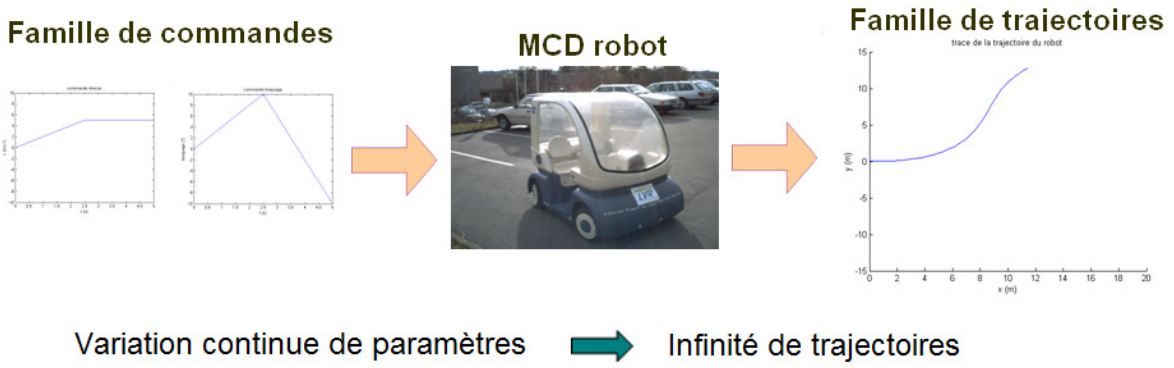

Figure 56 - Génération d'une famille de trajectoires en effectuant une variation continue des paramètres de la famille de commandes

V. Les contraintes

1. Notation générale

Dans la théorie de la commande prédictive, il existe deux types de contraintes bien distinctes mais qui se traitent de la même manière : les contraintes liées au problème et celles dites de conception.

Les contraintes liées au problème font partie de la définition même du problème et sont incontournables. Pour la commande d'un moteur, elles correspondent par exemple aux contraintes de saturation des actionneurs. Elles font physiquement partie du problème à résoudre et doivent impérativement être prises en compte explicitement dans la modélisation.

Les contraintes de conception n'existent pas physiquement ; mais elles peuvent être ajoutées au problème d'optimisation pour lui donner certaines propriétés comme la stabilité ou la robustesse.

Toutes ces contraintes sont notées de la même manière :

$$C_i(X_0,t_0,t,u) \le 0 \quad \forall t \in T_i \quad i \in \{1,...,n_c\} \qquad \text{Équation 39}$$

Où n_c est le nombre de contraintes et $T_i \subset [t_0, t_0 + T_p]$ est l'ensemble des instants où l'inégalité $C_i(.) \le 0$ doit être vérifiée.

Pour tenir compte de ces contraintes dans le problème d'optimisation, on les ajoute à la fonction coût ce qui donne la fonction de coût total Z_{tot}:

- soit en définissant des fonctions $B(C_i)$ telles que B croisse quand C_i s'accroît et est négatif, et telles que B tende vers l'infini quand C_i tend vers 0 (méthode des pénalités internes) ; cela permet de conserver la continuité de la nouvelle fonction Z_{tot} ainsi obtenue en fonction de l'état X.

$$Z_{tot}(X_0,t_0,T_p,u,C) = Z(X_0,t_0,T_p,u) + \sum_{i=1}^{n_c} B(C_i) \qquad \text{Équation 40}$$

- soit en les incorporant à la fonction coût de manière discontinue (pénalité infinie quand on sort du domaine défini par les contraintes) ; on perd la continuité de la fonction coût ce qui pose des

problèmes pour démontrer la convergence et la stabilité de la commande, en contrepartie cela permet d'inclure facilement différents types de contraintes dans le problème.

2. Spécification robotique mobile

Les contraintes que nous retrouvons habituellement dans les problèmes de robotique mobile sont :

- des contraintes liées à la commande u : saturation des actionneurs, comme la vitesse maximale de rotation de roues motrices, l'accélération et la décélération maximales, l'angle de braquage limite de direction qui, lui, est plutôt lié à la structure mécanique du robot ;
- des contraintes liées à l'état X du robot : obstacles dans le plan de navigation du robot, vitesses maximales recommandées dans une zone donnée, angle d'approche du véhicule.

Nous pouvons noter que les contraintes de non holonomie sont implicites dans la modélisation du robot car elles sont contenues dans le modèle cinématique et n'ont donc pas à être inclues dans la modélisation des contraintes du problème.

Les contraintes de saturation des actionneurs peuvent être inclues par l'intermédiaire de fonctions B_i hyperboliques. Si on considère une contrainte de saturation sur la commande u_1:

$$C_i : u_1 < u_{\max}$$
$$C_i : u_1 - u_{\max} < 0$$
$$B_i = -\frac{1}{u_1 - u_{\max}}$$

Équation 41

C'est la méthode dite des pénalités internes : B_i s'accroit au fur et à mesure qu'u1 se rapproche d'u_{max}, et tend vers l'infini lorsqu'u1 tend vers u_{max} (Figure 57).

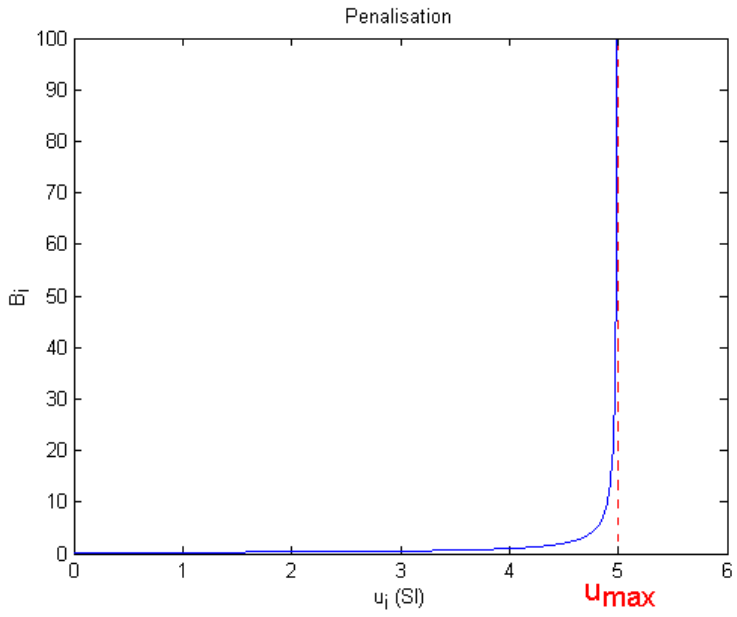

Figure 57 - Fonction de pénalisation des contraintes

Note : Il faudra cependant s'assurer, à l'initialisation de la résolution du problème, que ces contraintes sont bien respectées. En effet, il peut arriver par exemple que la limitation en

décélération du robot lui empêche de s'arrêter à l'itération de calcul suivante, entrainant une pénalité infinie à l'initialisation du problème. Ce genre de phénomène peut entrainer des problèmes de résolution sur certains algorithmes ; il est donc nécessaire de s'assurer de la faisabilité des commandes à l'initialisation de la résolution du problème.

Les obstacles eux, sont des contraintes définies dans un espace de dimension 2 ou plus (abscisse et ordonnée ou autre suivant la représentation choisie). De plus, suivant le système robotique étudié, le type de capteur extéroceptif utilisé, et l'analyse qui est faite des informations obtenues par ces capteurs, les obstacles peuvent être représentés de moult façons différentes (cartographie cartésienne, polaire, pas de cartographie proprement dite etc…).

Pour éviter qu'un robot n'entre en collision avec un obstacle, il est nécessaire de tenir compte de sa dimension. Plutôt que de vérifier que chaque point appartenant au robot n'entre pas en collision, on calcule une distance de sécurité qui tient compte de l'envergure du robot. Le test de collision consiste à vérifier que la distance entre l'obstacle et le point de contrôle du robot est bien supérieure à cette distance de sécurité. Cette distance de sécurité d_s est généralement prise égale à la distance maximale entre la périphérie et le point de contrôle du robot, à laquelle on rajoute une certaine marge de sécurité supplémentaire (pour compenser d'éventuelles erreurs de précision sur la localisation).

Nous présentons ici deux méthodes pour exprimer les contraintes liées aux obstacles ; une plutôt destinée aux cartographies de type segmentation des obstacles, et une pour les cartes de type grille d'occupation.

3. Cartes de type obstacles-segments

Une méthode pour introduire les obstacles consiste à calculer la distance euclidienne entre chaque obstacle et les points de la trajectoire du robot, nommée d. Ainsi, de la même manière que pour les contraintes de saturation, la méthode des pénalités internes peut être utilisée en posant :

$$d > d_s \qquad \text{Équation 42}$$

Dans certaines méthodes de cartographie, une segmentation des obstacles est effectuée. A partir des mesures issues des capteurs extéroceptifs, des segments représentatifs des obstacles sont extraits. L'accumulation de relevés, au cours des déplacements du robot, permet d'une part d'améliorer la robustesse des mesures et d'autre part de recaler la position du robot dans l'environnement. Ce type de méthode est courant, la Figure 58 est une carte obtenue par la méthode de segmentation de Canou implantée sur un robot mobile à roues différentielles de type unicycle [13].

Figure 58- Carte segmentée obtenue à partir de télémètres laser [13]

La carte des obstacles est dans ce cas caractérisée par un certain nombre de segments obstacles. La distance entre la trajectoire du robot et ces obstacles peut alors être déterminée en effectuant une projection orthogonale des points de la trajectoire du robot sur l'obstacle. Si le projeté orthogonal H est contenu dans le segment, alors la distance d entre le point P de la trajectoire et un segment obstacle AB est directement PH et vaut :

$$d = PH = \frac{\left|(x_B - x_A)(y_P - y_A) - (x_P - x_A)(y_B - y_A)\right|}{\sqrt{(xA - xB)^2 + (yA - yB)^2}}$$

Équation 43

Ce calcul revient simplement à diviser l'aire du triangle PAB par la base AB. Par contre si le projeté orthogonal H n'est pas compris dans le segment obstacle, alors la distance d est le minimum des deux distances entre le point de la trajectoire P et chacune des deux extrémités de l'obstacle (Figure 59). Un des soucis avec cette méthode est qu'elle est très gourmande en temps de calcul, puisque ce calcul devra être ré-effectué sur chacun des points de la trajectoire et par rapport à chaque obstacle.

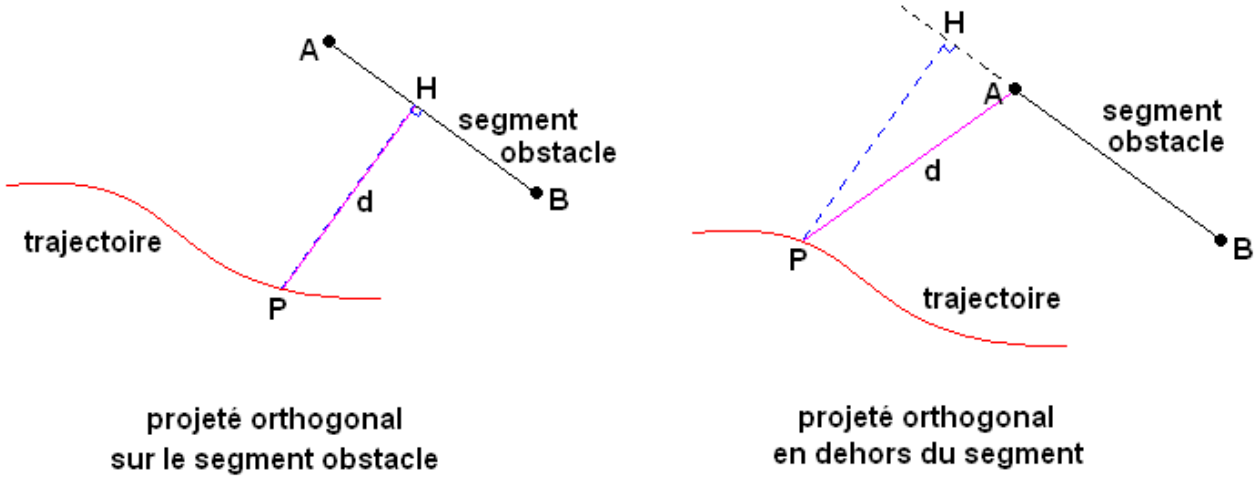

Figure 59- Distance entre un point de la trajectoire et l'obstacle

La fonction de coût Z_{tot} est alors pénalisée suivant une loi hyperbolique de la forme :

$$B_i = -\frac{1}{d - d_s} \qquad \text{Équation 44}$$

Et la fonction de cout total devient:

$$Z_{tot}(X_0, t_0, T_p, u, C_{cr}) = Z(X_0, t_0, T_p, u) + \sum^{n_r} B_r(C_{r,i}) + \sum^{n_o} B_o(C_{o,i}) \qquad \text{Équation 45}$$

4. Cartes de type grille d'occupation

Une autre manière d'introduire les obstacles sous forme de contraintes que nous proposons est plus adaptée à de représentations de type grille d'occupation (Figure 60). Il s'agit de traiter les contraintes sur les obstacles de manière discrète, en rajoutant une pénalité « infinie » à la fonction coût dans le cas où les contraintes sur les obstacles ne sont pas respectées. On perd la continuité de la fonction coût ce qui pose des problèmes pour démontrer certaines propriétés comme la convergence de notre système, mais en contrepartie nous améliorons la flexibilité d'adaptation à différents types de cartographie des obstacles.

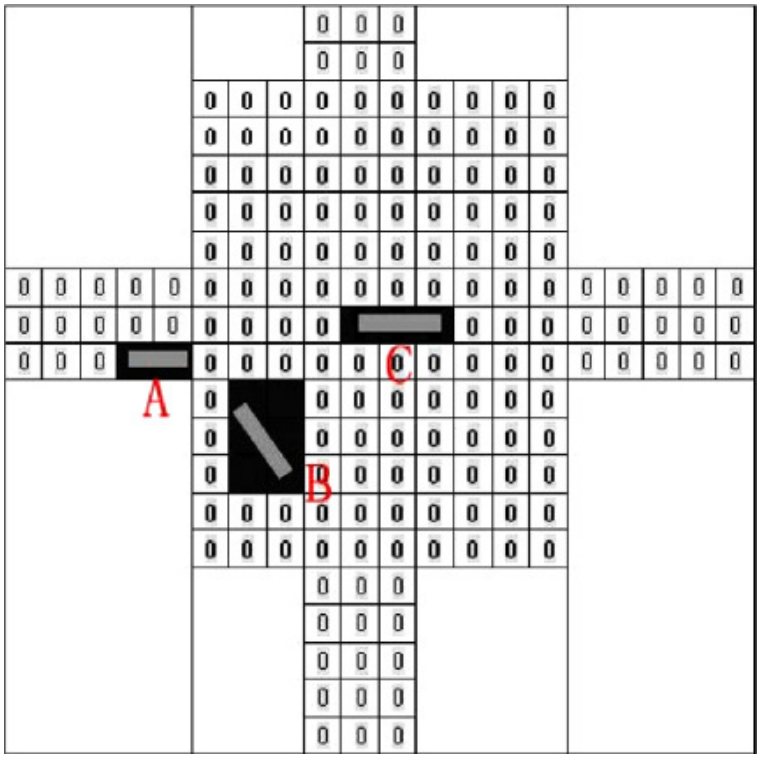

Figure 60- Grille d'occupation contenant 3 robots [66]

Le principe des grilles d'occupation est de discrétiser la carte locale sous la forme d'une grille, et d'attribuer à chaque case une valeur 1 si celle-ci est connue comme étant occupée par un obstacle, ou 0 dans le cas contraire. De manière à pouvoir assimiler la trajectoire du robot à la trajectoire d'un seul point (son point de contrôle), les obstacles sont élargis d'une distance égale à la demi-envergure du robot.

La famille de trajectoires générées est projetée dans cette grille. Une pénalité seuil Zseuil est ajouté dans le cas où les trajectoires du robot passent par des cases occupées. La valeur de est calculée suivant les paramètres du navigateur (taille de la grille, nombre de points considérés sur une trajectoire, paramètres de la fonction coût), de manière à éliminer les trajectoires correspondantes (les trajectoires non libres se voient associer un coût supérieur à n'importe quelle trajectoire libre).

$$Z_{tot}(X_0, t_0, T_p, u, C_{cr}) = Z(X_0, t_0, T_p, u) + \sum_{}^{n_r} B_r(C_{r,i}) + \begin{cases} 0 \\ Z_{seuil} \end{cases} \qquad \text{Équation 46}$$

VI. L'algorithme d'optimisation

A l'éclairage des sections précédentes, le problème de la commande prédictive revient à chercher les n_p paramètres de la fonction de commande u permettant de minimiser la fonction Z_{tot} définie sur $[t_0, t_0+T_p]$. Il existe un très grand nombre d'algorithmes développés pour résoudre les

problèmes d'optimisation**.** Mais il n'existe pas d'algorithme permettant de rechercher à coup sûr le minimum global (en s'assurant qu'il ne s'agit pas d'un minimum local) d'une fonction non convexe.

Pour l'application à la commande prédictive, ces algorithmes doivent être capables de fonctionner en temps réel (temps de calcul par rapport à la période d'échantillonnage T_e) et doivent donner la garantie que la solution donnée à n'importe quelle itération sera contenue dans le domaine des commandes admissibles pour le système (c'est-à-dire hors contrainte). Concrètement cela signifie que l'algorithme doit être capable de fournir une solution admissible à n'importe quel moment où on l'interrompt (contrainte inhérente au temps réel). Par contre, seule une solution approchée est demandée.

Les algorithmes d'optimisation sous contraintes peuvent être classés en deux catégories. Les algorithmes déterministes présentent généralement les meilleures propriétés de convergence, et permettent d'obtenir des solutions très précises très rapidement. Pour ces raisons, ils seront très efficaces pour trouver une trajectoire permettant de suivre au mieux la trajectoire de référence ou amener le robot sur une position précise. Par contre, ils sont très sensibles aux problèmes des minima locaux. Or l'introduction de contraintes discrètes dues aux obstacles a justement pour effet de casser la convexité de la fonction coût et introduire des minima locaux,.

La seconde catégorie d'algorithme est celle des algorithmes stochastiques. Ces algorithmes sont basés sur des approches probabilistes. Ils présentent l'inconvénient d'être moins précis et moins rapides à converger que les algorithmes déterministes (leur convergence n'est d'ailleurs pas toujours prouvée). Par contre, ils possèdent l'avantage d'être beaucoup moins sensibles aux problèmes de minima locaux, ce qui leur permet, dans le cadre de problèmes de navigation, de trouver des trajectoires de contournement d'obstacle qu'un algorithme de type déterministe ne saurait pas trouver.

Le choix du type d'algorithme est donc dépendant du type de terrain d'évolution (dégagé ou obstrué), et de la confiance que l'on a dans le planificateur concernant sa prise en compte des obstacles dans la génération de chemin. Le choix du type d'algorithme d'optimisation a fait l'objet d'une étude que nous présentons en annexe IV. Des résultats comparatifs entre un algorithme déterministe (Levenberg-Marquardt) et un algorithme stochastique (Recuit Simulé) sont également donnés dans le chapitre V. Pour l'application du navigateur en milieu encombré, et suite à ces études, nous avons utilisé l'algorithme recuit simulé de manière à tester la robustesse du navigateur à une planification insuffisante.

VII. Conclusion

Au cours de ce chapitre, nous avons développé un navigateur pour robot mobile, en reprenant le principe de la commande prédictive et en l'adaptant à notre problématique de navigation. En résumé, les différentes étapes réitérées à chaque cycle de notre méthode sont :

- 1- détermination d'une trajectoire de référence $X_{ref}(t)$, calculée à partir des informations fournies par le planificateur de mission, l'état courant X_0 du robot et ses contraintes cinématiques (vitesse maximale ou conseillée)
- 2- génération d'une trajectoire admissible par le robot, à partir d'une famille de fonctions d'entrée spécifique au type de robot et caractérisée par un jeu de n_p paramètres

- 3 - calcul d'une fonction de coût Z associée à cette trajectoire, caractérisant l'écart entre celle-ci et la trajectoire de référence, et intégrant des pénalités en fonction du respect des contraintes
- 4- optimisation de cette fonction Z en jouant sur les n_p paramètres, et retour à l'étape 2, à moins qu'une précision suffisante ait été atteinte

La routine des étapes 2-3-4 pouvant être interrompue à tout moment dès lors que le temps alloué au calcul de la trajectoire optimale est écoulé. La fonction de commande u associée, caractérisée par les n_p paramètres qui viennent d'être optimisés, est alors envoyée comme consigne au pilote. Rappelons que le pilote correspond au second niveau de notre architecture de contrôle, chargé d'appliquer les commandes en envoyant les consignes correspondantes aux actionneurs.

