View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by INRIA a CCSD electronic archive server

Etude d'un sous-systeme de navigation pour un robot mobile

E. Iooss, M. Boulle, P. Tournassoud

To cite this version:

E. Iooss, M. Boulle, P. Tournassoud. Etude d'un sous-systeme de navigation pour un robot mobile. RR-0783, INRIA. 1988. inria-00075768

HAL Id: inria-00075768

https://hal.inria.fr/inria-00075768

Submitted on 24 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INRIA

UNITÉ DE RECHERCHE
NRIA-ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél. (1) 39 63 55 11

Rapports de Recherche

N° 783

ETUDE D'UN SOUS-SYSTEME DE NAVIGATION POUR UN ROBOT MOBILE

Eric IOOSS
Marc BOULLE
Pierre TOURNASSOUD

FEVRIER 1988

Etude d'un sous-système de navigation pour un robot mobile

Local trajectory generation for a mobile robot

Eric Iooss

Marc Boullé

Pierre Tournassoud

INRIA

Domaine de Voluceau

BP 105, 78153 Le Chesnay Cedex

PAPIER RÉCUPÉRÉ ET RECYCLÉ

Etude d'un sous-système de navigation pour un robot mobile

Résumé : *Ce rapport décrit un algorithme local de navigation pour un robot mobile, basé sur un modèle de la cinématique du véhicule et du fonctionnement des capteurs ultrasonores embarqués. Nous montrons par des exemples qu'un système de règles de comportement simples permet d'améliorer l'efficacité des méthodes locales classiques. Nous proposons aussi de coupler l'algorithme local de navigation avec un planificateur global basé sur le graphe de connexité obtenu par décomposition de l'espace libre en cônes.*

Abstract : *In this report we propose an algorithm for local computation of the trajectories of a mobile robot. It is based on the kinematic model of the vehicle, and on a simulation of the operation of ultrasonic sensors. We show through examples that using a set of simple rules of behaviour significantly improves classical local techniques. We then propose to couple the local trajectory generation algorithm with a global planner working on a connectivity graph based on a decomposition of Free Space into cones.*

Introduction

Nous présentons dans ce rapport un système de navigation pour un robot mobile, basé sur les informations acquises par des capteurs ultrasonores. Le couplage avec un planificateur de trajectoires travaillant sur un modèle CAO de l'environnement est aussi ébauché.

L'approche classique pour la planification de trajectoires consiste à rechercher une trajectoire sûre dans l'*espace des configurations* du système, robot manipulateur dans l'espace tridimensionnel ou robot mobile dans le plan. On rappelle qu'une configuration du robot est tout ensemble de paramètres qui permet de décrire chacune des positions des corps mobiles. L'étape critique est la construction de l'*espace libre*, défini comme l'ensemble des configurations pour lesquelles le robot n'intersecte pas les obstacles. Cette construction est cependant très simple dans le cas d'un robot mobile représenté par un polygone qui se déplace uniquement en translation dans le plan encombré d'obstacles polygonaux : il suffit de calculer pour chacun des glissements élémentaires du mobile contre les obstacles sa contribution à la frontière de l'espace libre. Les trajectoires de longueur minimale sont alors contenues dans le *graphe de visibilité*. Ce graphe a pour nœuds, outre les positions initiales et finales, les sommets des obstacles transformés dans l'espace des configurations. Une arête relie deux nœuds quand le segment correspondant est tout entier dans l'espace libre. Pour un robot mobile représenté par un disque, on peut de même définir un graphe de visibilité généralisé, composé comme précédemment de segments mais aussi d'arcs de cercle pour représenter les rotations du disque autour des sommets des obstacles.

Le cas d'un robot mobile de symétrie circulaire est particulièrement séduisant puisqu'il permet d'ignorer l'orientation du mobile lors du calcul d'une trajectoire. Dès que l'orientation du mobile doit être prise en compte, l'espace des configurations est de dimension trois et une représentation exacte de l'espace libre ne peut pas être obtenue simplement. Avnaim et Boissonnat développent des représentations exactes de la frontière de l'espace libre [*AB88*]. Lozano-Pérez a proposé de discrétiser l'intervalle de variation de l'orientation, l'espace des configurations étant ainsi représenté par un empilement de tranches à orientation fixée [*LPW79*]. Afin de limiter le coût de la représentation de l'espace libre, Brooks et Lozano-Pérez ont développé un algorithme qui permet d'adapter la discrétisation de l'espace des configurations en fonction de la géométrie des obstacles entre les positions initiales et finales [*BLP85*]. On construit tout d'abord une partition grossière de l'espace des configurations, qui est

raffinée localement lorsque la trajectoire examinée traverse une cellule qui intersecte partiellement un obstacle.

Brooks a utilisé une représentation plus naturelle de l'espace libre, sous forme d'un ensemble de cônes généralisés qui représentent les couloirs dans lesquels le robot doit naviguer [*Bro83*]. Les déplacements considérés sont calculés parmi les rétractions du robot sur l'axe des couloirs. Plus précisément, à chaque couloir on associe un intervalle d'orientations possibles pour le mobile lorsque son centre (point de référence) est situé sur l'axe du cône. Le passage d'un couloir a un couloir adjacent est possible si les intervalles d'orientation correspondants s'intersectent. Une trajectoire est recherchée dans le graphe de connexité entre couloirs. Elle est constituée de translations à orientation fixée le long d'un couloir, et de rotations sur place à l'intersection de deux corridors. Bien sûr cette technique heuristique ne permet pas une exploration complète de l'ensemble des trajectoires entre des positions initiale et finale, mais donne des trajectoires simples et réalistes.

Si l'on s'attache avant tout à réaliser un système de navigation à partir des informations fournies par les capteurs embarqués, la décomposition en couloirs est d'autant préférable à une représentation par discrétisation arbitraire des orientations. Il sera en effet plus facile de recaler la position du robot dans le modèle CAO après intégration des erreurs de déplacement si le planificateur de trajectoires utilise des primitives qui sont naturelles pour décrire l'environnement. Notons que ce n'est pas le cas pour un robot manipulateur, dont la configuration peut toujours être connue précisément par simple lecture des variables articulaires.

Il reste un point important, qui n'a été jusqu'ici que peu considéré : les déplacements du robot sont contraints par l'architecture de la plateforme mobile. Bien que l'espace des configurations soit de dimension trois, le véhicule ne posséde en général que deux degrés de liberté. Par exemple, un robot mobile avec deux roues motrices indépendantes montées sur le même axe ne peut se déplacer que normalement à cet axe, une voiture a nécessairement son centre de rotation dans l'alignement de l'essieu arrière... Pour un robot mobile de symétrie circulaire qui peut tourner sur place, cette contrainte est sans influence sur le calcul d'une trajectoire. Pour un robot de géométrie arbitraire, les trajectoires licites ne constituent qu'un petit sous-ensemble des chemins de l'espace des configurations. Laumond a montré toutefois que cette contrainte, de même qu'une contrainte de minoration du rayon de braquage, ne modifiait pas l'analyse de l'existence d'une trajectoire. Cependant, les déplacements déduits des trajectoires calculées en ignorant la contrainte cinématique pourront comporter un grand nombre de points de rebroussements, c'est-à-dire de manœuvres de marche arrière [*Lau86*]. [*TJ88*] s'attaque au problème du contrôle du nombre de manœuvres nécessaires sur une trajectoire, en suivant l'optique de la décomposition de l'espace libre en cônes proposée par Brooks. Il est montré que la transition entre deux corridors adjacents peut s'effectuer sans manœuvre seulement si la trajectoire qui respecte la contrainte cinématique et reste en contact (en un point) avec les murs extérieurs au virage est sans collision. Dans le cas contraire, des manœuvres simples sont étudiées. Un algorithme heuristique de planification de

trajectoires, basé sur une décomposition de l'espace libre en cones, permet alors de contrôler le nombre de manœuvres sur une trajectoire.

Ici, plutôt que de nous concentrer sur des preuves d'existence d'une trajectoire entre deux configurations, nous décrivons une implantation pratique d'un algorithme de navigation. Nous avons découplé le problème du calcul d'une trajectoire comme suit. Le planificateur de trajectoire proprement dit est réduit au calcul d'un ensemble de consignes intermédiaires utilisées par un algorithme local de navigation. Ces consignes intermédiaires sont situées au centre des couloirs dans lesquels le robot peut naviguer, et sont extraites du diagramme de Voronoï construit à partir d'un modèle CAO des obstacles. Le générateur de trajectoire n'utilise au contraire que les informations locales fournies par les capteurs à ultra-sons embarqués. Il est fondé sur un jeu de règles de comportement simples (se rapprocher du but, suivre un mur, tourner face à un obstacle, ...), qui permettent au robot de réagir correctement lorsqu'il connaît son environnement immédiat, et d'atteindre ainsi la consigne intermédiaire. La classique *méthode des potentiels*, ainsi que la *méthode des contraintes* introduite pour des robots manipulateurs dans [*FT87a*], constituent des outils utiles qui sont précisés dans ce contexte.

Ce que nous proposons ici n'est qu'un outil *partiel* d'aide à la navigation, qui doit être complété par l'emploi de capteurs plus riches. Une division du travail s'impose entre capteurs, la stéréovision étant utilisée pour construire un modèle de l'environnement et calculer la position du robot (par mise en correspondance de la carte obtenue avec une carte CAO, ou recalage par rapport à quelques amers), les ultra-sons étant utilisés essentiellement pour les actions réflexe.

I. Cinématique du robot mobile

1. Caractéristiques du robot

Le robot mobile a deux roues motrices et une roue folle. On ne contrôle pas le rayon de giration en orientant les roues motrices qui restent dans l'axe du robot, mais en commandant indépendamment leurs vitesses de rotation. Ces vitesses, notées ω_1 et ω_2 par la suite, varient entre - Ω_{max} et Ω_{max}. La figure I.1 ci-dessous décrit la géométrie du robot.

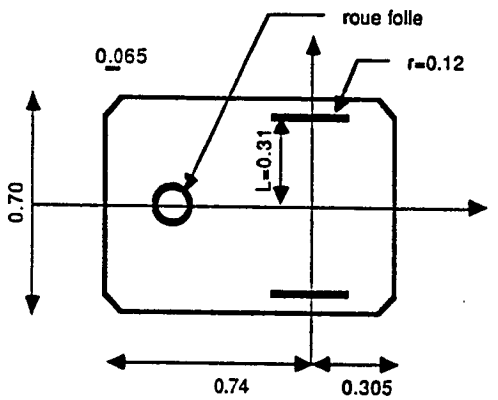


Figure I.1 : Dimensions du robot.
r est le rayon d'une roue, L le demi entre-axe.

Le triplet (x,y,θ) constitue un vecteur de configuration du mobile, avec x et y les coordonnées du point situé au milieu de l'axe des roues et θ l'orientation du robot (figure I.2).

Les équations du mouvement s'écrivent :

$$x' = \cos(\theta).(\omega_1 + \omega_2).r / 2$$

$$y' = \sin(\theta).(\omega_1 + \omega_2).r / 2$$

$$\theta' = (\omega_1 - \omega_2).r / 2.L$$

Nous définissons le rayon de giration par :

$$R = \frac{ds}{d\theta} = L \frac{\omega_1 + \omega_2}{\omega_1 - \omega_2}$$

Ainsi R est positif si le robot tourne à gauche.

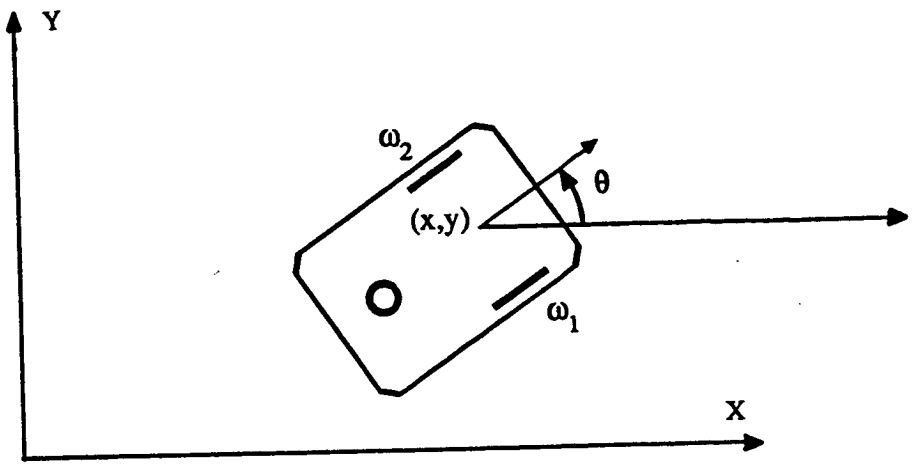


Figure I.2 : Variables cinématiques.

Le système décrit par les équations ci-dessus n'est pas holonome. En effet nous avons la relation :

$$dy = dx.tg(\theta),$$

qui signifie que la trajectoire est toujours normale à l'axe des roues.

Ainsi seuls quelques chemins de l'espace des configurations du robot sont réalisables physiquement. Les variables (ω_1,ω_2), qui sont les dérivées des variables odométriques, suffisent pour décrire le mouvement mais ne représentent pas une configuration du robot. Une trajectoire est cependant complètement décrite par la donnée de $(\omega_1(t),\omega_2(t))$ au cours du temps.

Il est utile de connaître le déplacement d'un point M lié au robot en fonction de (ω_1,ω_2). Il est exprimé par le jacobien J au point M, défini par la relation différentielle $dM / dt = J\,\omega$, où ω est le vecteur de commande de composantes ω_1 et ω_2.

Soient p et q les coordonnées du point M dans le repére lié au robot d'origine le milieu de l'axe des roues, et dont l'axe des x coïncide avec le cap du robot.

On rappelle :

$$\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \frac{r}{2} \begin{pmatrix} \cos(\theta) & \cos(\theta) \\ \sin(\theta) & \sin(\theta) \\ L^{-1} & -L^{-1} \end{pmatrix} \begin{pmatrix} \omega_1 \\ \omega_2 \end{pmatrix}$$

et puisque :

$$\begin{pmatrix} x'_M \\ y'_M \end{pmatrix} = J \begin{pmatrix} \omega_1 \\ \omega_2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & -p.\sin(\theta) - q.\cos(\theta) \\ 0 & 1 & p.\cos(\theta) - q.\sin(\theta) \end{pmatrix} \begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix}$$

nous obtenons :

$$J = \frac{r}{2}\begin{pmatrix} \cos(\theta) & \cos(\theta) \\ \sin(\theta) & \sin(\theta) \end{pmatrix} + \frac{r}{2.L}\begin{pmatrix} -p.\sin(\theta) - q.\cos(\theta) & p.\sin(\theta)+q.\cos(\theta) \\ p.\cos(\theta) - q.\sin(\theta) & -p.\cos(\theta)+q.\sin(\theta) \end{pmatrix}$$

Nous imposerons aussi par la suite une contrainte sur la variation des vitesses ω_1 et ω_2 pour simuler l'effet de l'inertie du robot. Considérant que les accélérations ω_1', ω_2' sont bornées, nous traduisons cette contrainte par les inégalités :

$$|\omega'_1| < \frac{\Omega_{max}}{\xi.dt} \qquad |\omega'_2| < \frac{\Omega_{max}}{\xi.dt}$$

2. Choix d'une trajectoire en espace libre

Puisqu'il qu'il n'y a pas de contrainte de minoration du rayon de braquage, tout lacet C^1 qui relie les configurations de départ et d'arrivée est une trajectoire possible. On en cherche une qui soit physiquement réaliste, c'est-à-dire sans discontinuité du rayon de courbure, et de temps de parcours et d'énergie raisonnables (le problème en temps et énergie minimaux n'a pas de solution explicite : voir en annexe).

La consigne recherchée, donnée par les fonctions $\omega_1(t)$ et $\omega_2(t)$, doit être simple pour pouvoir être calculée en temps réel.

Une première solution consisterait à tourner sur place jusqu'à viser la position finale, se déplacer en ligne droite, puis se réorienter en azimut une fois la position (x,y) finale atteinte : c'est la solution en distance minimale mais pas forcément la plus rapide.

Cette commande ne peut pas cependant être utilisée comme consigne pour un générateur de trajectoires en présence d'obstacles : en effet elle n'a pas de rôle moteur, le robot ayant toujours tendance à se réorienter au lieu d'avancer.

Kanayama et Miyake [*KM85*] ont montré qu'il est très satisfaisant de prendre des trajectoires pour lesquelles la courbure varie linéairement par morceaux. De telles trajectoires décrivent des clotoïdes (figure I.3).

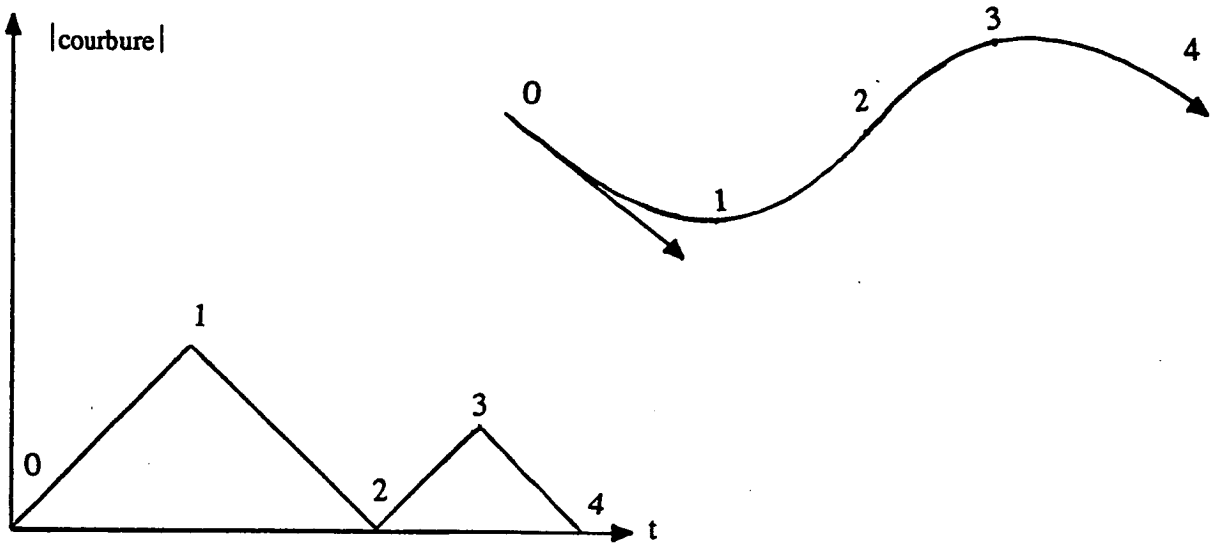


Figure I.3 : Exemple de clotoïdes.

Cependant il est plus rapide et plus simple de chercher la trajectoire comme réunion de deux arcs de cercle tangents entre eux, et de tangente l'angle θ pour les configurations initiales et finales. Le cas dégénéré de positions et orientations finales et initiales alignées, peu adapté à cette méthode, est traité séparément.

Le faisceau de trajectoires possibles est illustré par la figure I.4 ci-dessous :

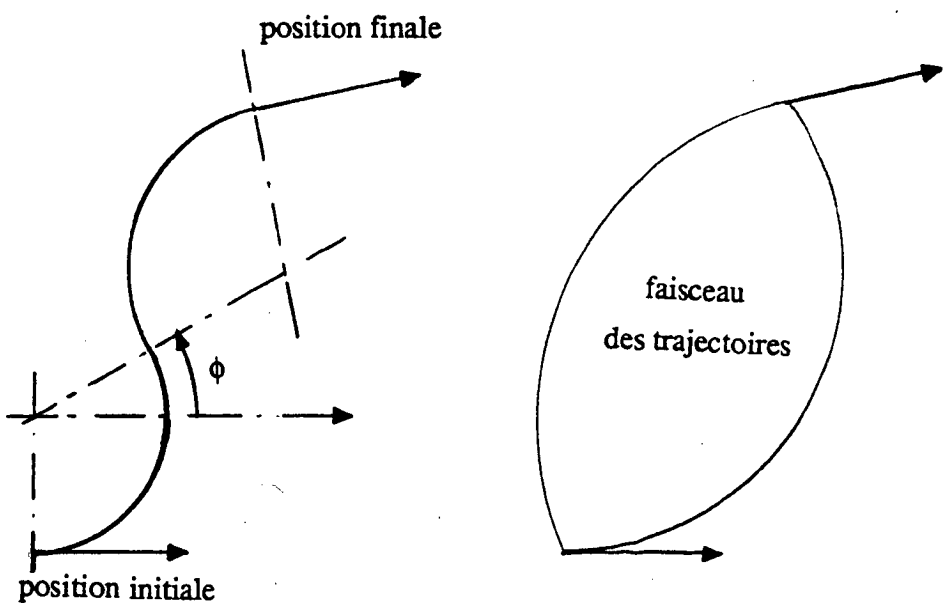


Figure I.4 : Faisceau des trajectoires composées de deux arcs de cercles.

Ecrivons les équations correspondantes en notant pour simplifier (0,0,0) la configuration initiale et (x,y,θ) la configuration objectif. Les variables *algébriques* du problème sont :

R_1, R_2 les rayons de courbure du premier et du second cercles (R est positif dans le cas d'un virage à gauche, négatif pour un virage à droite).

ϕ_1, ϕ_2 les angles du point de tangence vus du centre du premier et du second cercle.

La tangence des deux cercles s'écrit :

$$|R_1|.\cos(\phi_1) = x - R_2.\sin(\theta) + |R_2|.\cos(\phi_2)$$

$$R_1 + |R_1|.\sin(\phi_1) = y + R_2.\cos(\theta) + |R_2|.\sin(\phi_2)$$

La continuité de l'orientation de la tangente s' écrit :

si $R_1 . R_2 > 0$: $\phi_1 = \phi_2$, sinon : $\phi_1 = \phi_2 + \pi$ (figure I.5).

Il reste donc un paramètre que nous pouvons faire varier librement ; nous choisissons ϕ_1.

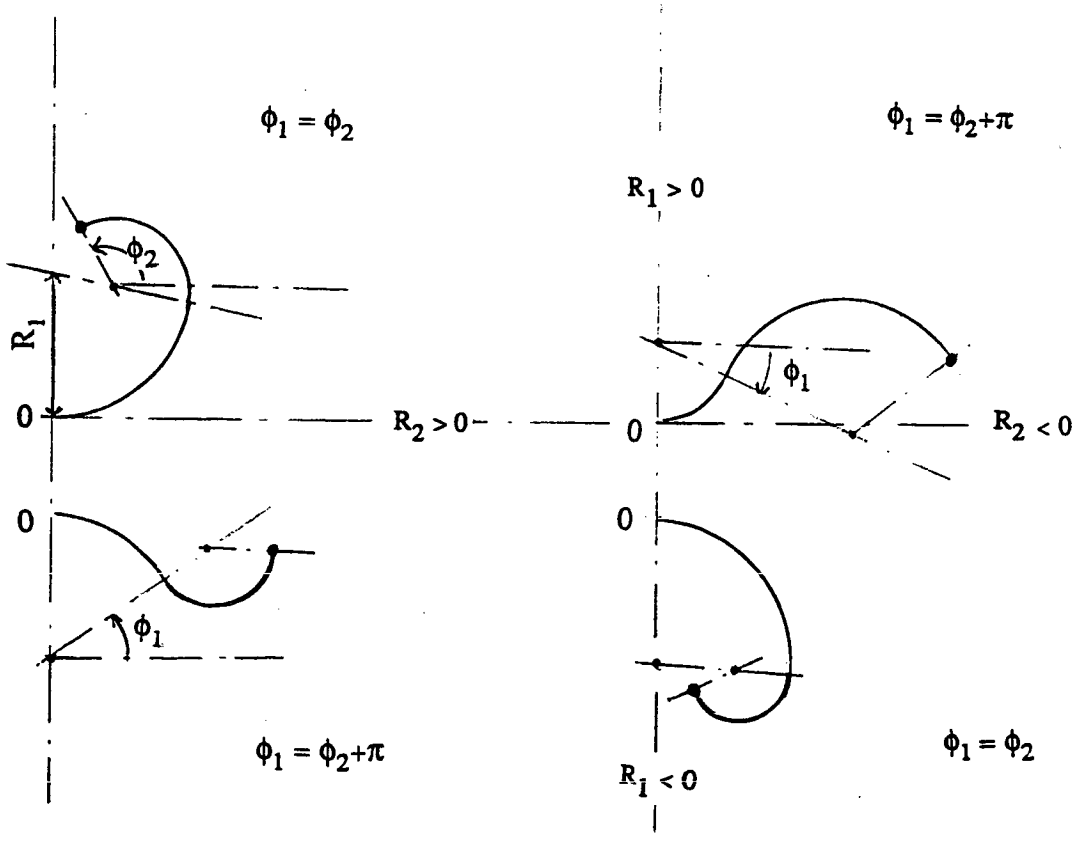


Figure I.5 : Exemples de trajectoires possibles.

• $R_1 < 0$

Les rayons valent :

$$R_1 = \Delta^{-1} \ (x (\sin \phi_1 - \cos \theta) - y (\sin \theta + \cos \phi_1))$$

$$R_2 = \Delta^{-1} \ (x (\sin \phi_1 - 1) - y \cos \phi_1)$$

avec :

$$\Delta = \cos \phi_1 (\cos \theta - 1) + \sin \theta (\sin \phi_1 - 1)$$

• $R_1 > 0$

Les rayons valent :

$$R_1 = \Delta^{-1} \ (- x (\sin \phi_1 + \cos \theta) - y (\sin \theta - \cos \phi_1))$$

$$R_2 = \Delta^{-1} \ (- x (\sin \phi_1 + 1) + y \cos \phi_1)$$

avec :

$$\Delta = \cos \phi_1 (1 - \cos \theta) - \sin \theta (\sin \phi_1 + 1)$$

Nous cherchons la consigne (ω_1,ω_2) à envoyer au robot pour qu'il suive une portion du premier cercle.

Le choix de la trajectoire à suivre dans le faisceau paramétré par ϕ_1 est effectué par la minimisation d'un critère comportant :

- d : la distance réelle à parcourir sur la trajectoire.
- $k \, | \, |R_1| - |R_2| \, |$: la discontinuité des rayons des deux cercles; ce terme permet d'éviter des consignes brutales, telles que tourner sur place puis se rendre à la position finale en un seul arc de cercle.
- $k' \, |R_1 - \underline{R_1}|$: la discontinuité du rayon du premier arc de cercle calculé par rapport à celui appliqué pour l'incrément précédent.

De la valeur de ϕ_1 ainsi calculée, nous déduisons R_1 puis :

$$\frac{\omega_1}{\omega_2} = \frac{R_1+L}{R_1 - L}$$

Nous désirons que le robot parcourt cet arc le plus vite possible ($\text{Max}(|\omega_1|, |\omega_2|) = \Omega_{max}$) et dans le bon sens. Ceci achève la détermination de (ω_1,ω_2), qui est la consigne que va suivre le robot jusqu'au prochain calcul. Si (ω_1,ω_2) diffère trop de la consigne précédente $(\underline{\omega_1},\underline{\omega_2})$, et donc si une des conditions $|d\omega_1|$, $|d\omega_2|$ bornées par $\Omega_{max}.\xi^{-1}$ est violée, nous choisissons d'appliquer une nouvelle consigne de la forme: $\alpha \, (\omega_1,\omega_2)+(1-\alpha) \, (\underline{\omega_1},\underline{\omega_2})$ qui vérifie les conditions précédentes.

Il peut sembler paradoxal que le robot suive toujours le premier arc de cercle, mais cet arc est réactualisé à chaque incrément de temps dt et la trajectoire suivie peut ne pas ressembler à deux arcs de

cercle tangents : elle est notamment plus lissée du fait du terme $k'.|R_1 - \underline{R}_1|$ dans le critère et parce que l'accélération est bornée. Quand l'objectif est très proche, ce type de solution est peu adapté; nous préférons alors atteindre le but en un seul arc de cercle puis nous recaler en azimut.

La figure I.7 montre des traces de trajectoires en espace libre.

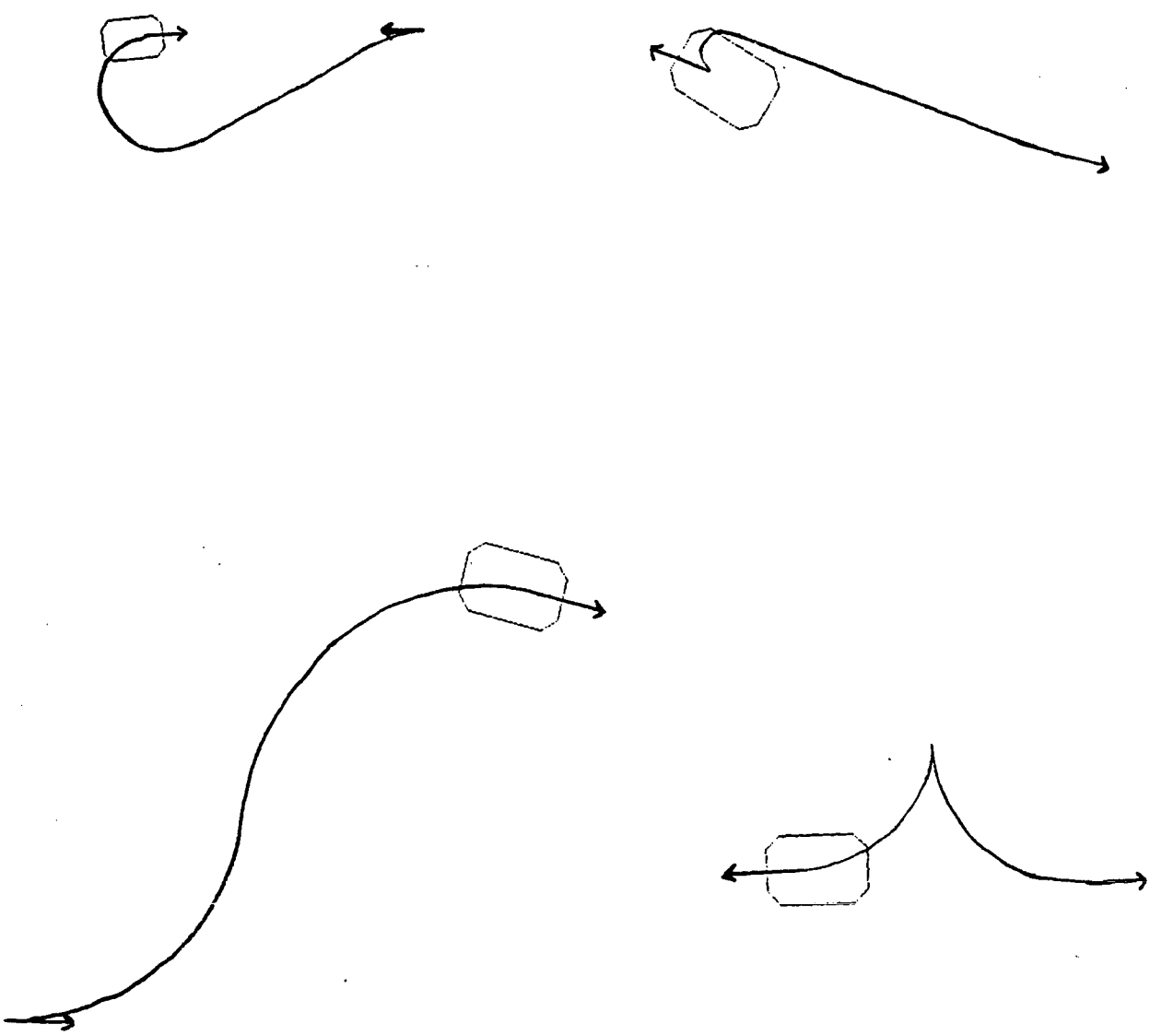


Figure I.7 : Exemples de trajectoires en espace libre ; les flèches dessinées correspondent aux orientations initiales et finales ; le robot est dessiné en position finale.