Ce processus complet est réitéré périodiquement toutes les T_e secondes. Le bouclage du Figure 61 présente la méthode sous forme de schéma de commande:

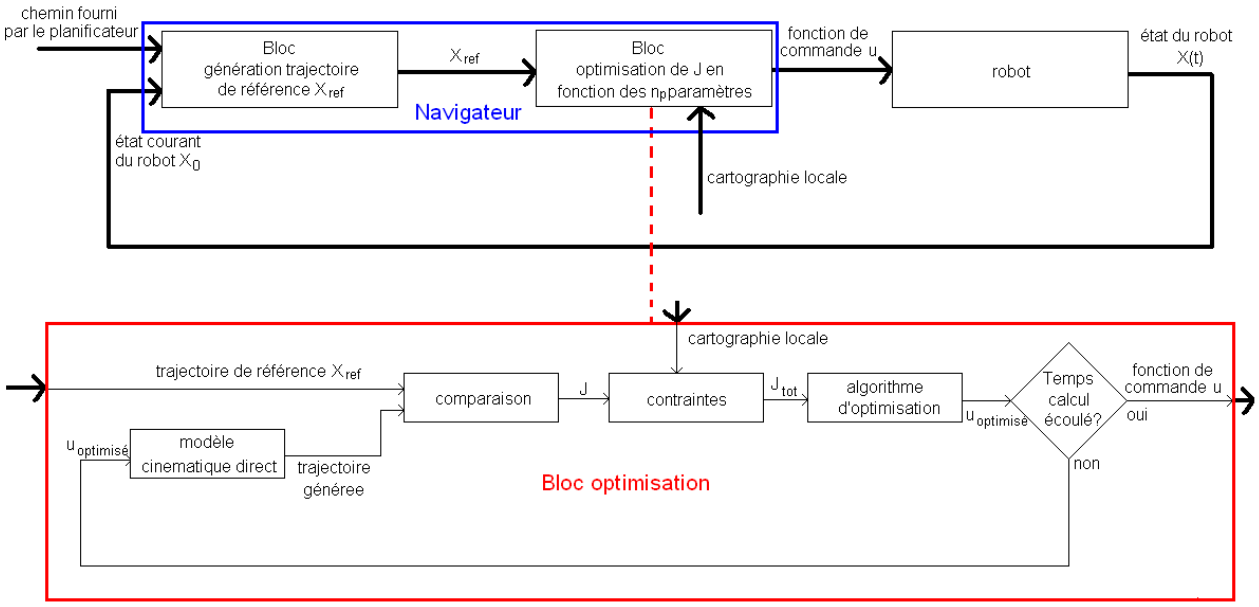

Figure 61- Représentation du principe de la commande

Cette méthode est essentiellement basée sur la connaissance du modèle cinématique direct du véhicule. L'optimisation de la fonction de coût Zest en effet réalisée en effectuant des projections successives de trajectoires réalisables par le robot, en faisant varier les n_p paramètres caractérisant la fonction de commande u, et en comparant des trajectoires ainsi générées avec la trajectoire de référence. Cette méthode de navigation se classe donc parmi les méthodes de projection par modèle direct.

Le navigateur que nous avons décrit au cours de ce chapitre traite du problème de navigation d'un robot mobile sur terrain plat. Le principe de cette méthode pourrait être étendu aux espaces d'évolution en 3D, en utilisant par exemple les courbes de Bézier 3D pour la détermination de la trajectoire de référence, et en définissant correctement les familles de trajectoires à utiliser pour donner assez de liberté au robot. La difficulté serait alors de trouver un algorithme d'optimisation capable de résoudre le problème posé de manière satisfaisante, en un temps suffisamment court pour être utilisable en ligne.

Les deux points importants sont d'une part écrire correctement le problème de navigation sous forme de problème d'optimisation (bien choisir le type de fonction de commande u, écrire correctement les contraintes en s'assurant que les contraintes dures soient bien respectées à

l'initialisation de l'algorithme d'optimisation), et d'autre part de trouver un algorithme capable de résoudre efficacement le problème en ligne (plus le nombre de paramètres n_p à optimiser est grand, plus cela devient délicat). En contrepartie, le point fort de cette méthode est son adaptabilité à n'importe quel type de robot, à partir du moment où l'on connaît son modèle cinématique direct.

En utilisant la commande prédictive pour la navigation d'un robot mobile, nous pouvons projeter continûment une famille de trajectoires d'un robot à travers un modèle direct sur un horizon temporel donné. De plus, la perception locale de l'environnement (obstacles fixes ou mobiles) est intégrée dans les contraintes de la commande prédictive. Parallèlement, nous définissons une trajectoire de référence construite à partir d'une courbe de Bézier paramétrée par les données fournies par le planificateur de chemin. La résolution de la fonction coût propose la trajectoire admissible libre d'obstacle la plus proche de la trajectoire de référence. En choisissant les familles de trajectoires soumises au navigateur, nous maitrisons le comportement du robot au cours de la mission.

CHAPITRE V
APPLICATION A UN VEHICULE A DEUX ESSIEUX DIRECTEURS

Le navigateur par commande prédictive est simulé sur un véhicule de type CyCab. Les contraintes cinématiques, spécifiées à partir de la description du robot, et une famille de trajectoire proposée sont intégrées dans la fonction coût. Deux algorithmes, un déterministe (Levenberg-Marquardt) et un stochastique (Recuit-Simulé), sont décrits et appliqués pour la résolution de cette fonction. Des simulations comparatives valident la méthode de navigation par modèle direct et commande prédictive. Elles permettent également d'évaluer la pertinence des algorithmes suivant la structure de l'environnement.

I. Présentation du CyCab

1. Historique

Le premier prototype de CyCab a été développé par l'INRIA Rhônes Alpes. C'est un véhicule électrique construit à partir de la base mécanique des voiturettes de golf Andruet. La particularité de ce véhicule électrique est son système de double braquage, à l'avant et l'arrière, qui permet de réduire le rayon de braquage du véhicule, améliorant ainsi sa manœuvrabilité dans les espaces restreints. Cette propriété en fait un véhicule très intéressant dans le cadre de la navigation en milieu urbain (Figure 62).

L'application visée par ce véhicule est de développer une alternative au transport public « de masse » que sont les bus de ville. L'idée était de créer un parc de véhicule en libre service, permettant de transporter 2 personnes sans pilote.

2. Modèle cinématique du robot

Dans cette partie nous déterminons le modèle cinématique direct du CyCab (Figure 63). On pose :

- G le centre de giration du véhicule ; ce point correspond au point d'intersection des axes des 4 roues, dans l'hypothèse d'un mouvement sans glissement sur le sol
- L l'empattement du véhicule
- F et R respectivement les point milieux des essieux avant et arrière ;
- ξ l'angle de braquage de l'essieu avant, équivalent à l'angle de braquage des roues avant, sur le point F ; et de la même $k\xi$ l'angle de braquage de l'essieu arrière au point R

- ρ_F et ρ_R les rayons de braquage des essieux avant et arrière respectivement, correspondant aux distances FG et RG

- enfin on définit le point C, que l'on appellera point de contrôle du véhicule, et qui correspond à la projection du centre de giration G sur la droite FC ; et on nommera ρ le rayon de braquage par rapport à ce point C ($\rho = GC$) .

Figure 62- Le robot CyCab

Nous commençons par calculer la position du point C sur l'axe FC, pour ce faire nous calculons les distances FC et RC :

$$\begin{cases} FC = \rho.\tan(\xi) \\ RC = \rho.\tan(k\xi) \end{cases}$$ Équation 47

On trouve le système :

$$\rho = \frac{FC}{\tan(\xi)} = \frac{RC}{\tan(k\xi)} \text{ et } FC + RC = L$$ Équation 48

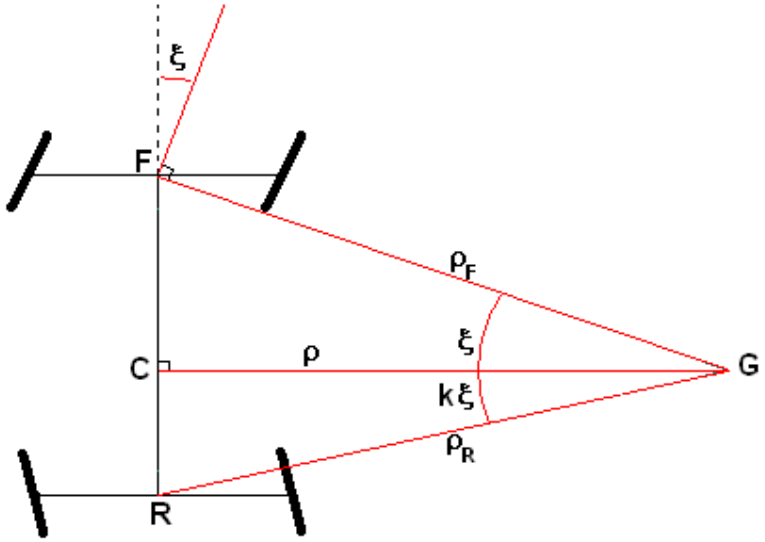

Figure 63- Schéma du système mécanique du CyCab

Pour $\tan(\xi) \neq 0$ et $\tan(k\xi) \neq 0$ soit $\xi \neq 0°$ et $\xi \neq 180°$. Vu qu'il s'agit d'angles de braquage, $\xi = 180°$ n'est pas admissible, et $\xi = 0°$ correspond à la situation où le robot ne tourne pas. Dans ce cas le modèle est directement donné par :

$$\begin{cases} \dot{x}_C = V.\cos(\theta) \\ \dot{y}_C = V.\sin(\theta) \\ \dot{\theta} = 0 \end{cases}$$ Équation 49

Le système Équation 48 a pour solution :

$$FC = L\frac{\tan(\xi)}{\tan(\xi)+\tan(k\xi)} \text{ et } RC = L\frac{\tan(k\xi)}{\tan(\xi)+\tan(k\xi)}$$ Équation 50

On pose $\chi = \frac{\tan(\xi)}{\tan(\xi)+\tan(k\xi)}$ ce qui nous donne :

$$FC = \chi.L \text{ et } RC = (1-\chi).L$$ Équation 51

Lors d'une rotation autour du point G, la vitesse de rotation en chaque point s'écrit :

$$\dot{\theta} = \frac{V_R}{\rho_R} = \frac{V_F}{\rho_F} = \frac{V_C}{\rho}$$ Équation 52

Géométriquement on trouve :

$$\rho = \frac{FC}{\tan(\xi)} = \frac{\chi.L}{\tan(\xi)}$$ Équation 53

En posant $V=V_C$, on en déduit le modèle cinématique direct du robot :

$$\begin{cases} \dot{x}_C = V.\cos(\theta) \\ \dot{y}_C = V.\sin(\theta) \\ \dot{\theta} = \dfrac{V}{\rho} = V\dfrac{\tan(\xi)}{\chi.L} = V\dfrac{\tan(\xi)+\tan(k\xi)}{L} \end{cases}$$ Équation 54

Le modèle obtenu est très proche du modèle cinématique d'un véhicule de type voiture. La seule différence se situe au niveau de la vitesse de rotation : pour un véhicule classique, avec seulement les deux roues directrices à l'avant, l'équation obtenue serait $\dot{\theta} = V\dfrac{\tan(\xi)}{L}$. Dans le cas du CyCab, et de tout véhicule à double braquage, cette vitesse de rotation est ainsi sensiblement augmentée. Si on se place dans le cas moyen où k=1, on note que la valeur de $\dot{\theta}$ sera doublée par rapport à un véhicule normal. Et si on se place dans le cas k=0, on retombe tout normalement sur les équations traditionnelles d'un véhicule de type voiture.

Le modèle ainsi obtenu montre bien les propriétés intéressantes de braquage de ce type de véhicule, lui conférant une maniabilité accrue par rapport à un véhicule classique à simple braquage.

II. Commande prédictive appliquée au CyCab

1. La fonction coût

Nous avons vu que l'architecture de commande qui contrôle notre robot est une architecture multi niveaux, qui intègre un module de navigation (un générateur de trajectoires) et un module de planification de chemin. Ce module est chargé de fournir un chemin à suivre au navigateur. La trajectoire de référence, ou courbe de rattrapage, que l'on appellera $X_{ref}(t)$, est calculée à partir de ce chemin.

Pour la construction de la courbe de rattrapage, nous utilisons une courbe de Bézier, qui est une courbe polynomiale de degré 3. Cette courbe possède 4 points de contrôle (P1 à P4), passe par P1 et P4, et a pour dérivée 3(P2-P1) en P1, et 3(P4-P3) en P4. Nous utilisons ces propriétés pour faire en sorte que cette courbe respecte les contraintes sur X_0 et le prochain point à atteindre fourni par le planificateur X_d soit :

$$X_d = \begin{pmatrix} x_d \\ y_d \\ \theta_d \\ \dot{s}_d \end{pmatrix}$$ Équation 55

Pour que la courbe de rattrapage concorde avec Xo et Xd, les points de contrôle de la courbe de Bézier doivent vérifier :

$$P1:\begin{cases} x_1 = x_0 \\ y_1 = y_0 \end{cases},\ P2:\begin{cases} x_2 = x_1 + \frac{1}{3}.\dot{s}_0.\cos(\theta_0) \\ y_2 = y_1 + \frac{1}{3}.\dot{s}_0.\sin(\theta_0) \end{cases},\ P3:\begin{cases} x_3 = x_4 - \frac{1}{3}.\dot{s}_d.\cos(\theta_d) \\ y_3 = y_4 - \frac{1}{3}.\dot{s}_d.\sin(\theta_d) \end{cases} \text{ et}$$

$$P4:\begin{cases} x_4 = x_d \\ y_4 = y_d \end{cases}$$

Équation 56

Enfin l'équation de la courbe de rattrapage avec les points de contrôle ainsi déterminés devient :

$$\begin{cases} x_r(l) = (1-l)^3 x_1 + 3l(1-l)^2 x_2 + 3l^2(1-l)x_3 + l^3 x_4 \\ y_r(l) = (1-l)^3 y_1 + 3l(1-l)^2 y_2 + 3l^2(1-l) y_3 + l^3 y_4 \end{cases} \text{ avec } 0 < l < 1$$

Équation 57

Il s'agit d'une courbe géométrique normalisée entre 0 et 1 et de longueur que le robot ne peut pas forcément parcourir pendant un temps égal à l'horizon de prédiction T_p. Ainsi, si la longueur de la courbe D est supérieure à $\dot{s}_d.T_p$, seule la partie entre $l = 0$ et $l = \frac{\dot{s}_d.T_p}{D}$ est conservée. A l'inverse, si le robot est capable d'atteindre le point X_d avant la fin de l'horizon temporel T_p, alors la trajectoire de référence est augmentée en utilisant le point de passage suivant fourni par le planificateur, comme nous l'avons décrit dans le chapitre IV-2.

Au final la trajectoire de référence X_{ref} est :

$$Xref(t) = \begin{pmatrix} x_{ref}(t) \\ y_{ref}(t) \\ \theta_{ref}(t) \\ s_{ref}(t) \end{pmatrix} = \begin{pmatrix} x_r(\frac{\dot{s}_d}{D}.t) \\ y_r(\frac{\dot{s}_d}{D}.t) \\ \arctan 2(\dot{x}_r, \dot{y}_r) \\ \dot{s}_d \end{pmatrix} \text{ avec } t \in [0, T_p] \text{ si } \frac{\dot{s}_d}{D}t \leq l$$

Équation 58

$$Xref(t) = \begin{pmatrix} x_{ref}(t) \\ y_{ref}(t) \\ \theta_{ref}(t) \\ s_{ref}(t) \end{pmatrix} = \begin{pmatrix} x_d \\ y_d \\ \theta_d \\ \dot{s}_d \end{pmatrix} \text{ avec } t \in [0, T_p] \text{ si } \frac{\dot{s}_d}{D}t > l \text{ et si } X_d \text{ est le point terminal}$$

Équation 59

$$X_{ref}(t) = \begin{pmatrix} x_{ref}(t) \\ y_{ref}(t) \\ \theta_{ref}(t) \\ s_{ref}(t) \end{pmatrix} = \begin{pmatrix} x_{r'}(\frac{\dot{s}_d}{D}.t) \\ y_{r'}(\frac{\dot{s}_d}{D}.t) \\ \arctan_2(\dot{x}_{r'}, \dot{y}_{r'}) \\ v_{\max} \end{pmatrix} \text{ avec } t \in [\frac{D}{v_{\max}}, T_p]$$

Équation 60

si X_d n'est pas le point terminal.

Maintenant que la courbe de rattrapage $X_{ref}(t)$ est définie, nous définissons la fonction coût de la manière suivante :

$$J(X_0,t_0,T_p,u) = K_{final}(X_{ref} - X(t_0+T_p,u))^2 + \int_{t_0}^{t_0+T_p} K_{pacours}(X_{ref}(\tau) - X(\tau,u))^2 d\tau$$ Équation 61

Où K_{final} et $K_{parcours}$ sont des matrices de pondération de la forme :

$$\begin{pmatrix} k_x & 0 & 0 & 0 \\ 0 & k_y & 0 & 0 \\ 0 & 0 & k_\theta & 0 \\ 0 & 0 & 0 & k_{v'} \end{pmatrix}$$ Équation 62

Concrètement, cette fonction coût calcule l'erreur quadratique entre la trajectoire de référence et la trajectoire du robot, en pondérant différemment les différentes composantes de l'état du robot, et en pondérant également différemment l'erreur terminale et l'erreur de suivi du parcours.

2. La fonction de commande et les trajectoires générées

Pour obtenir des bonnes possibilités de contournement d'obstacle, nous avons utilisé des fonctions de commande $u(v,\xi)$ de type linéaire (Figure 64) :

Ce sont des fonctions linéaires, dans lesquelles les paramètres p_1 et p_3 désignent respectivement la vitesse et le braquage à atteindre en $t_0 + T_p/2$, et les paramètres p_2 et p_4 la vitesse et le braquage à atteindre en $t_0 + T_p$.

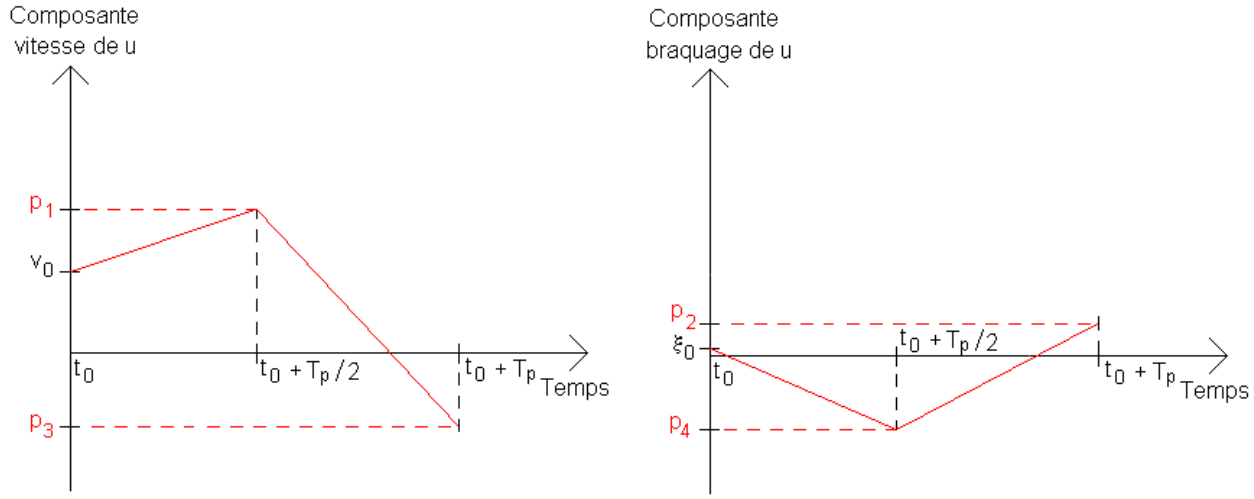

Figure 64- Famille de fonctions d'entrées u

Ces paramètres doivent respecter un certain nombre de contraintes par rapport aux vitesses et braquages maximaux et minimaux :

$$
\begin{aligned}
v_{\min} &\le p_1 \le v_{\max} \\
v_{\min} &\le p_3 \le v_{\max} \\
\xi_{\min} &\le p_2 \le \xi_{\max} \\
\xi_{\min} &\le p_4 \le \xi_{\max}
\end{aligned}
$$

Équation 63

Ils doivent également respecter des contraintes sur les accélérations, décélérations et vitesses de braquage maximales :

$$
\begin{aligned}
v_0 - decel_{\max} * \frac{T_p}{2} \le p_1 \le v_0 + accel_{\max} * \frac{T_p}{2} \\
p_1 - decel_{\max} * \frac{T_p}{2} \le p_3 \le p_1 + accel_{\max} * \frac{T_p}{2} \\
\xi_0 - \dot{\xi}_{\max} * \frac{T_p}{2} \le p_2 \le \xi_0 + \dot{\xi}_{\max} * \frac{T_p}{2} \\
p_2 - \dot{\xi}_{\max} * \frac{T_p}{2} \le p_2 \le p_2 + \dot{\xi}_{\max} * \frac{T_p}{2}
\end{aligned}
$$

Équation 64

A partir de cette famille de fonctions de commande (Figure 65), une famille de trajectoires est générée (Figure 66). En faisant simplement varier chaque paramètre selon 3 valeurs (son maximal, son minimal et sa moyenne), nous obtenons $3^4 = 81$ trajectoires différentes permettant de montrer des possibilités de mouvement très variées (grande ligne droite, braquage contre braquage, marche arrière puis demi-tour, etc…). Ces trajectoires sont représentées sur les figures ci-dessous, avec les fonctions d'entrées correspondantes.