II. Etude du comportement du robot mobile à partir d'un modèle CAO de l'environnement

1. Description du simulateur

Le simulateur est un programme graphique écrit en langage C tournant sur station de travail SUN.

Le but de ce simulateur est de visualiser sur un écran graphique le déplacement du robot mobile au sein d'un environnement complexe. La représentation de l'environnement est bidimensionnelle. Les obstacles sont modélisés par des polygones convexes pouvant éventuellement s'intersecter : ceci permet d'écrire des algorithmes rapides de calcul de distances entre le robot et les obstacles, et n'enlève rien au réalisme des situations rencontrées.

Une distance d'influence a été définie (d_i de l'ordre de 1 m) à partir de laquelle les obstacles sont détectés par le robot mobile, ainsi qu'une distance de sécurité (d_s environ 4 cm), distance minimale dont le robot peut s'approcher d'un obstacle (figure II.1).

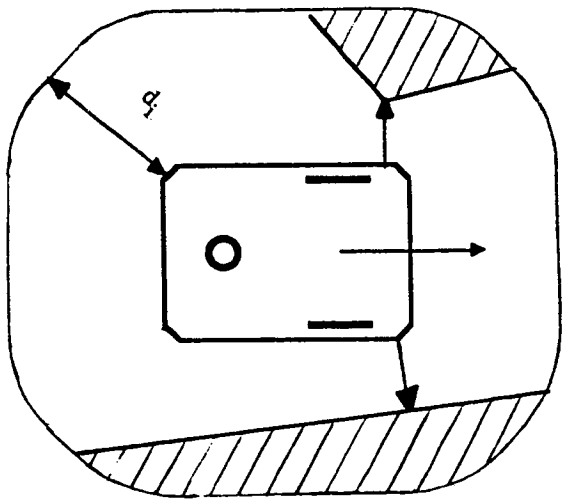


Figure II.1 : Distance d'influence des obstacles.

Nous ne modélisons pas dans cette partie le comportement des capteurs, mais nous supposons que le robot connaît pour chaque obstacle détecté la distance et l'azimut du point le plus proche sur l'obstacle.

2. Ecriture des contraintes générées par les obstacles dans l'espace des commandes

Soit M le point du robot le plus proche de l'obstacle, soit J le jacobien calculé en M et n_M le vecteur unitaire du segment reliant M à l' obstacle. On montre que la dérivée d' de la distance à l'obstacle s'écrit [*FT87a*] :

$$d' = < n_M, v_M > = < n_M, J\,\omega > = < J^t n_M, \omega >.$$

Une contrainte sur la variation de la distance à l'obstacle peut donc se traduire au premier ordre en une contrainte linéaire sur les paramètres de commande ω_1 et ω_2, les consignes licites étant représentées par un demi-espace de l'espace des commandes (figure II.2).

La distance du robot à chaque obstacle ne doit pas décroître trop rapidement quand la consigne est appliquée. Si l'on impose la contrainte suivante (amortisseur) sur la variation de la distance on assure l'absence de collision puisque la distance à l'obstacle reste minorée par une exponentielle décroissante :

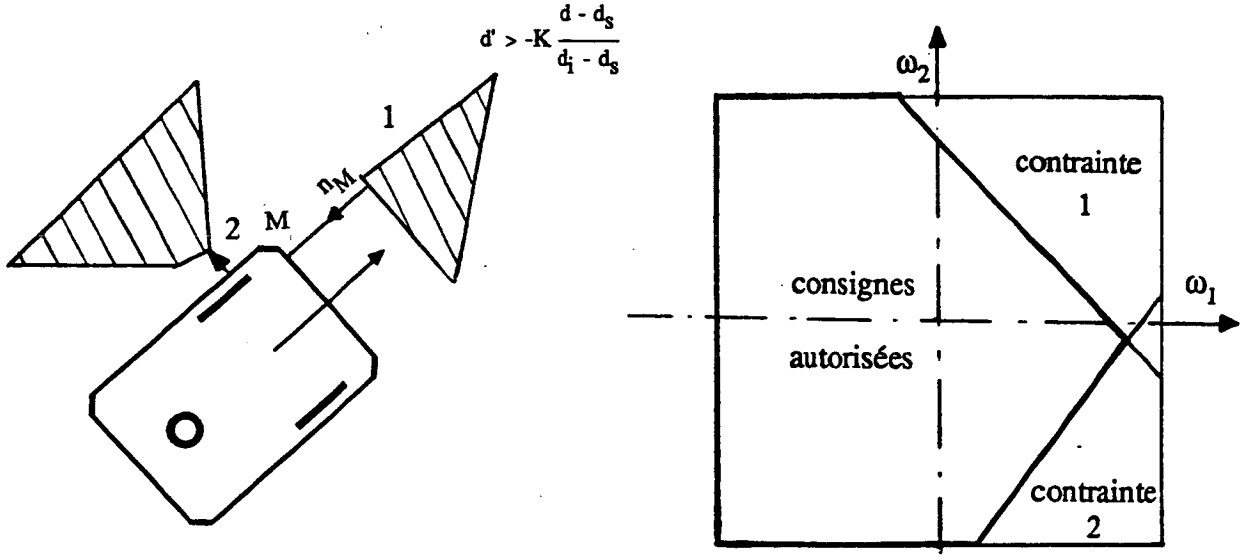


Figure II.2 : Les obstacles vus dans l'espace des commandes ; ils sont plus ou moins rapprochés en fonction de l'angle suivant lequel ils sont perçus par le robot.

Ces idées ont été introduites pour le calcul des trajectoires d'un robot manipulateur dans [*FT87a*]. La formule de l'amortisseur n'est cependant pas la mieux adaptée dans le cas où les accélérations sont bornées. On a dans le cas limite :

$$d'' = K^2 \frac{d - d_s}{(d_i - d_s)^2}$$

donc le robot freine d'autant plus qu'il est loin de l'obstacle.

Dans le cas simple où le robot ne tourne pas, la contrainte de l'accélération bornée impose :

$$d'' \leq \frac{V_{max}}{\xi . dt}$$

soit :

$$d' \geq -\sqrt{\frac{2.V_{max}.(d - d_s)}{\xi . dt}}$$

Nous employons cette formule quand le robot n'est pas trop près de l'obstacle ($d \geq 2.d_s$), mais nous gardons la formule de l'amortisseur, sûre pour les courtes distances, la formule précédente n'assurant pas l'absence de collision avec les obstacles. Le raccord entre ces deux formules est fait d'une part en ajustant le coefficient K de telle manière que d' reste une fonction continue de d, et d'autre part en s'assurant que les fonctions en escaliers de pas dt effectivement réalisées par le robot restent en deçà des fonctions limites ci-dessus.

Nous avons ainsi un ensemble convexe de consignes admissibles par l'intersection des demi-plans d'équation : $< J^t n_M, \omega > = d' \geq - f(d)$

La méthode des contraintes, décrite dans [*FT87a*], qui consiste à projeter la consigne calculée en espace libre sur le polygone convexe des consignes admissibles, assure l'évitement des collisions à coup sûr (figure II.3). Cette méthode n'a pas de rôle moteur : le robot est peu dévié de sa trajectoire calculée en espace libre, il s'approche le plus possible des obstacles en ralentissant mais sans les contourner. Elle doit être complétée par une méthode plus active.

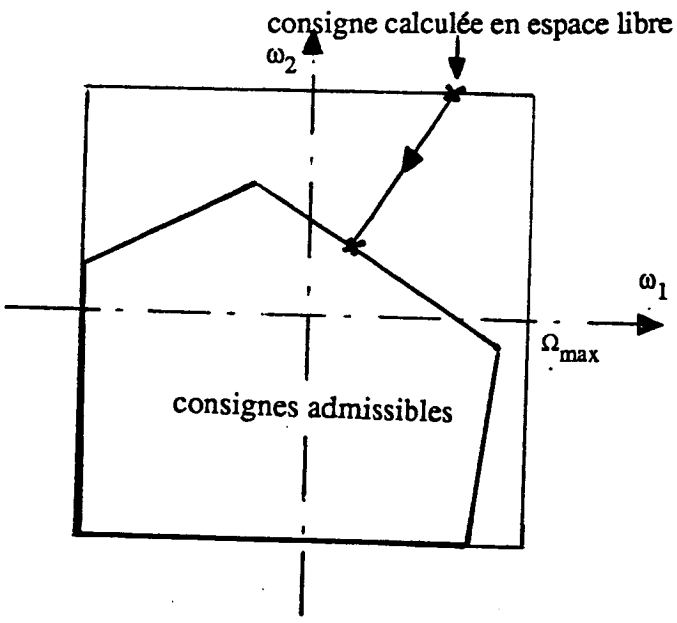


Figure II.3 : Méthode des contraintes.

3. Méthode des potentiels

Cette méthode a été initialement décrite par Khatib [*Kha78*] dans le cadre de la commande dynamique des robots manipulateurs. Le principe consiste à considérer que le mobile est plongé dans un champ de potentiel somme d'un champ l'attirant vers son objectif et de champs répulsifs de la part des obstacles. Le gradient de potentiel en un point du mobile est considéré comme une force fictive qui lui est appliquée. Cette méthode peut aussi être utilisée pour calculer une trajectoire de référence, le gradient du potentiel étant simplement interprété comme une vitesse ou une accélération.

Cette méthode n'est pas parfaitement transposable car nous ne connaissons que l'environnement immédiat du robot et non tous les obstacles comme le suppose Khatib.

A chaque obstacle nous associons un potentiel répulsif qui en éloigne le robot ; à partir de la donnée de la consigne en espace libre nous construisons un potentiel qui attire le robot vers son objectif. Ces potentiels seront écrits directement dans l'espace des commandes. On cherche une consigne réalisable (c'est à dire dans le polygone convexe des consignes admissibles) qui minimise la somme des potentiels répulsifs et attractif calculés précédement.

Le potentiel attractif est de la forme :

$$V(\omega_1, \omega_2) = ((\omega_1 - \underline{\omega}_1)^2 + (\omega_2 - \underline{\omega}_2)^2) / 2$$

avec $(\underline{\omega}_1 , \underline{\omega}_2)$ la consigne calculée en espace libre.

Nous avons utilisé comme potentiel répulsif généré par un obstacles un potentiel en $1/\delta^2(\omega)$ où $\delta(\omega)$ mesure la distance à la contrainte dans l'espace des commandes (figure II.4).

$$\delta(\omega) = | v_3 - v_1.\omega_1 - v_2.\omega_2 | / (v_1^2 + v_2^2)^{1/2}$$

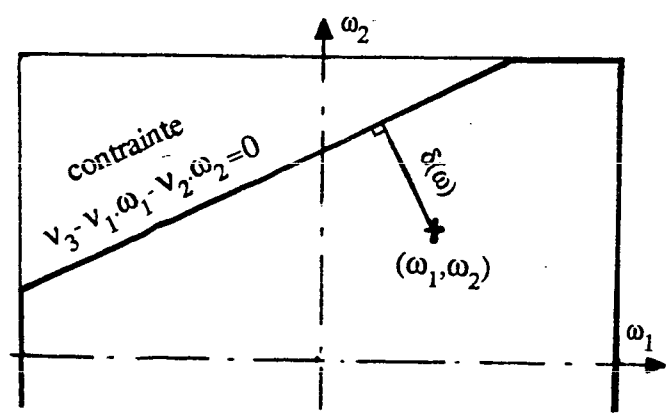


Figure II.4 : "Distance" à une contrainte dans l'espace des commandes.

Les coefficients utilisés pour pondérer le terme attractif et le terme répulsif du potentiel sont assez difficiles à ajuster : le robot rebondit ou tourne sur place s'il est face à l'obstacle, ou bien glisse sur celui-ci (figure II.5). La distance critique (supérieure à la distance de sécurité) en deçà de laquelle il interdit au robot de s'approcher de l'obstacle est difficile à contrôler.

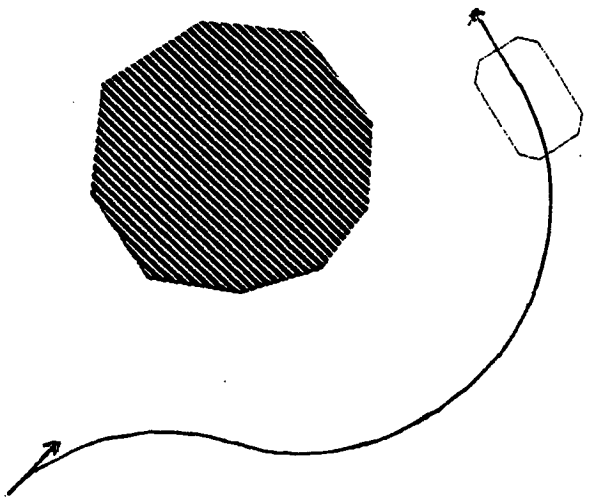


Figure II.5 : Le potentiel arrondit la trajectoire du robot.

Le deuxième potentiel testé comporte un terme de sécurité en $1/\delta^2(\omega)$ et un terme linéaire destiné à réorienter le robot (figure II.6).

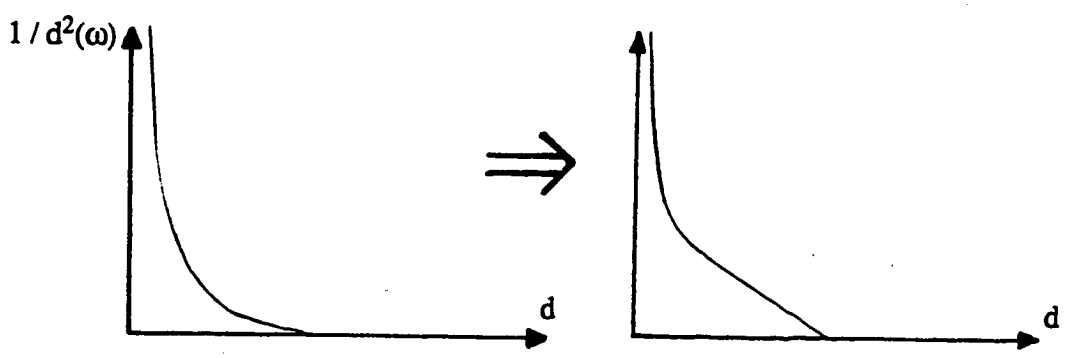


Figure II.6 : Amélioration de la forme du potentiel.