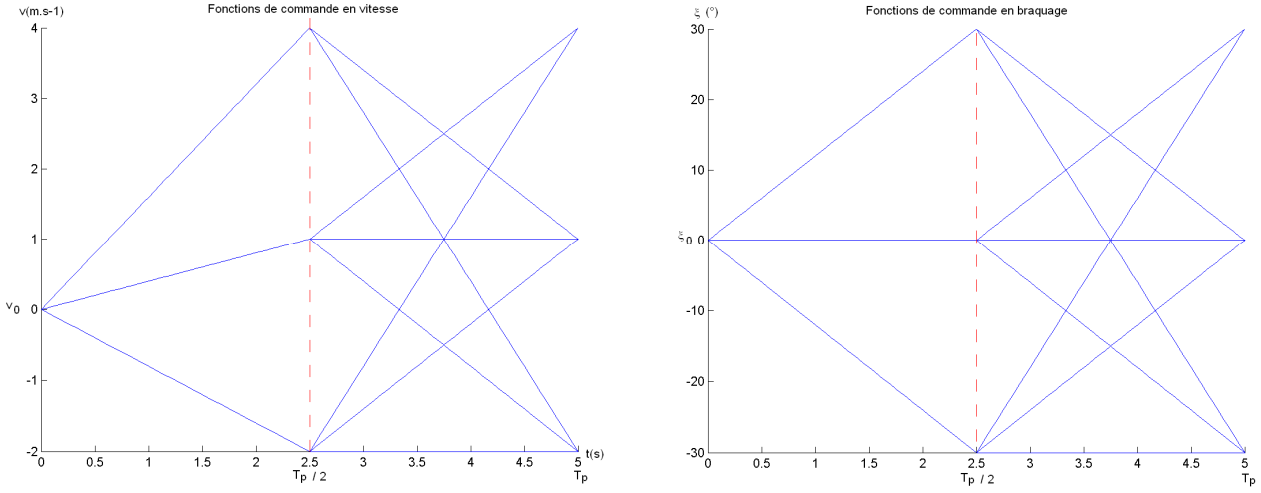

Figure 65- Fonctions de commande

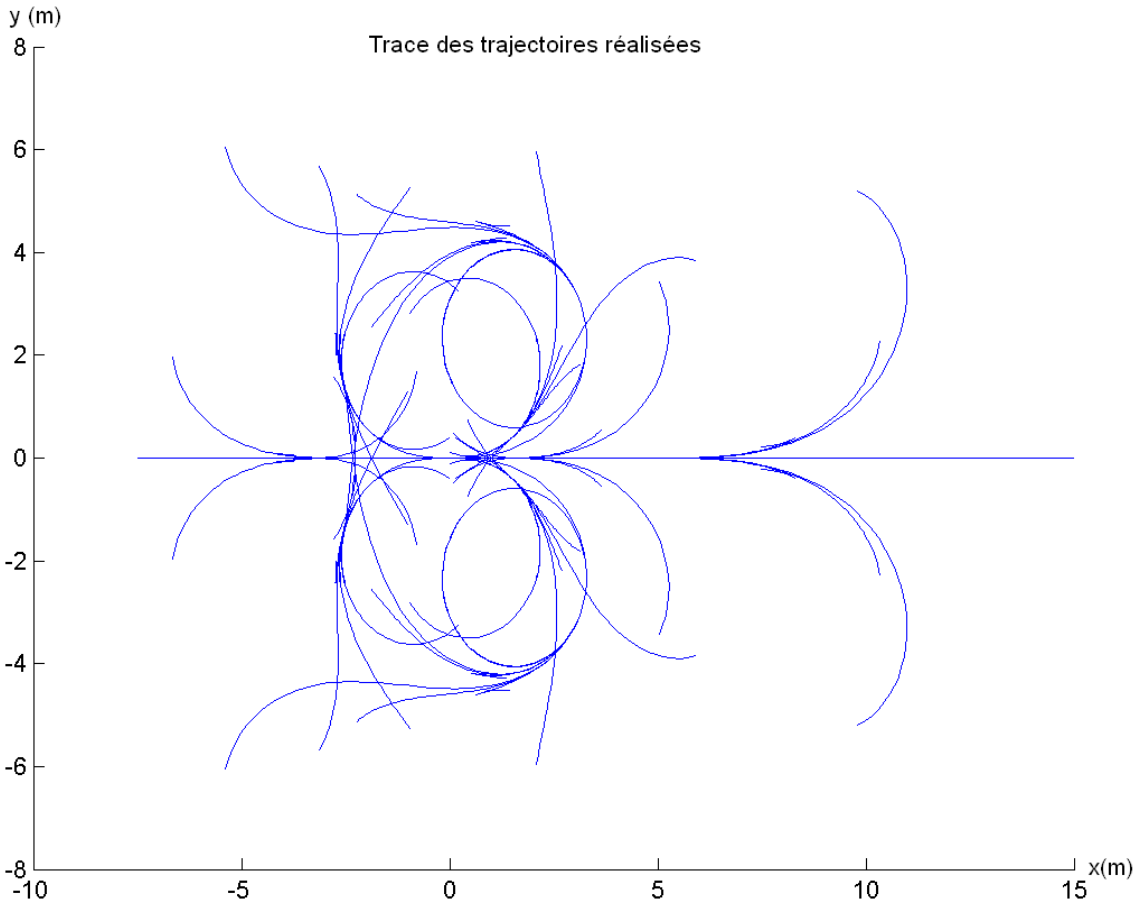

Figure 66- Trace des trajectoires du robot

3. L'algorithme d'optimisation

1- Levenberg Marquardt

Levenberg Marquardt est un algorithme éprouvé d'optimisation déterministe, il trouvera donc toujours la même solution pour un problème donné [46]. C'est une méthode qui utilise les informations contenues dans la fonction critère, à savoir le Hessien et le gradient de la fonction, pour converger le plus efficacement possible vers le minimum *local* de la fonction (Figure 67).

Le principe de fonctionnement de cet algorithme s'inspire de celui de la descente du gradient (recherche de la direction de plus forte pente descendante), mais utilise en plus les informations contenues dans Hessien de la fonction coût (dérivées secondes) pour adapter le pas de recherche à la dynamique de la fonction coût. Cela permet de converger beaucoup plus rapidement (en nombre d'itérations) vers le minimum de la fonction. L'expression générale du pas de Levenberg est :

$$\delta p_i = (H + \lambda.diag(H))^{-1} G \qquad \text{Équation 65}$$

Où H et G sont respectivement le hessien et le gradient de la fonction coût J. L'organigramme de l'algorithme que nous utilisons est présenté sur la figure Figure 67- organigramme d'optimisation.

Le coefficient λ permet à l'algorithme de s'adapter à la forme du critère. Pour un λ moyen, l'algorithme est une combinaison entre une descente de gradient et une approximation quadratique du critère. Si l'algorithme converge, la valeur de λ est augmentée de manière à augmenter l'influence du hessien, et ainsi accélérer la vitesse de convergence. De cette manière, dans les zones linéaires, on se déplace très rapidement vers le minimum de la fonction.

Dans les zones où le critère est moins linéaire, l'algorithme peut se mettre à diverger à cause du hessien qui va se mettre à varier très rapidement et de manière non monotone. Dans ce cas, la valeur de λ est diminuée de manière à baisser l'influence du hessien, et se rapprocher d'un algorithme de type descente de gradient classique.

Les avantages de cet algorithme sont :

- la convergence vers le minimum local du problème, donc la meilleure solution locale au sens du critère
- la très grande rapidité à converger (en quelques itérations l'algorithme est très proche de la solution locale)

En contrepartie, son principal inconvénient réside dans sa grande sensibilité aux problèmes de minima locaux, donc résultats tributaires de l'initialisation de l'algorithme. Dans les problèmes de contournement d'obstacle, il aura donc des difficultés à trouver une trajectoire de contournement.

Pour qu'un algorithme d'optimisation puisse être utilisé dans une application temps réel, il faut garantir que la solution obtenue si on l'arrête à n'importe quelle itération soit contenue dans le domaine des solutions admissibles pour le robot. D'autre part, il est souhaitable que l'algorithme converge le plus rapidement possible vers la solution optimale.

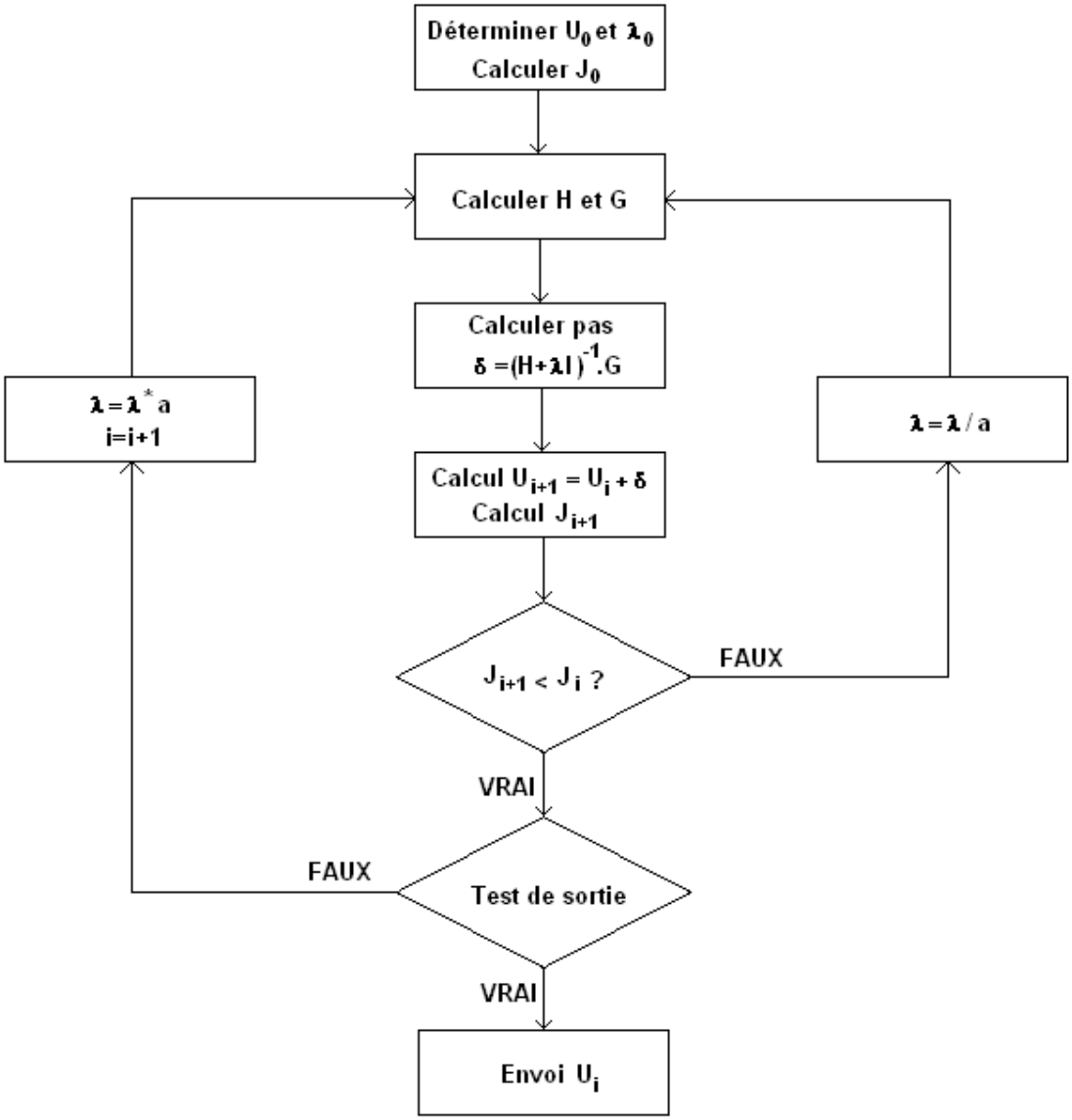

Figure 67- organigramme d'optimisation

2- Recuit simulé

Recuit simulé est un algorithme stochastique, basé sur une part d'aléatoire ; il trouvera généralement une solution différente si on lui représente le même problème. Il a été développé par Kirkpatrick, Gelatt et Vecchi en 1983 [35]. Le principe de cet algorithme s'inspire du principe physique du recuit des matériaux solides. Le chauffage du matériau à très haute température, puis son refroidissement lent conduisent à une réorganisation plus ordonnée des atomes constituant le matériau, et l'amenant ainsi vers un état plus stable (énergie plus faible).

L'algorithme va tester les critères à optimiser associés aux variables proches des variables d'initialisation. Si le critère associé à ces variables voisines est plus faible, on conserve ces variables comme référence pour l'itération suivante. Si le critère est plus élevé, il y aura une certaine probabilité que ces variables soient tout de même gardées comme référence. La loi de probabilité qui régit ces transitions vers un critère plus élevé, est une loi décroissante au fil du temps (s'inspirant ainsi du refroidissement d'un matériau). Cette probabilité suit une formule du type :

$$P = 1 - \exp(-\frac{\Delta J}{T}) \qquad \text{Équation 66}$$

Dans cette formule, ΔJ est l'écart entre le nouveau critère et l'ancien, et T joue le rôle d'une « température » qui diminue au fil du temps, selon une loi de refroidissement qu'il faut régler empiriquement. Ainsi au début du processus, de nombreux mouvements vers des positions associées à un critère plus élevé seront tolérés, permettant à l'algorithme d'explorer plus de régions et ainsi ne pas s'enfermer dans un minimum local.

Au fur et à mesure de l'évolution du processus d'optimisation, ces transitions vers des critères plus élevés seront de moins en moins permises, ce qui contraint l'algorithme à converger vers un minimum global.

Les avantages de cet algorithme sont :
- une grande capacité à éviter les minima locaux et à converger vers le minimum global
- il n'y a pas de condition particulière sur les particularités de la fonction coût

Par contre cet algorithme souffre des inconvénients suivants:
- l'algorithme converge vers une solution approchée et non une solution exacte
- la vitesse de convergence est plus faible au voisinage du minimum global

III. Simulations

1. Comparaison des algorithmes

Nous avons comparé les résultats obtenus de notre navigateur en utilisant un algorithme déterministe d'une part, et un algorithme stochastique de l'autre. L'algorithme déterministe testé est Levenberg Marquardt (LM), qui utilise les informations sur les dérivées premières et secondes de la fonction de coût J. L'algorithme stochastique est le Recuit Simulé (RC), qui s'inspire du refroidissement des matériaux en effectuant une recherche aléatoire, de plus en plus ordonnée au fil des itérations (ce qui permet d'explorer plus de régions du domaine des solutions admissibles dans un premier temps, avant de converger lentement vers un minimum dans un second temps).

La figure Figure 68 montre un cas où le robot est face à un obstacle, et où le planificateur de chemin lui fournit un point de passage situé de l'autre côté de l'obstacle. Nous regardons uniquement la trajectoire sélectionnée par chaque algorithme à l'instant t_0, le système n'est pas bouclé. Dans cette situation, les simulations montrent qu'en utilisant l'algorithme LM, le navigateur n'est pas capable de trouver une trajectoire de contournement, car il aura tendance suivre strictement la trajectoire de référence. Par contre, en utilisant RC, le navigateur sera capable de trouver des trajectoires de contournement.

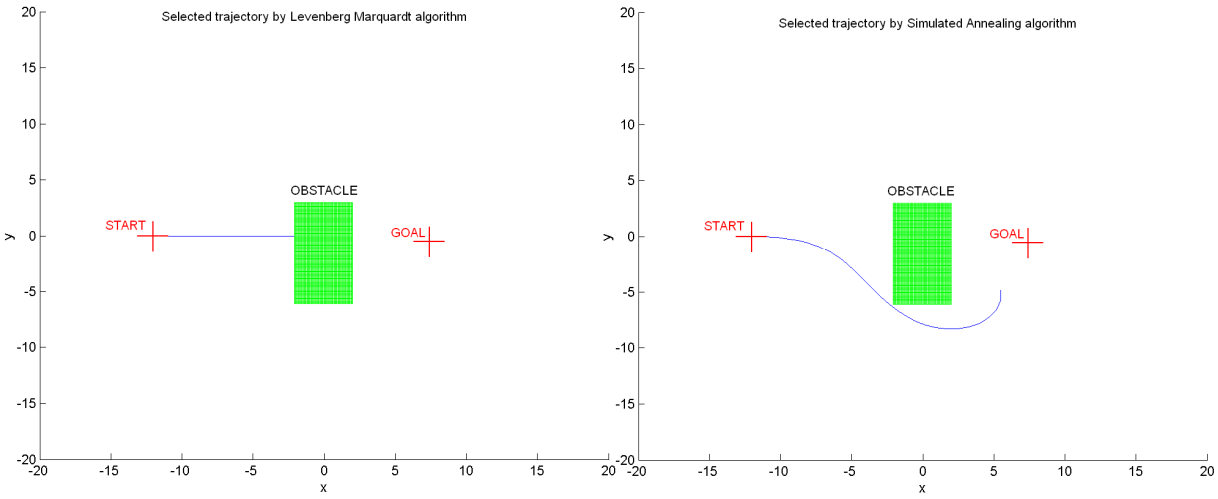

Figure 68- Evitement d'obstacle avec différents algorithmes d'optimisation (gauche : Levenberg-Marquardt, droite : Recuit-Simulé)

2. Simulation complete d'évitement d'obstacle

Les simulations suivantes montrent le comportement du robot dans le cas où la trajectoire de référence traverse un obstacle. Ces simulations permettent de montrer comment le navigateur que nous avons développé permet d'adapter la trajectoire du robot pour contourner l'obstacle, tout en suivant du mieux possible la trajectoire de référence. Sur ces simulations, nous avons utilisé l'algorithme du recuit simulé, qui comme nous l'avons montré précédemment est efficace pour trouver des trajectoires dans ce type de situation.

Sur les figures, la trace des déplacements effectués par le robot est représentée en bleu. Les croix bleues repèrent une nouvelle itération du navigateur. De ce fait, la trace représentée entre deux croix correspond aux déplacements effectués par le robot pendant une période d'échantillonnage. La trajectoire de référence est représentée en rouge, et la trajectoire prévue du robot entre $t_o + T_e$ et $t_o + T_p$ est tracée en vert. La fréquence d'échantillonnage T_e est fixée à 1s, et l'horizon de prédiction T_p à 5s.

Sur la figure Figure 69, le robot doit se rendre en (10,5) en partant de (-10,0). Sur cette simulation, le robot trouve bien une trajectoire pour contourner l'obstacle, et atteint la position souhaitée. Par contre, on remarque que la trajectoire sélectionnée par le navigateur s'écarte beaucoup de la trajectoire de référence. Cela s'explique d'une part par le réglage du poids du coût de parcours, que l'on a affaibli par rapport au poids du cout terminal, de manière à accroitre la liberté du robot à s'éloigner de la trajectoire de référence (et ainsi trouver plus facilement des trajectoires de contournement) ; et d'autre part par l'utilisation de l'algorithme du recuit simulé, qui nécessite beaucoup d'itérations pour converger efficacement vers la trajectoire optimale.

Sur la simulation représentée par la figure Figure 70, le point d'arrivée a été changé en (10,-6). Encore une fois, cette simulation met en évidence d'une part l'efficacité du navigateur couplé à l'algorithme du recuit simulé pour effectuer un contournement d'obstacle ; et d'autre part le manque de précision dans le choix d'une trajectoire optimale pour suivre une trajectoire de référence.

Les figures Figure 71 et Figure 72 présentent des simulations où le robot se rend de (-5,10) à (5,-10). Ces simulations mettent en évidence le fait que pour une même situation donnée, l'utilisation de l'algorithme du recuit simulé engendrera des choix de trajectoire qui peuvent être totalement différents. Cela est dû au fait que le recuit simulé fait partie de la classe des algorithmes

stochastiques. Dans tous les cas, la trajectoire générée par le navigateur permet de contourner l'obstacle et d'atteindre la bonne position sous une précision satisfaisante.

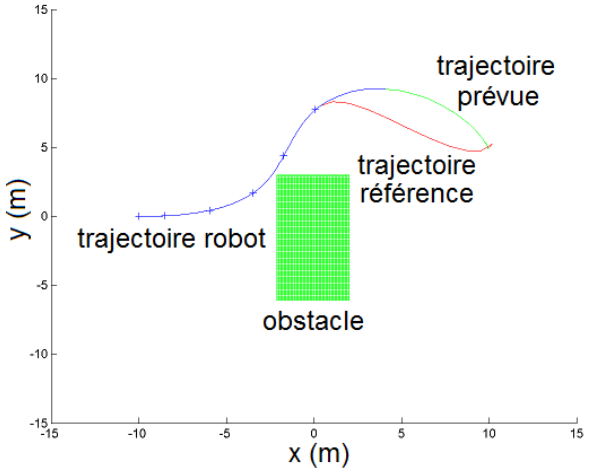

Figure 69- Contournement d'obstacle avec recuit simulé (a)

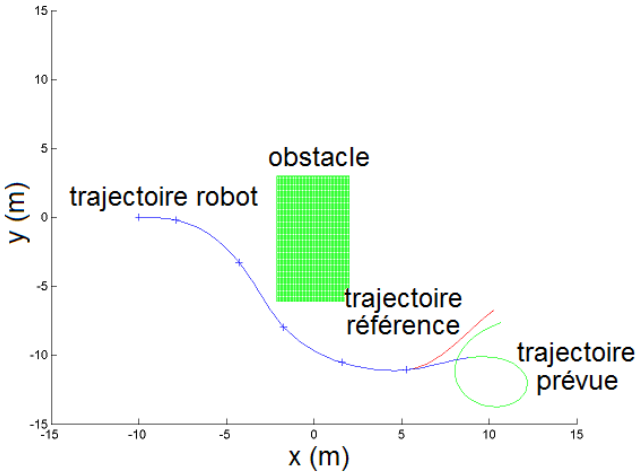

Figure 70- Contournement d'obstacle avec recuit simulé (b)

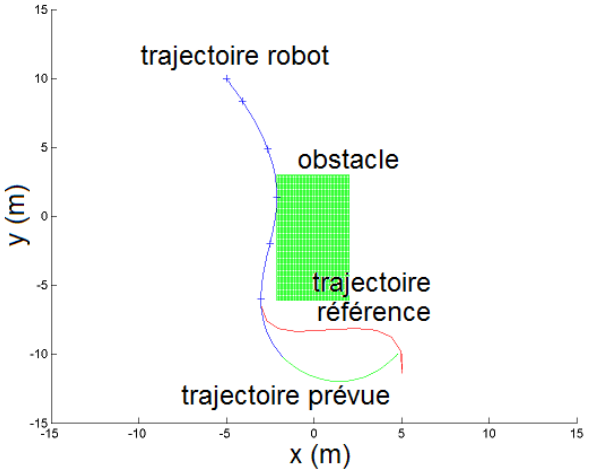

Figure 71- Contournement d'obstacle avec recuit simulé (c)

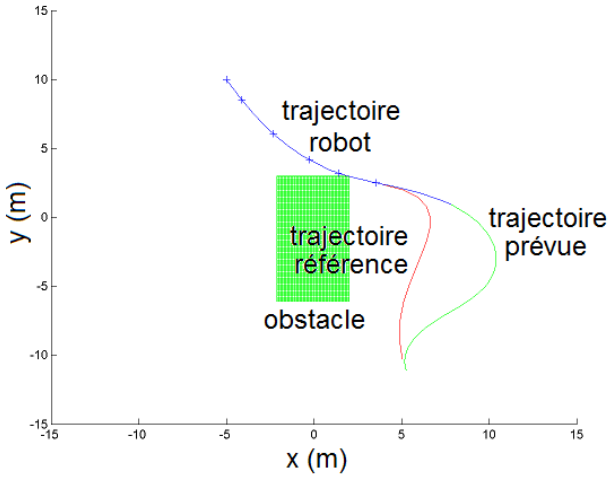

Figure 72- Contournement d'obstacle avec recuit simulé (d)

3. Influence de l'horizon de prédiction

Dans les simulations suivantes, nous tentons de mettre en avant l'effet de l'horizon de prédiction sur la capacité du navigateur à trouver des trajectoires de contournement d'obstacle. Sur les figures Figure 74 et Figure 73, l'objectif qui est proposé au robot est de se rendre du point (-5 ,10) au point (5,-10). Sur la figure Figure 73, l'horizon de prédiction est fixé à 5s, alors que sur la figure Figure 74 il est de 2.5s. Ces simulations montrent que si l'horizon de prédiction Tp est fixé à une valeur trop faible, le robot n'est pas capable de trouver une trajectoire satisfaisante. Il existe donc un compromis à faire entre la distance que le robot est capable de parcourir pendant T_p, et la taille caractéristique des obstacles D_o. Dans nos simulations, les meilleurs résultats ont été obtenus pour un rapport :

$$\frac{v_{\max} * T_p}{D_o} \approx 3 \qquad \text{Équation 67}$$

.

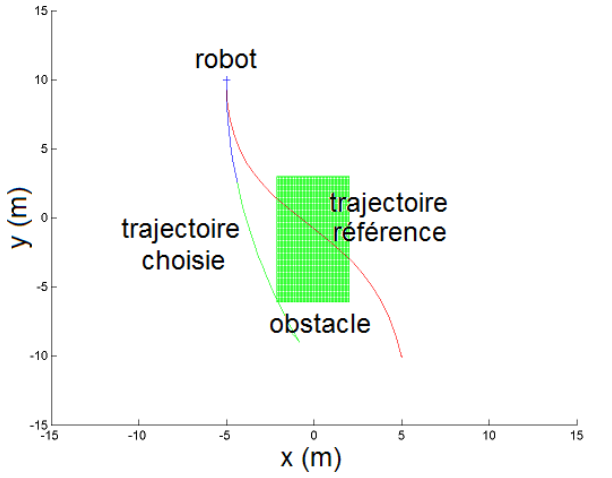

Figure 73- Horizon de prediction 5s

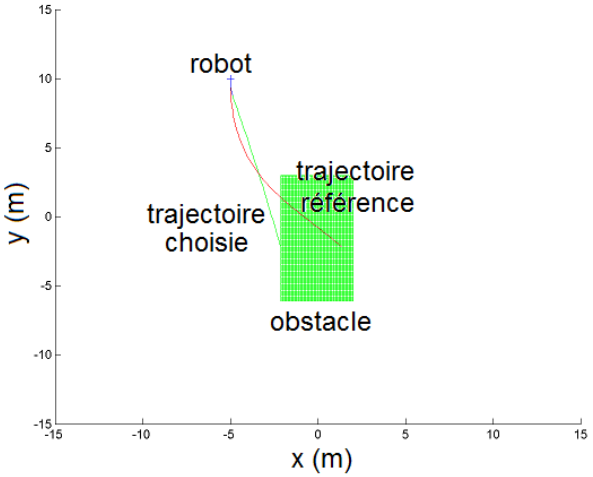

Figure 74- Horizon de prediction 2.5s

IV. Simulations complètes de navigation

Figure 75 et Figure 76 montrent une simulation de navigation pour les robot CyCab et RAOUL évoluant au milieu d'obstacles. En entrée, le robot reçoit un certain nombre de points de passage successifs (grosses croix rouges), et la position des obstacles sur une carte de type grille d'occupation de taille 40*40 m² et de résolution 10*10 cm² (les obstacles sont représentés par les zones vertes, et on considère qu'ils sont élargis pour tenir compte de la taille du robot). La trajectoire du robot est représentée par la ligne bleue, chaque petite croix marquant une nouvelle itération de l'algorithme de navigation (période Te). Enfin, la courbe rouge correspond à la trajectoire de référence calculée à partir de l'état courant du robot et du prochain point de passage, et la courbe verte est la trajectoire sélectionnée par l'algorithme d'optimisation sur l'horizon de prédiction suivant.

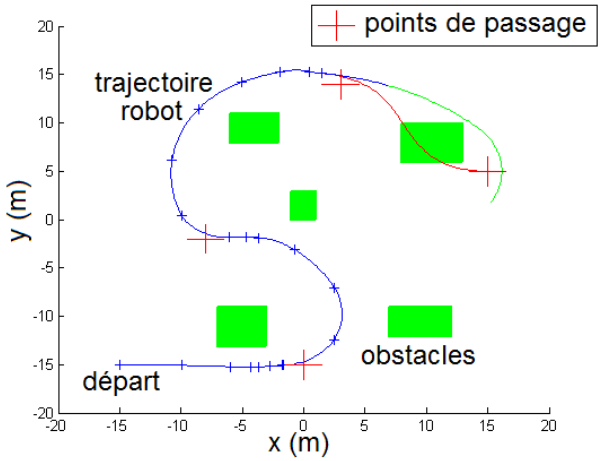

Figure 75 : Navigation du CyCab en milieu obstrué pour aller de (-15;-15) à (15;5)

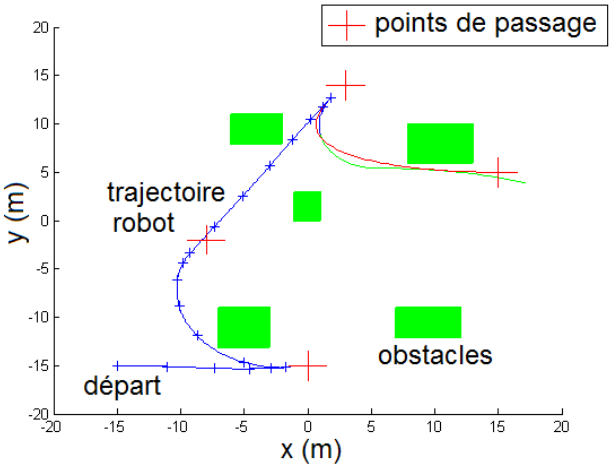

Figure 76: Navigation de RAOUL en milieu obstrué pour aller de (-15;-15) à (15;5)

Dans ce scénario, le robot se rend du point (-15,-15) au point (15,5) en passant par 3 points de passages successifs, sous une précision de 1m pour chacun d'entre eux (avant de passer au point suivant). Tout d'abord on remarque que le robot n'entre jamais en collision avec les obstacles, et qu'il parvient bien à rallier ses différents points de passage successifs. Si on regarde plus attentivement, on voit que la planification fournie au navigateur ne tient pas compte efficacement des différents obstacles, ce qui engendre des trajectoires de référence entrant en collision avec les obstacles. Malgré cela, le navigateur parvient à déterminer des trajectoires de contournement autour des obstacles, ce qui lui d'atteindre ses différents objectifs. Cela est rendu possible grâce d'une part à l'utilisation d'un algorithme de type recuit simulé (qui trouve des solutions en dehors des minima locaux induits par la cassure de convexité due à l'introduction des obstacles), et d'autre part grâce à l'utilisation d'un horizon temporel adapté permettant de générer des trajectoires suffisamment longues pour contourner les obstacles.

Les réglages des pondérations Kparcours et Kfinal sont les suivants :

$$K_{parcours} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} K_{final} = \begin{pmatrix} 100 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad \text{Équation 68}$$

Ce réglage fait que l'on cherche à contrôler la position (x,y) du robot, mais pas sa vitesse et son orientation, ce qui laisse de l'autonomie au robot dans le choix de ses trajectoires. Il en résulte que le robot à roues différentielles, RAOUL, qui possède des capacités cinématiques équivalentes en marche avant et marche arrière, utilisera aussi bien l'une que l'autre pour réaliser ses trajectoires (Figure 76).

Figure 77 et Figure 78 représentent les résultats d'un autre scénario de navigation dans lequel les robots partent du point (-18,12.5) et doivent se rendre en (-7,0) en passant par 4 points de passage successifs. Ce scenario présente plusieurs pièges en U pouvant induire un blocage du navigateur dû à une planification insuffisante. Grace à sa capacité d'anticipation sur un horizon temporel de 5s, le navigateur permet aux robots de choisir des trajectoires qui ne les bloquent pas dans l'un des pièges.

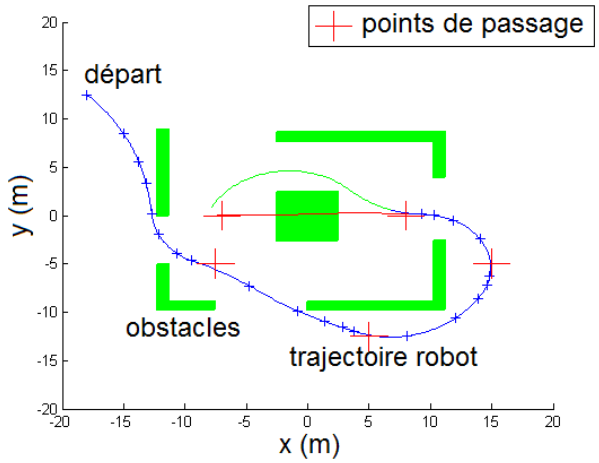

Figure 77 : Navigation du CyCab en milieu obstrué pour aller de (-18;12.5) à (-7;0)

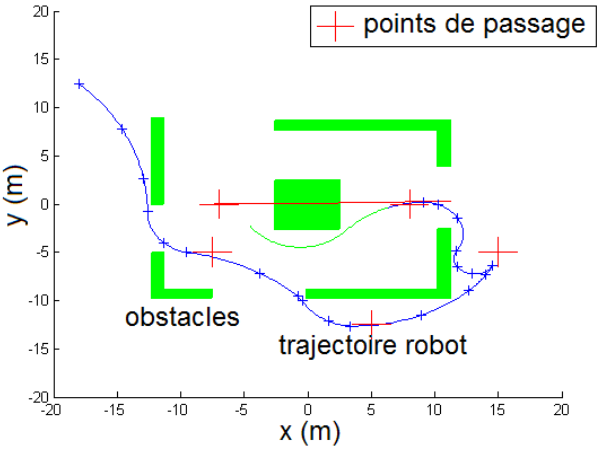

Figure 78 : Navigation de RAOUL en milieu obstrué pour aller de (-18;12.5) à (-7;0)

Figure 79 présente un scénario encore plus contraint (avec plusieurs pièges en U). On constate que la trajectoire complète du robot évite les obstacles et suit les points de passage, ce qui valide la méthode de navigation dans le cadre de navigation en milieux encombrés.

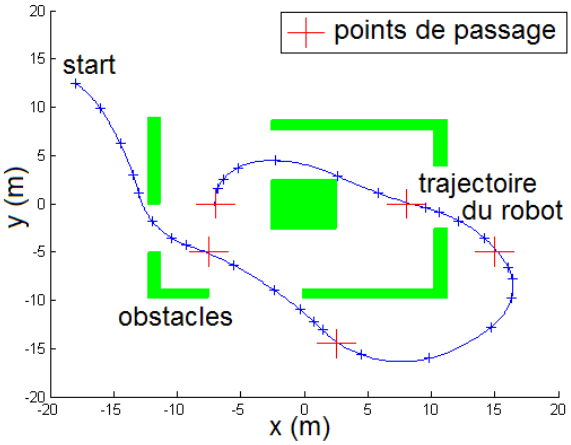

Figure 79: Navigation du robot CyCab en environnement encombré (obstacles en U)

Figure 80 présente le même scénario de navigation que le précédent, cependant cette fois ci la fonction coût a été modifiée de manière à inciter le robot à conserver une continuité dans la marche avant ou marche arrière. Concrètement, la fonction coût est la suivante :

$$Z_{tot}(X_0,t_0,T_p,u,C_{cr}) = Z(X_0,t_0,T_p,u) + \sum^{n_r} B_r(C_{r,i}) + \sum^{n_o} B_o(C_{o,i}) + \sum^{m} (u(t_0+T_e) - u(t_0))^2 \qquad \text{Équation 69}$$

Cette fonction est intéressante si l'on cherche à conserver une continuité dans la commande du robot, mais elle diminue la liberté du robot dans le choix de ses trajectoires en fonction de ses capacités cinématiques.

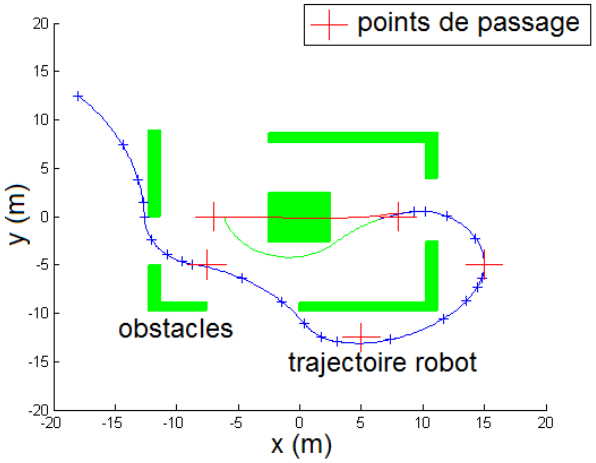

Figure 80 : Navigation de RAOUL en milieu obstrué pour aller de (-18;12.5) à (-7;0) avec le critère

Figure 81 montre une simulation de "planificateur fou" dans laquelle le navigateur se voit fournir des points de passage de manière aléatoire, sans tenir compte des obstacles. Cette simulation permet de montrer la robustesse du navigateur aux erreurs de planification, et sa capacité réactive, puisque le robot n'entre jamais en collision avec les obstacles.

Nous avons ensuite cherché à comparer les résultats obtenus par le navigateur par commande prédictive et le navigateur par lignes de fuite présenté dans le chapitre III. Figure 83 montre une simulation réalisée par le navigateur par lignes de fuite avec le robot CyCab, alors que Figure 82 présente les résultats obtenus par le navigateur par commande prédictive. Pour obtenir la même précision sur les points de passage successifs, le navigateur par commande prédictive a des temps de calcul diminué de 37% en moyenne. De plus, le robot rallie les différents points de passage successifs plus rapidement avec le navigateur par commande prédictive. Cela s'explique par le fait que comme le navigateur par lignes de fuite dispose d'un nombre de trajectoires restreints (dépendant de la discrétisation effectuée), il a plus de difficultés à passer près des points de passage et doit donc ralentir à chacun d'eux.

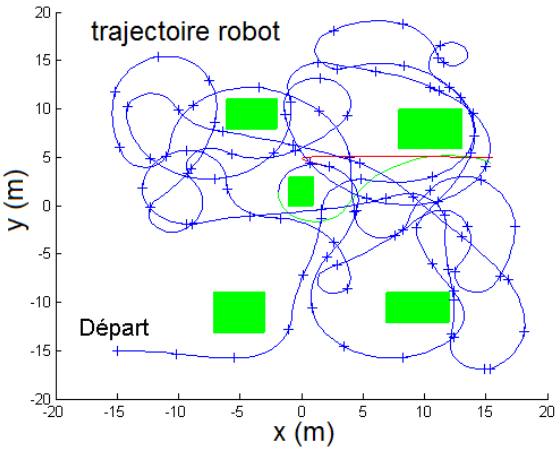

Figure 81 : planificateur fou et robustesse du navigateur à une planification défectueuse

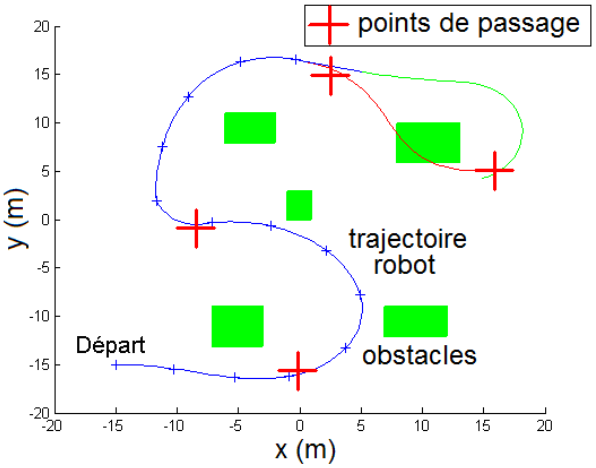

Figure 82 : navigation par commande prédictive

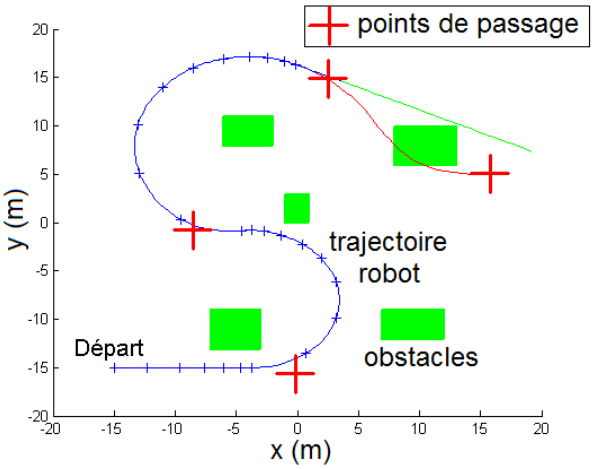

Figure 83 : navigation par lignes de fuite

V. Discussion

Les simulations valident l'efficacité de notre navigateur par commande prédictive en milieu encombré. Des trajectoires, calculées à partir du modèle cinématique direct du robot et respectant

ses contraintes cinématiques propres sont générées. Elles permettent au robot de contourner les obstacles se présentant sur son chemin, et qui ne seraient pas pris en compte par le planificateur pour diverses raisons (obstacle mobile, obstacle trop petit pour être pris en compte dans la cartographie globale…). Ce navigateur assure donc l'adaptation locale, aux contraintes de l'environnement et du robot, d'un chemin fourni par le planificateur.

La méthode a été appliquée dans notre étude à un robot mobile à deux essieux directeurs. En outre, elle peut aisément être appliquée à n'importe quel type de robot mobile dont le modèle cinématique direct est défini. Il suffit de modifier le modèle du robot et rajouter les contraintes cinématiques dans la formulation des contraintes générales. Les résultats pourraient alors être améliorés en adaptant au mieux les fonctions de commande définies au type de robot, de manière à tirer partie du maximum de ses capacités cinématiques spécifiques.

Un problème important sur lequel nous nous sommes heurtés, est celui du choix de l'algorithme d'optimisation à utiliser. L'emploi d'algorithmes déterministes tels que Levenberg-Marquardt, qui sont particulièrement efficaces lorsqu'il s'agit de suivre au mieux une trajectoire de référence en l'absence d'obstacles, devient contre indiqué lorsque des obstacles apparaissent devant le robot. En effet, ceux-ci se sont montrés particulièrement sensibles aux problèmes de minima locaux, avec notre modélisation. Face à ce type de contraintes, les algorithmes stochastiques se sont montrés beaucoup plus souples, en trouvant des trajectoires de contournement. Cependant, les trajectoires qu'ils génèrent restent moins précises que celles calculées par les algorithmes déterministes en absence d'obstacle. Ainsi, pour améliorer sensiblement la qualité des trajectoires générées dans n'importe quelle situation, une combinaison de ces deux types d'algorithmes est en cours d'étude, pour tenter d'allier les avantages de chacun d'eux.

Les résultats de ces simulations valident la méthode de navigation par modèle direct et commande prédictive pour un robot à double essieux directeur. L'environnement d'évolution du robot conditionne le choix de l'algorithme d'optimisation (déterministe ou stochastique) qui délivrera la meilleure trajectoire à suivre pour le robot sur l'horizon temporel choisi.

Conclusion Générale

L'autonomie d'un robot mobile reste une problématique de recherche ouverte, source de nombreuses thématiques.

D'un point de vue global, l'objectif de rendre un robot totalement autonome est réalisable par le découpage et la structuration de plusieurs tâches spécifiques. Elles se classent en deux grandes catégories : celles liées à la perception et celles liées à la commande. L'architecture de contrôle du robot structure ces différentes fonctions de manière à aboutir à un ensemble cohérent.

Dans ce cadre, nous avons choisi de développer notre travail sur la base de l'architecture de contrôle multi-niveaux proposée par Mourioux [51], dont le découpage hiérarchique et modulaire permet l'intégration de notre navigateur. Celui-ci devient alors un pivot qui fait le lien entre les capacités cinématiques du robot, la mission qui lui est assignée et l'environnement dans lequel il évolue. C'est donc une brique incontournable de l'objectif d'autonomie d'un robot mobile.

Nous proposons notamment une méthode qui permet d'accroitre l'autonomie du robot, en lui laissant le libre choix de ses trajectoires. Ainsi, si la situation du robot l'exige, par exemple si un obstacle imprévu apparaît devant lui, le robot est capable d'effectuer de lui-même des manœuvres d'évitement, telles que des marches arrière, sans que l'on ait à programmer un tel comportement. Cette propriété à adapter son comportement en fonction de la situation correspond à une capacité de prise de décision autonome, au niveau même du bloc de navigation.

Plus précisément, notre apport se situe dans le développement d'une méthode de détermination de trajectoires réalisables par le robot, avec l'évitement d'obstacles, et suivant un parcours fourni par un planificateur de chemin. Notre méthode réalise ainsi trois principales sous fonctions : le suivi d'un chemin de référence, l'adaptation aux contraintes cinématiques du robot et la gestion des obstacles.

Nous assurons le suivi de chemin par la construction d'une courbe de Bézier reliant la position courante du robot à sa destination, et respectant les orientations entre ces deux points. De plus, la continuité de la courbe générée est respectée. Cette courbe est ensuite paramétrée en temps de manière à obtenir une trajectoire de référence.

Pour la prise en compte de l'environnement, nous avons proposé une méthode continue et une méthode discrète d'intégration des obstacles, par segmentation ou par grille d'occupation. Notre navigateur peut ainsi s'adapter à ces différents types de cartographie.

Enfin, nous utilisons le modèle cinématique direct du robot, ce qui garantit la génération de trajectoires admissibles par le véhicule. Par l'ajout de contraintes de conception sur la commande, nous engendrons des familles de trajectoires spécifiques, qui permettent de maitriser le comportement du robot en s'adaptant à la situation (type de milieu d'évolution et de robot).

Cette utilisation du modèle direct constitue un autre point clé de notre travail pour le développement de notre navigateur. Elle a été motivée par la volonté de concevoir une méthode applicable simplement à n'importe quel type de robot mobile. Certaines méthodes par suivi de trajectoire, telles que les sorties plates ou les fonctions transverses, ont montré d'excellents résultats dans ce domaine, mais nécessitent la détermination de modèles inverses n'existant pas toujours pour le robot mobile étudié. A partir de cette constatation, notre démarche a consistée à se baser sur les mouvements réalisables du robot pour développer une méthode de navigation alternative.