Le résultat est meilleur mais le robot, pour se réorienter parallèlement à l'obstacle, manœuvre en reculant puis en avançant, ce qui est peu satisfaisant (figure II.7).

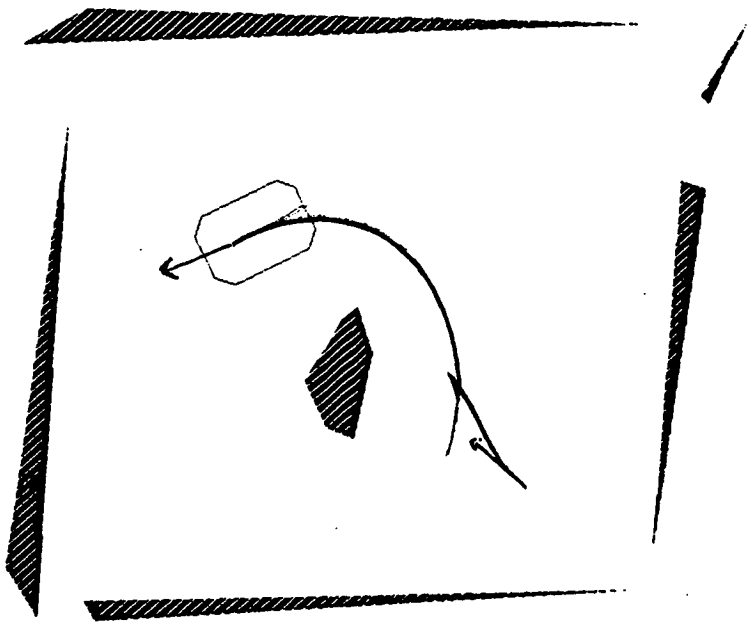


Figure II.7 : Le robot rebondit et manœuvre pour éviter l'obstacle.

En présence de plusieurs obstacles, on utilise classiquement comme potentiel généré par l'ensemble agrégé des obstacles détectés, la somme des potentiels engendrés par chaque obstacle. Cette méthode crée des cols de potentiel entre les obstacles. A nouveau il faut contrôler l'importance respective des termes attractifs et répulsifs. Dans notre cas, où nous ne connaissons que l'environnement proche du robot, ces cols peuvent se révéler infranchissables, par exemple dans le cas du passage d'une porte.

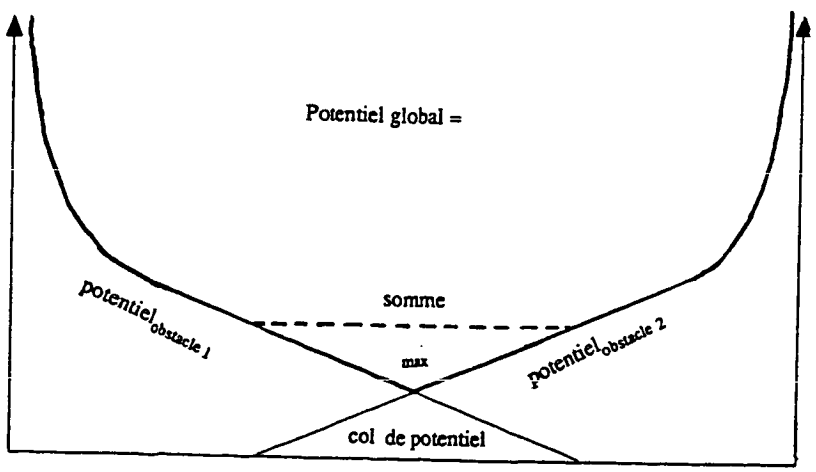


Figure II.8 : Comparaison de la somme et du maximum des potentiels.

De plus, par cette méthode, N obstacles très rapprochés engendrent un potentiel N fois plus important qu'un seul obstacle isolé.

Il peut ainsi être préférable de choisir pour potentiel global le potentiel généré par l'obstacle le plus proche du robot dans l'espace des consignes, c'est-à-dire le terme maximum parmi les potentiels répulsifs (figure II.8). Cependant cette fonction a un gradient non continu, ce qui entraîne des oscillations indésirables lorsque le robot navigue dans un couloir. Le passage d'une porte, bien que réalisable, reste aléatoire.

4. Méthode mixte avec modification de la consigne.

La méthode du champs de potentiels, comme la méthode des contraintes, est peu efficace lorsque l'attraction vers l'objectif est trop antagoniste avec la répulsion engendrée par les obstacles. Une solution à ce type de problèmes consiste à changer la consigne initiale en tenant compte des obstacles avant l'intervention de la méthode des potentiels. Par exemple, un nouveau terme est ajouté dans le critère de choix de la trajectoire en deux arcs de cercles, tel que :

$$\left| \frac{1}{|\alpha|} - \frac{1}{\pi/2} \right| \quad \text{si } |\alpha| < \pi / 2, \ 0 \text{ sinon,}$$

avec α l'angle entre la consigne testée et la contrainte la plus proche dans l'espace des commandes (figureII.9).

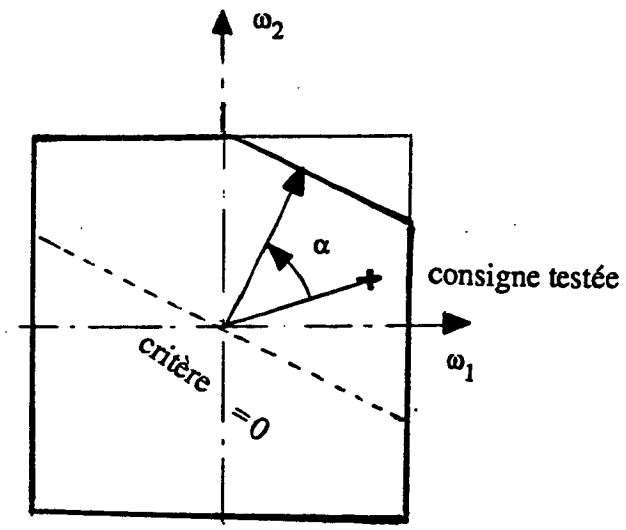


Figure II.9 : Prise en compte d'un obstacle pour le calcul de la consigne.

Ceci permet au robot de mieux anticiper la prise en compte des obstacles. Les résultats sont meilleurs (figure II.10), mais parce que la consigne n'est recherchée que parmi les trajectoires en deux arcs de cercles, le choix des trajectoires possibles peut être restreint et cette méthode ne l'élargit pas (figure II.11).

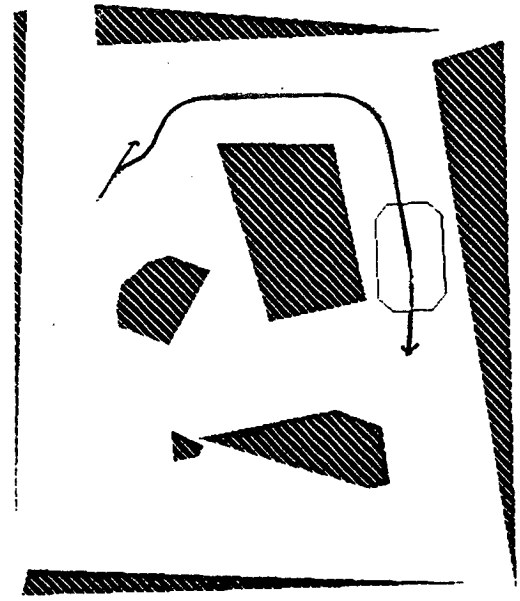


Figure II.10 : Méthode des potentiels dans un cas favorable.

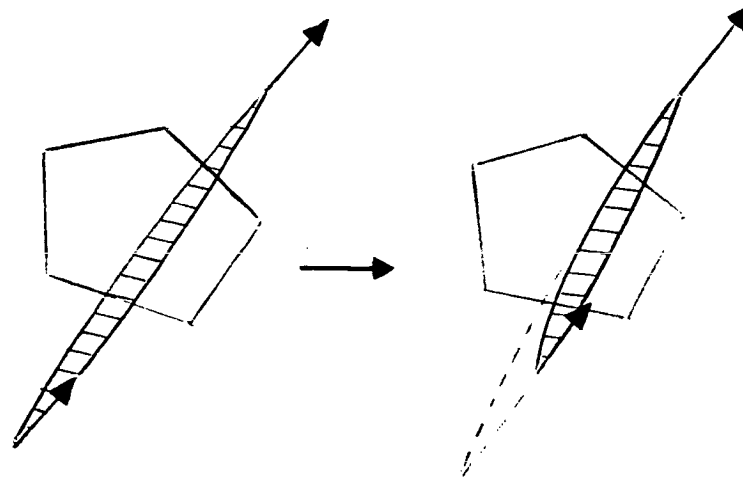


Figure II.11 : Le faisceau des consignes calculées est étroit, et ne s'élargit pas au cours du déplacement du robot.

En conclusion, il semble difficile de trouver une méthode générale s'adaptant aux multiples cas particuliers rencontrés : murs de face ou sur le côté, plus ou moins inclinés par rapport au robot, portes, petits obstacles, couloirs, coudes . . .

C'est pourquoi nous avons recherché des techniques qui sont mieux guidées par la géométrie locale des obstacles, pour agir sur la consigne plus efficacement.

5. Méthode utilisant un système de règles adaptées à la nature de l'environnement immédiat

La compatibilité de la consigne avec l'environnement proche est analysée comme suit. Dans un premier temps, nous prenons en compte l'obstacle frontal le plus proche. Suivant son inclinaison par rapport au robot, nous choisissons une consigne qui tend à réorienter le robot latéralement à l'obstacle (figure II.12).

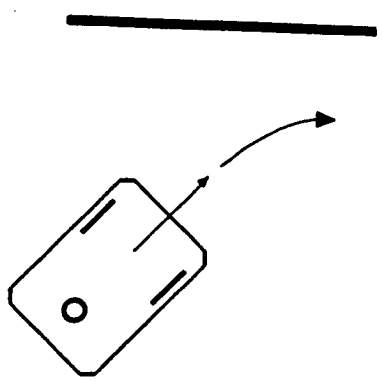


Figure II.12 : Obstacle frontal.

Dans un deuxième temps, si un obstacle est détecté à l'arrière du robot à moins d'une distance critique (environ 10 cm par exemple), une procédure de "décollement de mur" est exécutée. Cette procédure consiste à faire glisser le point du robot le plus proche de l'obstacle à distance constante de ce dernier (figure II.13).

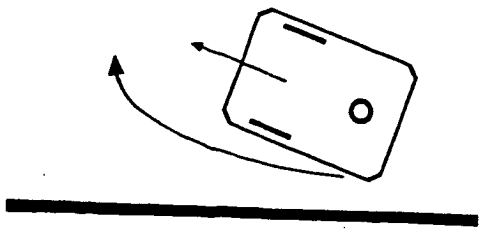


Figure II.13 : Obstacle dorsal, décollemement de mur.

Dans un troisième temps nous modifions la consigne ainsi obtenue pour éviter les discontinuités trop fortes sur les vitesses des roues.

Et enfin, un programme de sécurité (méthode des contraintes) empêche le robot d'entrer en collision avec les obstacles et le fait ralentir dans les situations délicates.

Cette méthode est satisfaisante par sa simplicité et son efficacité. Le robot résout les problèmes locaux simples, et même certains problèmes plus complexes (figure II.14).

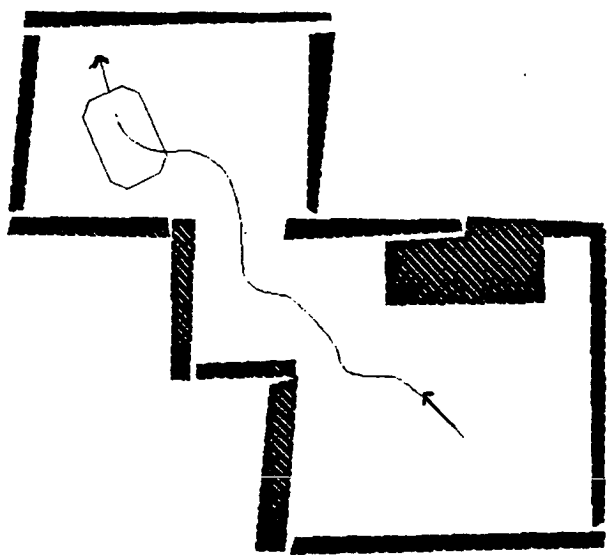


Figure II.14 : Méthode basée sur l'utilisation un système de règles, le passage d'une chicane.

C'est cette approche que nous allons développé dans la partie suivante, qui décrit un algorithme de navigation du robot à partir des données des capteurs à ultra-sons.

III. Un algorithme de navigation basé sur les données des capteurs embarqués

1. Description du simulateur

1.1. Modélisation des capteurs ultrasonores

Les capteurs sont du type capteur ultrasonore Polaroïd électrostatique. Chaque capteur émet dans une direction donnée un train d'onde dans la gamme 50 à 60 kHz, et reçoit quelques millisecondes plus tard l'écho de son signal. De la vitesse de propagation du son on peut déduire la distance du capteur à l'obstacle dans la direction visée. La portée va de quelques centimètres à 1 ou 5 mètres suivant la fréquence (et donc la résolution) choisie. Le capteur émet principalement dans un cône d'une ouverture de 15°, ce qui entraîne une imprécision en azimut sur les données calculées [*Dru85*].

Dans notre simulation, le comportement des capteurs est simplifié (figure III.1) : ils détectent tous les obstacles dans un cône de base le diamètre du capteur (6.5 cm) et d'une ouverture de 15°. On ne tient pas compte des réflexions qui sont très dépendantes de la nature des surfaces.

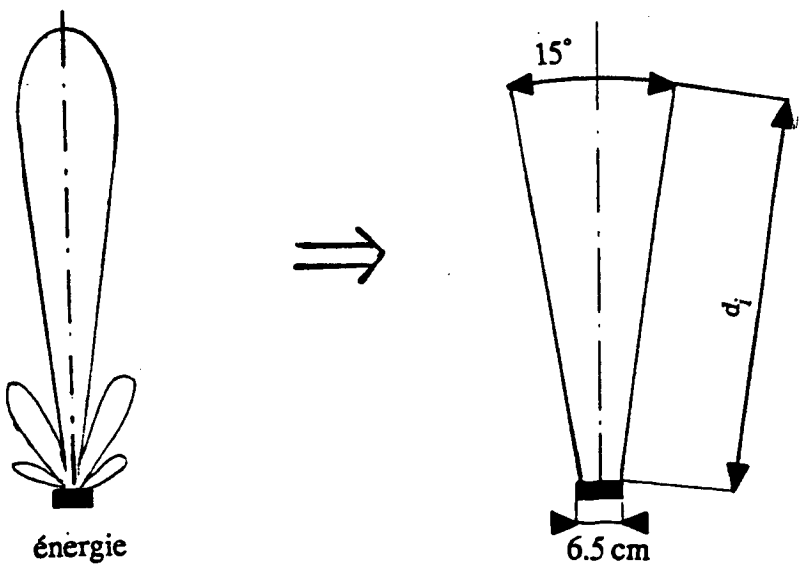


Figure III.1 : Modélisation des cônes des capteurs.