Un autre atout de cette approche est l'intégration aisée de paramètres supplémentaires, tels que les glissements inhérents au type de terrain, par leur simple prise en compte dans le modèle cinématique du véhicule. La méthode de navigation n'est en rien changée. C'est un point fort par rapport aux méthodes utilisant des modèles inverses, puisque celles-ci ne sont plus valides dès lors que l'on rajoute des paramètres variant avec le type de terrain d'évolution.

Par ailleurs, d'un point de vue contrôle, notre problématique peut se traduire en termes suivants : prévoir les trajectoires, réalisables par le robot, par l'intermédiaire d'un modèle cinématique direct, et calculées sur un horizon temporel glissant. Dans ce cadre, nous avons utilisé une approche de commande éprouvée de l'automatique, la commande prédictive, correspondant naturellement à notre problème, tout en l'adaptant aux contraintes spécifiques de la robotique mobile.

Nous avons ainsi posé notre problème de navigation sous la forme d'un problème d'optimisation sous contraintes, dans lequel : la trajectoire de référence est calculée à partir du chemin à suivre par le robot ; l'écart entre cette trajectoire et celle prévue du robot constitue la fonction coût ; et enfin, les obstacles sont des contraintes supplémentaires à respecter.

La formalisation du problème de navigation sous cette forme par commande prédictive est un apport important par rapport aux méthodes par modèle directes existantes. En effet, elle nous permet d'utiliser des outils tels que les algorithmes d'optimisation, pour calculer la meilleure commande à appliquer.

Les simulations ont validé notre méthode en montrant son efficacité pour générer des trajectoires de contournement d'obstacle réalisables par le robot.

Comme nous l'avons souligné, ce travail s'inscrit dans le cadre d'une architecture de contrôle multi niveaux, et nous avons réalisé un navigateur performant permettant de suivre un parcours fourni par le planificateur de chemin, tout en respectant les contraintes du robot et de l'environnement dans lequel il évolue. Nous apportons une solution simple de développement et de mise en œuvre, adaptable à de nombreux types de robots.

En termes de perspectives, l'amélioration de notre méthode se situe principalement au niveau d'une intégration encore plus effective dans un système complètement autonome. En effet, l'adaptabilité de notre navigateur à tout type de robot et d'environnement peut encore être accrue, notamment en élaborant une bibliothèque de familles de trajectoires permettant d'adapter au mieux le comportement du robot à la situation à laquelle il est confronté : type de mission (suivi d'une route, exploration, phase de parking…) et type de milieu d'environnement (encombré ou non, terrain glissant…). L'algorithme d'optimisation peut aussi être amélioré, en travaillant au développement d'un moyen de combiner les avantages des méthodes déterministes (efficacité de la convergence en l'absence d'obstacles) à ceux des méthodes stochastiques (gestion des minima locaux induits par les obstacles).

De plus, l'étape nécessaire à l'obtention d'une autonomie complète est le développement du planificateur de chemin. L'idée est de lui donner la charge de déterminer une série de points de passage pour le robot, en s'appuyant sur une perception globale du milieu dans lequel le robot évolue. L'intégration de ces modules sur un robot permettra alors de valider expérimentalement notre approche.

References

[1] Adachi, Y.; Saito, H.; Matsumoto, Y.; Ogasawara, T.*, "Memory-based navigation using data sequence of laser range finder*", Computational Intelligence in Robotics and Automation, 2003. Proceedings. 2003 IEEE International symposium on Volume 1, Issue , 16-20 July 2003 Page(s): 479 - 484 vol.1

[2] Agirrebeitia, J., Aviles, R., de Bustos, I.F. and Ajuria, G. *, «A new APF strategy for path planning in environments with obstacles »* ; Mechanism and Machine Theory, Volume 40, Issue 6, June 2005, Pages 645-658

[3] J. M. AHUACTZIN LARIOS, « Le Fil d'Ariane: une méthode de planification générale. Application à la planification automatique de trajectoires» ; thèse soutenue le 29 septembre 1994, INRIA

[4] Alamir M.*,"Stabilization of Nonlinear Systems Using Receding-Horizon Control Schemes : A Parametrized Approach for Fast Systems*". Lecture Notes in Control and Information Sciences, Springer, London, ISBN 1-84628-470-8. 2006

[5] Alamir, M.*,"La commande predictive pour systemes non lineaires a dynamiques rapides",* in JNRR 07, 9-12 October 2007

[6] Andrews, J. R. and Hogan, N., "Impedance Control as a Framework for Implementing Obstacle Avoidance in a Manipulator." Control of Manufacturing Processes and Robotic Systems, Eds. Hardt, D. E. and Book, W., ASME, Boston, 1983, pp. 243-251.

[7] Arkin, R. C., "Motor Schema-Based Mobile Robot Navigation." *The International Journal of Robotics Research*, August 1989, pp. 92-112.

[8] Baille, G., Garnier, P., Mathieu, H., Pissard-Gibollet, R., "Le cycab de l'INRIA Rhône-Alpes", rapport technique Avril 1999

[9] Belker, T., Schulz, D. « Local Action Planning for Mobile Robot Collision Avoidance», I*ntelligent Robots and System, 2002. IEEE/RSJ International Conference on Volume 1, 30 Sept.-5 Oct. 2002 Page(s):601 - 606 vol.1*

[10] Bonnafous, D. Lacroix, S. Simeon, T. , « Motion generation for a rover on rough terrains", IROS 2001, Volume: 2, On page(s): 784-789 vol.2, Meeting Date: 10/29/2001 - 11/03/2001 Location: Maui, HI, USA ISBN: 0-7803-6612-3

[11] Brooks, R. A., "A Robust Layered Control System for a Mobile Robot." *IEEE Journal of Robotics and Automation*, Vol. RA-2, No. 1, 1986, pp. 14-23

[12] G. Campion, G. Bastin et B. D'Andréa-Novel. *Structural Properties and Classification of Kinematic and Dynamic Models of Wheeled Mobile Robots*. IEEE Transactions on Robotics and Automation, vol. 12, no. 1, pages 47–62, 1996.

[13] Canou J., Novales C., Poisson G., Marché P. Quick primitives extraction using inertia matrix on measures issue from an ultrasonic network ICRA 2001-IEEE, International Conference on Robotics and Automation, Séoul, Corée, 21-26 Mai, 2001

[14] J. Canou, G. Mourioux, C. Novales et G. Poisson A local map building process for a reactive navigation of a mobile robot Proceedings of IEEE International Conference on Robotics and Automation, pp. 4839-4844, ICRA 2004, New Orleans, Louisiane USA, 26 avril-1er mai 2004.

[15] J. Canou, D. Sallé, M. Traonmillin, V. Dupourqué: The Anglet Experiment: A Cybercar on the Beach. EUROCAST 2007: 1066-1072

[16] Cappelle, C. , El Badaoui El Najjar, M., Pomorski, D., Charpillet, F., *"Détection, suivi et géo-localisation d'obstacles à l'aide d'un modèle 3D géo-référencé, une caméra et un GPS : validation avec un lidar"*, Conférence Internationale Francophone d'Automatique, CIFA'08, Bucarest, Roumanie, 3-5 Septembre, 2008

[17] Chatterjee, R., Matsuno, F., *« Use of single side reflex for autonomous navigation of mobile robots in unknown environments »* ; Robotics and Autonomous Systems, Volume 35, Issue 2, 31 May 2001, Pages 77-96

[18] J. Courbon, Y. Mezouar, L. Eck, P. Martinet, "A generic framework for topological navigation of urban vehicle", ICRA09 Workshop on Safe navigation in open and dynamic environments Application to autonomous vehicles, ICRA09, Kobe, Japan, May 12th, 2009

[19] Courbon, J., Mezouar, Y., Lequievre, L., Eck, L., *"Navigation of urban vehicle: An efficient visual memory management for large scale environments"*, IROS 2008: 1817-1822

[20] Courtial, E. "Commande predictive et estimation d'état de systèmes non linéaires », rapport de thèse soutenue le 30 avril 1996, chapitre II, pp 38-41, Université Claude Bernard - Lyon 1

[21] Creuze, V., Parodi, O., Xiang X., «Design, Simulation and Experimental Results of Taipan 300, a New Autonomous Underwater Vehicle Prototype", published in OCEANS'09 IEEE, Bremen, Germany : (2009)

[22] Dufour, P., "Contribution à la commande predictive des systèmes à paramètres répartis non linéaires », rapport de thèse soutenue le 17 mai 2000, Université Claude Bernard - Lyon 1

[23] Ellouze, M. and Andr´ea-Novel, B.d'. 2000. Control of unicycle-type robots in the presence of sliding effects with only absolute longitudinal and yaw velocities measurement. European Journal of Control, 6(6):567–584.

[24] Er, M.J., Tien Peng Tan, Sin Yee Loh, *« Control of a mobile robot using generalized dynamic fuzzy neural networks »* ; Microprocessors and Microsystems, Volume 28, Issue 9, 2 November 2004, Pages 491-498

[25] P. Fabiani, V. Fuertes, A. Piquereau, R. Mampey, et F. Teichteil-Konigsbuch. Autonomous flight and navigation of VTOL UAVs : from autonomy demonstrations to out-of-sight flights. Aerospace Science and Technology, 11(2-3) :183–193, 2007

[26] Fliess, M., Lévine, J., Martin, P., and Rouchon, P. "*Flatness and defect of non-linear systems : introductory theory and examples"*. International Journal of Control, 61 :1327–1361, 1995

[27] Foret, J.; Bruneau, O.; Fontaine, J.G., "Unified approach for m-stability analysis and control of legged robots", 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. Volume 1, Issue , 27-31 Oct. 2003 Page(s): 106 - 111 vol.1

[28] Fraisse, P., Gil. A., Zapata, R., *« Stratégie de commande décentralisée d'une flottille de robots mobiles terrestres téléopérés. »*, JNRR 05, Guidel, October 5-7

[29] Fruchard, M., « Méthodologies pour la commande de manipulateurs mobiles non-holonomes », thèse soutenue le 23 septembre 2005 à l'école nationale supérieure des mines de paris - sophia antipolis , préparée à l'INRIA Antipolis

[30] Ghommam, J., Poisson, G., « Motion coordination control of multiple marine crafts », 10th IEEE International Workshop on Advanced Motion Control, 2008. AMC '08. Publication Date: 26-28 March 2008 On page(s): 44-49

[31] M.Y. IBRAHIM, A. FERNANDES, « Study on mobile robot navigation techniques », 2004 IEEE International Conference on Industrial Technology , Volume 1, 8-10 Dec. 2004 Page(s):230 - 236 Vol. 1

[32] Jetto, L., Longhi, S., Vitali, D., « Localization of a wheeled mobile robot by sensor data fusion based on a fuzzy logic adapted Kalman filter", Control Engineering Practice 7, 763-771, 1999

[33] Khatib, O., "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots." *1985 IEEE International Conference on Robotics and Automation*, St. Louis, Missouri, March 25-28, 1990, pp.500-505..

[34] M. AL-KHATIB, J. J. SAADE, « An efficient data-driven fuzzy approach to the motion planning problem of a mobile robot » ; Fuzzy Sets and Systems, Volume 134, Issue 1, 16 February 2003, Pages 65-82

[35] Kirkpatrick, S.; C. D. Gelatt, M. P. Vecchi (1983-05-13). "Optimization by Simulated Annealing". *Science*. New Series **220** (4598): 671–680. ISSN 00368075

[36] Klancar, G., Skrjanc, I., "Tracking-error model-based predictive control for mobile robots in real time", Robotics and Autonomous Systems, Volume 55 , Issue 6 (June 2007), Pages 460-469, ISSN:0921-8890, 2007

[37] Koren, Y.; Borenstein, J. **, "**Potential field methods and their inherent limitations for mobile robot navigation"; *Robotics and Automation*, Sacramento, California, April 7-12, 1991, pp. 1398-1404 vol.2

[38] Lebedev, D.V., Steil, J.J. and Ritter, H.J. « The dynamic wave expansion neural network model for robot motion planning in time-varying environments» ; *Neural Networks, Volume 18, Issue 3, April 2005, Pages 267-285*

[39] Lenain, R. Thuilot, B. Cariou, C. Martinet, P., "Rejection of sliding effects in car like robot control: application to farm vehicle guidance using a single RTK GPS sensor", Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. , 27-31 Oct. 2003 Volume: 4, On page(s): 3811- 3816 vol.3, ISBN: 0-7803-7860-1

[40] *Lenain* R., Thuilot B., Cariou C., Martinet P. "*High accuracy path tracking for vehicles in presence of sliding. Application to farm vehicle automatic guidance for agricultural tasks*." In Autonomous robots 21(1):79–97, 2006.

[41] Lettvin, J.Y., Maturana, H.R., McCulloch, W.S., & Pitts, W.H., *What the Frog's Eye Tells the Frog's Brain",* Proceedings of the IRE, Vol. 47, No. 11, pp. 1940-51, 1959

[42] LIAFA-Jussieu (Laboratoire d'Informatique Algorithmique: Fondements et Applications), « Courbes de Bézier et B-splines », http://www.liafa.jussieu.fr/~carton/Enseignement/InterfacesGraphiques/MasterInfo/Cours/Swing/splines.html,, septembre 2005

[43] V. J. LUMESKY, A. A. STEPANOV, « Path planning strategies for a point mobile automaton moving amist unknown obstacles for arbitrary shape ». Algoritmica, Vol 2, n° 4 (1987) pp 403-440

[44] Mayne D.Q.; Rawlings J.B.; Rao C.V; Scokaert P.O.M., *"vii. Constrained model predictive control: stability and optimality"*, Automatica, Vol. 36, No. 6. (June 2000), pp. 789-814

[45] J. MINGUEZ, L. MONTANO, « Sensor-based robot motion generation in unknown, dynamic and troublesome scenarios » ; Robotics and Autonomous Systems, Volume 52, Issue 4, 30 September 2005, Pages 290-311

[46] Marquardt , D., "An Algorithm for Least-Squares Estimation of Nonlinear Parameters". SIAM Journal on Applied Mathematics 11: 431–441, 1963

[47] Morette N., Novales C., Josserand L., *« Escape Lanes Navigator to control "RAOUL" Autonomous Mobile Robot »*, international conference on informatics in control, automation and robotics, Angers, France, May 9-12, 2007, pages 45-52

[48] Morette N., Novales C., Vieyres P., « Inverse versus Direct Kinematics Model based on Flatness and Escape Lanes to control CyCab Mobile Robot », IEEE-ICRA 2008, Pasadena, USA, May 19-23, 2008

[49] Morin, P. et Samson, C. « Trajectory tracking for non-holonomic vehicles : overview and case study ». Dans Kozlowski, K., éditeur, 4th Inter. Workshop on Robot Motion Control (RoMoCo), pages 139–153.

[50] Morin, P. et Samson, C. (2001d). « Practical stabilization of driftless systems on Liegroups ». Dans IEEE Conf. on Decision and Control (CDC), pages 4272–4277.

[51] Mourioux G., "Proposition d'une architecture multifonctions pour l'autonomie globale des robots », rapport de these soutenue le 22 novembre 2006, université d'Orléans

[52] Mourioux, G., Novales, C., Poisson, G. et Vieyres, P.: Omni-directional robot with spherical orthogonal wheels : concepts and analyses. Proceedings of IEEE International Conference on Robotics and Automation, ICRA 2006, Orlando, Floride USA, 15-19 mai 2006

[53] Novales, C. "Pilotage par actions reflexes et navigation locale de robots mobiles rapides", rapport de these soutenue le 20 octobre 1994, université Montpellier II

[54] Novales, C., Pallard, D., Laugier, C., «Controlling the Motions of an Autonomous Vehicle using a Local Navigator» ; ISRAM 1996, International Symposiumon RoboticsAnd Manufacturing, 27-30 mai 1996, Montpellier, France

[55] Ouadah, N., Azouaoui, O., Hamerlain, M., "Implémentation d'un contrôleur flou pour la navigation d'un robot mobile de type voiture », *Troisième Congrès francophone, Majecstic 2005, 16-18 Novembre 2005, Rennes (France).*

[56] Pin, F.G., Killough, M.S.*, « A New Familly of Omnidirectional and Holonomic Wheeled Platforms for Mobile Robots",* IEEE transactions on robotics and automation, Vol 10, N°4, August 1994

[57] Pivtoraiko, M., Kelly, A, and Rander R., "Efficient Braking Model for Off-Road Mobile Robots." Field and Service Robots, Ocober 2005, Port Douglas, Australia

[58] Pradalier, C., Hermosillo, J., Koike, C., Braillon, D., Bessière, P., Laugier, C., *"The CyCab: a car-like robot navigating autonomously and safely among pedestrians"*. Robotics and Autonomous Systems (RAS) 50(1):51-67 (2005)

[59] Pradel, G., Hoppenot, P., "Symbolic trajectory description in mobile robotics" - Journal of Intelligent & Robotics Systems, vol. 45, 157-180, 2006.

[60] Propoi, A. I., "Use of linear programming methods for synthesizing sampled-data automatic systems", Automation and Remote Control **24** 7 (1963), pp. 837–844.

[61] Pruski, A., "Robotique mobile, la planification de trajectoire", ED. Hermes, Paris 1996

[62] Qin S.J.; Badgwell T.A., *"A survey of industrial model predictive control technology",* Control Engineering Practice, Volume 11, Number 7, July 2003 , pp. 733-764(32)

[63] G. RAMIREZ, S. ZEGHLOUL, « Exponential control law for a multi-degree of freedom mobile robot » ; Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on Volume 1, 14-19 Sept. 2003 Page(s):509 - 514 vol.1

[64] Richalet, J., Rault, A., Testud, J.L., Papon, J.*, "Model Predictive Heuristic Control : Applications to Industrial processes"*, Automatica, 14, 413-428 (1978)

[65] E.D. SONTAG, « Mathematical Control Theory – Deterministic Finite Dimensional Systems », Editeur : Springer-Velag, New-York, 1990

[66] Tengda Sun, Jinfeng Wang, "A traffic cellular automata model based on road network grids and its spatial and temporal resolution's influences on simulation Simulation Modelling Practice and Theory", Volume 15, Issue 7, August 2007, Pages 864-878

[67] Tournassoud, P., "Planification et contrôle en robotique, application aux robots mobiles et manipulateurs", ED Hermes, Paris 1992

[68] Wang, X.; Hou, Z.; Zou, A.; Tan, M.; Cheng, L. , "A behavior controller based on spiking neural networks for mobile robots" Neurocomputing Volume 71 , Issue 4-6 (January 2008) Pages 655-666

[69] Wongngamnit, C., Angluin., D., "Robot localization in a grid", Information Processing Letters archive. Volume 77 , Issue 5-6 (March 2001) table of contents. Pages: 261 - 267

[70] Xu, W.L., « A virtual target approach for resolving the limit cycle problem in navigation of a fuzzy behaviour-based mobile robot » ; *Institute of Technology and Engineering, College of Sciences, Massey University, Palmerston North, New Zealand ;* Received 22 June 1998 ; accepted 20 August 1999

[71] Yang Cheng, Maimone, M.W., Matthies, L. *, "Visual odometry on the Mars exploration rovers - a tool to ensure accurate driving and science imaging",IEEE* Robotics & Automation Magazine, Volume 13, Issue 2, June 2006 Page(s):54 – 62

Annexe I : Intégration sur le CyCab

Nous présentons l'intégration technique du circuit de commande pour le véhicule Cycab, ainsi que les différents éléments qui composent le robot et leur principe de fonctionnement.

1. Les moteurs

Les 4 roues du véhicule sont motrices, et sont actionnées indépendamment les unes des autres, par des moteurs à courant continu de type MP 8165, équipés de codeurs incrémentaux HEDS-9140. Chaque moteur est alimenté par un hacheur de puissance, fonctionnant sous le 48V fourni par les batteries du circuit de puissance.

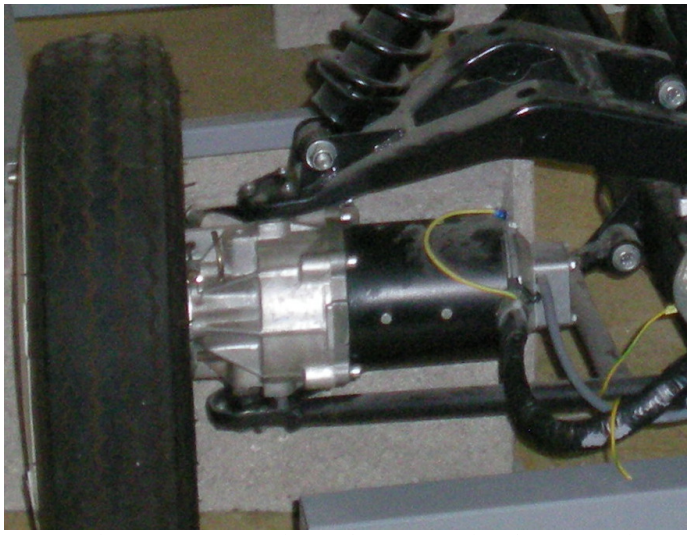

Figure 84- Moteur de traction avec codeur incrémental

2. Les hacheurs

Les hacheurs fournissent la tension nécessaire aux moteurs, suivant la vitesse et le sens de rotation souhaités. L'information sur la vitesse est fournie au hacheur par l'intermédiaire d'un signal PWM. Le signal PWM (pulse width modulation) est un signal en créneau dont le rapport cyclique α est proportionnel à la vitesse de rotation souhaitée. Le hacheur reçoit également un signal binaire lui indiquant le direction. Suivant le sens de direction souhaitée, la polarité de la tension fournie par le hacheur est inversée, de manière à ce que le moteur tourne dans le bon sens.

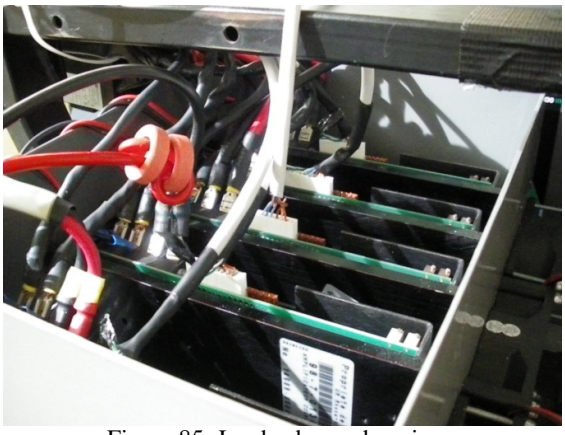

Figure 85- Les hacheurs de puissance

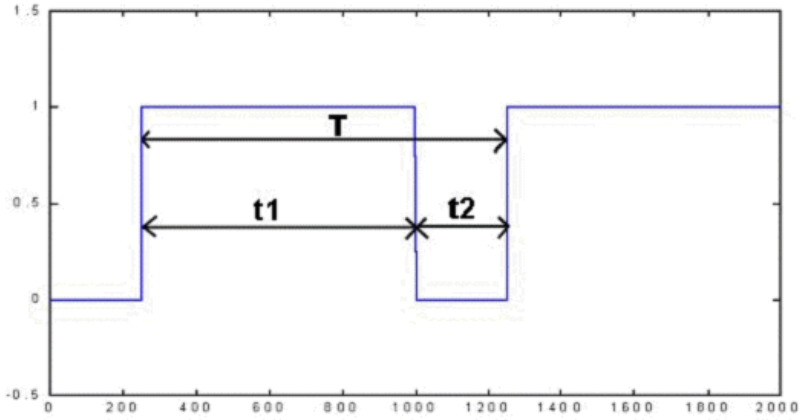

Figure 86-Le signal PWM

3. Les cartes d'axe

Chaque hacheur est piloté par une carte d'axe indépendante. Ces cartes d'axe sont des LM 629 CS fabriquées par la société Sidena. Chaque carte pilote un moteur de traction différent. Pour ce faire elles reçoivent les signaux des ch A et B des codeurs incrémentaux et génèrent des signaux PWM et direction qui sont envoyés aux hacheurs. Les 4 cartes sont fixées sur un même rail, le rail de traction, qui comprend en plus une carte d'alimentation et une carte LAN permettant d'assurer la liaison, sous protocole TCP-IP, avec l'ordinateur de contrôle.

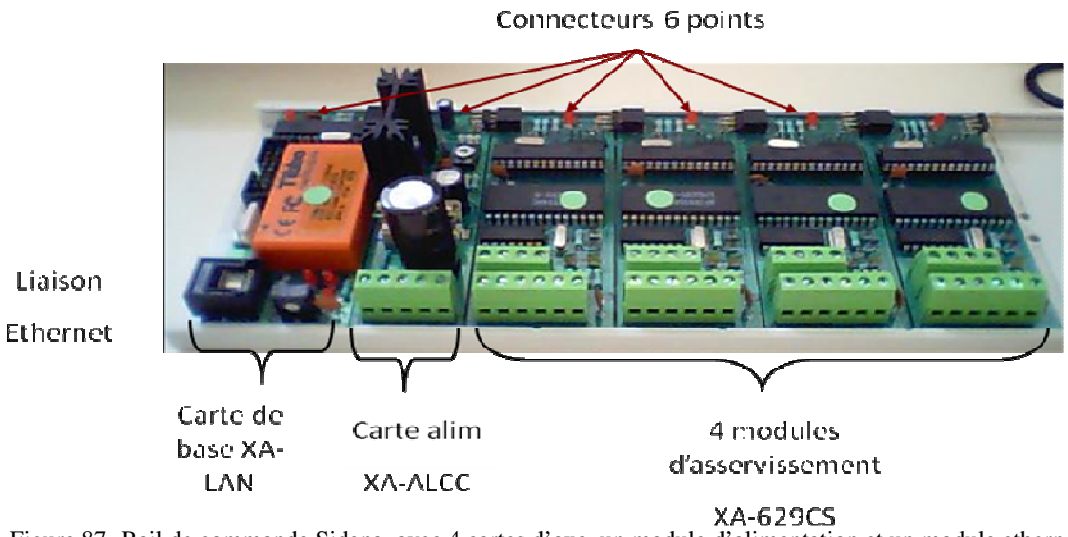

Figure 87- Rail de commande Sidena, avec 4 cartes d'axe, un module d'alimentation et un module ethernet

Pour le mécanisme de direction du véhicule, le volant et la crémaillère ont été remplacés par un vérin électrique. De manière similaire aux moteurs de traction, le vérin de direction est alimenté par un hacheur, lui-même piloté par une carte LM 629 Un codeur optique incrémental permet d'effectuer le contrôle le braquage, et un capteur de déplacement rectiligne de type REC-TIP12 de MCB industrie, un potentiomètre linéaire, positionné au dessus du vérin, sert de capteur de position absolue permettant de tenir compte de l'offset de position initiale du vérin. La mesure de ce capteur est lue par une carte d'entrée analogique positionnée sur un deuxième rail, le rail de direction, comprenant la carte d'axe correspondant au vérin de direction, une carte d'alimentation, une carte réseau et donc la carte d'entrée analogique.

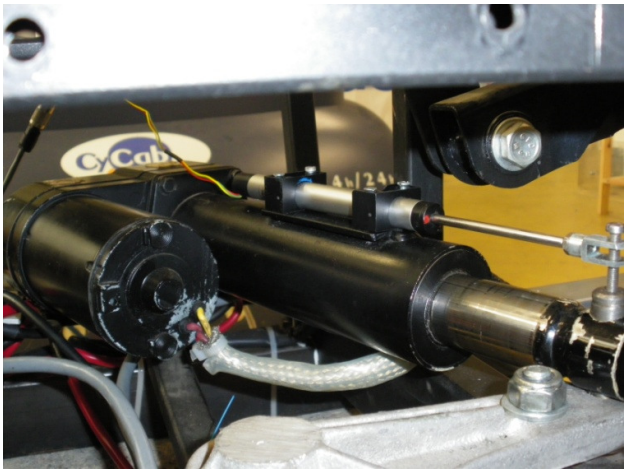
Figure 88- Vérin de direction équipé d'un potentiomètre linéaire

Chacun des deux rails (traction et direction) possède une adresse IP propre, lui permettant de communiquer avec le PC. La communication entre les cartes contenues dans un même rail, se fait par l'intermédiaire d'un bus I2C, qui sert à la fois de support d'alimentation et de communication à l'intérieur du rail.

Chaque roue est équipée d'un frein à tambour mécanique, commandés par un moteur de type essuie-glace de voiture. Ce moteur fonctionne en tout ou rien, avec deux positions remarquables : une ou il tire sur le câble, et une ou il ne tire pas.

L'architecture globale est présentée sur la figure Figure 89- Schéma général du véhicule.

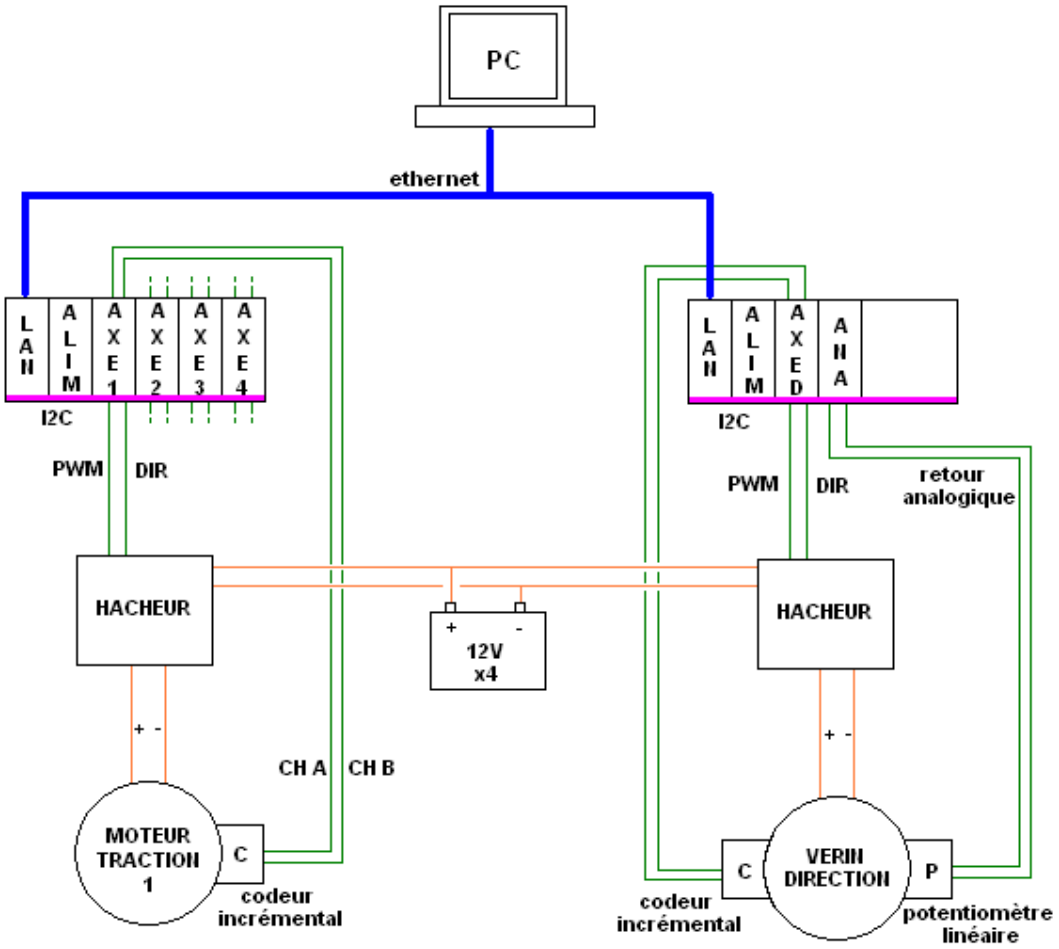

Figure 89- Schéma général du véhicule

Annexe II : Modèle inverse plat du CyCab

VI. Point de contrôle du robot

Figure 90 représente la structure mécanique du véhicule CyCab. Sur ce schéma, nous avons représenté la position d'un axe fixe virtuel, situé entre les essieux avants et arrières. Si l'on considère le cas d'un véhicule parfait évoluant sans glissement, cet axe virtuel passe par le centre instantané de rotation G. L correspond à l'empattement (distance entre les essieux avant et arrière), et l'essieu virtuel passe par le point C à une distance χL de l'essieu avant.

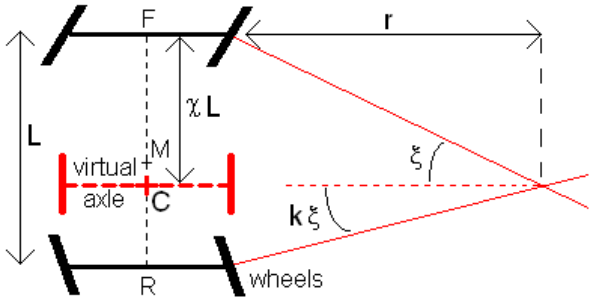

Figure 90 : Géométrie du CyCab

Géométriquement, on trouve :

$$\chi L = r \tan(\xi)$$
$$(1-\chi)L = r \tan(k\xi)$$

A partir de ces équations sur le position du point de contrôle C et de l'essieu virtuel fixe, nous étudions 3 différents cas : k = 0, k = 1 et 0 < k < 1.

VII. Cas k = 1 : même angle de braquage à l'avant qu'à l'arrière

Dans ce cas χ = 0.5 et le point de contrôle C reste fixe, sur le barycentre des 4 roues du véhicule (appelé M). Dans ce cas, on se retrouve avec un véhicule possédant des capacités de rotation doublées par rapport à une voiture classique:

$$\dot{x}_M = v.\cos\theta$$

$$\dot{y}_M = v.\sin\theta$$

$$\dot{\theta} = 2v.\tan\left(\frac{\xi}{L}\right)$$

x, y, θ et ξ désignent respectivement la position, l'orientation et le braquage du robot.

Un système plat est un système qui devient équivalent à un système linéaire si l'on utilise un bouclage endogène. Le modèle obtenu étant différentiellement plat, nous pouvons exprimer les entrées (v, ξ) du robot en utilisant des sorties plates z_1, z_2 suivantes et un nombre fini de leurs dérivées :

$$z_1(t) = x_C(t)$$
$$z_2(t) = y_C(t)$$

Nous obtenons le modèle cinématique inverse plat du robot :

$$\theta = \arctan\left(\frac{\dot{z}_2}{\dot{z}_1}\right)$$
$$v = \sqrt{\dot{z}_1^{\,2} + \dot{z}_2^{\,2}}$$
$$\xi = \arctan\left(L.\frac{\dot{z}_1\ddot{z}_2 - \ddot{z}_1\dot{z}_2}{2.\left(\dot{z}_1^2 + \dot{z}_2^2\right)^{\frac{3}{2}}}\right)$$

VIII. Cas k =0 : même angle de braquage à l'avant qu'à l'arrière

Dans ce cas nous retombons exactement sur le modèle de type voiture, dont le modèle inverse bien connu est le suivant :

$$\theta = \arctan\left(\frac{\dot{z}_2}{\dot{z}_1}\right)$$
$$v = \sqrt{\dot{z}_1^{\,2} + \dot{z}_2^{\,2}}$$
$$\xi = \arctan\left(L.\frac{\dot{z}_1\ddot{z}_2 - \ddot{z}_1\dot{z}_2}{\left(\dot{z}_1^2 + \dot{z}_2^2\right)^{\frac{3}{2}}}\right)$$

IX. Cas k ≠ 1

Dans ce cas, la position du point C n'est plus fixe sur le robot, elle varie en fonction de l'angle de braquage. χ devient une fonction trigonométrique de l'angle de braquage ξ. La relation entre le barycentre M et le point de contrôle C est alors donnée part :

$$x_C = x_M + (\chi - 0.5).L.\cos\theta$$

$$y_C = y_M + (\chi - 0.5).L.\sin\theta$$

L'expression du modèle inverse du véhicule devient :

$$\theta = \arctan\left(\frac{\dot{z}_2}{\dot{z}_1}\right)$$

$$v = \sqrt{\dot{z}_1^{\,2} + \dot{z}_2^{\,2}}$$

$$\xi = \arctan\left(\chi.L.\frac{\dot{z}_1\ddot{z}_2 - \ddot{z}_1\dot{z}_2}{\left(\dot{z}_1^2 + \dot{z}_2^2\right)^{\frac{3}{2}}}\right)$$

χ dépendant de ξ, il n'est plus possible d'exprimer les sorties plates générales du système. Cependant, la Figure 91 montre que pour des angles de braquage compris entre –30° et +30°, et 0.5 < k < 2, la position du point de contrôle C varie peu en fonction du braquage et dépend essentiellement de k et donc de la structure mécanique du robot.

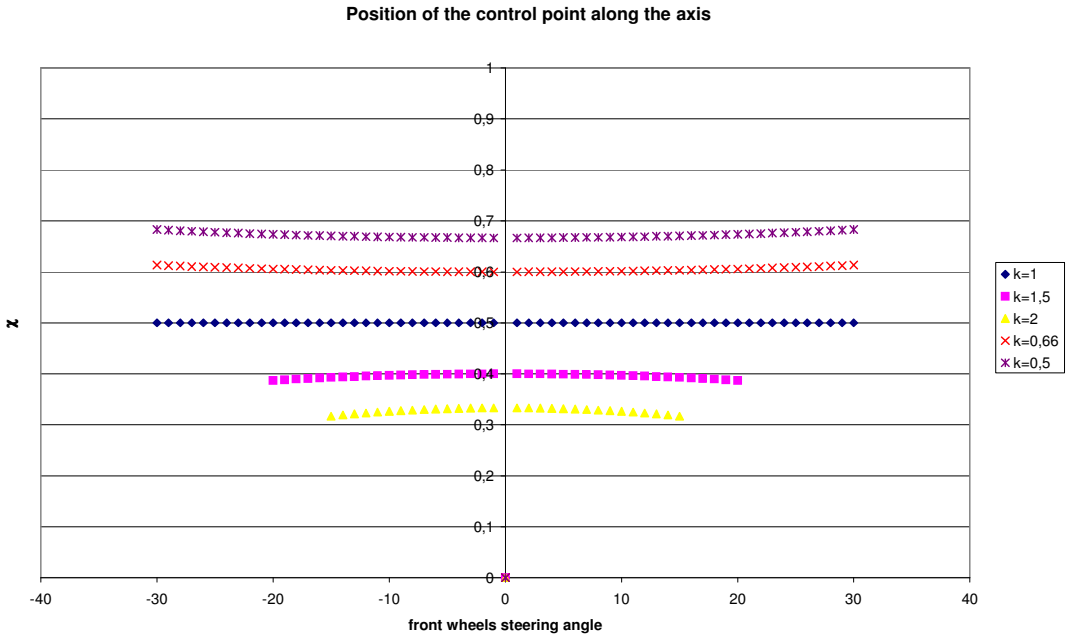

Figure 91 : position du point de contrôle C en fonction du braquage et de k

Comme nous observons de faibles variations de la position du point de contrôle, nous négligeons ses variations en fonction de l'angle de braquage et l'exprimons en fonction de k, en utilisant un développement limité :

$$\tan\xi = \xi + \sigma(\xi^3)$$

$$\chi = \frac{\xi}{\xi + k\xi} + \sigma(\xi^3)$$

$$\chi = \frac{1}{1+k} + \sigma(\xi^3)$$

Ainsi nous aboutissons au modèle inverse plat général pour les robots de type voiture à deux essieux directeurs :

$$\theta = \arctan\left(\frac{\dot{z}_2}{\dot{z}_1}\right)$$

$$v = \sqrt{\dot{z}_1{}^2 + \dot{z}_2{}^2}$$

$$\xi = \arctan\left(L.\frac{(\dot{z}_1\ddot{z}_2 - \ddot{z}_1\dot{z}_2)}{(k+1).\left(\dot{z}_1^2 + \dot{z}_2^2\right)^{\frac{3}{2}}}\right)$$

X. Validation du modèle

Pour valider ce modèle, nous l'avons confronté au modèle cinématique direct du robot. A partir de trajectoires obtenues lors d'une simulation de navigation par modèle direct et lignes de fuite, nous avons recalculé les entrées $(\hat{v}, \hat{\xi})$ correspondantes grâce au modèle cinématique inverse. Nous avons ensuite comparé ces entrées avec celles injectées en entrée du modèle cinématique.

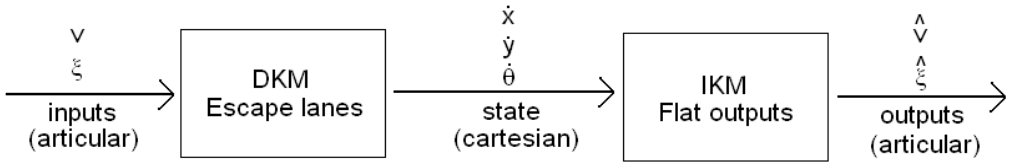

Figure 92 : principe de la comparaison

montre les résultats obtenus pour k = 2. Les vitesses v et $\hat{v}$ au long de la navigation y sont représentées, et montre l'écart entre les deux vitesses.

Nous avons observé que l'écart entre les deux vitesses restait faible, et que l'approximation sur la position du point C constant en fonction de l'angle de braquage ne générait pas d'erreurs importantes (inférieures à 0.3 $m.s^{-1}$ dans le pire des cas, correspondant aux grandes accélérations / décélérations).

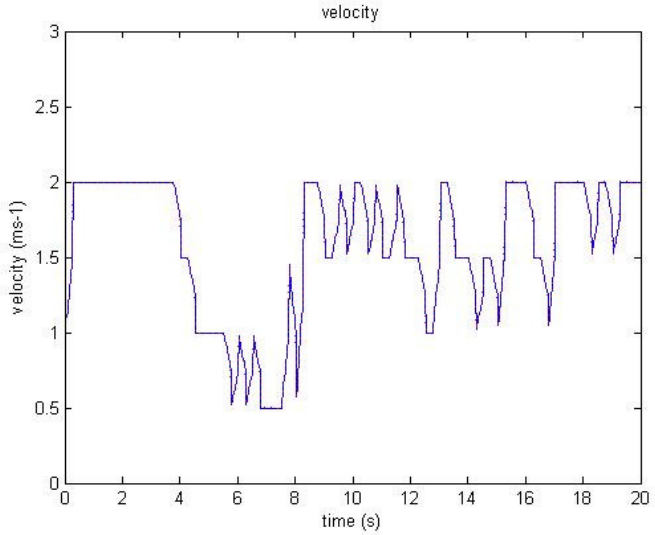

Figure 93 : Evolution de la vitesse calculée le long du parcours

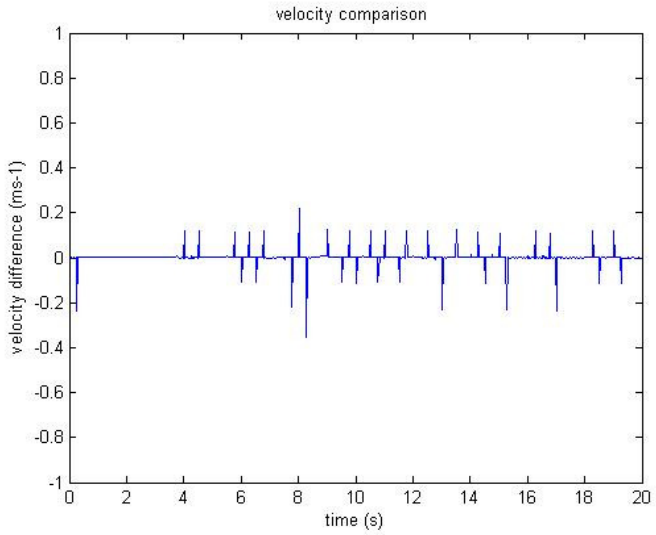

Figure 94 : Ecart entre la vitese injectée en entrée de MCD et calculée par le MCI

Annexe III : Etude de différents algorithmes d'optimisation

XI. Principe des algorithmes

Un algorithme d'optimisation est un processus cherchant à optimiser une fonction critère (c'est-à-dire la minimiser ou la maximiser), en déterminant les paramètres d'entrée correspondants. Par exemple, pour une fonction z = f(x,y), l'algorithme cherchera à déterminer le couple (x,y) minimisant z.