1.2. Disposition des capteurs

La configuration testée comporte 24 capteurs (24 x 15° = 360°). Dans la réalité les capteurs sont disposés sur le bord de la carrosserie du robot, à différentes hauteurs, pour pouvoir détecter des obstacles de formes et de tailles variées. On se ramène par projection à une description bidimensionnelle des obstacles. Une disposition de l'ensemble des capteurs sur une tourelle centrale [*Fly85*] a été envisagée car elle assure une couverture totale de l'environnement, mais d'une part elle n'est pas pratiquement réalisable, et d'autre part les obstacles étant détectés plus loin dans le cône, les données azimutales sont moins précises. La disposition choisie présente cependant l'inconvénient de comporter des angles morts et d'importantes zones proches du robot ne sont pas couvertes (figure III.2).

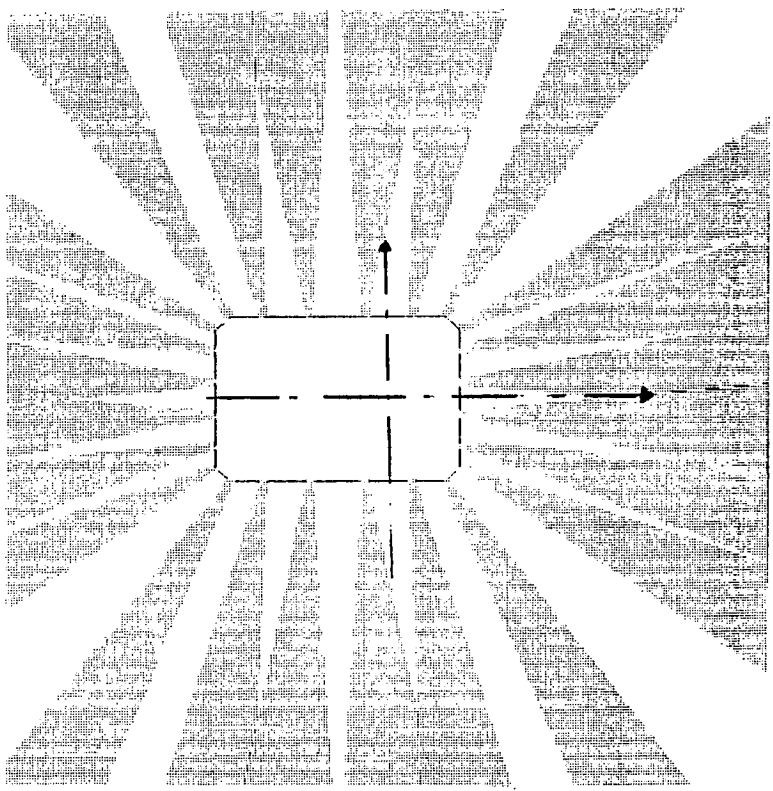


Figure III.2 : Zones couvertes par les capteurs ;
positions et orientations des capteurs.

1.3. Simulation des données des capteurs

La simulation prend en compte les obstacles présents dans le cône du capteur considéré, et calcule la distance du plus proche de ceux-ci au capteur. Le simulateur se contente donc de renvoyer une distance par capteur, la seule information en azimut étant l'orientation du capteur.

Nous avons choisi une portée maximale égale à la distance d'influence de 1 m. Cette portée est suffisante pour éviter les obstacles à temps pour une vitesse maximum de 1 m/s, une réactualisation des données tous les dt = 0.15 s, et un coefficient d'inertie ξ fixé à 8 (il faut ξ.dt secondes pour que le robot s'immobilise dans le pire des cas).

2. Traitement des données des capteurs

2.1. Calcul des positions des obstacles

Il est difficile à partir des données d'un capteur de connaître la position exacte d'un obstacle. Nous utilisons comme approximation le point médian du segment parallèle à la base du capteur, à la distance mesurée par le capteur. Nous déduisons pour les 24 points correspondants leur distance et leur azimut relativement au robot.

L'orientation des capteurs a été choisie approximativement perpendiculaire à la carosserie du robot. En effet, un autre choix entraîne une grande imprécision en distance, comme indiqué par la figure III.3.

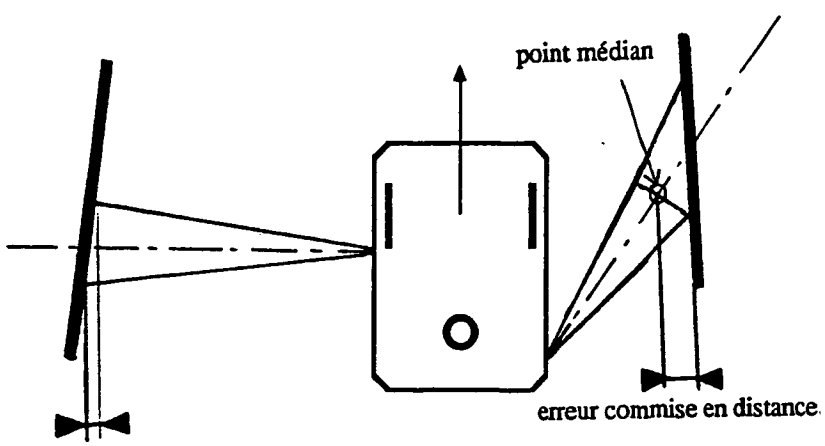


Figure III.3 : Orientation des capteurs.

2.2. Sélection des données

L'univers du robot est divisé en quatre régions se recoupant deux à deux (figure III.4).

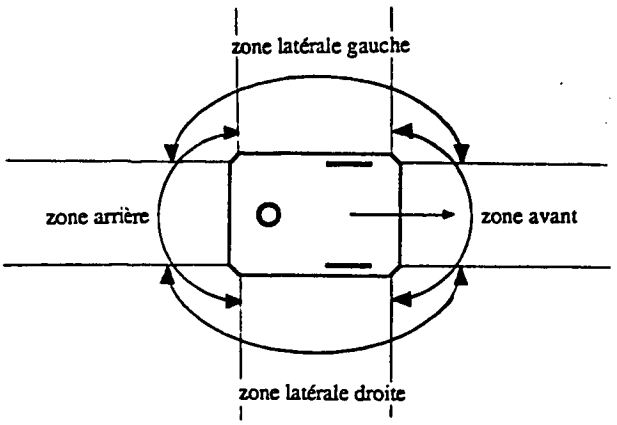


Figure III.4 : Découpage de l'espace autour du robot.

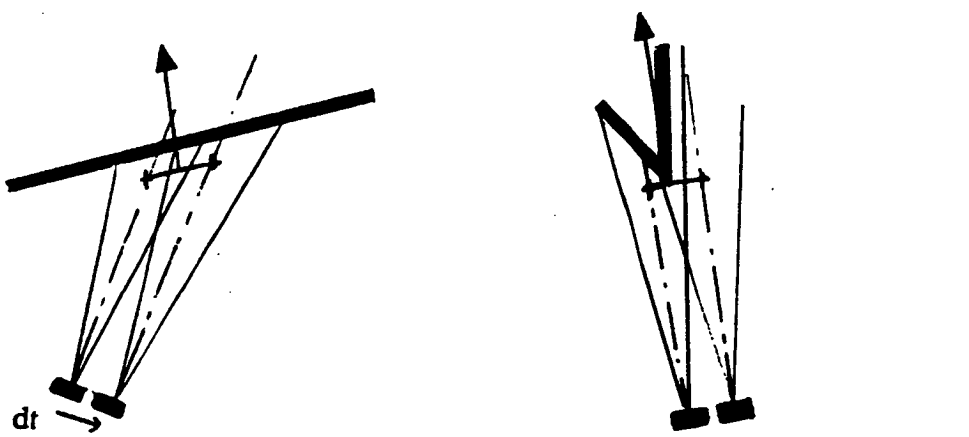
Dans chaque région, nous sélectionnons l'obstacle le plus proche du robot. Nous mémorisons pour chacun des quatre obstacles ainsi pris en compte le point correspondant dans un repère de référence, ainsi que le numéro du capteur qui a effectué la mesure.

2.3. Traitement des angles morts

Un mur, de part son étendue, est toujours détecté par au moins un des capteurs du robot. Par contre, un point anguleux peut subitement quitter un cône de détection sans entrer immédiatement dans le suivant, à cause de la présence d'angles morts (cf. figure III.2). Pour pallier ce problème, nous mémorisons pour chacune des zones définies au paragraphe précédent, l'obstacle le plus proche du robot, parmi l'union des obstacles actuellement vus par les capteurs et de l'obstacle le plus proche précédemment mémorisé. Ce traitement permet de signaler la présence d'un point anguleux proche du robot même si celui-ci n'est pas détecté, et est peu coûteux. Cependant lorsqu'on quitte le monde de la simulation pour le monde réel, les données des capteurs doivent de plus subir un traitement statistique pour éliminer les informations incohérentes.

2.4. Calcul de l'orientation de la frontière des obstacles

La connaissance de la normale aux obstacles permet d'affiner les procédures de traitement ultérieures. Cette normale est approximativement connue par le calcul de la normale au segment joignant les points actuellement et précédemment détectés par un même capteur. Ce calcul est acceptable dans le cas où l'obstacle détecté est un mur, mais est inexact dans le cas d'un point anguleux (figure III.5). Dans ce dernier cas, la normale calculée se rapproche de la normale au capteur. Quand ce calcul semble injustifié (si les deux points détectés sont trop proches), nous choisissons comme normale à l'obstacle la normale au capteur concerné. En effet quand nous avons pour seule donnée la distance du point de l'obstacle le plus proche du capteur, la pire situation advient lorsque toute la frontière de l'obstacle est à cette distance, c'est à dire lorsque la normale à la frontière est égale à la normale au capteur.



même capteur à t et t+dt

la normale calculée se rapproche de la normale au capteur

Figure III.5 : Calcul de la normale à l'obstacle.

3. Calcul de la trajectoire en espace contraint

3.1. Principe de la méthode

La méthode employée utilise une base de règles de comportement, adaptées à la perception de l'environnement du robot par des capteurs ultrasonores. Le découpage de l'espace du robot en quatre régions et la connaisance du sens de déplacement du robot permet de classer simplement les obstacles en obstacles frontaux, latéraux et dorsaux. A chaque type de configuration relative du robot et de l'obstacle correspond une procédure de traitement adaptée. La consigne calculée en espace libre est alors modifiée en prenant successivement en compte les obstacles frontaux, latéraux puis dorsaux. Au cours de ces

modifications de la consigne, le sens de déplacement du robot reste le même. Seule une commande venant d'un niveau supérieur peut demander au robot de faire marche arrière pour se débloquer.

3.2. Obstacles frontaux

Seul l'obstacle frontal le plus proche, sélectionné lors du traitement des données des capteurs, est pris en compte. La procédure n'est exécutée que si l'obstacle se trouve à une distance inférieure à la distance d'influence d_i.

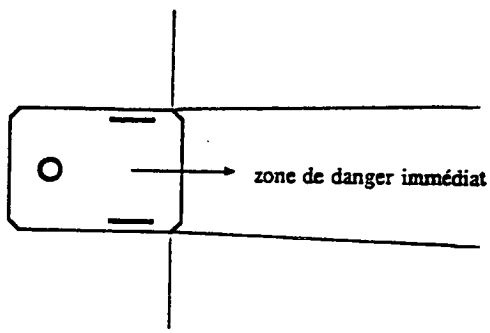


Figure III.6 : Zone avant du robot.

Si l'obstacle détecté est dans la zone de danger immédiat (figure III.6), le robot modifie sa consigne en espace libre afin de s'orienter parallèlement à celui-ci. Quand l'obstacle a une orientation pratiquement perpendiculaire au robot, ce dernier tourne dans le sens coïncidant avec celui de sa consigne en espace libre. Dans le cas contraire, le robot "glisse" sur la contrainte sans tenir compte de sa consigne initiale.

Pour ce faire, d'une part le robot ralentit en fonction de sa distance à l'obstacle, et d'autre part le robot tourne d'autant plus vivement que l'obstacle est proche ou dangereux (faible azimut relatif par rapport au robot).

Quand l'obstacle a quitté la zone de danger immédiat, le sens de rotation du robot est cette fois bien déterminé et la procédure achève de faire tourner le robot pour le rendre pratiquement parallèle à la contrainte.

Cette procédure modifie donc la consigne jusqu'au moment où les obstacles perçus initialement comme frontaux seront considérés comme latéraux.

Un exemple de contournement d'obstacle (figure III.7) visualise l'action de cette procédure.

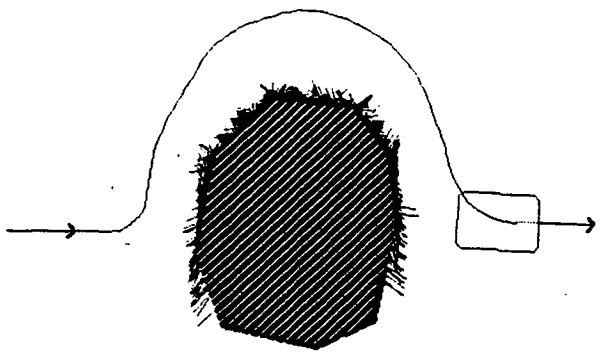


Figure III.7 : Contournement d'obstacle (les segments renvoyés par les capteurs sont représentés).

3.3. Obstacles latéraux

Les obstacles latéraux "droit" et "gauche" sélectionnés sont examinés dans l'ordre inverse de priorité, l'obstacle le plus éloigné étant traité en premier lieu.

Les deux zones latérales sont à leur tour subdivisées en quatre sous-zones : zones avant, latérale avant, latérale arrière, arrière (figure III.8).

Si l'obstacle détecté est dans la zone avant à une distance inférieure à environ la moitié de d_i, la procédure agit sur la consigne de façon semblable à la procédure traitant les obstacles frontaux, jusqu'à ce que le robot soit réorienté parallèlement à l'obstacle. Les formules utilisées sont similaires à celles traitant les obstacles frontaux, et assurent la continuité entre les deux procédures.

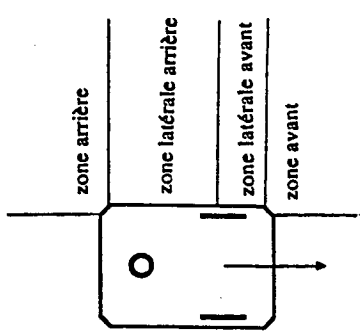


Figure III.8 : Zone latérale du robot.

Les cas latéraux avant et arrière ne sont traités que si l'obstacle est proche du robot (par exemple à moins de 3 fois la distance de sécurité), ce qui permet à celui-ci de se rapprocher des obstacles non immédiatement dangereux.

Dans le cas où l'obstacle détecté est dans la zone latérale avant, si la consigne écarte le robot de l'obstacle, elle demeure inchangée. Dans le cas contraire, nous imposons une consigne éloignant légèrement le robot de la parallèle à l'obstacle.