Il n'existe pas d'algorithme permettant de déterminer à coup sûr le minimum global d'une fonction (en dehors de l'énumération exhaustive des couples d'entrée). On distingue principalement deux types d'algorithmes : les méthodes déterministes, et les méthodes stochastiques (c'est-à-dire probabilistes).

Nous avons cherché à déterminer quel type d'algorithme nous permettrait d'obtenir les meilleures performances pour notre navigateur par commande prédictive. Nous avons étudié un algorithme de chacune des deux catégories mentionnées : l'algorithme Levenberg-Marquardt pour le déterministe, et l'algorithme Recuit Simulé pour le stochastique.

1. Levenberg Marcquardt

LM est un algorithme d'optimisation déterministe, il trouvera toujours la même solution pour un problème donné. C'est une méthode qui utilise les informations contenues dans la fonction critère, à savoir le hessien et le gradient de la fonction, pour converger le plus efficacement possible vers le minimum *local* de la fonction.

Les avantages de cet algorithme sont :
- la convergence vers la solution **exacte** du problème
- la très grande rapidité à converger (en 2/3 itérations on est très proche de la solution optimale)

Les inconvénients :
- très grande sensibilité aux problèmes de minima locaux, donc résultats tributaires de l'initialisation de l'algorithme
- peu d'itérations pour converger certes, mais une itération demande un grand nombre de calculs (gradient et hessien numérique, inversion de matrice)

2. Recuit simulé

RC est un algorithme stochastique, c'est-à-dire qu'il est basé sur une part d'aléatoire et trouvera une solution différente si on lui représente le même problème. Le principe de cet algorithme s'inspire du principe physique du recuit des matériaux solides. Le chauffage du matériau à très haute température, puis son refroidissement lent conduisent à une réorganisation plus

ordonnée des atomes constituant le matériau, et l'amenant ainsi vers un état plus stable (énergie plus faible).

A partir d'une position (x,y) initiale, l'algorithme va passer de manière aléatoire à une position proche. Si le critère associé à cette position est plus faible, on conserve cette position. Si le critère est plus élevé, on pourra tout de même conserver cette position, mais selon une loi de probabilité décroissante au fil du temps (mimant ainsi le refroidissement d'un matériau). La probabilité de passage vers un critère plus élevé suit une formule du type :

$$P = 1 - \exp(-\frac{\Delta J}{T})$$

Dans cette formule, ΔJ est l'écart entre le nouveau critère et l'ancien, et T joue le rôle d'une « température » qui diminue au fil du temps, selon une loi de refroidissement qu'il faut régler empiriquement. Ainsi au début du processus, de nombreux mouvements vers des positions associées à un critère plus élevé seront tolérés, permettant à l'algorithme d'explorer plus de régions et ainsi ne pas s'enfermer dans un minimum local.

Au fur et à mesure de l'évolution du processus d'optimisation, ces transitions vers des critères plus élevés seront de moins en moins permises, ce qui contraint l'algorithme à converger vers un minimum.

Les avantages de cet algorithme sont :
- une grande capacité à éviter les minima locaux et à converger vers le minimum global
- il n'y a pas de condition particulière sur les particularités de la fonction cout

Les inconvénients sont :
- l' algorithme converge vers une solution approchée et non une solution exacte
- vitesse de convergence plus faible au voisinage du minimum global

XII. Test de l'algorithme Levenberg-Marquardt

1. Tests avec une grille d'occupation

Le principe de la « grille d'occupation » est le suivant : on rajoute tout simplement une pénalité « infinie » à la fonction z pour un certain nombre de cases de la grille (représentées en verte). En dehors de ces cases on ne rajoute aucune pénalité. Par pénalité infinie on entend une valeur largement supérieure à n'importe quelle valeur de z dans le domaine de recherche.

a) Minimum hors contrainte

Dans ce test, on demande à l'algorithme de trouver le minimum de la fonction, qui est en dehors de l'espace des contraintes. La difficulté est que l'on l'initialise de telle manière qu'entre le point initial de l'algorithme, et le point minimum, il y a la région des obstacles.

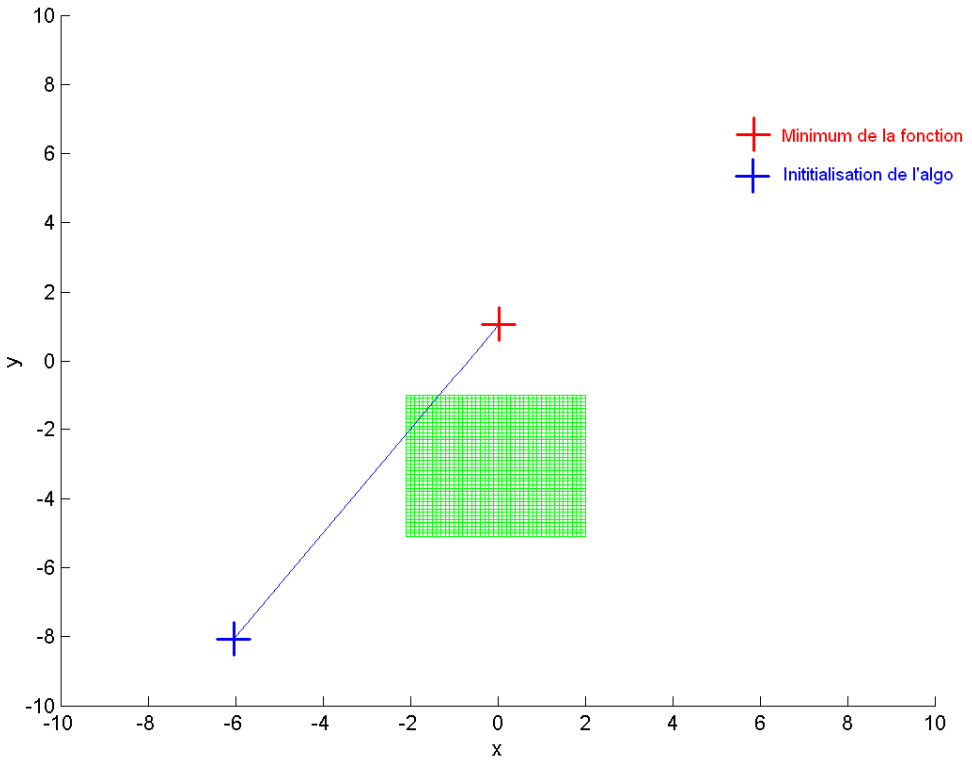

Figure 95

Dans ce test (Figure 95), on voit que l'algorithme a bien fonctionné : il a réussi à trouver le minimum global de la fonction, malgré la présence de contraintes sur son chemin de recherche. (on rappelle qu'il ne s'agit pas d'une trajectoire de robot, peu importe que le « chemin » fait par l'algorithme passe dans les contraintes, ce qu'on veut c'est qu'il trouve le minimum de la fonction).

En navigation, ce type de cas intervient lorsque la trajectoire idéale du robot (celle qui se rapproche le plus de la trajectoire de référence) ne heurte pas les obstacles, mais que par contre l'algorithme est optimisé de « mauvaise » manière. Donc à priori, quelque soit la manière dont il est optimisé, l'algorithme trouvera la solution optimale si celle-ci est en dehors des contraintes.

b) Minimum dans les contraintes, bonne initialisation de l'algorithme

Dans ce test, le minimum de la fonction z est situé dans le domaine des contraintes. En navigation, ce cas correspond à celui où la trajectoire de référence heurte les obstacles. (la trajectoire de référence est calculée on le rappelle sans tenir comptes des obstacles mais uniquement par rapport aux points fournis par le planificateur et la position courante du robot). Dans ce cas la mission de l'algorithme serait de déterminer la trajectoire qui se rapproche le plus de la trajectoire de référence, mais sans heurter les obstacles. Dans notre test, cela correspond à trouver le minimum de la fonction qui soit en dehors des contraintes.

Sur le test, on voit que cette mission a bien été remplie (Figure 96). Par contre le point d'initialisation de l'algorithme lui était plutôt favorable pour trouver ce minimum (il ne traversait pas les grilles occupées).

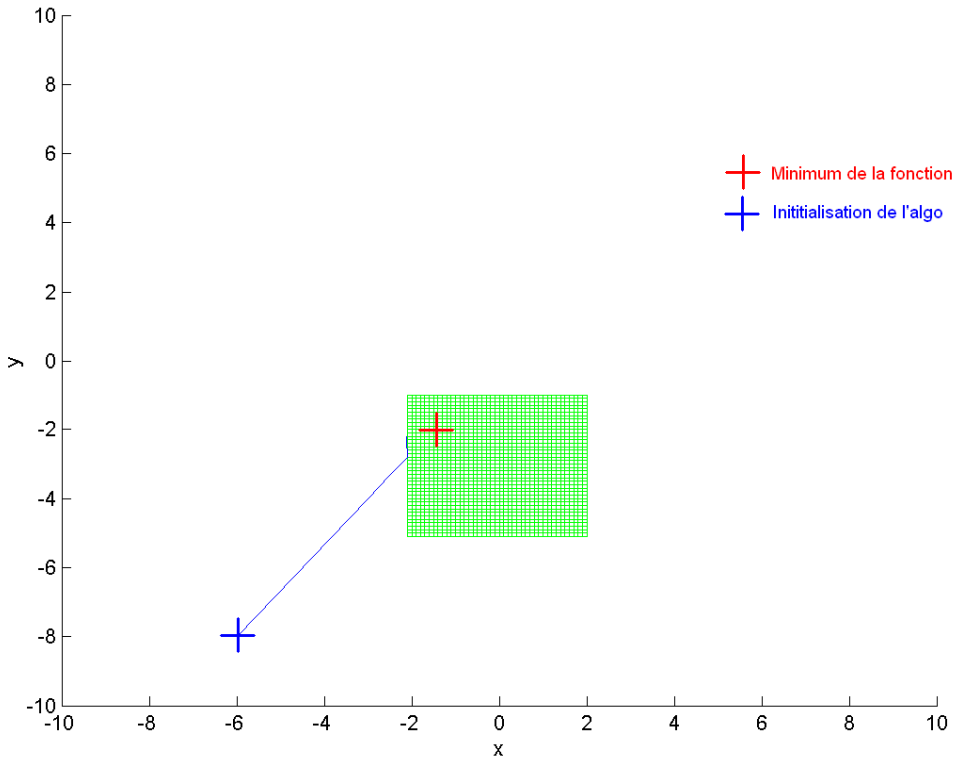

Figure 96

c) Minimum dans les contraintes, et mauvaise initialisation

Ce problème est le même que le précédent, sauf que cette fois ci on a volontairement mal initialisé l'algorithme. Cette situation correspond à celle où en navigation, on se présente face à un obstacle : la trajectoire de référence passe à travers l'obstacle, et l'algorithme ne part pas sur uns solution de contournement, mais sur une ligne droite (cas classique, on initialise toujours l'algorithme avec un braquage nul).

Dans ce cas on voit que l'algorithme a échoué (Figure 97), il n'a pas réussi à trouver la même solution que dans le cas 2. Il est tout simplement tombé dans un minimum local. Dans ce cas, si il s'agissait de guider le robot, l'algorithme lui dirait d'aller jusqu'à l'obstacle, et de s'y arrêter. C'est donc un échec.

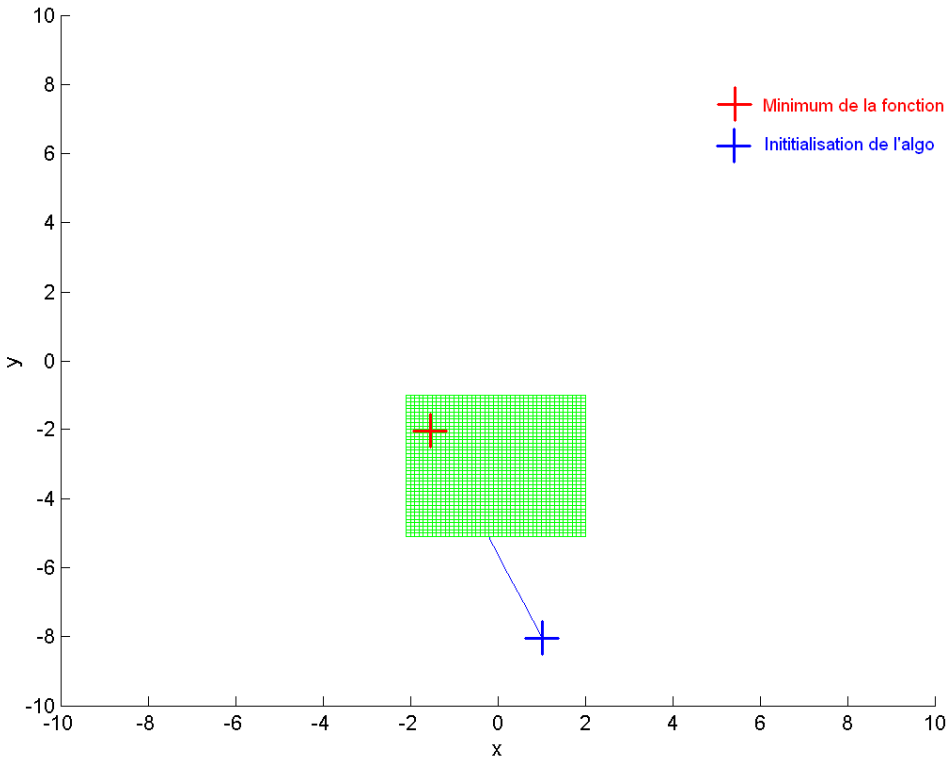

Figure 97

d) Conclusion

Dans le 3e cas, si on cherche à savoir pourquoi l'algorithme n'a pas « contourné » l'espace des contraintes, on peut penser que cela est dû à la nature discrète des contraintes. En effet, vu que l'algorithme se base sur le calcul du Hessien et du gradient pour se guider vers le minimum, la discontinuité introduite par ce type de contrainte peut poser problème. C'est ce que l'on a cherché à vérifier en travaillant sur des contraintes continues dans la prochaine série de tests.

2. Tests avec des contraintes continues

Ici on traite les contraintes comme des obstacles segments, c'est-à-dire qu'on calcule pour chaque point (x,y) la distance d entre ce point et l'obstacle le plus proche, et on pénalise la fonction z par une fonction hyperbolique :

$$z = (x-a)^2 + (y-b)^2 + \frac{1}{d}$$

Pour toutes les points contenus à l'intérieur de l'obstacle, on fixe la distance d à une valeur « infiniment » petite.

a) 1) Minimum hors contrainte

Le test est le même que dans le 1.1 : trouver le minimum de la fonction, qui est en dehors de l'espace des contraintes, en partant d'un point « mal placé ».

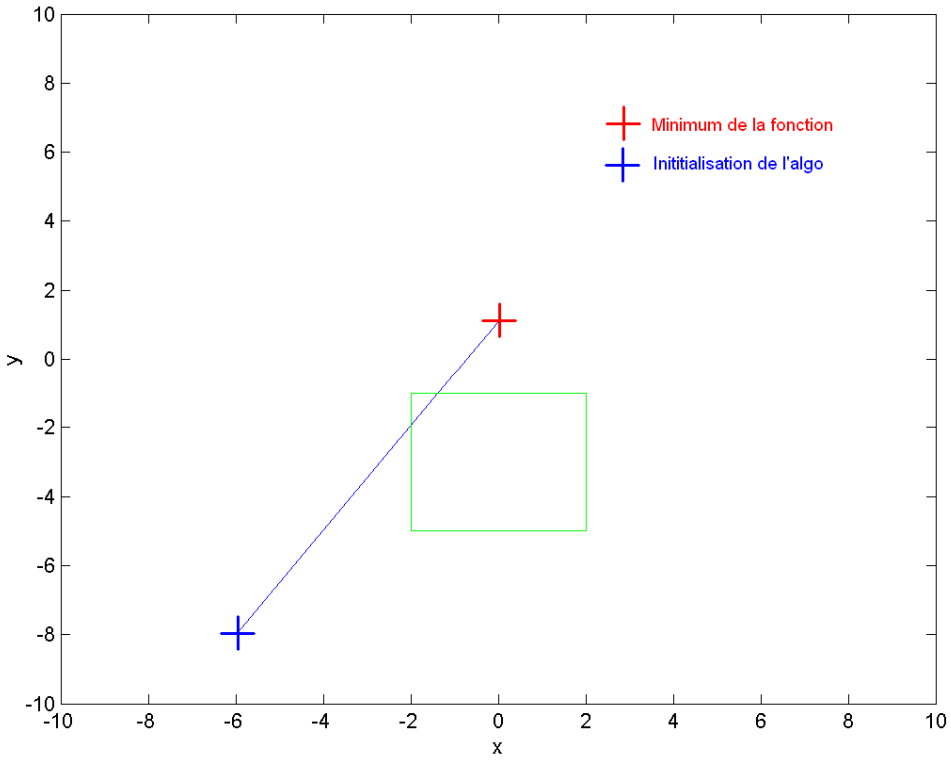

Figure 98

Les résultats obtenus sont les même qu'avec la grille d'occupation : l'algorithme parvient à trouver le minimum local de la fonction z (Figure 98).

b) 2) Minimum dans les contraintes, bonne initialisation de l'algorithme

Ce test est le même que dans la section 1.2. Le minimum de la fonction z est situé dans le domaine des contraintes. Il s'agit donc de trouver le minimum de la fonction qui soit en dehors des contraintes. L'objectif a été atteint (Figure 99), même si on remarque que le point trouvé est moins proche du vrai minimum local. Cela est dû à la pénalité qui devient trop forte dès lors qu'on s'approche des contraintes

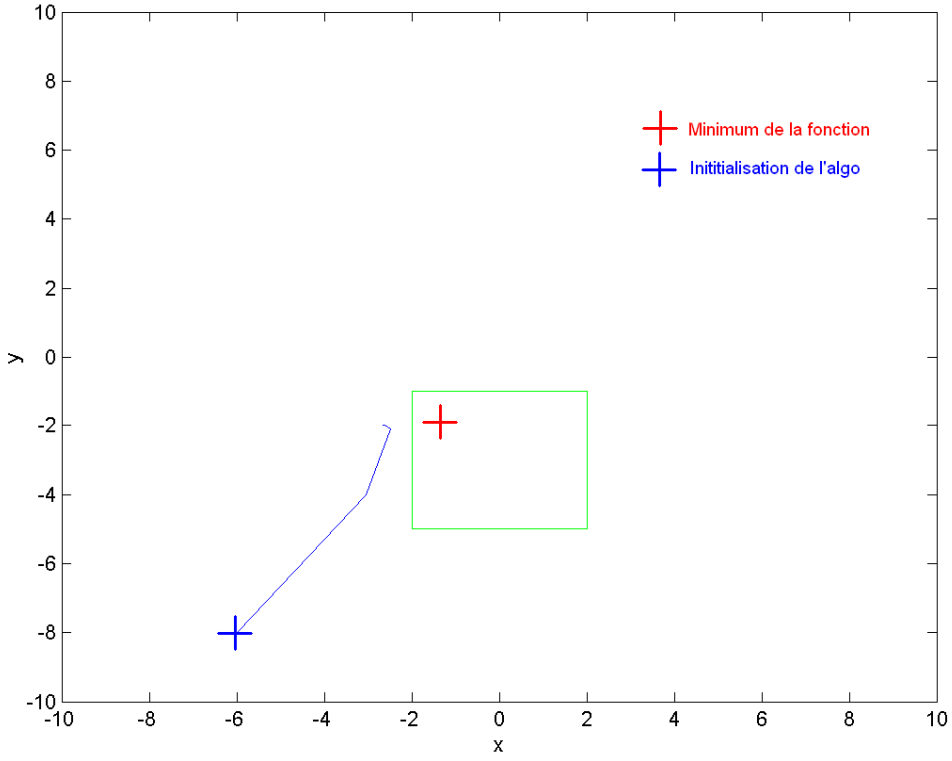

Figure 99

c) 3) Minimum dans les contraintes, et mauvaise initialisation

Ce problème est le même que dans la section 1.3, c'est-à-dire même problème que le précédent mais avec une mauvaise initialisation. L'algorithme n'a pas mieux réussi à se rapprocher du minimum local, malgré la continuité dans la fonction à optimiser (Figure 100).

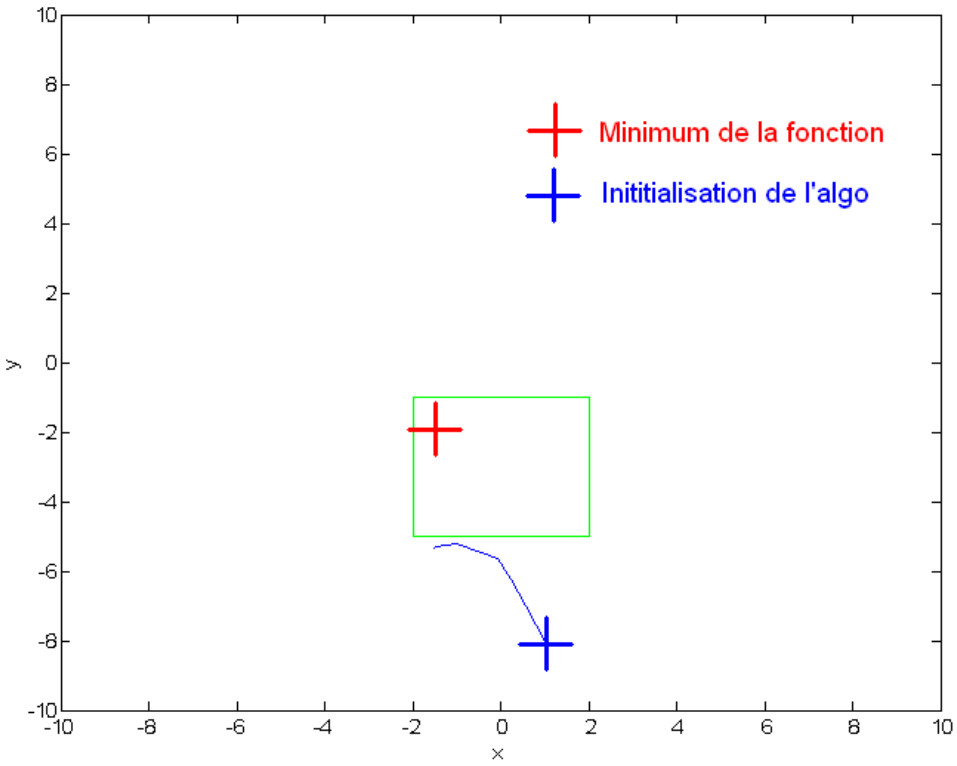

Figure 100

d) Conclusion

Cette série de tests a permis de démontrer que l'algorithme Levenberg-Marquardt n'était pas capable de converger correctement vers le minimum global de la fonction, dans le cas où le minimum de la fonction cout non pénalisée par les contraintes appartenait au domaine des contraintes. Ce problème correspond, en navigation, à celui où la trajectoire de référence passe au milieu des obstacles. Ces simulations ont permis de montrer que dans ce genre de situation, l'algorithme pourra ou non trouver le minimum global, qui correspond à une solution de contournement d'obstacle, suivant la manière dont on l'initialise.