De même, dans le cas où l'obstacle détecté est dans la zone latérale arrière, on agit sur la consigne comme dans le cas latéral avant, sans toutefois chercher à éloigner le robot de l'obstacle.

Enfin si l'obstacle détecté est dans la zone arrière, la consigne n'est pas modifiée.

Cette procédure modifie donc la consigne jusqu'au moment où les obstacles perçus initialement comme latéraux sont considérés comme dorsaux.

Sur la figure III.9, on voit ainsi le robot se rapprocher suffisament près de l'obstacle, puis avancer en restant à distance constante de celui-ci, et commencer à tourner au passage du coin anguleux dès que l'axe des roues a dépassé le sommet de l'obstacle.

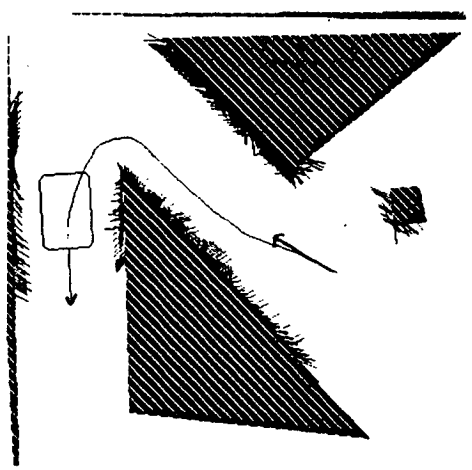


Figure III.9 : Exemple d'obstacle latéral.

3.4. Obstacles dorsaux

Les obstacles dorsaux ne sont traités que s'ils sont à distance très faible (inférieure à environ 2.5 fois la distance de sécurité), et une procédure de décollement de mur est alors appliquée. Le problème est de maintenir l'arrière du robot à distance constante du mur, ce qui revient à imposer au point correspondant à l'arrière du robot de suivre une droite parallèle au mur (figure III.11). Dans le cas où le robot est en marche avant, cette condition est réalisée par la formule :

$$R = \frac{c}{tg(\theta)} - a \quad \text{c'est-à-dire} \quad \frac{\omega_2}{\omega_1} = \frac{c - (a+L).tg(\theta)}{c - (a - L).tg(\theta)}$$

où a est la demi-largeur du robot, b (respectivement c) la distance entre l'axe des roues et l'avant

(l'arrière) du robot, et θ l'orientation du robot par rapport au mur.

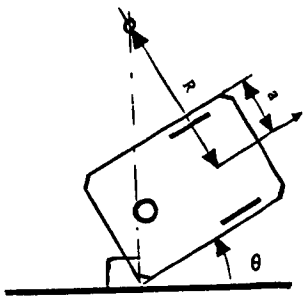


Figure III.10 : Décollement de mur.

Ce programme agit dès que la consigne tend à rapprocher l'arrière du robot de l'obstacle, et rend la main aux autres procédures quand le robot peut suivre sa trajectoire sans danger.

La figure III.11 ci-dessous illustre cette procédure.

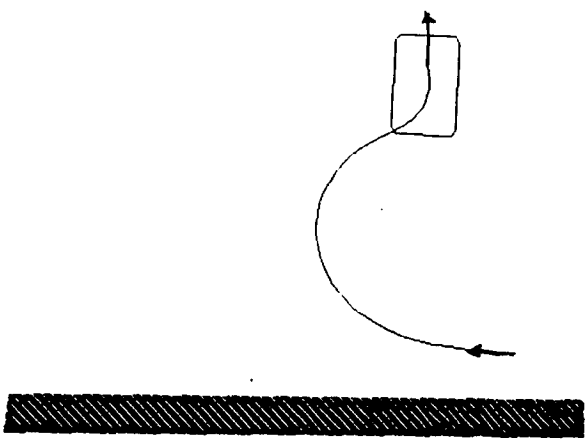


Figure III.11 : Trajectoire de décollement de mur.

3.5. Coopération entre les règles

Les différentes règles utilisées sont imbriquées et complémentaires. Il y a d'abord un ordre de priorité dans l'examen des règles et un ordre d'importance inverse lié aux distances d'action de chacune d'elles. Les obstacles dorsaux à moins de 2.5 d_s ont le dernier mot, sinon ce sont les obstacles latéraux à moins de 0.5 d_i, sinon les obstacles frontaux à moins de d_i. En effet la distance nécessaire au robot pour réagir lorsqu'il se trouve face à un obstacle est supérieure à la distance requise dans le cas d'un obstacle sur le côté ou à l'arrière. D'autre part quand la procédure concernant les obstacles dorsaux est exécutée, cela signifie que les autres procédures n'ont soit pas de raison d'être, soit agissent depuis longtemps et

donc ont déjà largement influé sur la trajectoire.

Le passage d'un coude dans un couloir (figure III.12) permet d'observer un exemple de coopération entre les règles employées. Dans un premier temps le robot achève de s'orienter parallèlement au mur intérieur du couloir (obstacle frontal). Ensuite le robot se rapproche légérement de ce mur et avance en conservant une faible distance par rapport à celui-ci. Dès que l'axe des roues a passé le sommet de l'obstacle intérieur au couloir, le robot entame son virage, et le poursuit jusqu'à ce que l'arrière du robot glisse contre le mur extérieur au coude (obstacle latéral). Le robot "décolle" de ce mur : il tourne le plus vite possible sans que l'arrière se rapproche du mur (obstacle dorsal). Le mur de devant, perçu comme obstacle frontal, a pour effet d'achèver la réorientation du robot, qui se dirige désormais vers son objectif en restant approximativement au milieu du couloir (obstacles latéraux).

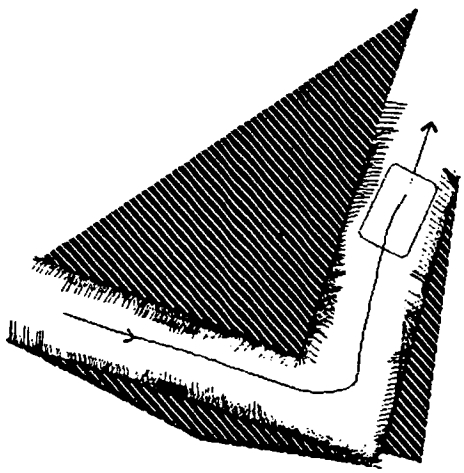


Figure III.12 : Passage d'un coude dans un couloir.

4. Vitesse de parcours de la trajectoire et inertie du robot

La méthode des contraintes utilisée de façon brute est peu adaptée aux données renvoyées par les capteurs, en raison de l'imprécision en azimut due aux capteurs. La méthode simpliste qui consiste à ralentir de façon isotrope en fonction de la distance au plus proche obstacle est peu satisfaisante : le robot freine même quand un obstacle est dans son dos.

La méthode utilisée est un compromis des deux techniques ci-dessus. Nous avons besoin de caractériser simplement une zone dans laquelle peut se mouvoir le robot en toute sécurité. Nous considérons le plus grand rectangle ne contenant pas d'obstacles et de côtés parallèles au robot. Nous l'obtenons aisément compte tenu de la connaissance de la distance du robot au plus proche obstacle dans chacune des quatre zones définies au III.2.2 (devant,derrière,latérales).

Nous calculons la vitesse de rapprochement du robot à chaque côté du rectangle : $< J^t n_M, \omega >$ avec M le point du robot qui se rapproche le plus du côté considéré, J la jacobienne calculée en M et n_M la normale au robot en M, orientée de l'obstacle vers le robot. Par exemple si le robot avance et tourne à gauche, alors il se rapprochera en un incrément dt :

de devant de la distance : $r.dt.\left(\frac{\omega_1+\omega_2}{2} + \frac{a}{L}\frac{\omega_1 - \omega_2}{2}\right)$

de derrière de la distance : $r.dt.\left(-\frac{\omega_1+\omega_2}{2} + \frac{a}{L}\frac{\omega_1 - \omega_2}{2}\right)$

du coté droit de la distance : $r.dt.\frac{c}{L}\frac{\omega_1 - \omega_2}{2}$

du côté gauche de la distance : $r.dt.\frac{b}{L}\frac{\omega_1 - \omega_2}{2}$

On rappelle que a est la demi-largeur du robot, b (respectivement c) la distance entre l'axe des roues et l'avant (respectivement l'arrière) du robot.

Ces vitesses de rapprochement au rectangle majorent pour chaque zone concernée les vitesses de rapprochement réelles du robot aux obstacles. Nous avons calculé dans la partie précédente à quelle vitesse maximale peut rouler le robot, sachant qu'il voit un obstacle devant lui à la distance d, et que son inertie l'empêche de s'arrêter instantanément. Nous trouvons pour les valeurs numériques $\xi = 8$ et $v_{max} = 1$ m/s :

$$v \leq 1.1 \sqrt{d - d_s} \qquad m/s$$

Si cette formule est respectée pour les quatre vitesses de rapprochement calculées, on est assuré de l'absence de collision. Pour ce faire, comme v est linéaire en ω, nous divisons si nécessaire homothétiquement la commande (ω_1, ω_2) jusqu'à vérifier les quatre inéquations. Quand l'obstacle est très proche nous utilisons par sécurité un frein isotrope, fonction linéaire de la distance, qui prolonge de manière continue le frein précédent.

5. Performances et perspectives de la méthode

5.1. Résultats

Le robot sait manœuvrer, passer une porte (figure III.13), tourner dans un couloir (figure III.12), suivre un mur (figure III.9), éviter de petits obstacles, se décoller d'un mur (figure III.11). Il ralentit à l'approche d'une difficulté et évite les collisions de manière sûre.

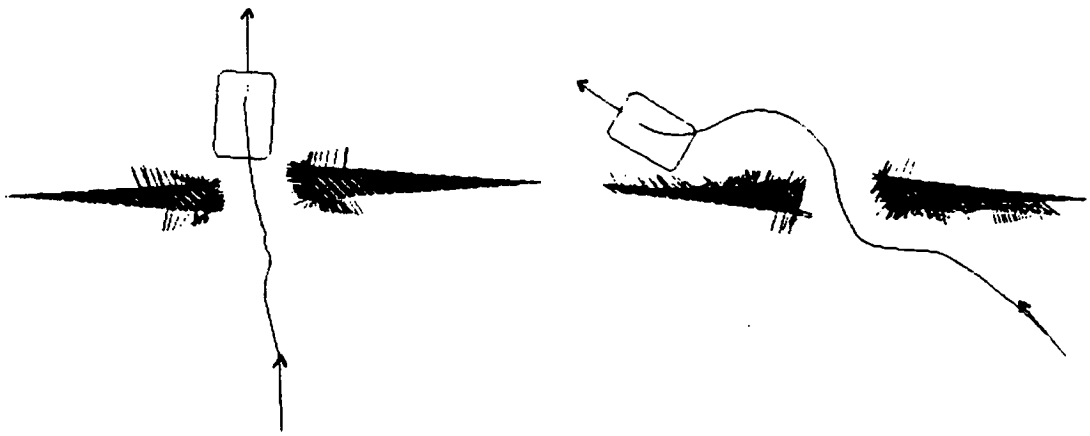


Figure III.13 : Passage d'une porte.

La nature des obstacles, contrairement à une méthode du type potentiel, ne semble pas trop géner le robot (figure III.14).

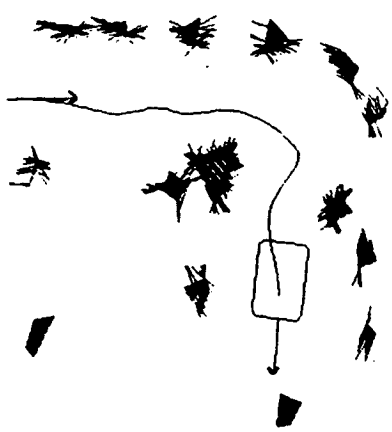


Figure III.14 : Passage d'un couloir à angle droit en environnement "morcelé".

On peut noter que le programme résout certains problèmes "globaux" alors que le robot ne connait que son environnement immédiat (figures III.15 et III.16). En effet les règles de comportment utilisées incitent le robot à contourner naturellement les obstacles.

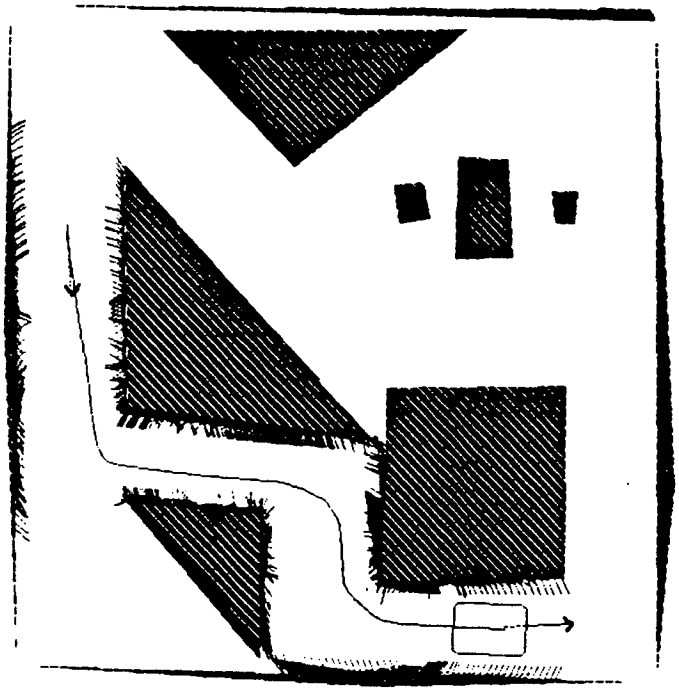


Figure III.15 : Passage d'une chicane.

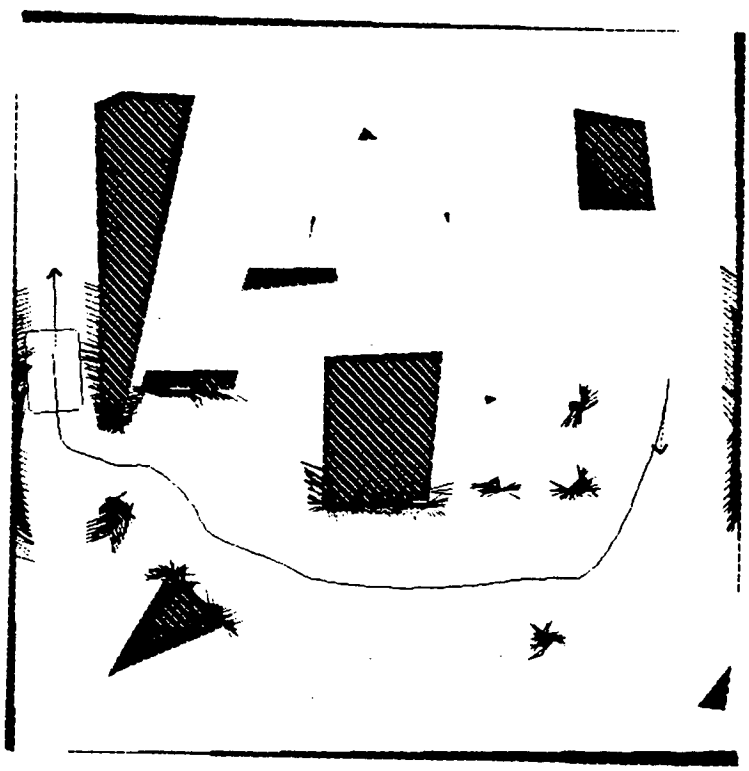


Figure III.16 : Navigation dans un environnement complexe.