XIII. Recuit simulé

L'algorithme de recuit simulé a été soumis au test numéro 3 du rapport sur l'algorithme Levenberg Marquardt (« minimum » de la fonction situé dans les contraintes, et mauvaise initialisation). Dans ce test, nous avons vu que l'algorithme LM s'enfermait dans un minimum local, quelque soit manière d'introduire les contraintes (méthode continue ou méthode discrète).

1. 1 obstacle-contrainte

Comme il s'agit d'une méthode probabiliste, pour que le test soit significatif il faut faire un grand nombre d'essais et donner les résultats sous forme statistique. Nous avons donc effectué 1000 tirages successifs du même problème, pour regarder les solutions obtenues. Un exemple de résultat du processus d'itération est montré sur la Figure 101.

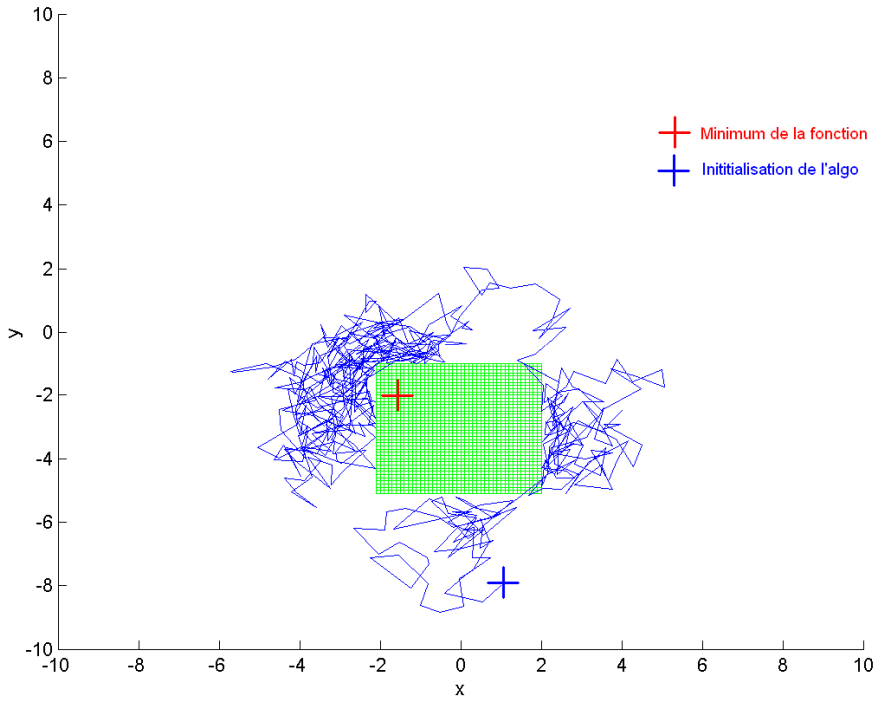

Figure 101 - Recherche du minimum global par l'algorithme

Le graphe ci-dessus représente les déplacements de l'algorithme lors de sa recherche dans l'espace (x ,y). Ce graphe est difficile à lire tel quel, ce que l'on remarque surtout c'est que l'algorithme a visité un peu partout au cours de sa recherche.

Les graphes ci-dessous (Figure 102, Figure 103 et Figure 104) permettent de mieux voir la manière dont l'algorithme a convergé : l'évolution du critère, de x et de y sont représentés en fonction des itérations de l'algorithme.

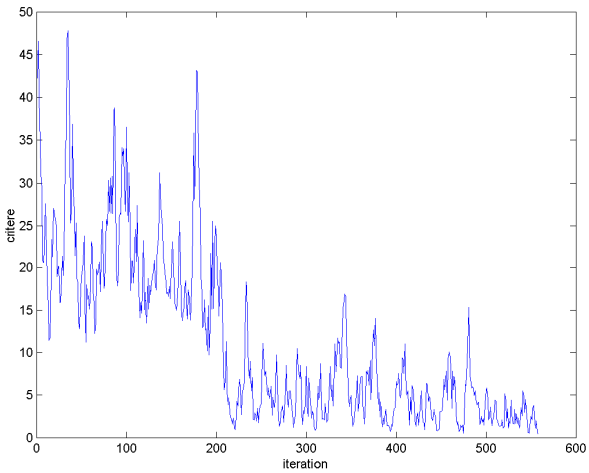

Figure 102 - Evolution du critère en fonction des itérations

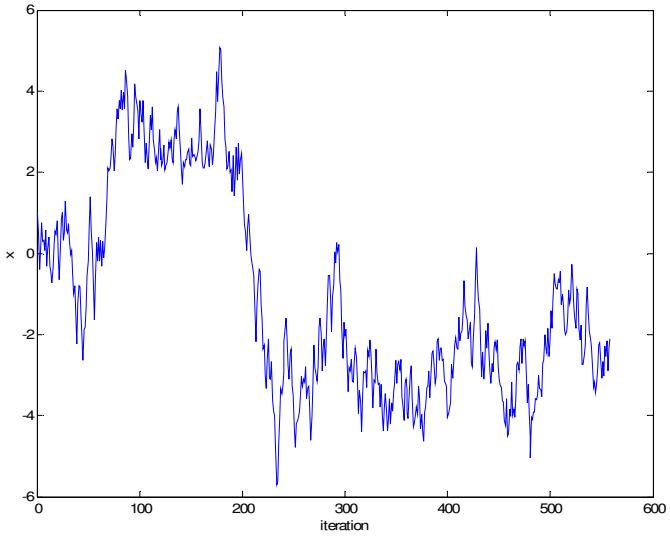

Figure 103 - Evolution de x

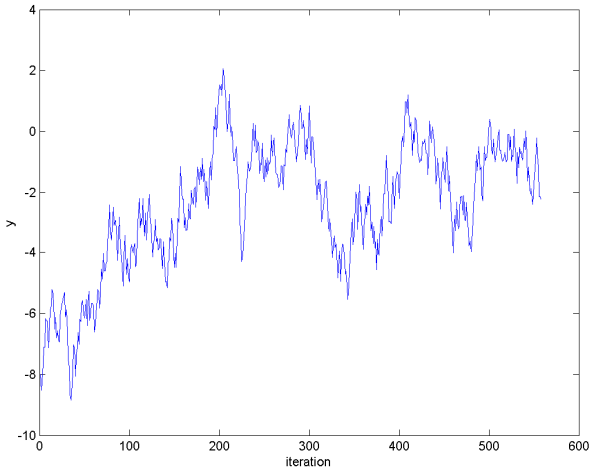

Figure 104 - Evolution de y

Ces résultats permettent de constater que malgré le désordre apparent dans la manière de l'algorithme d'avancer, en réalité il s'est bien déplacé vers la solution optimale située en (-1.5, -2).

Sur un grand nombre d'essais (1000), on a obtenu les résultats suivants :

- 98.4% des essais ont trouvé le minimum global de la fonction
- 75.0% des essais ont trouvé le minimum avec une précision inférieure à 1 sur x et sur y (donc 23% ont trouvé le minimum global mais sous une précision assez mauvaise).

Bien sûr ces résultats sont tributaires de la manière de régler l'algorithme (notamment la loi de refroidissement). Cependant cela montre que cet algorithme est efficace pour déterminer un minimum global sur ce type de problème (là où l'algorithme LM tombait dans les minima locaux).

2. 2 obstacles-contraintes

Nous avons effectué des tests sur l'algorithme en rajoutant un second obstacle dans le plan des solutions admissibles. La fonction à minimiser est z=(x+4)*(x+4)+(y-6)*(y-6), et on regarde, sur

1000 tirs, en combien d'itérations en moyenne l'algorithme trouve la solution optimale, sous une précision donnée.

Un exemple de résultat est donné sur la Figure 105, et montre la convergence de la fonction z sur cet exemple (Figure 106).

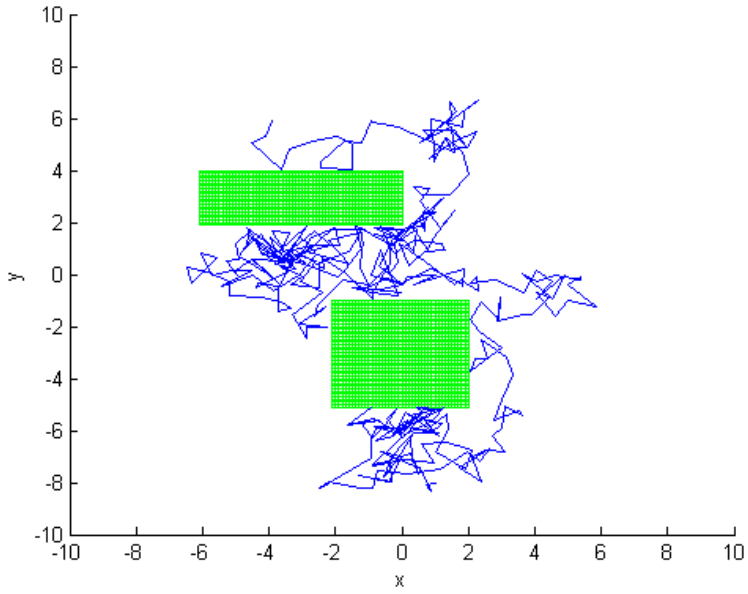

Figure 105

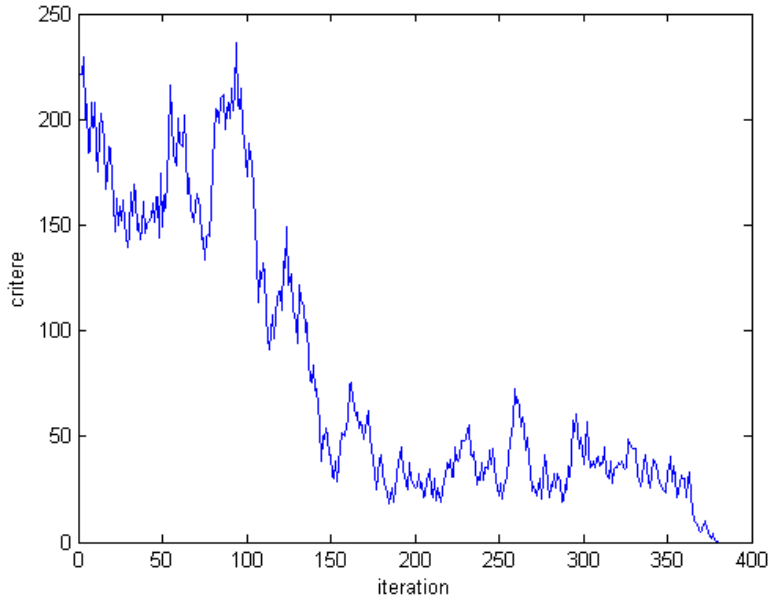

Figure 106

Sur les 1000 essais, nous avons obtenu une moyenne de 628 itérations pour atteindre la solution globale. Nous avons comparé ce résultat au nombre d'essais obtenus pour le même scénario mais avec 1 seul obstacle : cette fois ci la moyenne était de 471 itérations pour trouver la solution globale.

Ces résultats montrent que même avec deux contraintes venant casser la convexité de la fonction à minimiser, l'algorithme recuit simulé parvient à trouver le minimum global. Cependant, le nombre d'itérations (et donc le temps de calcul) nécessaire pour atteindre le résultat optimal avec la même précision augmente de 33%.

XIV. Tests de génération de trajectoire et navigation

Tout le problème de navigation est contenu dans le critère Z_{tot} qu'il faut minimiser. Le travail de l'algorithme d'optimisation est de trouver l'ensemble des paramètres p_{1v}, p_{2v}, $p_{1\xi}$ et $p_{2\xi}$ minimisant ce critère. Cela revient à déterminer la commande u permettant de générer le mouvement du robot le plus proche de la trajectoire de référence, et respectant les différentes contraintes.

Les algorithmes d'optimisation sous contraintes peuvent être classés en deux catégories. Les algorithmes déterministes présentent généralement les meilleures propriétés de convergence, et permettent d'obtenir des solutions plus précises et plus rapidement. Par contre, ils sont très sensibles aux problèmes des minima locaux. Or l'introduction de contraintes discrètes dues aux obstacles a justement pour effet de casser la convexité de la fonction coût et d'introduire des minima locaux.

La seconde catégorie d'algorithme est celle des algorithmes stochastiques. Ces algorithmes sont basés sur des approches probabilistes. Ils présentent l'inconvénient d'être moins précis et moins rapides à converger que les algorithmes déterministes (leur convergence n'est d'ailleurs pas toujours prouvée). Par contre, ils possèdent l'avantage d'être beaucoup moins sensibles aux problèmes de minima locaux, ce qui leur permet, dans le cadre de problèmes de navigation en milieu encombré, de trouver des trajectoires de contournement d'obstacle qu'un algorithme de type déterministe ne saurait pas trouver.

Pour illustrer cela, la Figure 107 présente une situation dans laquelle le robot est face à un obstacle en U. La trajectoire de référence ne tient pas compte de la position de l'obstacle, et passe en plein milieu de celui-ci. Dans l'espace paramétrique, cela engendre un minimum local. Suivant la manière dont sont initialisés les paramètres, l'algorithme déterministe tombe dans ce minimum local (figures de gauche), alors que l'algorithme stochastique lui parvient à trouver les paramètres optimaux quelque soient les paramètres initiaux, permettant au robot de trouver une trajectoire de contournement d'obstacle. La trajectoire complète de contournement est représentée sur la Figure 108.

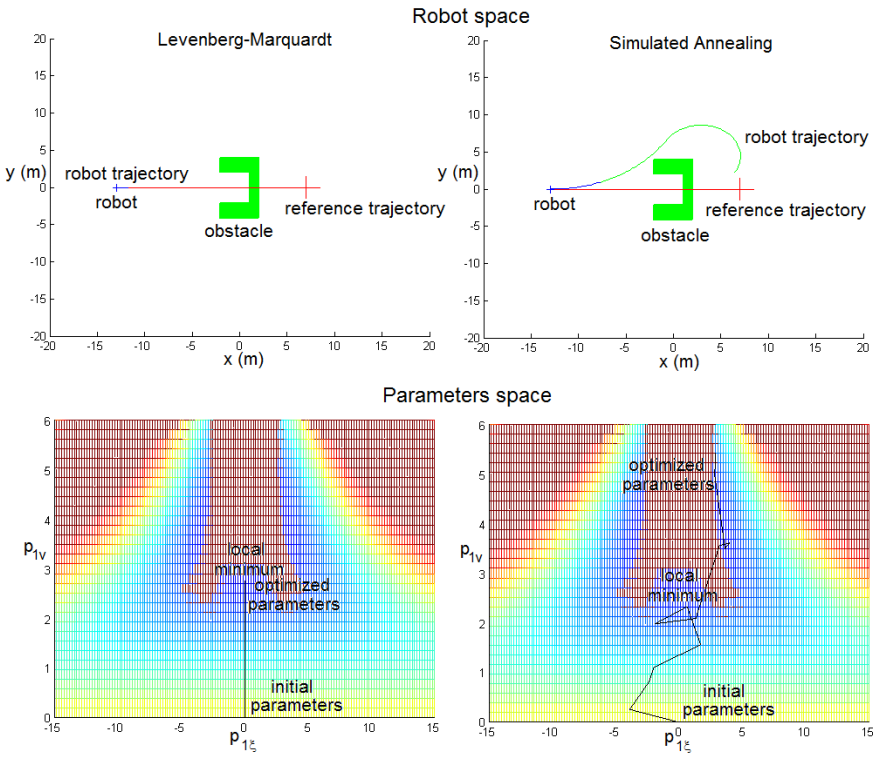

Figure 107

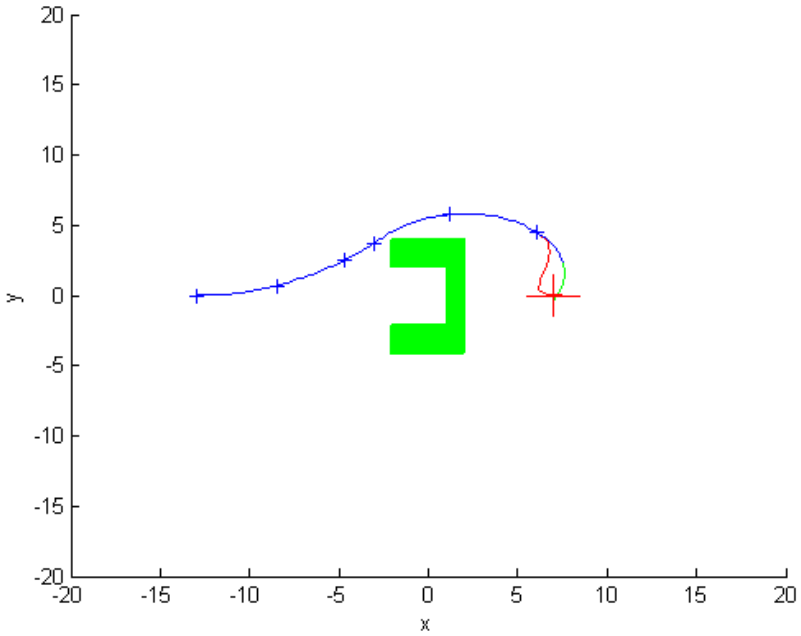

Figure 108

Le choix du type d'algorithme est donc dépendant du type de terrain d'évolution (dégagé ou obstrué), et de la confiance que l'on a dans le planificateur concernant sa prise en compte des obstacles dans la génération de chemin. Pour l'application du navigateur en milieu encombré, nous avons utilisé un algorithme stochastique, l'algorithme recuit simulé, de manière à tester la robustesse du navigateur à une planification incomplète.

XV. Discussion

Ces résultats montrent qu'un algorithme tel que le recuit simulé est capable de résoudre la problématique du contournement d'obstacles. En effet le problème que nous avons posé ici aux algorithmes, semble proche de celui du contournement d'obstacle avec une trajectoire de référence qui passerait dans les obstacles : une trajectoire « optimale » (celle qui suivrait le mieux la trajectoire de référence) qui n'est pas admissible car heurtant les obstacles (donc minimum de la fonction ne respectant pas les contraintes).

Cependant, une combinaison des deux types de méthodes pourrait améliorer encore d'avantage les résultats, soit en utilisant les deux successivement (premières itérations en RC, et affinage en LM), mais dans ce cas les temps de calculs seraient doublés ; soit en switchant l'algorithme d'optimisation suivant le type de terrain d'évolution (encombré d'obstacles ou non). On encore en activant l'algorithme RC lorsque le robot a détecté qu'il se trouvait dans une impasse.

Nicolas MORETTE

Contribution à la Navigation de robots mobiles : approche par modèle direct et commande prédictive

Résumé :

L'autonomie d'un robot mobile autonome requiert la réalisation coordonnée de tâches de commande et de perception de l'environnement. Parmi celles-ci, la navigation joue un rôle de pivot dans l'interaction du robot avec son terrain d'évolution. Elle consiste en la détermination de trajectoires réalisables par le robot pour suivre un chemin préétabli, tout en assurant la non collision avec les obstacles, mobiles ou fixes. Pour effectuer cette tâche, notre approche s'appuie sur le modèle cinématique direct du véhicule pour générer des trajectoires admissibles par le robot. En premier lieu, une trajectoire de référence est construite à partir du chemin à suivre. Le problème de navigation est alors modélisé sous la forme d'un problème d'optimisation sous contraintes dont la fonction coût quantifie l'écart entre la trajectoire prédite du robot et la trajectoire de référence. Les obstacles sont intégrés sous forme de contraintes en pénalisant le critère, et sa minimisation détermine la commande optimale à appliquer. Cette navigation par commande prédictive nous permet d'anticiper les mouvements de contournement d'obstacles sur l'horizon de prédiction choisi, tout en gardant une certaine réactivité vis-à-vis de la dynamique des obstacles et du robot. En outre, l'utilisation de familles de trajectoires paramétrées permet de maitriser le comportement du véhicule.

Mots clés : robotique mobile, navigation, modèle direct

Contribution to mobile robots Navigation : direct model and model based control approach

Summary :

Autonomous robots have to perform both control and perception tasks coordinately. Among these ones, the navigation task is a key in the interaction between the robot and its environment. It consists of determining the trajectories which the robot can follow in order to negotiate correctly around static and dynamic obstacles, assuming that it is programmed to map out its environment and situate itself within that environment. To perform this task, our approach rests on the direct kinematics model of the robot to generate admissible trajectories for the robot. Firstly, a reference trajectory is computed from the reference path provided by a path planer. Then the navigation task is modelized as an optimization under constraints problem, whose the cost function quantify the gap between the reference trajectory and the predicted trajectory of the robot. The obstacles are taken into account as constraints, and the minimization of the resulting cost function determinate the optimal control for the robot on a prediction horizon. This predictive navigation allows the robot to anticipate bi-pass movements on the chosen prediction horizon, Moreover, the behaviour of the robot is mastered by the use of parametered trajectories families.

Keywords : mobile robot, navigation, direct model

PUCVL
Pôle Universités
Centre Val de Loire

Institut PRISME, IUT de Bourges
63, Avenue de Lattre de Tassigny, 18020
Bourges

UNIVERSITE D'ORLEANS]