5.2. Améliorations futures

La méthode employée qui repose sur un système de règles s'adaptant à l'environnement immédiat du robot est simple et modulaire. Il est possible par exemple de multiplier le nombre de zones d'influence du robot (quatre actuellement), de mémoriser les données sur plusieurs coups afin de se livrer à une reconstruction partielle de l'environnement (voir par exemple [*KTB87,Cro85,Gir84*]), d'ajouter de nouvelles règles compatibles avec l'ensemble des règles déjà existantes (actuellement les obstacles ne sont pas pris en compte s'ils se trouvent à une distance du robot supérieure à 1 m alors que la portée des capteurs est de 5 m).

IV. Couplage de l'algorithme de navigation avec un programme de planification globale

L'algorithme de navigation ci-dessus n'utilise qu'une connaissance très locale de l'environnement. Dans le cas où l'objectif final du robot est isolé de sa position initiale par un ensemble concave d'obstacles, le robot peut se trouver bloqué avant d'atteindre son but.

Pour remédier à ce problème, nous indiquons au robot une liste de positions intermédiaires atteignables : le programme local considère alors successivement chaque position de cette liste comme une position finale temporaire, qu'il abandonne pour passer à la position intermédiaire suivante quand il s'en est approchée suffisamment.

Les objectifs intermédiaires sont calculés à partir d'un modèle CAO des obstacles. Dans un premier temps nous construisons le diagramme de Voronoï de l'ensemble des obstacles. On rappelle que ce diagramme est constitué de l'ensemble des points situés à égale distance de deux obstacles. Il représente ainsi la topologie de l'espace libre (Figure IV.1).

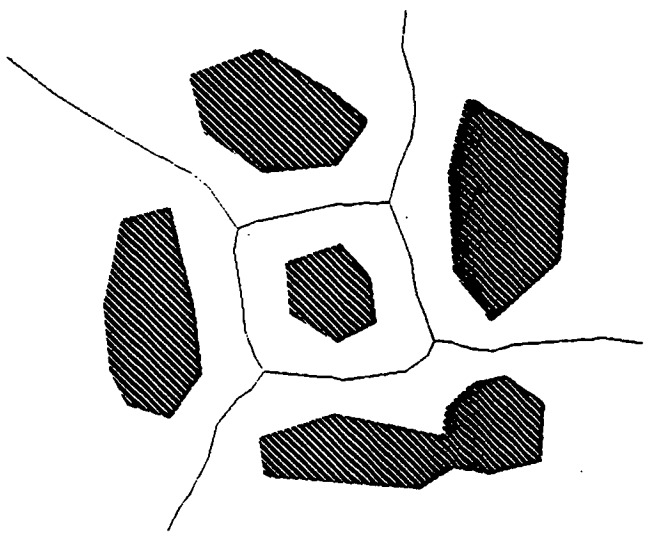
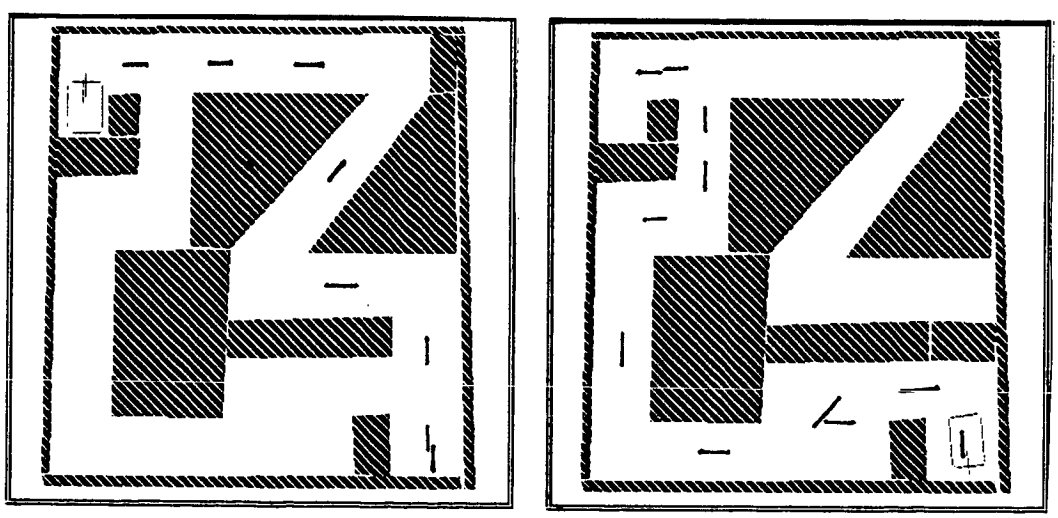


Figure IV.1. Diagramme de Voronoï d'un ensemble de polygones.

Un tel diagramme est composé de segments entre deux arêtes ou deux sommets des obstacles, et d'arcs de parabole entre un segment et un sommet. Pour la construction du diagramme, on se reportera à [*Kir79*] et [*LD81*].

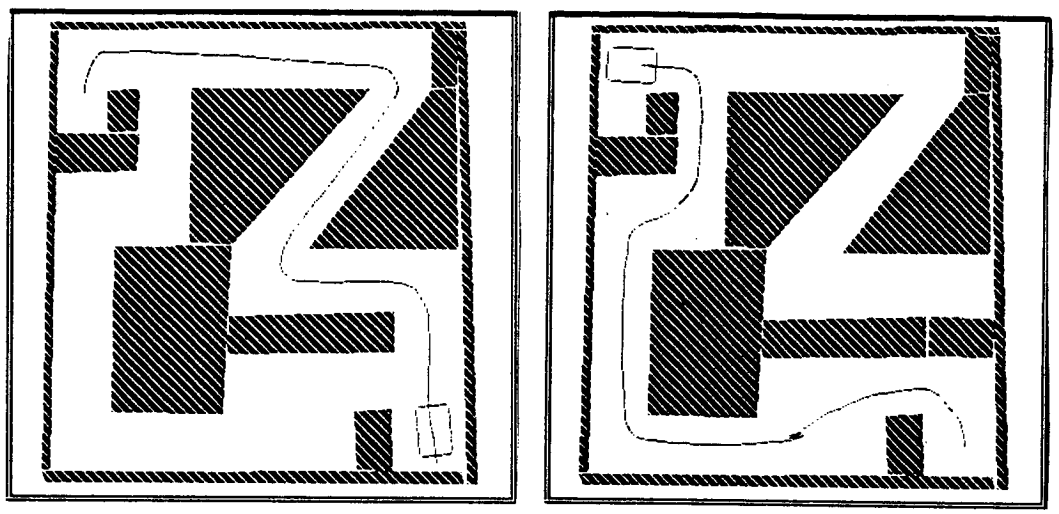
Dans un deuxième temps, compte tenu de la géométrie du robot, les sections de ce diagramme où il est clair que le robot ne peut pas passer sont éliminées (par exemple entre deux arêtes dont la distance est inférieure à la largeur du robot). Enfin, on construit un graphe de connexité dont les nœuds sont les configurations (x,y,θ) correspondant au milieu des segments du diagramme (ce qui correspond au centre des corridors dans la terminologie de Brooks), le coût de transition entre deux nœuds voisins étant calculé comme la distance sur le diagramme de Voronoï. Un chemin dans le graphe est donc représenté par une liste de configurations intermédiaires, qui sont successivement traitées par l'algorithme de navigation. On se reportera à [*TJ88*] pour une analyse détaillée du problème de la planification des trajectoires d'un robot mobile avec contrainte cinématique.

La figure IV.2 montre une séquence de positions intermédiaires calculées par le planificateur, et la figure IV.3 la trajectoire qui en résulte.



(a) (b)

Figure IV.2 : Configurations intermédiaires calculées par le programme de planning.



(a) (b)

Figure IV.3 : Trajectoire suivie par le robot.

Bien sûr, un tel découplage de la recherche d'une trajectoire entre un planificateur global et un générateur local de trajectoires basé sur des données fournies par les capteurs rend aléatoire la prédiction de bonne terminaison du déplacement dans un cas complexe. Dans [*FT87b*] une méthode systématique d'apprentissage de stratégies par un planificateur global, à partir des résultats de l'exécution d'une trajectoire calculée localement, a été introduite.

Bibliographie

[AB88] **F.Avnaim, J.D.Boissonnat.** "Polygon Placement under Translation and Rotation", Symp. on Theoretical Aspects of Computer Science, 1988.

[BLP85] **R.A.Brooks, T.Lozano-Pérez.** "A Subdivision Algorithm in Configuration Space for Findpath with Rotation", IEEE Trans. on Systems, Man and Cybernetics (SMC-15), pp 224-233, 1985.

[Bro83] **R.A.Brooks.** "Solving the Find-Path Problem by Good Representation of Free Space", IEEE Trans. on Systems, Man and Cybernetics (SMC-13), pp 190-197, 1983.

[Cro85] **J.Crowley.** "Navigation for an Intelligent Mobile Robot", IEEE Journal of Robotics and Automation, RA-1, pp 31-41, 1985.

[Dru85] **M.Drumheller.** "Mobile Robot Localization Using Sonar", A.I. Memo 826, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, 1985.

[Fly85] **A.M.Flynn.** "Redundant Sensors for Mobile Robot Navigation", Technical Report 859, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, 1985.

[FT87a] **B.Faverjon, P.Tournassoud.** "A Local Based Approach for Path Planning of Manipulators With a High Number of Degrees of Freedom", in Proceedings IEEE Int. Conf. on Robotics and Automation, pp 1152-1159, Raleigh, 1987, aussi rapport de recherche INRIA No 621.

[FT87b] **B.Faverjon, P.Tournassoud.** "The Mixed Approach for Motion Planning : Learning Global Strategies from a Local Planner", in Proceedings Int. Joint Conf. on Artificial Intelligence, pp 1131-1137, Milan, 1987.

[Gir84] **G.Giralt, R.Chatila, M.Vaisset.** "An Integrated Navigation and Motion Control System for Autonomous Multisensory Mobile Robots", Robotics Research 1, Brady et Paul éditeurs, MIT Press, pp 191-214, 1984.

[Kha78] **O.Khatib, J.F.Lemaître.** "Dynamic Control of Manipulators Operating in a Complex Environment", in Proceedings 3rd CISM-IFToMM, Udine, 1978.

[Kir79] **D.G.Kirkpatrick.** "Efficient Computation of Continuous Skeletons", 20ème FOCS, pp 18-27, 1979.

[KM85] **Y.Kanayama, N.Miyake.** "Trajectory Generation for Mobile Robots", in Proceedings 3rd International Symposium of Robotics Research, Gouvieux, 1985.

[KT86] **H.Krogh, E.Thorpe.** "Integrated Path Planning and Dynamic Steering Control for

Autonomous Vehicules", Robotics Institute, Carnegie-Mellon University, Pittsburg, 1986.

[KTB87] **D.J.Kriegman, E.Triendl, T.O.Binford.** "A Mobile Robot : Planning and Locomotion", in Proceedings IEEE Int. Conf. on Robotics and Automation, Raleigh, 1987.

[Lau86] **J.P.Laumond.** "Feasible Trajectories for Mobile Robots With Kinematic and Environment Constraints", in Proceedings Int. Conf. on Intelligent Autonomous Systems, Amsterdam, 1986.

[LD81] **D.T.Lee, R.L.Drysdale.** "Generalization of Voronoi Diagrams in the Plane". SIAM Journal of Computers, (10):73-87, 1981.

[LPW] **T.Lozano-Pérez, M. Wesley.** "An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles", Communications of ACM, pp 560-570, 1979.

[TJ88] **P.Tournassoud, O.Jehl.** "Motion Planning for a Mobile Robot with a Kinematic Constraint", in Proccedings IEEE Int. Conf. on Robotics and Automation, Philadelphia, 1988.

Annexe : Recherche d'une trajectoire en temps et énergie minimaux.

1. Problème posé

Partant d'une configuration du robot $(x,y,\theta) = (0,0,0)$ et visant la configuration (x_f,y_f,θ_f), est-il possible de trouver explicitement une trajectoire correspondant à un minimum de temps et d'énergie ?

2. Equations

Nous cherchons $\omega_1(t)$ et $\omega_2(t)$, vitesses de rotation des roues, telles que :

$$\theta' = (\omega_1 - \omega_2).r / (2.L)$$

$$x' = \cos(\theta).(\omega_1+\omega_2).r / 2$$

$$y' = \sin(\theta).(\omega_1+\omega_2).r / 2$$

$$x(0) = 0,\ y(0) = 0,\ \theta(0) = 0$$

$$x(T) = x_f,\ y(T) = y_f,\ \theta(T) = \theta_f,\ 0 < T$$

$$|\omega_1(t)| \text{ et } |\omega_2(t)| \leq \Omega_{max}$$

$$\inf J(\omega_1, \omega_2) = \gamma.T + \int_{0 \text{ à } T} (\alpha.(\omega_1(t)+\omega_2(t))^2+\beta.(\omega_1(t) - \omega_2(t))^2)\, dt$$

α est le coefficient de pondération pour l'énergie de translation,

β est le coefficient de pondération pour l'énergie de rotation,

γ est le coefficient de pondération pour la durée d'exécution.

La formulation de J est très générale. Par exemple si $\alpha = \beta$, l'intégrale s'interprète comme l'énergie dépensée par les moteurs.

3. Application du principe de Pontryagin (condition nécessaire)

L'hamiltonien vaut : $H = L + < p, A > = \alpha.(\omega_1+\omega_2)^2+\beta.(\omega_1 - \omega_2)^2 + < p, A >$

avec $p=(p_1, p_2, p_3)$ le vecteur adjoint,

et A le vecteur de composantes : $(\omega_1 - \omega_2).r / (2.L)$, $\cos(\theta).(\omega_1+\omega_2).r / 2$, $\sin(\theta).(\omega_1+\omega_2).r / 2$.

On a : $p' = -A^t_{(x,y,\theta)}\, p - L_y$

(ω_1, ω_2) réalise le minimum de $H(\omega_1, \omega_2, x, y, \theta, p, t)$

$H(T) = -\gamma$

De la première équation matricielle on déduit :

p_2 constante, p_3 = constante, et

$p_1' = (p_2.\sin(\theta) - p_3.\cos(\theta)).(\omega_1+\omega_2).r / 2.$

Trois cas sont à distinguer :

- le cas où $|\omega_1(t)|$ et $|\omega_2(t)| < \Omega_{max}$,
- le cas où $|\omega_1(t)| = \Omega_{max}$ et $|\omega_2(t)| < \Omega_{max}$ et le cas symétrique,
- le cas où $|\omega_1(t)| = \Omega_{max}$ et $|\omega_2(t)| = \Omega_{max}$.

Le troisième cas est un cas dégénéré, qui correspond à aller en ligne droite ou tourner sur place. Mais on peut tout de même remarquer que H reste constant dans ce cas.

Nous pouvons déjà remarquer que :

- $\omega_1(t)$ et $\omega_2(t)$ sont C^0 : en effet nous pouvons montrer directement en utilisant des théorèmes d'analyse numérique, que du fait de la forme du problème, $\omega_1(t)$ et $\omega_2(t)$ sont continues.
- le principe de Pontryagin assure la continuité du vecteur adjoint p(t).
- l'hamiltonien H est continu car composé de fonctions continues. D'autre part, nous pouvons directement vérifier sur les équations que dans chaque cas H est constant par morceaux, et, comme H est continu, H est constant globalement au cours du mouvement.

4. Calcul dans le cas : $|\omega_1(t)|$ et $|\omega_2(t)| < \Omega_{max}$

Le principe se traduit par :

p_2 = constante, p_3 = constante

$H(T) = -\gamma$

$p_1' = (p_2.\sin(\theta) - p_3.\cos(\theta)).(\omega_1+\omega_2).r / 2$

$dH(\omega_1, \omega_2) = 0$

On en déduit :

$$\alpha.(\omega_1(t)+\omega_2(t))^2+\beta.(\omega_1(t) - \omega_2(t))^2 = \gamma$$

$$r^2.(p_2.\cos(\theta)+p_3.\sin(\theta))^2 / (64.\alpha.\beta.L^2)+ \theta'^2 = r^2.\gamma/ (4.L^2.\beta)$$

et des conditions aux limites qui déterminent les constantes p_2, p_3 et T.

Le mouvement doit s'effectuer à énergie constante, cette constante étant déterminée par l'importance que l'on donne à la durée d'exécution (coefficient γ). L' angle θ suit l'équation d'un oscillateur harmonique.

5. Calcul dans le cas : $\omega_1(t) = \Omega_{max}$, $|\omega_2(t)| < \Omega_{max}$

Les trois autres cas symétriques se traitent de la même façon.

Le principe se traduit par :

$$p_2 = \text{constante}, \; p_3 = \text{constante}$$

$$H(T) = -\gamma$$

$$p_1' = (p_2.\sin(\theta)- p_3.\cos(\theta)).(\omega_1+\omega_2).r / 2$$

$$\partial_{\omega 2} H (\omega_1, \omega_2) = 0 \text{ et } \omega_1(t) = \Omega_{max}$$

La dernière équation s' écrit :

$$2.\alpha.(\omega_2+ \Omega_{max})+2.\beta.(\omega_2- \Omega_{max})+r.(-p_1 / L + p_2.\cos(\theta)+p_3.\sin(\theta)) / 2 = 0$$

En la dérivant par rapport à t :

$$\omega_2' = r^2. \Omega_{max}.(p_2.\sin(\theta) - p_3.\cos(\theta)) / (4.L(\alpha+\beta))$$

Or : $\theta'' = - \omega_2'.r / 2.L$, donc l'angle θ suit l'équation d'un oscillateur harmonique.

6. Raccordements

Dans le cas $|\omega_1(t)|$ et $|\omega_2(t)| < \Omega_{max}$, on a :

$$\alpha.(\omega_1(t)+\omega_2(t))^2+\beta.(\omega_1(t) - \omega_2(t))^2 = \gamma$$

donc la commande décrit une ellipse. Ω_{max} est faible et le temps est un facteur prépondérant, nous pouvons donc supposer que : $4.\Omega_{max}^2.Max(\alpha,\beta) < \gamma$. Donc le carré $|\omega_1(t)| = |\omega_2(t)| = \Omega_{max}$ est inclus dans l'ellipse et, seul le cas décrit en 5 est réalisable.

La trajectoire peut être composée de plusieurs arcs du type 5. Comme $\omega_1(t)$ et $\omega_2(t)$ sont continues, ces changements d' arcs ne peuvent s'effectuer que dans deux cas :

- si la courbure est nulle ($\omega_1 = \omega_2 = \Omega_{max}$)
- ou si le rayon de courbure est nul ($\omega_1 = -\omega_2 = {}^{+}_{-} \Omega_{max}$). Ceci correspond le plus souvent à un point de rebroussement sur la trajectoire.

7. Conditions aux limites

Dans le cas d'un seul arc nous avons trois inconnues (p_2,p_3,T) et trois équations implicites aux limites :

$$\theta(T) = \theta_f$$

$$x_f = r/2 . \int_{0 \text{ à } T} \cos(\theta).(\omega_1(t)+\omega_2(t)).dt$$

$$y_f = r/2 . \int_{0 \text{ à } T} \sin(\theta).(\omega_1(t)+\omega_2(t)).dt$$

$\omega_1(t)$ et $\omega_2(t)$ ne dépendent que de p_2,p_3.

8. Etude du rayon de courbure des trajectoires

8.1. Paramétrage de la trajectoire

Le rayon de courbure algébrique est défini par :

$$R(\theta) = L \frac{\omega_1+\omega_2}{\omega_1 - \omega_2}$$

et la trajectoire est donnée par :

$$\begin{cases} \dfrac{dx}{d\theta} = \cos(\theta).R(\theta) \\ \dfrac{dy}{d\theta} = \sin(\theta).R(\theta) \end{cases}$$

On pose pour la suite : $p_2.\cos(\theta)+p_3.\sin(\theta) = A.\sin(\theta+\phi)$

$$f(\theta) = L \left(\frac{2.\Omega_{max}.\sqrt{\alpha+\beta}}{\sqrt{4.\alpha.\Omega_{max}^2+\gamma+r.A.\Omega_{max}.\sin(\theta+\phi)}} - 1 \right)$$

8.2. Rayons de courbure

Premier cas : $\omega_1(t) = \Omega_{max}$, $|\omega_2(t)| < \Omega_{max}$

Des équations $H = -\gamma$ et $\partial_{\omega 2} H = 0$, et en éliminant p_1, nous obtenons :

$(\alpha+\beta).(\Omega_{max} - \omega_2)^2 = 4.\alpha.\Omega_{max}^2 + \gamma + r.\Omega_{max}.A.\sin(\theta+\phi)$

$R(\theta) = f(\theta)$

Deuxième cas : $\omega_1(t) = -\Omega_{max}$, $|\omega_2(t)| < \Omega_{max}$

$(\alpha+\beta).(\Omega_{max} + \omega_2)^2 = 4.\alpha.\Omega_{max}^2 + \gamma - r.\Omega_{max}.A.\sin(\theta+\phi)$

$R(\theta) = f(\theta+\pi)$

Troisième cas : $\omega_2(t) = \Omega_{max}$, $|\omega_1(t)| < \Omega_{max}$

$(\alpha+\beta).(\Omega_{max} - \omega_1)^2 = 4.\alpha.\Omega_{max}^2 + \gamma + r.\Omega_{max}.A.\sin(\theta+\phi)$

$R(\theta) = -f(\theta)$

Quatrième cas : $\omega_2(t) = -\Omega max$, $|\omega_1(t)| < \Omega max$

$(\alpha+\beta).(\Omega_{max} + \omega_1)^2 = 4.\alpha.\Omega_{max}^2 + \gamma - r.\Omega_{max}.A.\sin(\theta+\phi)$

$R(\theta) = -f(\theta+\pi)$

8.3. Etude de f(θ)

$f(\theta)$ est définie si : $4.\alpha.\Omega_{max}^2 + \gamma + r.\Omega_{max}.A.\sin(\theta+\phi) > 0$.

Le long d' un même arc le signe de $f(\theta)$ ne change pas :

- Si $f(\theta) \geq 0$ alors $-4.\alpha.\Omega_{max}^2 - \gamma \leq r.\Omega_{max}.A.\sin(\theta+\phi) \leq 4.\beta.\Omega_{max}^2 - \gamma \leq 0$
- Si $f(\theta) \leq 0$ alors $r.\Omega_{max}.A.\sin(\theta+\phi) \geq 4.\beta.\Omega_{max}^2 - \gamma \leq 0$

8.4. Etude des raccords

A et ϕ sont conservés (p(t) continu).

Si le raccord consiste à passer de $\omega_1(t) = \Omega_{max}$ à :

- $\omega_2^*(t) = -\Omega_{max}$: la continuité des ω_i au raccord s' écrit :

  $\omega_2 = -\Omega_{max}$ et $\omega_1^* = \Omega_{max}$, d' où : $r.\Omega_{max}.A.\sin(\theta+\phi) = 0 = 4.\beta.\Omega_{max}^2 - \gamma$

  ce qui n' est pas possible.

- $\omega_1^*(t) = -\Omega_{max}$: c'est impossible.

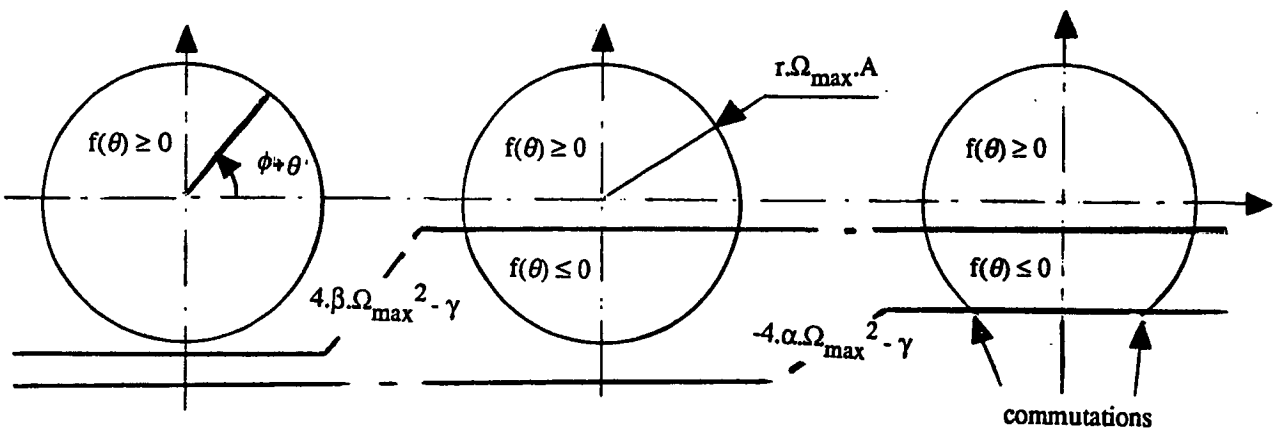
- $\omega_1^*(t) = \Omega_{max}$: c' est toujours le même arc.

- $\omega_2^*(t) = \Omega_{max}$: $\omega_2 = \Omega_{max}$ et $\omega_1^* = \Omega_{max}$, d' où :

  $r.\Omega_{max}.A.\sin(\theta+\phi) = -4.\alpha.\Omega_{max}^2 - \gamma$: ce cas est possible et correspond à un raccord à courbure nulle où $R(\theta)$ change de signe.

8.5. Résultat sur le nombre de raccords

Nous venons de montrer que les seuls raccords possibles sont à courbure nulle et que le rayon change alors de signe. Les cas possibles sont représentés ci-dessous :



θ est monotone le long d'un même arc, ce qui signifie qu'il est impossible de changer de sens sur les cercles ci-dessus, sauf aux points de commutations. D'autre part θ ne peut pas prendre les valeurs correspondant aux points de rebroussement. Donc il y a au plus un raccord, et dans le cas où il y en a un, chaque arc ne peut pas être parcouru en θ de plus de $\pi/2$. Ceci confirme l'intuition qui consiste à chercher une trajectoire comportant deux arcs de cercle ou deux tronçons de clotoïdes.

Dans le cas où il y a une commutation, l'accélération des roues n'est pas continue (de même qu'en utilisant des clotoïdes). Si θ_c est l'angle de la tangente au raccord et si R(θ) est le rayon sur le premier arc, les conditions aux limites s'écrivent :

$$x_f = \int_{\theta \text{ à } \theta c} \cos(\theta).R(\theta).dt - \int_{\theta c \text{ à } \theta 1} \cos(\theta).R(\theta).dt$$

$$y_f = \int_{\theta \text{ à } \theta c} \sin(\theta).R(\theta).dt - \int_{\theta c \text{ à } \theta 1} \sin(\theta).R(\theta).dt$$

$$r.\Omega_{max}.A.\sin(\theta_c+\phi) = -4.\alpha.\Omega_{max}^2 - \gamma$$

Ces trois équations permettent de calculer les trois variables θ_c, ϕ, A.

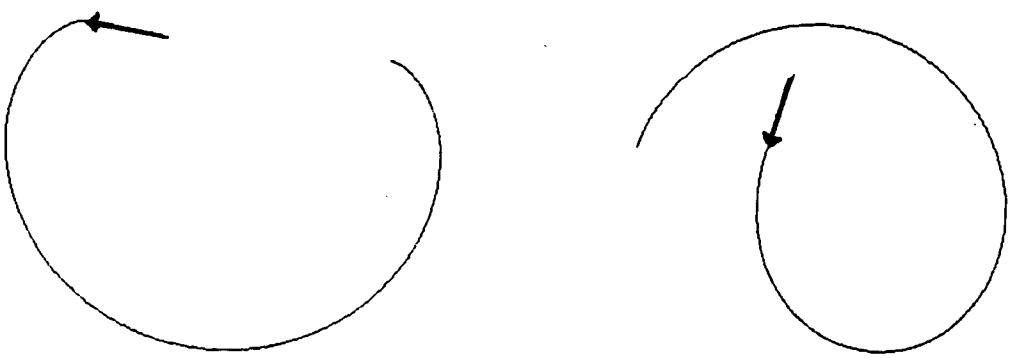
Le problème restant ouvert est de déterminer dans quels cas il y a une commutation.

9. Allures des trajectoires

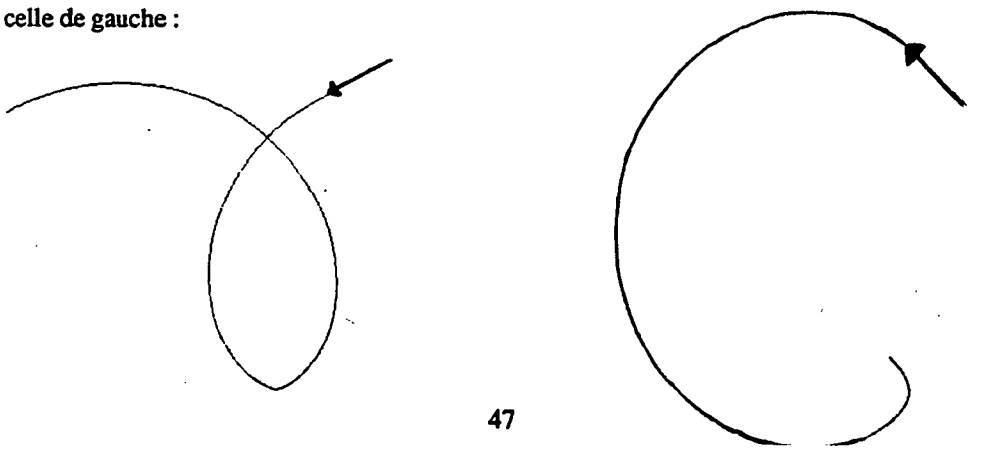
Des tronçons de trajectoire en temps et énergie minimum sont représentés ci-dessous.

Tronçons de trajectoires s'arrêtant au point de commutation :

Tronçons de trajectoires sur lesquelles il est impossible de commuter ; les courbes dessinées représentent une variation de θ d'une amplitude de 2π :



Les deux courbes ci-dessous ont les mêmes paramètres, sauf ϕ =0 pour celle de droite et ϕ = 1.5 pour celle de gauche :



Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique