Projet de Fin d'Etudes

CONCEPTION D'UN PLANIFICATEUR DE TRAJECTOIRES POUR UN ROBOT MOBILE

Matthieu Massonneau

Février 2003

Professeurs responsables du PFE :

ENSAM : M. Christophe GIRAUD-AUDINE
M. Jean-Luc BAUCHAT

SUPELEC : M. Stéphane VIALLE

Remerciements

Mes remerciements vont tout particulièrement à :

Monsieur Stéphane VIALLE, chef de travaux à SUPELEC, pour ses conseils en programmation

Monsieur Christophe GIRAUD-AUDINE, chercheur et enseignant à l'ENSAM, pour son aide sur le logiciel Matlab

Monsieur Jean-Luc BAUCHAT, directeur adjoint et enseignant à l'ENSAM, pour son aide sur la partie mathématique

Monsieur Ali SIADAT, chercheur et enseigant à l'ENSAM, pour ses conseils d'un point de vue robotique

Monsieur Hervé FREZZA-BUET, chercheur et enseignant à SUPELEC, pour ses conseils sur l'utilisation de la bibliothèque de commandes du Koala

Monsieur Olivier MENARD, thésard à SUPELEC, pour son aide diverse et variée

L'équipe technique de SUPELEC, tout particulièrement Monsieur FERNANDES, pour son aide quant à la conception du système de mesure d'erreur

L'équipe système informatique de SUPELEC

Table des matières

Vue d'ensemble du projet

Sujet

Ce Projet de Fin d'Etudes réalisé en collaboration entre le campus de Metz de SUPELEC et l'ENSAM de Metz a pour objet l'étude d'un système de planification de trajectoire pour un robot mobile Koala de la société K-Team.

Le robot doit être capable de calculer sa trajectoire entre son point de départ et une porte située dans un des murs de l'environnement.

La localisation de son point de départ par le robot peut réutiliser les travaux effectués jusqu'à maintenant sur le Koala, à savoir l'étude sur la détection de marques de recalage réalisée par *Vincent Rieger* [2] lors de son Studienarbeit, puis l'étude sur la localisation de *Cyrille Roussel* [1] au cours de son stage de DEA.

Contexte

L'environnement dans lequel évolue le robot est structuré ($2m \times 2m$) et plan. Il est délimité par des murs d'environ 30cm de hauteur. Le sol est recouvert d'un matériau s'apparentant au linoléum, donc relativement glissant.

L'environnement (cf. figure 1) est inconnu pour le robot mais balisé grâce à des marques P-Similaires situées sur les murs. La porte est simulée grâce à un pourtour sombre par rapport à la couleur des murs. Cet environnement ne comporte pas d'obstacles.

Le robot Koala de la société K-Team (cf. figure 2) dispose de six roues qui lui permettent de se déplacer à la manière d'un tank. Les pneus sont en caoutchouc souple. Pour prévenir toutes collisions avec l'environnement, le robot est équipé de seize capteurs infrarouges disposés sur son pourtour. Le Koala possède également une caméra C.C.D, montée sur une tourelle deux axes. Cette caméra fournit des images à la carte d'acquisition de l'ordinateur embarqué (PC104) ou externe (PC classique). Enfin un capteur odométrique permet de donner la position du robot par rapport à sa position précédente. L'odométrie est une méthode relative fondée sur la relation entre la rotation des roues et le déplacement du robot. La vitesse de rotation de roues est

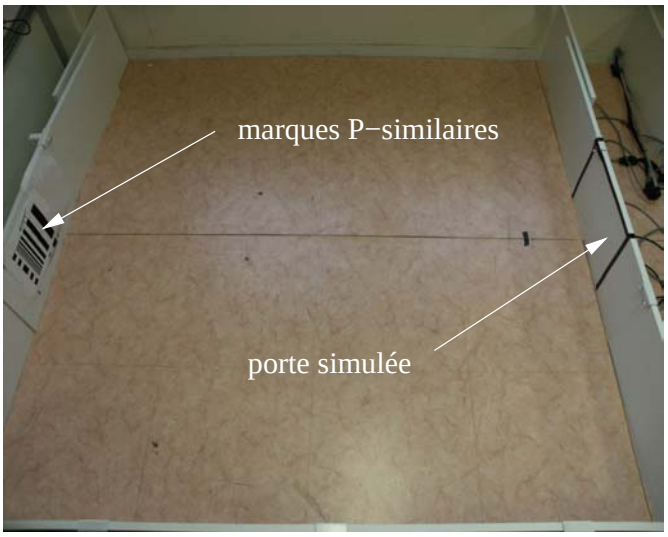


FIG. 1 – Environnement dans lequel évolue le robot

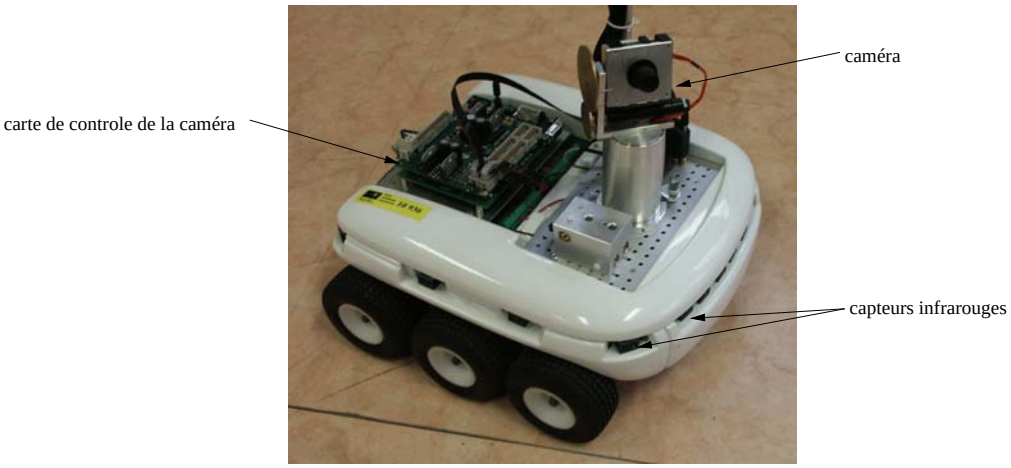


FIG. 2 – Robot Koala

intégrée afin d'en déduire le déplacement. On verra apparaître une erreur de localisation lorsque les roues glissent sur le sol, car le capteur odométrique relève un déplacement alors que le robot n'a pas bougé.

D'un point de vue navigation, le robot Koala est un système en boucle ouverte. Cela implique que l'erreur due au capteur odométrique n'est pas corrigeable.

Position du problème

Le programme de planification de trajectoire doit permettre au Koala de se rendre devant une porte, en minimisant l'erreur de position à l'arrivée, erreur provoquée par le glissement des roues sur le sol. Le critère d'optimisation de la trajectoire est donc la minimisation du glissement des roues.

Pour cela, quelques hypothèses ont été faites :
- la trajectoire doit avoir un rayon de courbure minimum ;
- la trajectoire doit être lisse, c'est à dire être continue, dérivable et ne comporter que des accélérations bornées [9],[10] ;
- le robot n'effectue pas de marche arrière.

Travail à effectuer

Le travail à effectuer se divise en cinq parties :
- la recherche de trajectoires réalisables par le Koala et adaptées à l'environnement ;
- la construction des trajectoires retenues ;
- la simulation des trajectoires pour valider les algorithmes de construction ;
- l'implantation du planificateur ;
- les tests pour valider les solutions retenues et mesurer l'erreur à l'arrivée.

Chapitre 1

Planification de trajectoires : état de l'art

Un problème important en robotique mobile est la planification de trajectoires permettant au robot d'atteindre une position particulière à partir de sa position actuelle. La trajectoire représente l'ensemble des points par lesquels passe le centre du robot, au cours de son mouvement de sa position initiale à sa position finale. Or il peut exister plusieurs trajectoires pour se rendre à cette position finale en fonction des contraintes imposées par l'environnement ou par le robot. Le choix et la construction de cette trajectoire sont appelés *planification de trajectoire*.

Il existe différentes méthodes de planification [3]. Certaines sont plus particulièrement utilisées dans le cas d'environnements à la forme complexe, ou comportant des obstacles. Deux cas de figure se présentent alors :

- L'environnement est connu à priori. On communique au robot les données concernants les limites de l'environnement, la position des obstacles, la forme des obstacles, ...
- L'environnement n'est pas connu à priori. Le robot se crée lui-même une base de données pour garder des traces de la trajectoire accomplie et des obstacles localisés.

Les algorithmes permettant de définir les points de passage pour construire la trajectoire sont différents en fonction du cas de figure.

1.1 Planification avec connaissances à priori

Dans ce cas l'environnement est représenté par un *modèle géométrique* [3] qui contient des informations comme la forme ou la position des obstacles. Ce modèle est discrétisé pour obtenir un *modèle topologique*, les éléments discrets correspondant à un ensemble de placements possibles du robot. Ce

modèle permet de construire un graphe ayant pour noeuds les placements possibles du robot. Ces noeuds sont choisis du point de départ au point d'arrivée. Le choix entre les noeuds est fait en utilisant un algorithme spécifique. Par exemple, l'algorithme exhaustif de Dijkstra [5] développe toutes les possibilités qui s'offrent à partir d'un noeud, et choisit le suivant en minimisant un critère de coût (la longueur à parcourir, la consommation d'énergie, ...).

Au contraire, les algorithmes heuristiques, comme l'algorithme A* [4], ne vont développer qu'un seul noeud parmi les noeuds suivants possibles. Ils considèrent une fonction d'évaluation *f(n)* correspondant au coût du chemin passant par ce noeud. Le noeud développé en priorité est celui dont la fonction d'évaluation est la plus faible. Cette fonction est composée de deux termes : *g(n)* est une fonction de coût du chemin reliant le noeud source au noeud n et *h(n)* une fonction de coût caractérisant la connaissance du coût minimal du chemin du noeud n au but. On parle de fonction heuristique. En général le critère d'optimisation utilisé constitue la distance minimale. La fonction heuristique estimée utilisée est la distance euclidienne entre le noeud n et le but, qui est connu en général. Cet algorithme a l'avantage de ne pas développer toutes les solutions possibles, d'où un gain de temps et d'espace mémoire utile.

1.2 Planification sans connaissances à priori

Dans ce cas, on planifie la trajectoire dans un environnement inconnu. La planification correspond alors à gérer dynamiquement les informations issues des capteurs d'environnement au cours du déplacement du mobile afin d'atteindre le plus rapidement possible un but connu dans un référentiel global ou local. Au départ le robot se dirige vers le point d'arrivée jusqu'à rencontrer un obstacle. Deux algorithmes se distinguent alors.

L'algorithme BUG1 [3] fait faire au robot le tour complet de l'obstacle et lui fait repérer le point du périmètre le plus proche du point d'arrivée. Le robot continue à tourner autour de l'obstacle jusqu'à revenir à ce point, et de là il repart en ligne droite vers l'objectif. S'il rencontre un autre obstacle, le robot refait la même opération.

L'algorithme BUG2, quant à lui, fait suivre au robot la droite reliant le point de départ au point d'arrivée dès qu'il la rencontre. Si le robot rencontre un obstacle, il le contourne jusqu'à rencontrer de nouveau la droite. Et ainsi de suite jusqu'à arriver au but.

Cependant les algorithmes présentés ci-dessus ne tiennent pas compte des contraintes cinématiques liées au robot. Il existe pour cela des méthodes de planification adaptées.

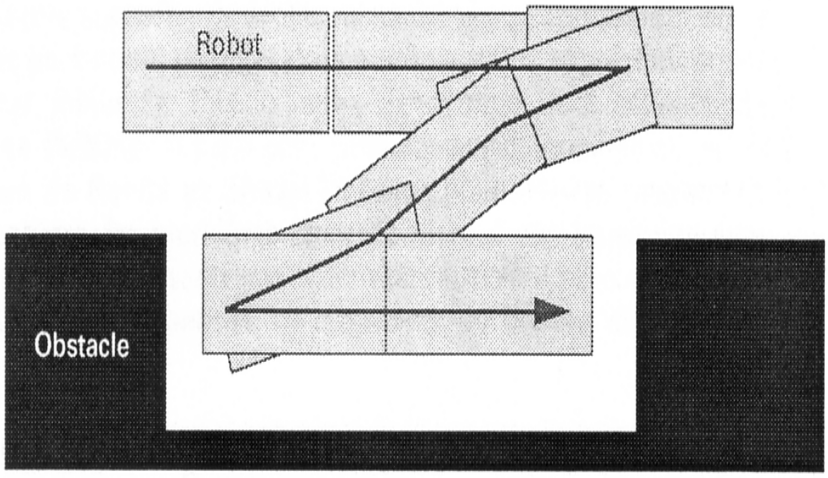


FIG. 1.1 – Exécution d'un créneau pour un véhicule non-holonome

1.3 Planification sous contraintes cinématiques

En général, le nombre de trajectoires possibles entre deux positions quelconques est limité par les contraintes cinématiques du robot (pour certains mobiles, par exemple, la rotation est obligatoirement dépendante d'une translation). La cinématique des robots à roues ne permet pas de suivre une trajectoire quelconque. Cela provient du fait que le nombre de variables définissant les configurations du mobile est inférieur au nombre de degrés de liberté. Un tel système est dit *non-holonome*. Le terme *configuration* désigne la position du robot mobile (deux ou trois coordonnées suivant que le robot se déplace dans le plan ou dans l'espace) et son orientation par rapport au référentiel de l'environnement (un ou trois angles). Ce genre de robot n'est pas capable de réaliser tous les types de trajectoire. C'est le cas d'un véhicule automobile forcé d'exécuter un "créneau" pour se garer (cf. figure 1.1).

En plus de la non-holonomie, le robot est affecté d'une contrainte supplémentaire : la trajectoire doit avoir un rayon de courbure minimum. Ceci est dû au fait que certains robots ne peuvent pas effectuer de rotation sur place, ou qu'une variation brutale des vitesses pour effectuer un arc arc de cercle de faible rayon est préjudiciable à la mécanique et à la précision du mouvement.

Parmi les trajectoires respectant ces contraintes, on distinguera plus particulièrement :

– les trajectoires lisses (cf. paragraphe 1.3.1) ;

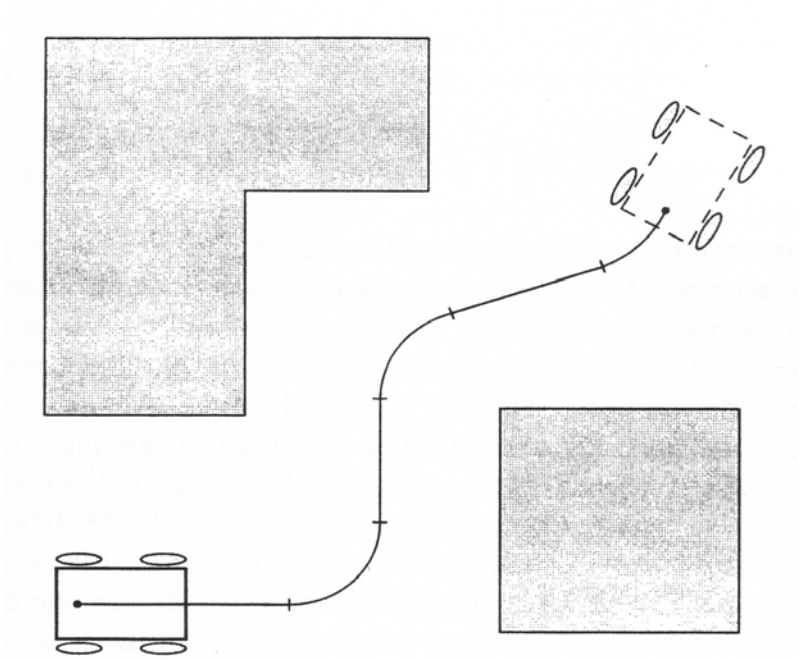


FIG. 1.2 – Exemple de trajectoire lisse

– les trajectoires construites à partir de courbes de *Reeds et Shepp* [7].

1.3.1 Les trajectoires lisses

Une trajectoire lisse doit satisfaire aux critères suivant [9],[10] :
– la trajectoire est continue ;
– la trajectoire est dérivable et ne comprend donc pas de points anguleux ;
– les accélérations en translation et en rotation sont bornées.

La continuité de la trajectoire est une condition évidente. Si la condition de dérivabilité n'est pas respectée et que la trajectoire comporte des points anguleux, le robot est soumis à des variations brusques de vitesse, qui peuvent provoquer le glissement des roues. C'est pour cette même raison que les accélérations sont bornées.

Ces trajectoires sont constituées de séquences d'éléments de base. Ceux-ci sont le plus souvent des segments de droite raccordés entre eux par des segments curvilignes (cf. figure 1.2). L'avantage d'utiliser ces éléments de base est que la commande du robot est alors simple à mettre en place.

Etudions maintenant quelques courbes pouvant servir à construire une trajectoire en utilisant ce principe.

FIG. 1.3 – Exemple de courbe de Dubins

Courbes de Dubins

Une courbe de Dubins [6] est une trajectoire entre deux points (départ et arrivée) associés à deux orientations, et décrite par un *mot*. Un *mot* est un ensemble de lettre C et S (C désigne un arc de cercle, S un segment de droite) (cf. figure 1.3).

Dubins adopte la notation suivante :
- un arc de cercle dans le sens trigonométrique par rapport à l'orientation du vecteur de départ est représenté par la lettre L ;
- un arc de cercle dans le sens antitrigonométrique par la lettre R ;
- chaque lettre majuscule S, L et R sont suivies par une lettre minuscule indiquant la longueur du segment ou de l'arc.

Dubins prouve qu'entre deux configurations de départ et d'arrivée, il existe une courbe sans point de rebroussement et de longueur minimale constituée de la succession : arc de cercle, segment de droite, arc de cercle (Ct Su Cv), ou de la succession de trois arcs de cercle (Ct Cu Cv).

Ces deux types de trajectoire définissent six mots qui permettent de relier deux points (avec une orientation associée) de manière optimale en distance (sans manoeuvres) : LtSuRv LtSuLt RtSuLv RtSuRv LtRuLv RtLuRv.

La détermination de la courbe consiste à tracer les quatre cercles tangents aux configurations initiales et finales, à calculer les tangentes entre ces cercles et à garder la trajectoire de longueur minimale réalisable sans rotation du robot sur lui-même (cf. figure 1.3). La rayon de courbure de la trajectoire correspond alors au rayon des cercles de construction.

Dans un environnement comportant des obstacles, chaque sommet d'obstacle est considéré comme un sous-but à atteindre. On peut ensuite discrétiser le problème pour obtenir un ensemble de configurations représentant chacune un noeud d'un graphe. Les noeuds sont reliés entre eux par des

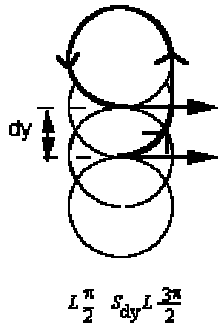


FIG. 1.4 – Déplacement dy par une courbe de Dubins

courbes de Dubins. Lorsque le graphe est construit, on utilise un algorithme de recherche de chemin, de type Dijkstra par exemple. La trajectoire obtenue est sans manoeuvres, mais elle n'est pas globalement de longueur minimale.

Sur l'exemple de la figure 1.4, nous remarquons les limites des courbes de Dubins ; en effet, pour un petit déplacement dy des configurations de départ et d'arrivée ayant la même abscisse, la courbe associée n'est pas optimale en longueur. Lorsque dy tend vers 0 la longueur de la courbe de Dubins devrait tendre aussi vers 0, ce qui n'est pas le cas.

Courbes de Mouaddib

Les courbes de Mouaddib [8] permettent aussi de construire des trajectoires lisses. Elles sont constituées d'une succession d'un segment de droite, d'un arc de cercle et d'un segment. Tout comme pour une courbe de Dubins, les segments de droite et l'arc de cercle sont tangents, ce qui définit une trajectoire sans discontinuités et dérivable en tout point (cf. figure 1.5). Le rayon de courbure de la trajectoire est celui du cercle, comme pour les courbes de Dubins.

L'inconvénient majeur de telles courbes est qu'elles ne définissent pas de trajectoires lisses pour toutes les configurations de départ. Dans l'exemple de la figure 1.6, le robot devra effectuer une rotation sur lui-même avant de parcourir la trajectoire jusqu'au but. Pour un robot ne pouvant pivoter sur lui-même, il lui est impossible de rejoindre le point d'arrivée prévu.

Courbes à courbure continue

Pour les courbes présentées ci-dessus, la courbure n'est pas continue. Aux raccordements, elle passe brutalement de zéro pour un segment de droite à $1/R$ pour un arc de cercle de rayon R. Pour assurer la continuité de la cour-

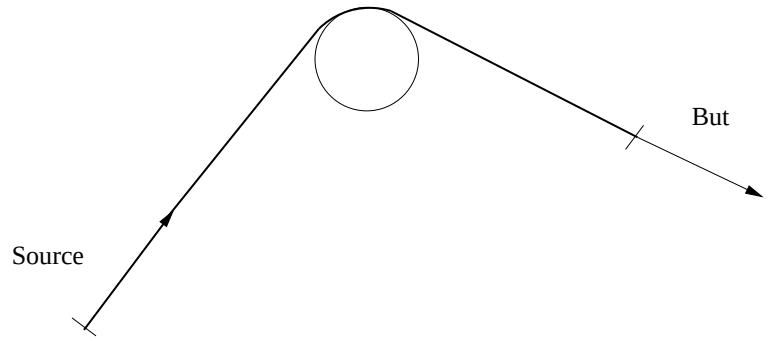


FIG. 1.5 – Exemple de courbe de Mouaddib

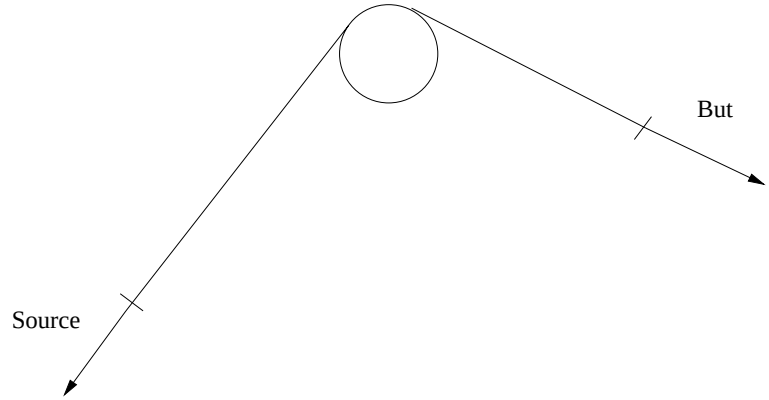


FIG. 1.6 – Limites des courbes de Mouaddib

bure, on peut remplacer les arcs de cercle par des arcs de spirale [9]. Un type de spirale souvent utilisé est la *clothoïde*. Sa courbure est une fonction linéaire de la distance le long de la trajectoire. En assemblant deux *clothoïdes*, on obtient une courbe avec une courbure qui croît linéairement dans sa première moitié de zéro à une valeur maximale, et ensuite décroît dans le sens inverse dans sa seconde moitié. Ainsi, deux clothoïdes permettent des jonctions avec des segments rectilignes sans discontinuité de la courbure.

Un autre type de trajectoire curviligne à courbure continue est le *spline polaire* dont les points sont facilement calculables en coordonnées polaires.

1.3.2 Courbes de Reeds et Shepp

Les courbes de Reeds et Shepp [7] définissent une trajectoire de longueur minimale en acceptant les manoeuvres. Ces courbes sont constituées au maximum de cinq éléments avec en plus deux points de rebroussement. En considérant des arcs gauches ou droits selon les directions, on obtient un

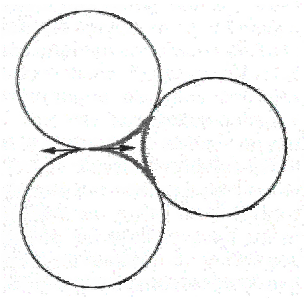


FIG. 1.7 – Exemple de courbe de Reeds et Shepp

nombre de 48 courbes possibles. Par exemple la figure 1.7 représente une courbe de Reeds et Shepp permettant de réaliser un demi-tour.

Reeds et Shepp reprennent la notation de Dubins ; à chaque arc de cercle est associé un sens de déplacement, C+ pour une marche avant par rapport au robot, et C- pour une marche arrière. Cette notation est étendue pour les segments de droite (S : S+, S-) qui décomposent les courbes. Reeds et Shepp représentent les points de rebroussement à l'aide de deux barres verticales.

Ct peut signifier un arc de cercle en marche avant à gauche Lt+, ou en marche arrière Lt-, ou le corollaire à droite Rt+, Rt-.

Chapitre 2

Planification de la trajectoire dans le cas du Koala

Ce sont l'environnement et la structure cinématique du robot qui vont déterminer laquelles des méthodes de planification présentées au chapitre 1 est la plus adaptée au cas qui nous intéresse.

2.1 Choix d'une courbe pour construire la trajectoire

L'environnement dans lequel évolue le Koala ne comporte pas d'obstacles et ses limites sont rectangulaires, donc de forme simple. Il n'est pas nécessaire d'utiliser un algorithme de recherche de chemin entre les obstacles. Le robot doit juste se rendre du point initial au point final en respectant les contraintes cinématiques et en essayant de minimiser l'erreur de position à l'arrivée. On se trouve donc dans le cas de la *planification sous contraintes cinématiques* [4].

Les courbes les plus intéressantes sont celles présentées au paragraphe 1.3, à savoir :

- les courbes permettant de construire des trajectoires lisses (courbes de Dubins, courbes de Mouaddib et courbes avec arcs de spirale) ;
- les courbes de Reeds et Shepp.

La trajectoire doit s'efforcer de minimiser l'erreur du capteur odométrique, en évitant le glissement des roues sur le sol. Pour cela on impose les conditions suivantes :

- Le rayon de courbure est minoré (borne à déterminer expérimentalement) ; pour la courbe servant à construire la trajectoire, le rayon de courbure doit donc être une fonction connue.

	Avantages	Inconvénients
Trajectoire de type 1	- longueur optimisée	- départ et arrivée en virage
Trajectoire de type 2	- arrivée et départ en ligne droite - un seul virage	- trajectoire non réalisable dans tous les cas

TAB. 2.1 – Tableau comparatif des trajectoires de type 1 et de type 2

– Les manoeuvres ne sont pas autorisées.

Les courbes de Reeds et Shepp nécessitant des manoeuvres, elles ne sont pas utilisables dans le cadre de cette étude.

En revanche les trajectoires lisses sont intéressantes pour réduire le glissement des roues (pas de discontinuité, dérivables en tout point et le rayon de courbure minimum est connu car imposé lors de la construction). Il faut maintenant choisir parmi les courbes de Dubins, les courbes de Mouaddib et les courbes à courbure continue, celles qui sont le plus adaptées.

Les courbes à courbure continue ne sont pas exploitables dans le cas du robot Koala. En effet pour obtenir un mouvement fluide, il a été décidé d'utiliser la commande en vitesse pour piloter le robot. Or, dans ce cas, le Koala ne permet pas de contrôler l'accélération de ses roues : elle est toujours à sa valeur maximale. De ce fait une courbe comme un arc de spirale, comportant beaucoup de variations du rayon de courbure, donc de vitesse des roues, engendrera plus d'erreur due au glissement qu'un arc de cercle (cf. paragraphe 5.2).

Enfin on notera que les courbes de Bézier ont été écartées pour deux raisons : tout d'abord le contrôle du rayon de courbure d'une courbe de Bézier est encore à l'état d'étude, et ensuite, si on arrive à contrôler le rayon de courbure, on se retrouve face au même problème que dans le cas des courbes à courbure continue du point de vue de la commande du Koala car le rayon de courbure n'est pas constant.

Le choix doit donc se faire entre les courbes de Dubins, et les courbes de Mouaddib. Le tableau 2.1 compare les principales caractéristiques des trajectoire de type 1, construites à partir des courbes de Dubins, et des trajectoires de type 2, construites à partir des courbes de Mouaddib.

Les trajectoires qui, a priori, engendreraient le moins de glissement, donc le moins d'erreur de position à l'arrivée, sont les trajectoires de type 2. Cela est dû au fait que le robot commence et termine sa trajectoire en ligne droite, et qu'il n'effectue qu'un seul virage.

Cependant ces trajectoires ne permettent pas de se rendre au but pour toutes les configurations de départ. Pour ne pas réduire le nombre de confi-

gurations de départ acceptant une trajectoire, il est nécessaire d'utiliser les trajectoires de type 1 en plus des trajectoires de type 2.

Il faut alors permettre au robot de pouvoir choisir lequel des deux types de trajectoire sera le plus approprié au cas dans lequel il se trouve. La démarche à suivre est la suivante : construction de chacun des types 1 et 2 de trajectoire grâce à deux algorithmes, puis choix d'une trajectoire.

2.2 Algorithme de construction d'une trajectoire de type 1

L'algorithme développé permet de construire, à partir des configurations initiale et finale, une trajectoire de type 1 et de vérifier son appartenance à l'environnement (cf. figure 2.1).

Les paragraphes suivant détaillent les étapes de la construction.

2.2.1 Données

Les données transmises à l'algorithme de planification sont :
- la configuration initiale (cf. figure 2.2). Elle représente :
  - la position initiale du centre du robot dans l'environnement, soit le point $I(x_I, y_I)$,
  - l'orientation initiale du robot, soit l'angle orienté θ_I entre l'axe des abcisses et le vecteur d'orientation initiale du robot $\overrightarrow{i}$.
- la configuration finale (cf. figure 2.2). Elle représente :
  - la position finale du centre du robot, soit le point $F(x_F, y_F)$,
  - l'orientation finale du robot, soit l'angle orienté θ_F.

2.2.2 Choix des cercles

La première étape de la construction est le choix des cercles que nous appelerons *cercle de départ* et *cercle d'arrivée* (cf. figure 2.3).

Le cercle de départ est choisi parmis les deux cercles tangents à la droite $(I, \overrightarrow{i})$ au point I (cf. figure 2.3). Le cercle de départ sera le cercle dont le centre est situé dans le même demi-plan, par rapport à la droite $(I, \overrightarrow{i})$, que le point F. Ce cercle est désigné par l'indice i. On a :

i=1 si le premier arc de cercle est de type R
et
i=2 si le premier arc de cercle est de type L

La démarche est la même pour trouver le cercle d'arrivée. Les deux cercles possibles sont les cercles tangents à l'axe des abscisses au point F. Le cercle d'arrivée sera le cercle dont le centre est situé dans le même demi-plan, par rapport à l'axe des abscisses, que le point I. Ce cercle est désigné par l'indice j. On a cette fois :

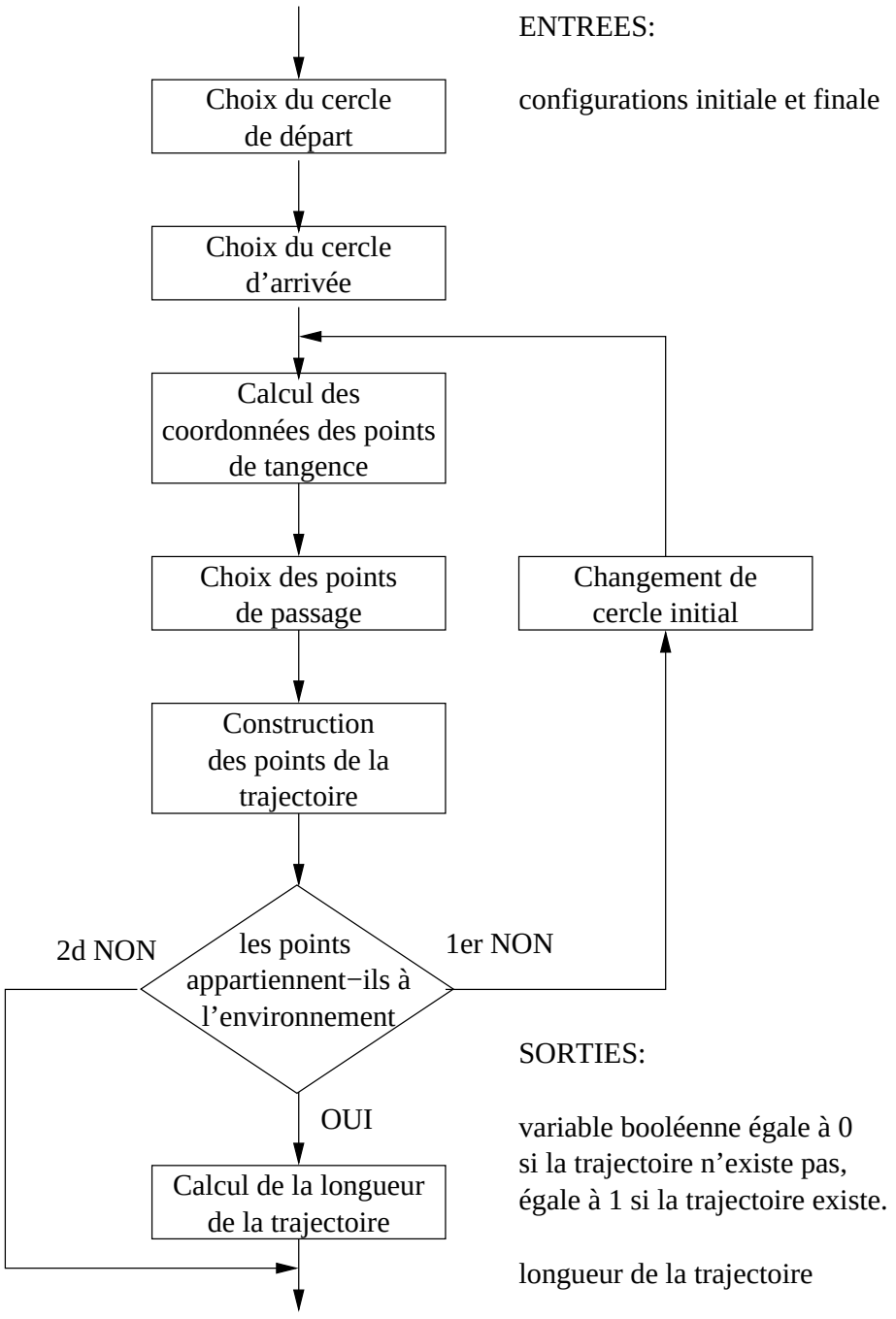


FIG. 2.1 – Algorithme de construction d'une trajectoire de type 1

j=3 si le second arc de cercle est de type R
et

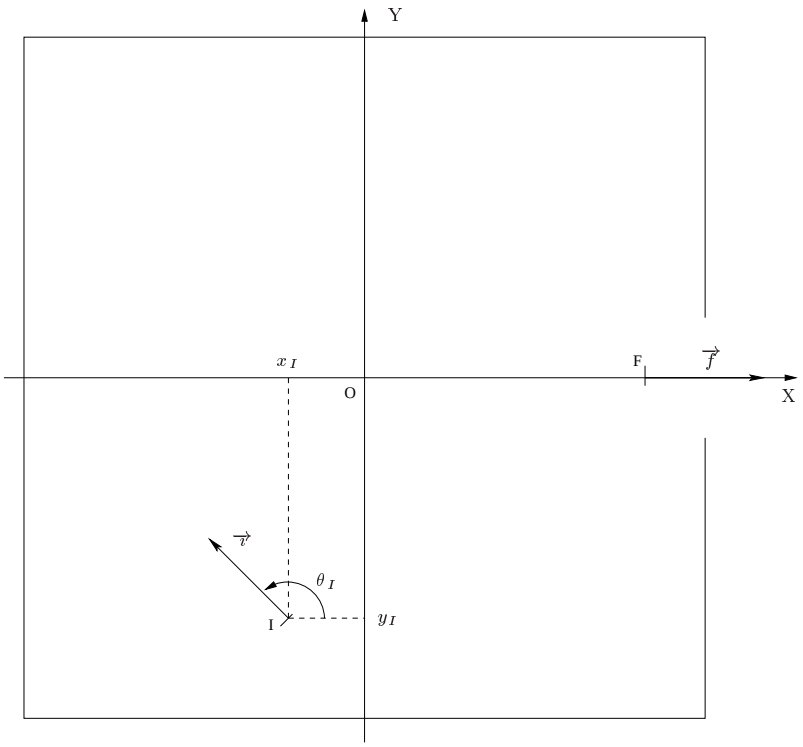


FIG. 2.2 – Données

j=4 si le second arc de cercle est de type L

2.2.3 Construction des cercles tangents

Une fois les deux cercles choisis, il faut déterminer les coordonnées de leurs centres (cf. figure 2.4) pour les construire. On note $C(i)(x_{C(i)}, y_{C(i)})$ le centre du cercle d'indice i, et $C(j)(x_{C(j)}, y_{C(j)})$ le centre du cercle d'indice j.

A partir de $\overrightarrow{OC(i)} = \overrightarrow{OI} + \overrightarrow{IC(i)}$ on en déduit pour $i = 1$ et $i = 2$:

$$\begin{cases} x_{C(1)} &= x_I + R\sin\theta_I \\ y_{C(1)} &= y_I - R\cos\theta_I \end{cases} \tag{2.1}$$

$$\begin{cases} x_{C(2)} &= x_I - R\sin\theta_I \\ y_{C(2)} &= y_I + R\cos\theta_I \end{cases} \tag{2.2}$$

Et de même pour $j = 3$ et $j = 4$:

$$\begin{cases} x_{C(3)} &= x_F \\ y_{C(3)} &= y_F - R \end{cases} \tag{2.3}$$

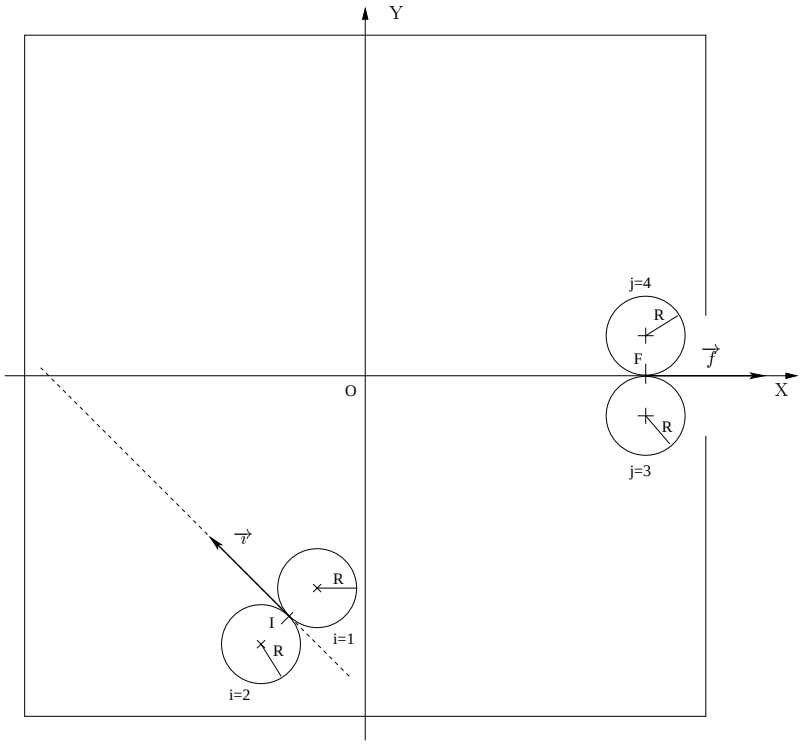


FIG. 2.3 – Choix des cercles de départ et d'arrivée

$$\begin{cases} x_{C(4)} &= x_F \\ y_{C(4)} &= y_F + R \end{cases} \tag{2.4}$$

R représente le rayon de courbure, imposé comme paramètre de la trajectoire.

2.2.4 Tracé des tangentes entre les cercles choisis

Entre les deux cercles obtenus, quatre trajectoires sont possibles pour se rendre du point initial au point final devant la porte (cf. figure 2.5). Ces quatre possibilités correspondent aux quatre tangentes qui existent entre les deux cercles.

Cependant une seule de ces trajectoires permet au robot d'arriver devant la porte sans effectuer de rotation sur lui-même. Deux trajectoires l'obligent à effectuer une rotation. La quatrième le contraint à pivoter deux fois sur lui-même.

Avant de pouvoir déterminer quelle trajectoire est réalisable sans pivotement, il est nécessaire de connaître les coordonnées des huit points de tangence entre chaque droite et les cercles.

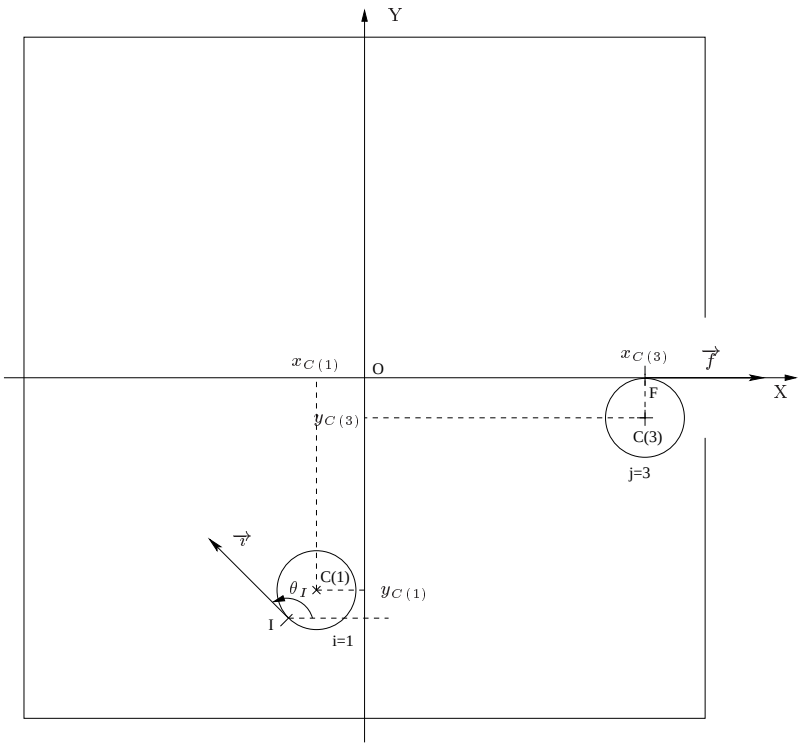


FIG. 2.4 – Construction des cercles tangents aux configurations initiales et finales

On note (T_1, T_3), (T_2, T_4), (S_1, S_3), et (S_2, S_4), les couples de points qui servent à construire chacune des quatre tangentes entre les cercles. Ces huits points existent pour chaque couple de cercles de départ et d'arrivée possible. Le couple de points choisi définira les points de passage du robot sur la trajectoire.

On calcule les coordonnées des points S_1, S_2, S_3 et S_4 à l'aide des relations suivantes :

$$\begin{cases} x_{S_1} &= x_{C(i)} + R\cos(\alpha_2 + \beta) \\ y_{S_1} &= y_{C(i)} + R\sin(\alpha_2 + \beta) \end{cases} \tag{2.5}$$

$$\begin{cases} x_{S_2} &= x_{C(i)} + R\cos(\alpha_2 - \beta) \\ y_{S_2} &= y_{C(i)} + R\sin(\alpha_2 - \beta) \end{cases} \tag{2.6}$$

$$\begin{cases} x_{S_3} &= x_{C(j)} + R\cos(\alpha_2 + \pi + \beta) \\ y_{S_3} &= y_{C(j)} + R\sin(\alpha_2 + \pi + \beta) \end{cases} \tag{2.7}$$

$$\begin{cases} x_{S_4} &= x_{C(j)} + R\cos(\alpha_2 + \pi - \beta) \\ y_{S_4} &= y_{C(j)} + R\sin(\alpha_2 + \pi - \beta) \end{cases} \tag{2.8}$$

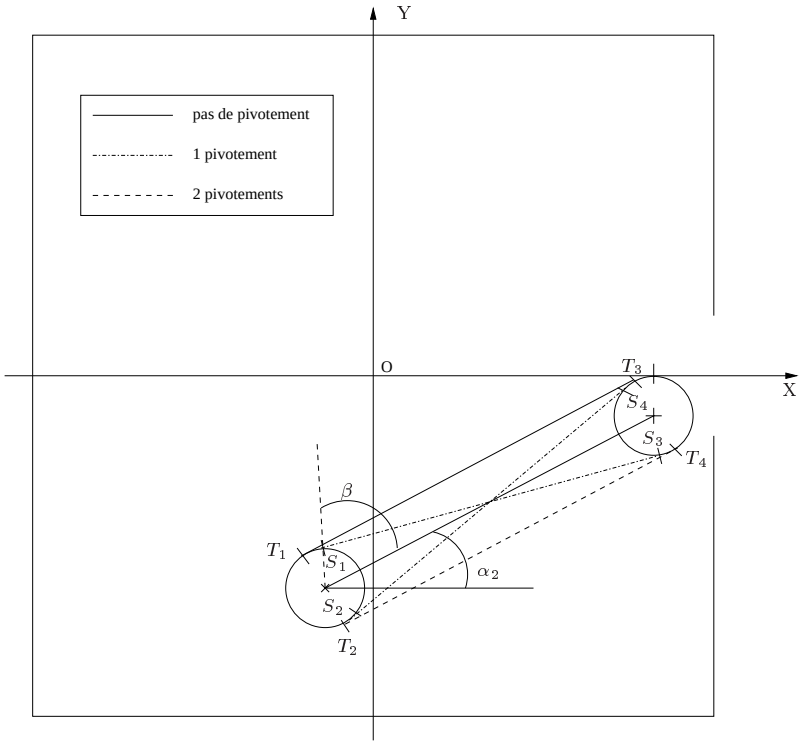


FIG. 2.5 – Représentation des quatre tangentes entre deux cercles

Et celles des points T_1, T_2, T_3 et T_4 grâce aux relations :

$$\begin{cases} x_{T_1} &= x_{C(i)} - R\sin\alpha_2 \\ y_{T_1} &= y_{C(i)} + R\cos\alpha_2 \end{cases} \tag{2.9}$$

$$\begin{cases} x_{T_2} &= x_{C(i)} + R\sin\alpha_2 \\ y_{T_2} &= y_{C(i)} - R\cos\alpha_2 \end{cases} \tag{2.10}$$

$$\begin{cases} x_{T_3} &= x_{C(j)} - R\sin\alpha_2 \\ y_{T_3} &= y_{C(j)} + R\cos\alpha_2 \end{cases} \tag{2.11}$$

$$\begin{cases} x_{T_4} &= x_{C(j)} + R\sin\alpha_2 \\ y_{T_4} &= y_{C(j)} - R\cos\alpha_2 \end{cases} \tag{2.12}$$

2.2.5 Recherche de la trajectoire sans pivotement

Après plusieurs essais de détermination du nombre de pivotement le long de la trajectoire par calcul, et pour des configurations de départ et d'arrivée quelconques, il s'est révélé que cette méthode n'aboutissait pas à des résultats convaincants car aucun cas général, pouvant prendre en compte toutes les configurations de départ et d'arrivée, n'a pu être identifié. La mise au point d'un algorithme prenant en compte toutes les configurations de départ et d'arrivée, pour trouver la trajectoire sans pivotement dans tous les cas, nécessiterait une étude géométrique approfondie.

Par contre si on se limite aux cas où la porte est centrée sur l'axe des abscisses et perpendiculaire à ce même axe, il est facile d'envisager les différentes situations dans lesquelles le robot peut se trouver. Pour chacunes d'entre elles, on détermine le couple de points de passage pour lequel le robot ne pivote pas pour atteindre la configuration finale.

Pour que l'algorithme de construction développé fonctionne, deux conditions sont à satisfaire :

- la porte doit être centrée sur l'un des murs ;
- le repère global de l'environnement doit être choisi de manière à ce que la porte soit centrée sur l'axe des abscisses et qu'elle soit perpendiculaire à ce même axe.

2.2.6 Trajectoire de type 1 finale

La trajectoire du robot est circulaire de rayon de courbure R entre I et le premier point de passage A, rectiligne entre le point A et le second point de passage B, et circulaire entre le point B et le point F.

En sortie de l'algorithme, on obtient une variable booléenne qui traduit l'existence de la trajectoire (variable égale à 0 si la trajectoire n'existe pas, égale à 1 si une trajectoire est trouvée), ainsi que la longueur de la trajectoire si elle existe.

2.3 Algorithme de construction d'une trajectoires de type 2

Les trajectoires de type 2 sont, elles aussi, des trajectoires lisses. Elles sont construites à partir des courbes de Mouaddib, et se composent des même éléments de base que les trajectoires de type 1 mais dans un ordre différent. Une trajectoire de type 2 est en effet constituée de la succession d'un segment de droite, d'un arc de cercle et d'un segment de droite (cf. figure 2.7).

Comme précisé dans la description des courbes de Mouaddib, cette trajectoire n'est pas réalisable dans toutes les configurations. La condition pour

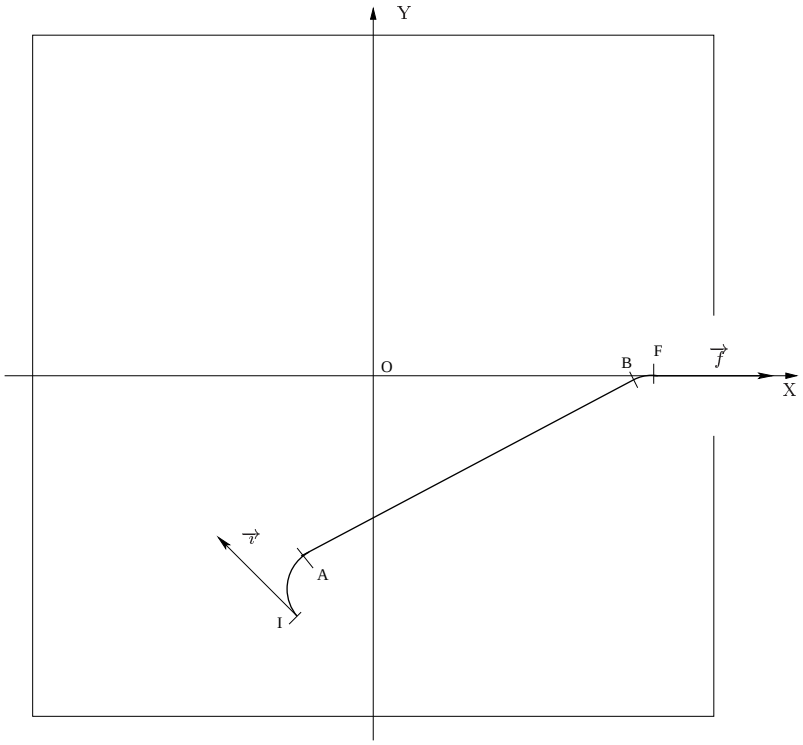


FIG. 2.6 – Représentation d'une trajectoire de type 1

avoir une trajectoire de type 2 est détaillée au paragraphe 2.3.4.

On construit la trajectoire de type 2 en passant par les étapes suivantes, les données initiales étant les même que pour la trajectoire de type 1 (cf. figure 2.2).

2.3.1 Calcul des coordonnées du point G

Le point G est le point d'intersection entre les droites $(I, \overrightarrow{i})$ et $(F, \overrightarrow{f})$. Ses coordonnées sont données par la résolution du système :

$$\left\{ \begin{array}{lcl} x_G & = & x_I + k_1 \cos\theta_I = x_F + k_2 \cos\theta_F \\ y_G & = & y_I + k_1 \sin\theta_I = y_F + k_2 \sin\theta_F \end{array} \right. \tag{2.13}$$

On trouve k_1 et k_2 et on en déduit les coordonnées du point G.

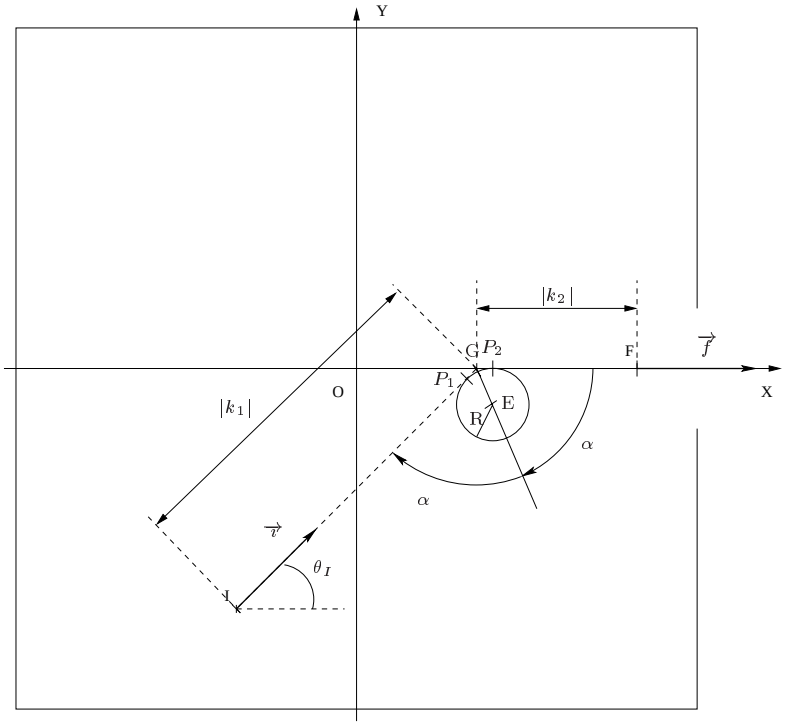


FIG. 2.7 – Représentation d'une trajectoire de type 2

2.3.2 Calcul des coordonnées du point E

On calcul ensuite les coordonnées du point E, centre du cercle tangent aux droites $(I, \vec{i})$ et $(F, \vec{f})$. On a :

$$\text{Si } 0 < \theta_I < \pi \qquad \begin{cases} x_E &= x_G - k_5 \cos\alpha \\ y_E &= y_G - k_5 \sin\alpha \end{cases} \tag{2.14}$$

$$\text{Si } -\pi < \theta_I < 0 \qquad \begin{cases} x_E &= x_G + k_5 \cos\alpha \\ y_E &= y_G + k_5 \sin\alpha \end{cases} \tag{2.15}$$

$$\text{Avec } k_5 = \frac{R}{sin\alpha}$$

2.3.3 Calcul des coordonnées des points P_1 et P_2

On calcule enfin les coordonnées des points de passage P_1 et P_2 dans le repère du cercle de centre E et de rayon R.

$$\begin{cases} x_{P_1} &= x_I + k_6 \cos\theta_I \\ y_{P_1} &= y_I + k_6 \sin\theta_I \end{cases} \tag{2.16}$$

$$\begin{cases} x_{P_2} &= x_F + k_7 \\ y_{P_2} &= y_F \end{cases} \tag{2.17}$$

$$\text{Avec } k_6 = k_1 - \frac{R}{|tan\alpha|} \text{ et } k_7 = k_2 + \frac{R}{|tan\alpha|}$$

2.3.4 Condition d'existence de la trajectoire de type 2

L'existence d'une trajectoire de type 2 dépend de la position du point P_1 sur la droite $(I, \overrightarrow{i})$. Si la coordonnée du point P_1 dans le repère de la droite $(I, \overrightarrow{i})$ est supérieure à zéro, alors il existe une trajectoire de type 2 qui permet, en partant de la configuration initiale, d'arriver à la configuration finale. Sinon il n'existe pas de trajectoire de type 2 et il faudra utiliser une trajectoire de type 1 s'il en existe une.

2.3.5 Trajectoire de type 2 finale

On obtient les coordonnées des points de passage du robot, soit dans l'ordre : I, P_1, P_2 et F. La trajectoire est rectiligne entre I et P_1, circulaire de rayon de courbure R entre P_1 et P_2, puis rectiligne entre P_2 et F.

2.4 Algorithme de choix d'une trajectoire

Comme précisé ci-dessus, la trajectoire de type 2 n'existant pas pour toutes les configurations de départ, il est nécessaire d'utiliser en plus les trajectoires de type 1. Suivant les cas rencontrés, le robot devra parfois choisir entre les deux trajectoires, parfois il n'aura qu'une trajectoire possible, et enfin, dans certains cas, il n'aura aucune trajectoire faisable pour se rendre à la porte. Il est donc indispensable de doter le robot d'un algorithme de choix de trajectoire (cf. figure 2.8) qui prend en compte l'ensemble des cas pouvant être rencontrés.

Le cas le plus problématique se présente lorsque le robot a le choix entre les deux types de trajectoire. Il faut définir un critère de comparaison entre les deux trajectoires.

La critère de comparaison choisi est la longueur des trajectoires. Si la longueur de la trajectoire de type 2 est inférieure à la longueur de la trajectoire de type 1 multipliée par un coéfficient k, on choisit la trajectoire de type 2. En effet, comme on l'a vu précédemment, c'est la trajectoire qui, a priori, réduit le plus l'erreur de glissement des roues sur le sol.

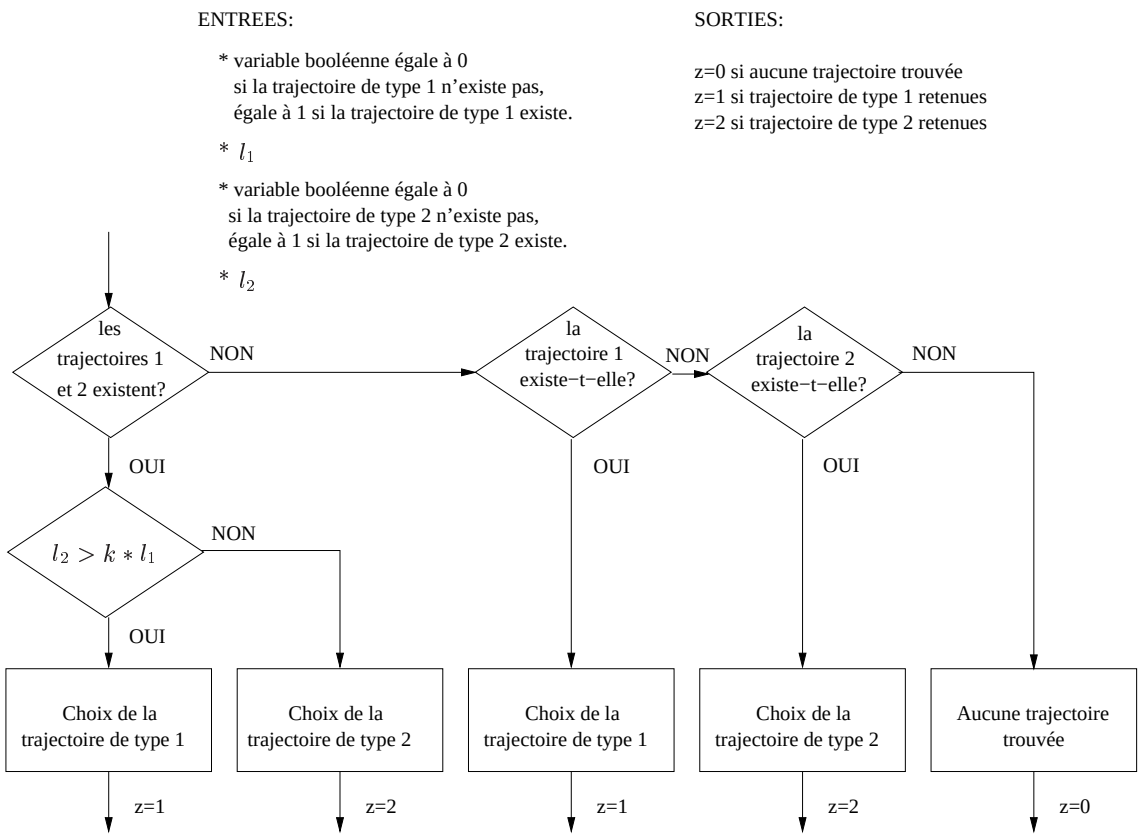


FIG. 2.8 – Algorithme de choix de trajectoire

En revanche, si la longueur de cette trajectoire devient supérieure à une limite (représentée à l'aide du coéfficient k), il est possible que l'erreur due aux glissements soit supérieure à celle d'une trajectoire de type 1, dont la longueur est k fois plus courte.

La valeur du coéfficient k est à déterminer expérimentalement car aucun moyen ne nous permet d'évaluer le glissement théorique, et surtout l'erreur qu'il provoque.

Chapitre 3

Simulation à l'aide du logiciel Matlab

Après avoir établi les algorithmes qui serviront à la planification des trajectoires pour le robot, et avant de les implanter sur le robot, il est intéressant de les évaluer par la simulation. Les objectifs de cette simulation sont :

- vérifier que les algorithmes de construction aboutissent aux trajectoires désirées ;
- vérifier que l'algorithme de choix fonctionne ;
- établir les limites des trajectoires choisies en déterminant l'ensemble des configurations de départ qui n'admettent pas de solution pour se rendre à la porte sans collision avec l'environnement ;
- lors de la programmation en C, comparer les résultats de la simulation avec ceux du programme C pour vérifier la validité du programme.

Pour la simulation des algorithmes de choix et de construction de la trajectoire, notre choix s'est porté sur le logiciel Matlab pour plusieurs raisons :

- Matlab permet de visualiser les trajectoires issues des algorithmes de construction et de choix ;
- on peut lancer le programme de simulation en boucle pour plusieurs positions de départ (*maillage de l'environnement*), et ainsi obtenir une représentation de l'*espace libre des configurations de départ* ;
- le langage de programmation est relativement rapide à assimiler.

3.1 Structure du programme Matlab

La structure du programme a été adaptée des algorithmes de construction et de choix (cf. figure 3.1). Les fonctions *trajectoire 1* et *trajectoire 2* correspondent aux algorithmes de construction, et la fonction *choixtraj* correspond à l'algorithme de choix de trajectoire. La fonction *main* contient les

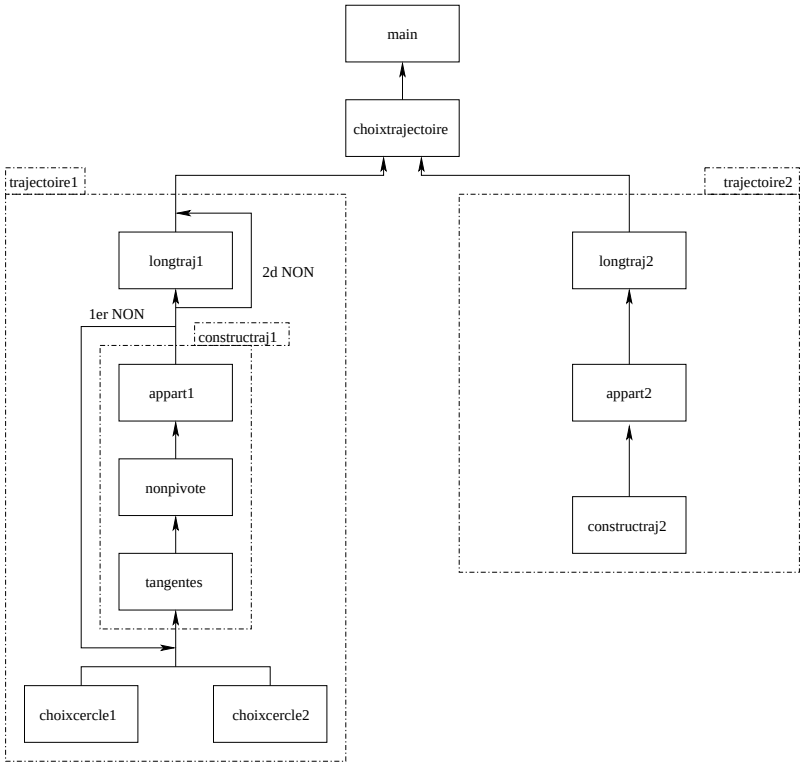


FIG. 3.1 – Structure du programme Matlab

informations pour tracer les courbes, et gère les choix de l'utilisateur.

Ce programme permet de réaliser un maillage de l'environnement : pour une orientation donnée, on peut simuler le comportement du robot pour différentes positions de départ en faisant varier x_i et y_i. Les noeuds du maillage sont les points de départ du robot. Le pas de ce maillage est paramétrable par l'utilisateur.

L'utilisation du programme Matlab, ainsi que les différentes fonctions, sont décrites dans l'annexe A.

3.2 Résultats de la simulation

La simulation a permi de vérifier le fonctionnement des algorithmes de construction et de choix des trajectoires et de corriger les erreurs de programmation.

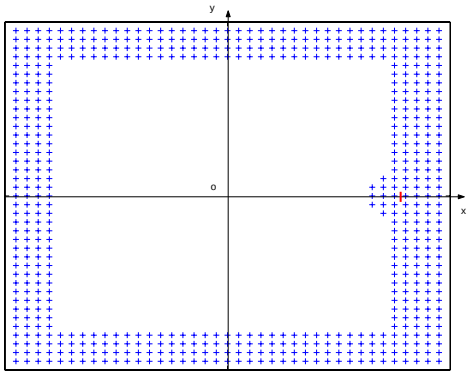
Elle a ensuite permi d'établir l'ensemble des configurations de départ n'admettant pas de solution pour se rendre devant la porte. On a pour cela

utilisé le programme en réalisant un maillage pour chaque orientation entre 0 et $\frac{7\pi}{4}$, tous les $\frac{\pi}{4}$. On obtient les graphiques de la figure 3.2, où les croix représentent les points de départ problématiques. La porte est située à droite de l'environnement sur chaque figure.

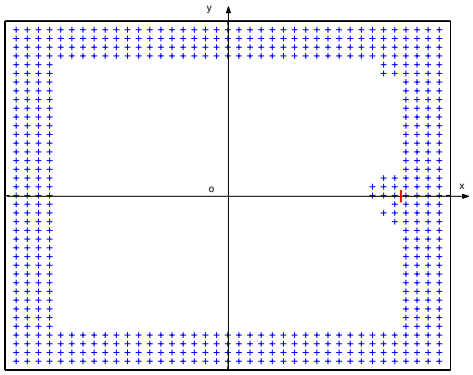
Si on superpose ces graphiques, on obtient la figure 3.3. Les points marqués d'une croix représentent les positions de départ qui, pour au moins une orientation initiale, n'admettent aucune trajectoire pour se rendre jusqu'à la porte. On remarque logiquement que les positions situées à proximité des murs posent problème. Le robot n'a pas assez d'espace à ces endroits pour effectuer de virages. Cette surface représente une bande d'environ 250mm de large située tout autour de l'environnement.

L'autre endroit où l'on remarque des positions problématiques est devant la porte. Ces positions ont été vonlontairement exclues pour les courbes de type 1. En effet lorsque le point initial est trop proche du point final, les cercles de départ et d'arrivée sont sécants, et seules deux tangentes existent entre ces cercles. L'algorithme pour obtenir les trajectoires dans ces cas étant complètement différent de l'algorithme précédent, il aurait fallu le reprogrammer. Le nombre de positions exclues étant relativement faible, il a été décidé d'exclure ces positions de départ.

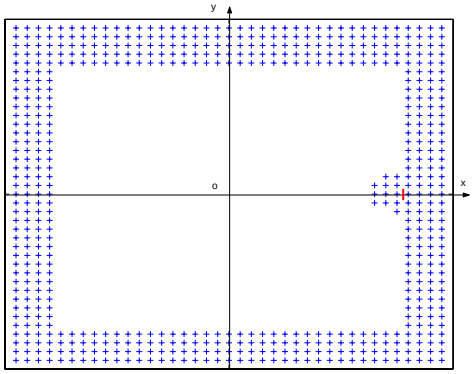
Si la surface problématique semble importante pour le cas présenté (environnement de 2 mètres par 2 mètres), il est à noter que la largeur de la bande occupée par les positions problématiques tout autour de l'environnement reste inchangée quelques soient les dimensions de l'environnement. La dimension de cette zone ne dépend que des caractéristiques du robot (longueur et largeur).



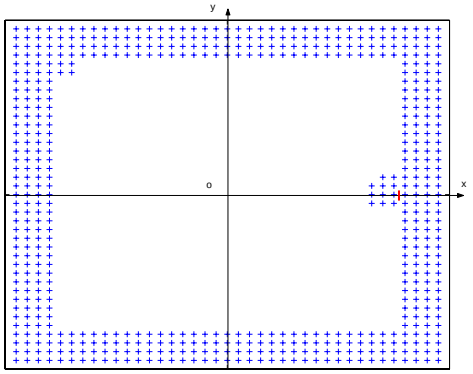
Orientation initiale : $\theta_I = 0$



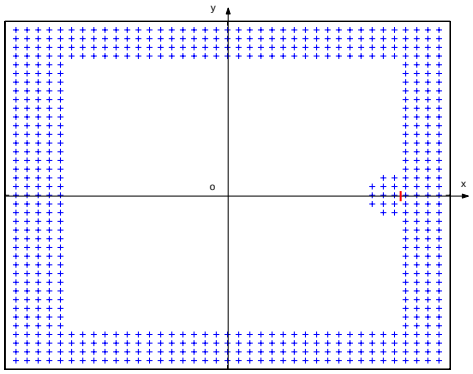
Orientation initiale : $\theta_I = \frac{\pi}{4}$



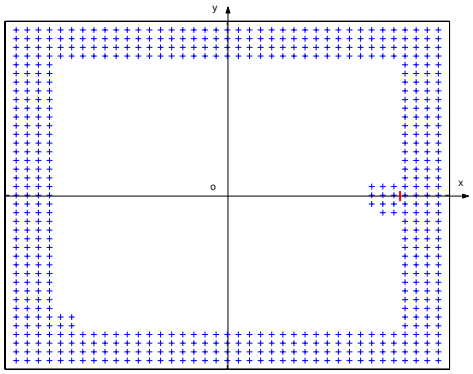
Orientation initiale : $\theta_I = \frac{\pi}{2}$



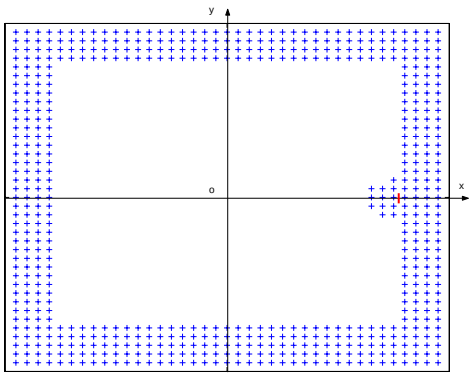
Orientation initiale : $\theta_I = \frac{3\pi}{4}$



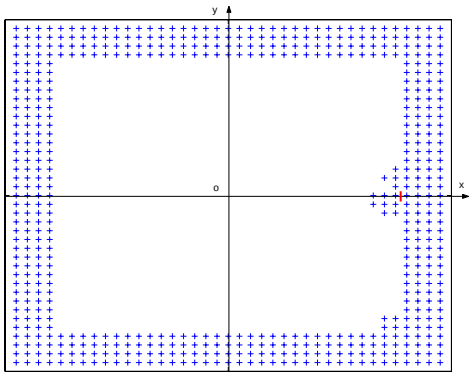
Orientation initiale : $\theta_I = \pi$



Orientation initiale : $\theta_I = \frac{5\pi}{4}$



Orientation initiale : $\theta_I = \frac{3\pi}{2}$



Orientation initiale : $\theta_I = \frac{7\pi}{4}$

FIG. 3.2 – Résultat de la simulation pour différentes orientations initiales

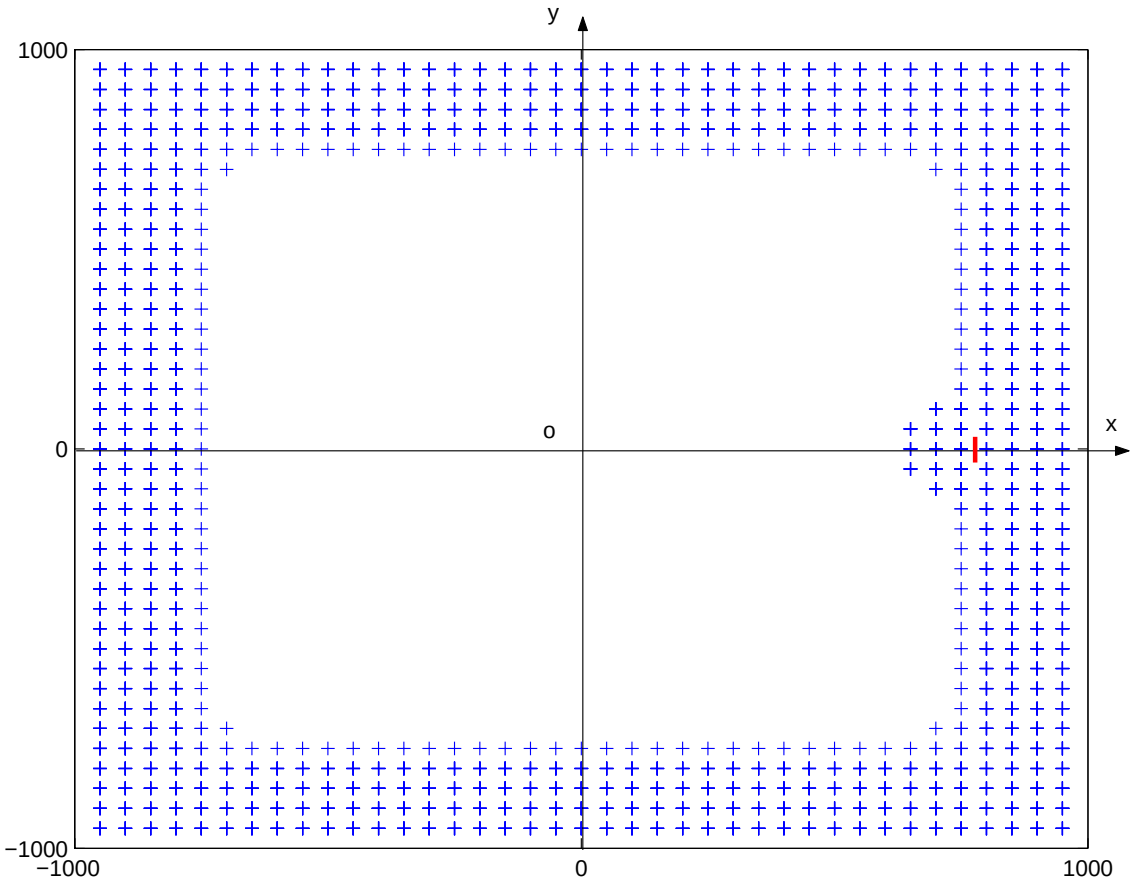


FIG. 3.3 – Positions de départ qui n'admettent aucune trajectoire pour au moins une orientation initiale

Chapitre 4

Implantation du planificateur

Après cette étape de simulation, les algorithmes précédents ont été programmés en langage C afin d'être implantés sur le robot. Mais avant, il a fallu déterminer la stratégie de commande à utiliser.

4.1 Stratégie de commande du Koala

Le Koala dispose de deux stratégies de commande, en position ou en vitesse [11].

Lors d'une commande en position, on indique au robot la position finale à atteindre pour la roue, la vitesse maximale et l'accélération de la roue. Lorsque la roue a atteint cette position, elle s'arrête. Si les positions finales et les profils de vitesse des deux roues motrices sont identiques, le robot se déplace en ligne droite, s'ils sont différents, il décrit un arc de cercle. La commande en position permet donc de réaliser des trajectoires de type 1 et de type 2. Cependant le robot s'arrêtera à chaque point de raccordement entre les éléments de base, c'est-à-dire deux fois le long de la trajectoire. Le mouvement n'est pas fluide et les deux redémarrages supplémentaires engendrés par ces arrêts peuvent être une source de glissement des roues.

Pour une commande en vitesse, on indique à la roue la vitesse de consigne à conserver. On annule cette consigne lorsque la roue à parcourue la distance donnée comme paramètre. Dans ce cas l'accélération n'est pas réglable, et elle est identique pour les deux moteurs. L'autre différence avec la commande en position est que le changement de la vitesse de consigne ne nécessite pas d'arrêt du robot. Ainsi il peut décrire les trajectoires de type 1 et 2 sans s'arrêter aux points de raccordement entre les éléments de base.

La fluidité du robot étant recherchée, et malgré le fait que l'accélération ne soit pas réglable, la commande en vitesse est la plus adaptée au cas qui nous intéresse.

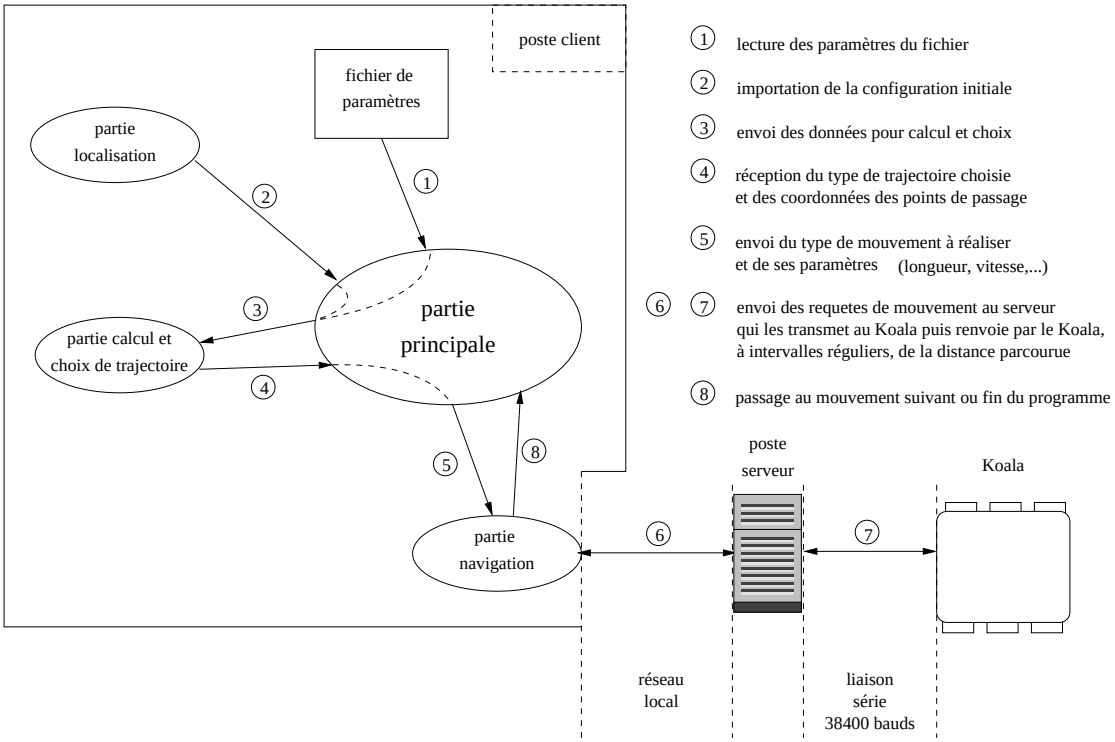


FIG. 4.1 – Implantation du planificateur de trajectoire dans le système

4.2 Implantation du planificateur

Le planificateur a été programmé en langage C afin d'être implanté sur le robot. La programme C se décompose en 15 fichiers pouvant être regroupés en quatre parties : la partie calcul et choix de la trajectoire, la partie localisation, la partie navigation, et la partie principale qui gère toutes les autres (cf. figure 4.1).

La partie calcul et choix de trajectoire reprend la structure du programme Matlab présentée sur la figure 3.1. Les noms des fichiers sont identiques pour qu'il n'y ai aucune ambiguité. La principale différence se situe au niveau de la gestion des variables, qui est maintenant faite à partir de tableaux statiques définis comme variables globales pour être accessibles à partir de tous les fichiers.

La partie localisation gère pour l'instant la saisie au clavier des coordonnées de départ du robot (x_I et y_I) et son orientation initiale (θ_I). Elle a été séparée de la partie principale car, à terme, cette partie contiendra le programme de localisation développé par Cyrille Roussel.

La partie navigation contient les fonctions qui gèrent la commande en vitesse du Koala. Elles permettent de faire décrire au Koala les éléments de

base qui constituent les trajectoires de type 1 et 2, à savoir des segments de droite et des arcs de cercle. Les paramètres de la fonction indiquant au Koala de réaliser un segment de droite sont la vitesse du centre du robot, et la longueur de ce segment. Les paramètres de la fonction générant des arcs de cercle sont la vitesse du centre, la longueur de l'arc, le rayon de courbure et le sens du virage (droite ou gauche).

La partie principale lit dans un premier temps les données contenues dans un fichier texte. Ces données sont les paramètres de la trajectoire qui sont susceptibles d'être fréquemment modifiés lors de la phase de test : le rayon de courbure, la vitesse du centre dans les lignes droites, la vitesse du centre dans les virages et le coéfficient de choix entre trajectoires de type 1 et 2. Ensuite elle utilise la partie localisation pour connaître la configuration initiale du robot. Elle fait appel à la partie calcul et choix pour savoir dans quel ordre et avec quels paramètres utiliser les fonctions de la partie navigation : arc de cercle de longueur l_{arc_1}, segment de longueur l_{segm} et arc de cercle de longueur l_{arc_2} pour une trajectoire de type1, et segment de longueur l_{segm_1}, arc de cercle de longueur l_{arc} et segment de longueur l_{segm_2} pour une trajectoire de type 2. L'utilisation du programme est détaillée dans l'annexe B.

Chapitre 5

Optimisation du planificateur de trajectoire

Le critère d'optimisation de la trajectoire est la minimisation de l'erreur de position à l'arrivée. Pour réduire cette erreur, il faut pouvoir la mesurer. C'est pourquoi un système de mesure d'erreur a été installé sur le Koala.

5.1 Mesure de l'erreur de position

5.1.1 Système de mesure de l'erreur de position

Le système suivant utilise trois diode laser d'un milliwatt de puissance. Deux sont disposées parallèlement à l'axe principal du robot, et la troisième perpendiculairement (cf. figure 5.1).

La mesure de l'erreur se fait en relevant les distances entre les traces des faisceaux laser et des marques fixes représentées sur les murs (cf. figure 5.2). On cherche trois erreurs, ε_x l'erreur selon l'axe des abscisses, ε_y l'erreur selon l'axe des ordonnées, et ε_θ l'erreur d'orientation du robot, il faut donc relever trois distances pour résoudre le système.

On mesure les distances l_{mes}, Δ_{x_1} et Δ_{y_1}, et on calcule les trois erreurs en résolvant le système suivant :

$$\left\{ \begin{array}{lcl} cos(\varepsilon_\theta) & = & \frac{c}{l_{mes}} \\ tan(\varepsilon_\theta) & = & \frac{\Delta_x - \varepsilon_x}{y_{envir} + \varepsilon_y} \\ tan(\varepsilon_\theta) & = & \frac{\Delta_y - \varepsilon_y}{x_{envir} - x_{F_{th}} - \varepsilon_x} \end{array} \right. \tag{5.1}$$

$$\text{Avec } \Delta_x = \Delta_{x_1} + e \cdot \left(1 - \frac{1}{cos\varepsilon_\theta}\right) \text{ et } \Delta_y = \Delta_{y_1} - d \cdot \left(1 - \frac{1}{cos\varepsilon_\theta}\right)$$

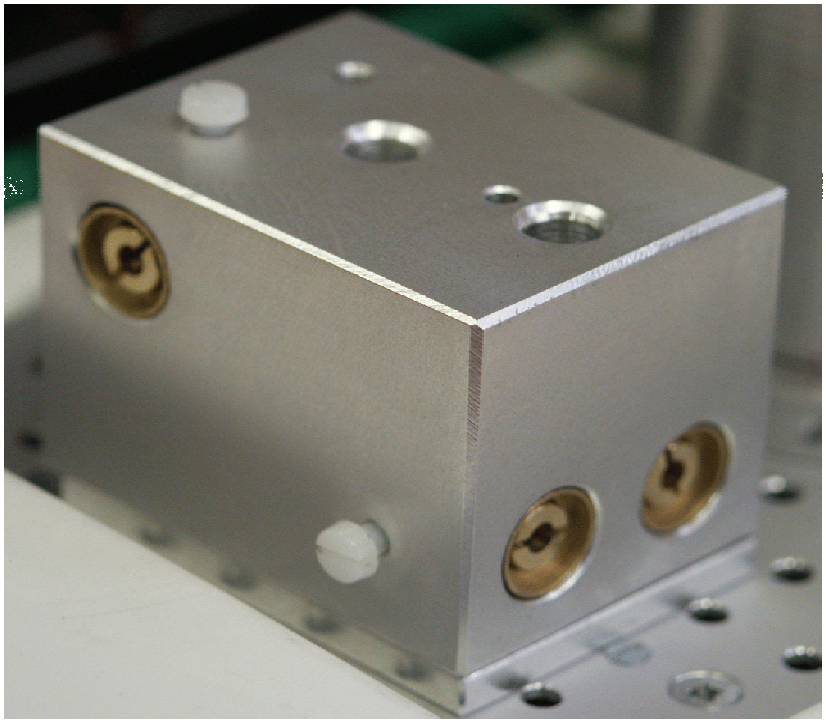


FIG. 5.1 – Dispositif de mesure de l'erreur de position

x_{envir} et y_{envir} sont les coordonnées des murs, et $x_{F_{th}}$ est l'abscisse du point final de la trajectoire. La résolution de ce système donne :

$$\varepsilon_\theta = arcos\left(\frac{c}{l_{mes}}\right) \tag{5.2}$$

$$\varepsilon_x = \frac{\Delta_x - tan\varepsilon_\theta(y_{envir} + \Delta_y) + tan^2\varepsilon_\theta(x_{envir} - x_{F_{th}})}{1 + tan^2\varepsilon_\theta} \tag{5.3}$$

$$\varepsilon_y = \frac{\Delta_y - tan\varepsilon_\theta(x_{envir} - x_{F_{th}} - \Delta_x) - tan^2\varepsilon_\theta y_{envir}}{1 + tan^2\varepsilon_\theta} \tag{5.4}$$

5.1.2 Précision des mesures et exploitation des résultats

Pour les tests, le robot est plaçé manuellement au point de départ de la trajectoire en utilisant des marquages au sol. La précision du positionnement est donc relative et elle entraîne des erreurs à l'arrivée. Cependant cette imprécision (de l'ordre du $1/2cm^2$) est inférieure à l'imprécision de la localisation à l'aide de marques P-similaires [1] qui est de l'ordre de $2cm^2$, et qui sera utilisée à terme pour connaître la configuration initiale.

De plus l'utilisation du système de mesure d'erreur n'a pas permis d'effectuer les mesures comme prévu. En effet le relevé de la valeur l_{mes} ne peut

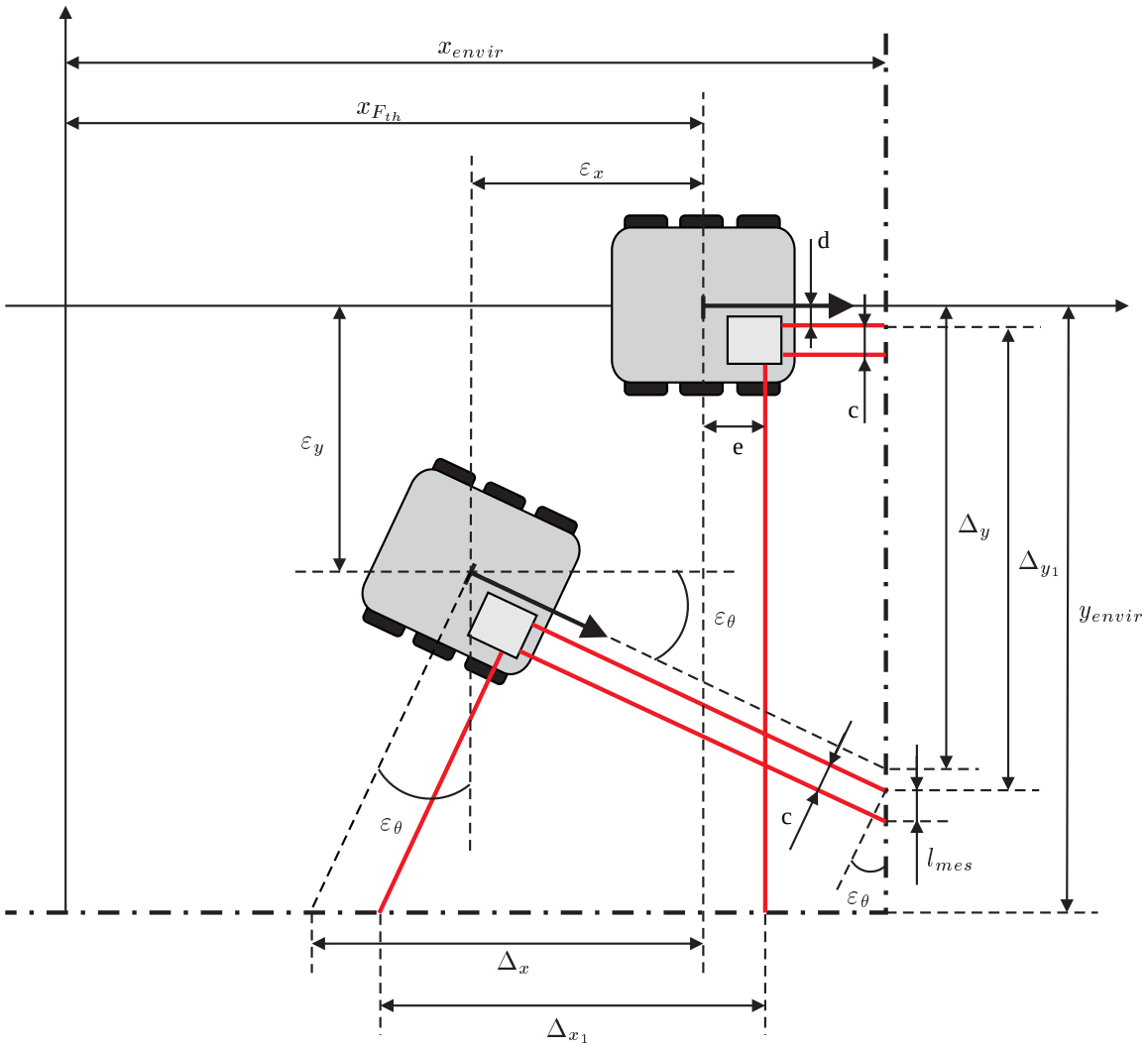


FIG. 5.2 – Représentation schématique de l'erreur de position à l'arrivée

pas se faire avec les modules laser utilisés, les points des faisceaux projetés sur les murs étant trop gros. Pour des mesures précises, il faudrait utiliser des modules comportant un réglage de la distance focale pour pouvoir modifier l'épaisseur du point.

Finalement les relevés de ε_θ ont donc été faits manuellement.

Les grandeurs qui seront utilisées par la suite pour représenter l'erreur de position à l'arrivée seront :

- la valeur absolue de l'erreur d'orientation à l'arrivée $|\varepsilon_\theta|$;
- la surface du disque de centre le point d'arrivée et de rayon $r = \sqrt{{\varepsilon_x}^2 + {\varepsilon_y}^2}$, soit $S = \pi \cdot r$ (cf. figure 5.3). Plus l'erreur de position

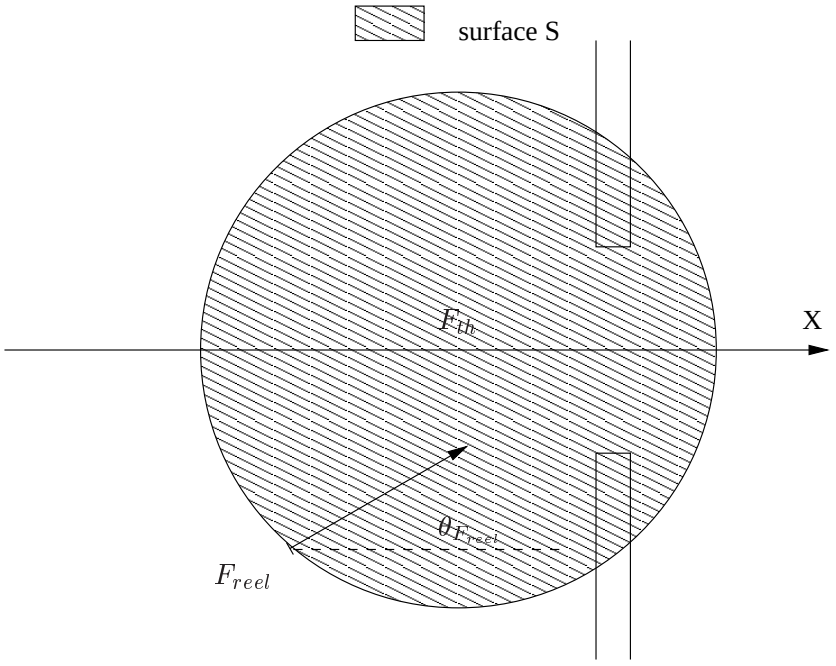


FIG. 5.3 – Représentation de la surface S

est grande, plus la valeur de S augmente.

Quand par la suite des choix devront être faits à partir des relevés de S et de $|\varepsilon_\theta|$, une importance plus grande sera donné à S. En effet l'erreur de mesure de ε_θ aura une influence directe sur la valeur de $|\varepsilon_\theta|$, alors que S dépend seulement de la tangente de ε_θ et de son carré.

5.2 Etude cinématique du Koala

5.2.1 Modèle cinématique

Les six roues du Koala sont motrices, reliées mécaniquement trois par trois. Les trois roues situées d'un même coté sont reliées au même moteur. Lors des virages, cela engendre un glissement des deux paires de roues situées à l'avant et à l'arrière du centre du robot. Les roues centrales sont donc prépondérantes dans les virages. Le modèle cinématique utilisé est le même que celui choisi par Cyrille Roussel [1], c'est-à-dire le modèle du robot différentiel : on considère que seules les roues centrales sont motrices (cf. figure 5.4).

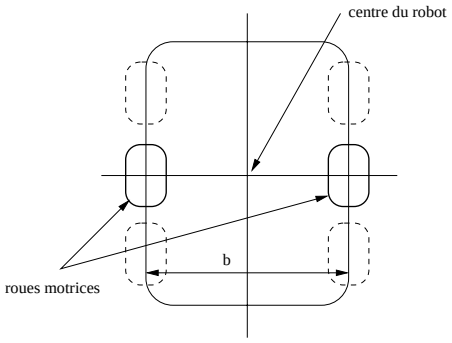


FIG. 5.4 – Modèle du robot différentiel

5.2.2 Relations cinématiques

Soit V_{ext} et V_{int} les vitesses respectivement des roues extérieure et intérieure lors d'un virage, soit b la voie du robot, soit R le rayon de courbure du virage, et soit R_{int} et R_{ext} les rayons de courbure des roues respectivement intérieure et extérieure (cf. figure 5.5). On considère que V_{ext} et V_{int} sont constantes tout au long du virage. On a la relation suivante :

$$\begin{cases} \frac{V_{ext}}{R_{ext}} = \frac{V_{int}}{R_{int}} \\ R_{ext} - R_{int} = b \end{cases} \tag{5.5}$$

$$\text{Avec } R_{int} = R - \frac{b}{2} \text{ et } R_{ext} = R + \frac{b}{2}$$

D'où :

$$\frac{V_{ext}}{V_{int}} = \frac{2R + b}{2R - b} = cste \tag{5.6}$$

Si on veut que le robot décrive un virage à droite, il faut que les vitesses des roues vérifient la relation :

$$\frac{V_L}{V_R} = \frac{2R + b}{2R - b} \tag{5.7}$$

Avec V_L et V_R vitesses des roues gauche et droite respectivement.

Et inversement pour un virage à gauche, pour lequel on doit avoir :

$$\frac{V_R}{V_L} = \frac{2R + b}{2R - b} \tag{5.8}$$

En utilisant la vitesse V du centre du robot, ces relations donnent :

$$V_{int} = \frac{2V}{1 + \frac{2R+b}{2R-b}} \text{ et } V_{ext} = \frac{2V}{1 + \frac{2R+b}{2R-b}} \frac{2R + b}{2R - b} \tag{5.9}$$

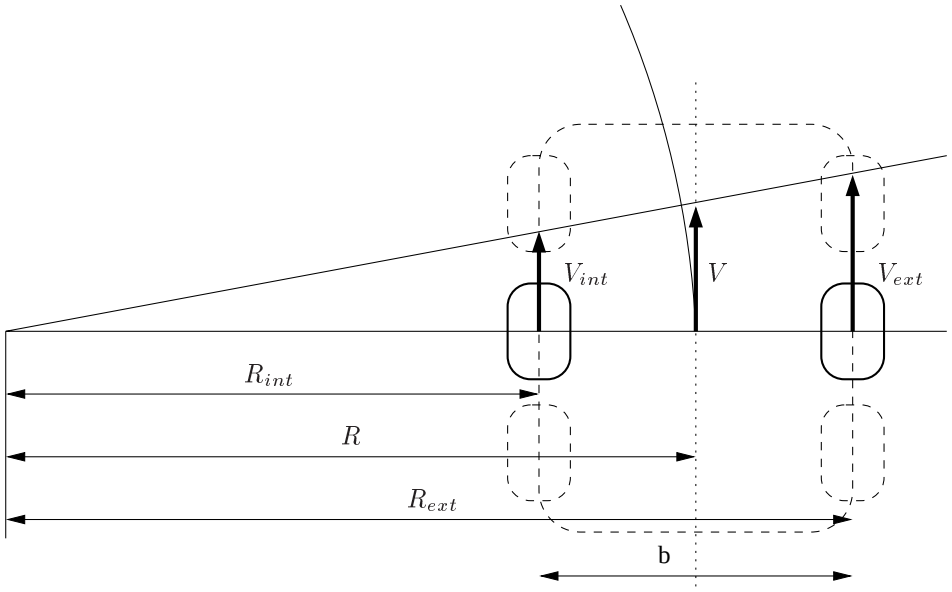


FIG. 5.5 – Cinématique du robot dans un virage de rayon de courbure donné

$$\text{Avec } V = \frac{V_{ext} + V_{int}}{2}$$

Pour un virage à droite, on doit donc vérifier :

$$V_R = \frac{2V}{1 + \frac{2R+b}{2R-b}} \text{ et } V_L = \frac{2V}{1 + \frac{2R+b}{2R-b}} \frac{2R+b}{2R-b} \tag{5.10}$$

Et pour un virage à gauche :

$$V_L = \frac{2V}{1 + \frac{2R+b}{2R-b}} \text{ et } V_R = \frac{2V}{1 + \frac{2R+b}{2R-b}} \frac{2R+b}{2R-b} \tag{5.11}$$

Ces relations permettent de commander le robot en vitesse pour qu'il décrive un cercle de rayon R avec une vitesse moyenne V (vitesse du centre du robot).

Pour que le Koala se déplace en ligne droite, il suffit d'imposer :

$$V_R = V_L = V \tag{5.12}$$

5.3 Optimisation des paramètres de la trajectoire

Les paramètres de la trajectoire à optimiser pour réduire l'erreur de position à l'arrivée sont le rayon de courbure et la vitesse du centre du robot.

Dans un premier temps, il a été decidé de vérifier l'influence du rayon de courbure sur l'erreur d'orientation.

5.3.1 Influence du rayon de courbure pour les virages

Pour identifier le comportement du robot dans les virages, en fonction du rayon de courbure, on a fait décrire au Koala des cercles de rayon variable. Les tests ont été fait pour des vitesses variant de 0,1m/s à 0,4m/s avec un pas de 0,05m/s, et pour des vitesses de 0,5ms/, 0,6m/s et 0,7m/s.

Les erreurs selon x et y (ε_x et ε_y) ont été négligées vu leurs faibles valeurs (au maximum 5mm). Seule l'erreur d'orientation (ε_θ) est représentée en fonction du rayon de courbure. La vitesse du centre du robot est un paramètre. Les courbes obtenues sont représentées sur la figure 5.6.

On remarque sur chaque courbe un comportement similaire pour les faibles rayons de courbure : l'erreur d'orientation est très importante. En effet lorsque le rayon de courbure est trop faible pour une vitesse donnée, le robot n'arrête pas sa rotation. Cela peut venir des codeurs incrémentaux, qui calculent la distance parcourue, dont la résolution ne serait pas assez élevée. La programme de commande du robot a été programmé tel que toutes les 3ms, le robot relève la position des codeurs et la transforme en distance parcourue par le centre. Il compare ensuite cette distance à la distance théorique à parcourir, calculée par le planificateur. Si la vitesse de rotation de la roue est élevée (comme c'est le cas quand on combine faible rayon de courbure et grande vitesse du centre du robot) et que la résolution du codeur est trop faible, on aura une perte d'informations. Par exemple au lieu de relever trois pulse, le codeur n'en relèvera qu'une seule, ce qui faussera le calcul de la distance, qui sera inférieure à la distance réelle parcourue. C'est pourquoi dans ces cas le robot arrête sa rotation trop tard, voire jamais. Cependant cela reste une hypothèse car la référence des codeurs utilisés n'a pas été trouvée dans le manuel du Koala, et on n'a donc pas pu vérifier leur résolution.

De plus on remarque pour chaque vitesse l'existence d'un rayon de courbure optimal pour lequel l'erreur d'orientation est minimale. La courbe de la figure 5.7 représente le rayon de courbure optimal (R_{opt} en mm) en fonction de la vitesse (V en m/s).

Ce phénomène a uniquement été constaté de manière empirique, et il n'a pas pu être vérifié par la théorie. Vu le nombre de facteurs qui interviennent (résolution des codeurs incrémentaux, différence d'adhérence entre les roues droite et gauche, différence de performance entre les moteurs,...), une telle loi de comportement est difficile à justifier. Malgré tout le comportement constaté a été admis pour la suite des expérimentations.

5.3.2 Influence du rayon de courbure pour une trajectoire complète

Après avoir constaté que chaque vitesse admet un rayon de courbure optimal quand le robot décrit un cercle, on a vérifié si on pouvait de même

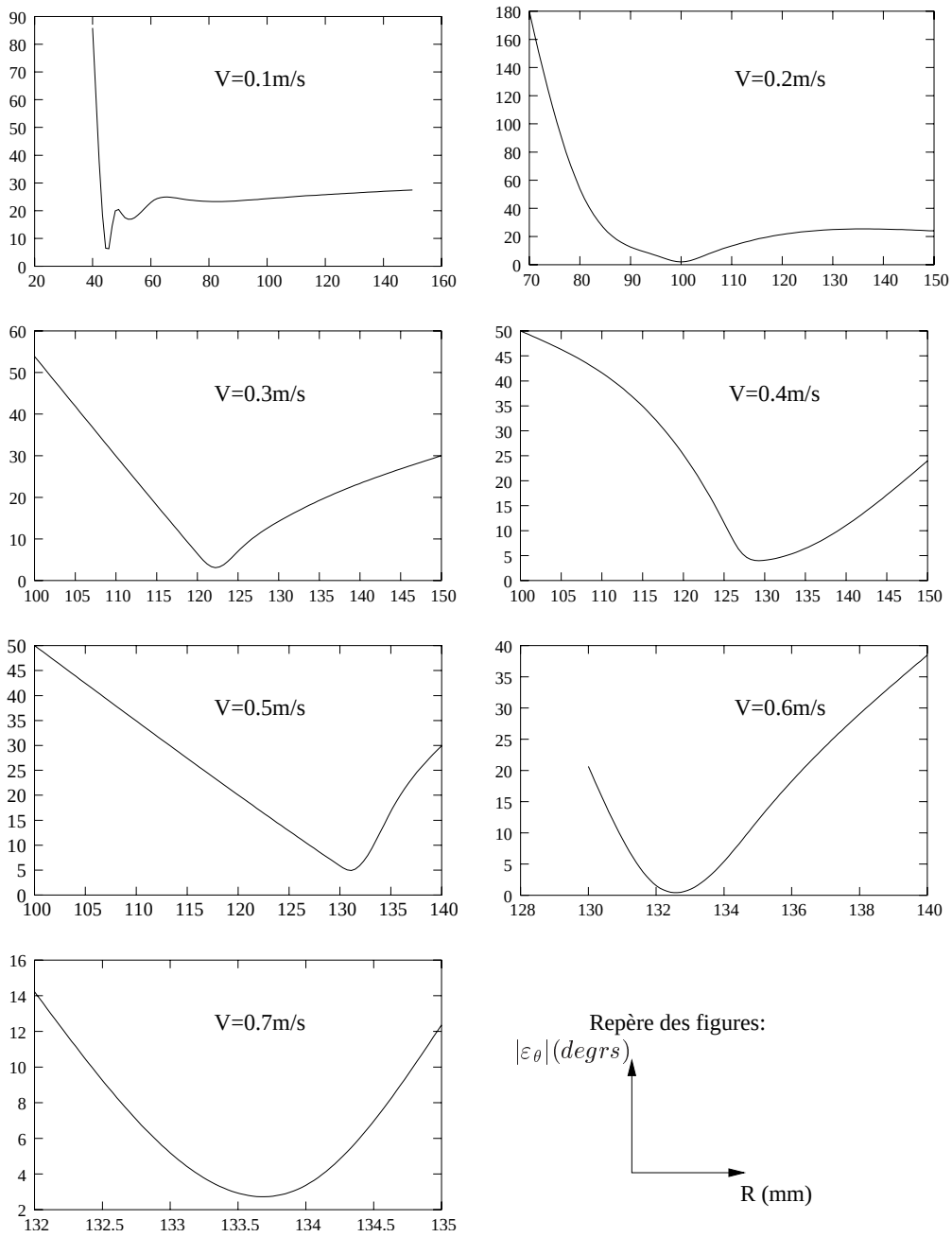


FIG. 5.6 – Variation de l'erreur d'orientation en fonction du rayon de courbure, pour différentes vitesses

trouver un rayon de courbure optimal pour une trajectoire complète, et en ne négligeant plus les erreurs selon x et y.

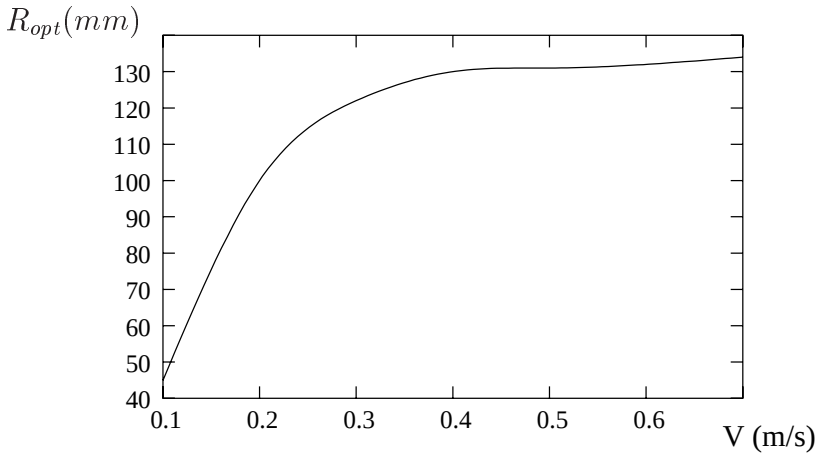


FIG. 5.7 – Rayon de courbure optimal en fonction de la vitesse du centre du robot

Pour faire ce test on a choisi une vitesse du centre du robot égale à 0,3m/s. La position de départ a été choisie assez éloignée de la porte (x_I=-500 et y_I=-500) pour que la différence entre les erreurs mesurées soit significative. En effet si, pour une trajectoire de type 1 par exemple, on a une erreur après le premier virage, celle-ci va augmenter proportionnellement à la distance parcourue en ligne droite.

Les trajectoires de type 1 et de type 2 ont toutes les deux été testées.

Influence du rayon de courbure pour une trajectoire de type 1

Pour vérifier l'existence d'un rayon de courbure optimale, l'orientation initiale importe peu. On a donc choisi arbitrairement $\theta_I = 3\pi/2$. Les résultats obtenus sont représentés sur la figure 5.8.

On remarque que le phénomène, observé lorsque le robot décrit un cercle, se produit également lorsque le Koala parcourt une trajectoire de type 1. Pour un rayon de courbure $R = 125mm$, la surface S et l'erreur d'orientation sont minimales. Ce rayon de courbure de 125mm est le rayon de courbure optimal pour la configuration de départ choisie.

On a dans un second temps décidé de vérifier si le rayon de courbure optimal est le même pour une configuration de départ symétrique par rapport à l'axe des abscisses. En effet pour deux configurations de départ symétriques, les trajectoires décrites sont elles aussi symétriques par rapport à ce même axe. On a donc refait le même test pour la configuration de départ suivante : $x_I = -500$, $y_I = 500$ et $\theta_I = \pi/2$. La figure 5.9 montre que pour un rayon de courbure de 125mm, S est minimale et $|\varepsilon_\theta|$ est très proche de la valeur

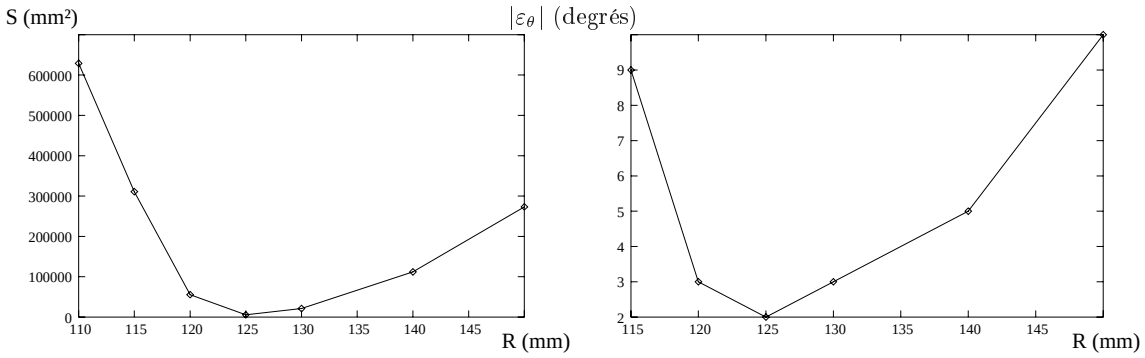


FIG. 5.8 – Variations de S et ε_θ pour une trajectoire de type 1, pour la configuration initiale $(-500; -500; 3\pi/2)$

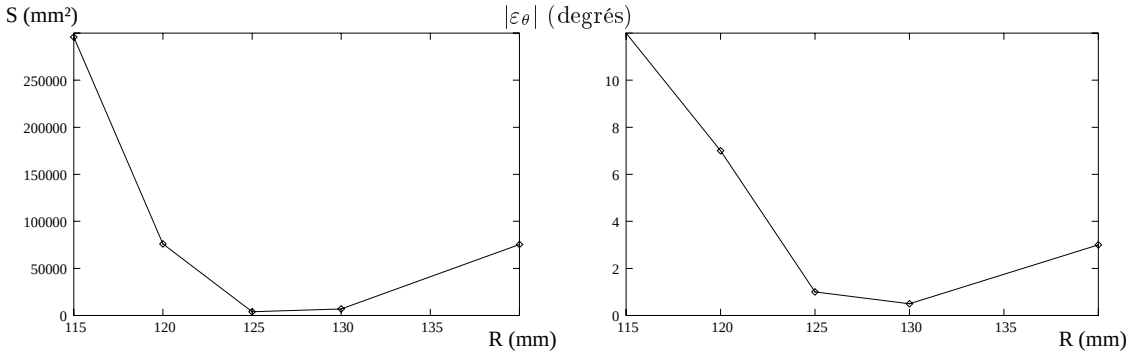


FIG. 5.9 – Variations de S et ε_θ pour une trajectoire de type 1, pour la configuration initiale $(-500; 500; \pi/2)$

minimale (environ un demi degré d'écart).

On peut donc en conclure que le comportement du robot est bien symétrique par rapport à l'axe des abscisses.

Influence du rayon de courbure pour une trajectoire de type 2

Après avoir vérifié l'existence d'un rayon de courbure optimal pour une trajectoire de type 1, les tests ont été faits pour une trajectoire de type 2. La configuration de départ est $x_I = -500$, $y_I = -500$ et $\theta_I = \pi/2$. Sur la figure 5.10 on voit que la surface S est minimale pour $R = 122mm$, alors que l'erreur d'orientation est, elle, minimale pour $130mm < R < 140mm$. Comme on prévilégie la valeur de S (cf. paragraphe 5.1.2), la rayon de courbure considéré comme optimal pour cette configuration de départ est $R = 122mm$.

Pour la configuration initiale symétrique par rapport à l'axe des abscisses

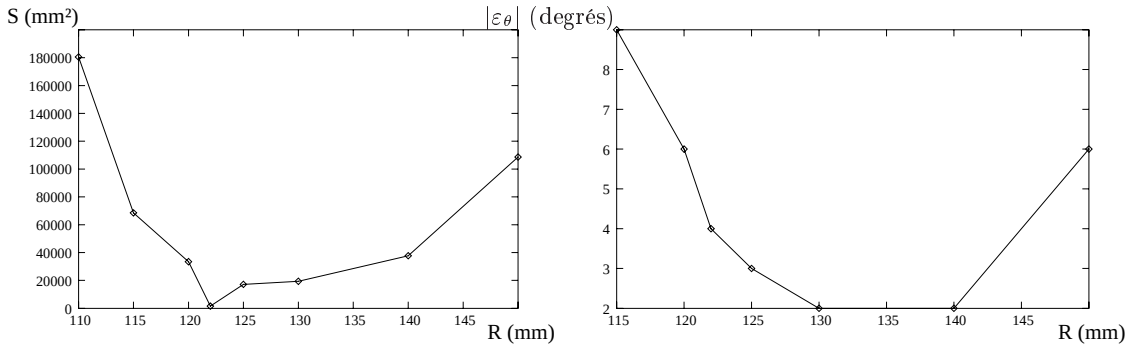


FIG. 5.10 – Variations de S et ε_θ pour une trajectoire de type 2, pour la configuration initiale $(-500; -500; \pi/2)$

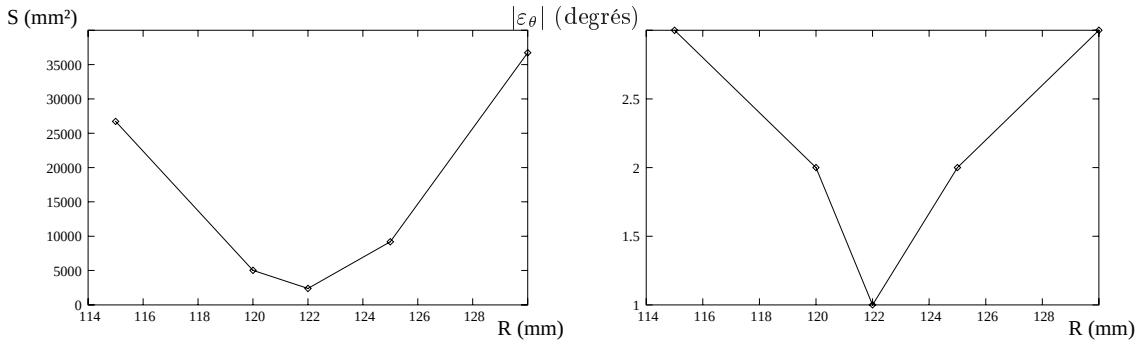


FIG. 5.11 – Variations de S et ε_θ pour une trajectoire de type 2, pour la configuration initiale $(-500; 500; 3\pi/2)$

($x_I = -500$, $y_I = 500$ et $\theta_I = 3\pi/2$), on obtient les courbes de la figure 5.11. Le rayon de courbure optimal pour la surface S et pour l'erreur d'orientation est $R = 122mm$, le même que pour la position symétrique.

Conclusions

On peut tirer deux conclusions de ces huits courbes : pour une vitesse du centre du robot fixée, il existe bien un rayon optimal qui réduit les erreurs de position et d'orientation à l'arrivée, et le comportement du Koala est symétrique par rapport à l'axe des abscisses. On peut désormais déduire de mesures faites pour une configuration de départ un comportement similaire pour la configuration symétrique par rapport à l'axe des abscisses.

5.4 Optimisation du choix du type de trajectoire

Cette étape doit permettre de fixer la valeur du coefficient k,utilisé par l'algorithme de choix pour opter pour un type de trajectoire ou l'autre.

Les tests ont été faits pour les configurations de départ suivantes, représentées sous la forme du triplet $(x_I\ ;y_I\ ;\theta_I)$: $(-500; -500; \pi/2)$, $(-500; -500; \pi/4)$, $(0; -500; \pi/4)$, $(0; -500; \pi/2)$ et $(0; -500; 3\pi/4)$. La vitesse est de 0,3m/s.

En ce qui concerne le rayon de courbure, il devait initialement être le même pour les deux types de trajectoire. Or on a remarqué que le rayon de courbure optimal variait en fonction du type de trajectoire utilisé et de la configuration de départ.

En ce qui concerne la variation en fonction de la trajectoire utilisée, le but de ce test étant de comparer la précision de chaque trajectoire, il fallait se placer dans les conditions optimales pour les deux trajectoires. On a donc utilisé un rayon de courbure différent pour chaque trajectoire.

La variation du rayon de courbure optimal en fonction de la configuration initiale a entrainé l'utilisation du *rayon optimal moyen* pour chacune des trajectoires. Le *rayon optimal moyen* est le rayon de courbure qui minimise la moyenne des S pour les cinq configurations de départ testées. Pour la trajectoire de type 1, on obtient un rayon optimal moyen égal à 130mm, et pour la trajectoire de type 2 égal à 122mm.

Les mesures ont ensuite été faites pour les deux types de trajectoire pour les configurations de départ précisée plus haut, et on obtient les résultats représentés sur la figure 5.12.

On voit sur cette figure que quelque soit la configuration de départ, la valeur de S est plus petite pour des trajectoires de type 2. La différence entre les deux types de trajectoire est la plus significative pour les configurations $(0; -500; \pi/2)$ et $(0; -500; 3\pi/4)$. L'utilisation d'une trajectoire de type 2 permet une réduction moyenne de 44% de S.

En ce qui concerne l'erreur d'orientation, elle est en général plus faible pour les trajectoires de type 1. Cependant la réduction moyenne de cette erreur n'est que de 25% en utilisant une trajectoire de type 1.

On en déduit que, dès que cela est possible, il est préférable d'utiliser une trajectoire de type 2 pour réduire l'erreur. Cependant pour certaines orientations initiales telle que 0, π, $5\pi/4$, $3\pi/2$ ou $7\pi/4$, il n'existe pas de trajectoire de type 2. Il faut donc trouver un rayon optimal moyen pour les trajectoires de type 1 pour ces orientations de départ, afin de réduire l'erreur à l'arrivée.

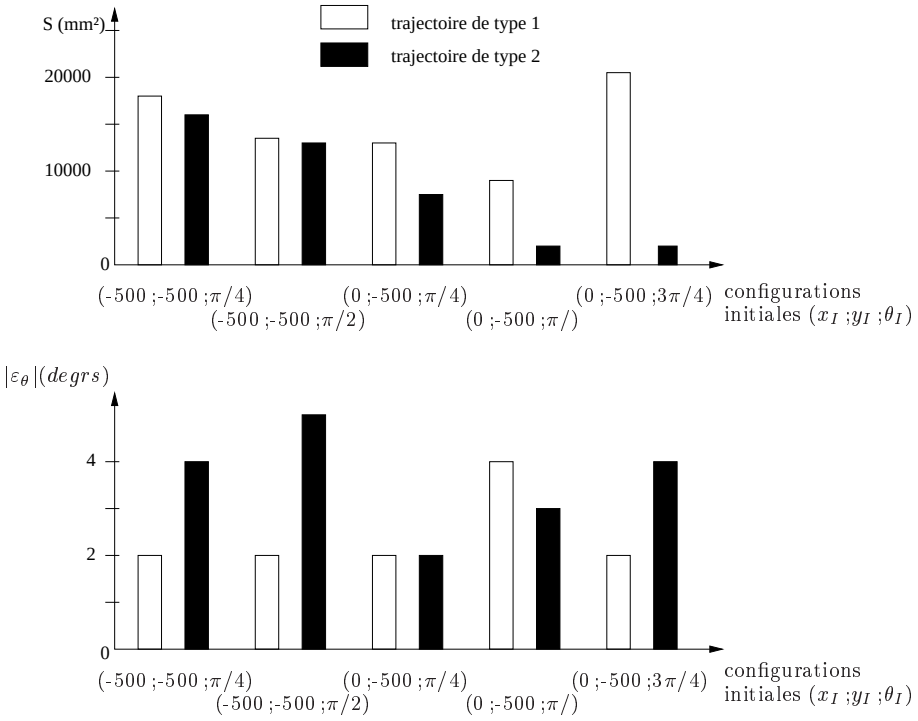


FIG. 5.12 – Comparaison des erreurs à l'arrivée pour des trajectoires de type 1 et de type 2

5.5 Recherche d'un rayon optimal moyen pour les trajectoires de type 1

Etant donné que le rayon optimal varie en fonction de la configuration de départ, on a cherché un rayon optimal moyen qui pourrait être utilisé par les trajectoires de type 1 quand il n'existe pas de trajectoire de type 2 pouvant permettre au robot de se rendre au point final.

Le rayon de courbure optimal n'étant pas facile à déterminer avec précision, on recherche l'intervalle de $\pm 5mm$ dans lequel il se trouve. Cet intervalle est plus facile à déterminer car il a été constaté que l'erreur d'orientation change de signe quand le rayon optimal est dépassé. En incrémentant de $\pm 5mm$ le rayon de courbure en fonction du sens de variation de ε_θ, on peut trouver l'intervalle de R contenant le rayon optimal.

Dans un premier temps, on a cherché cet intervalle pour 15 configurations de départ n'admettant que des trajectoires de type 1 pour se rendre devant la porte.

Etant donné que le comportement du robot est symétrique par rapport à l'axe des abscisses, l'ordonnée du point de départ y_I a été fixée à -500. Par

	$x_I = -500$	$x_I = 0$	$x_I = 500$
π	[125,130]	[125,130]	[120,125]
$5\pi/4$	[120,125]	[115,120]	[115,120]
$3\pi/2$	[125,130]	[125,130]	[105,110]
$7\pi/4$	[125,130]	[130,135]	[130,135]
0	[135,140]	[135,140]	[125,130]

TAB. 5.2 – Récapitulatif des intervalles contenant le rayon optimal

symétrie on pourra en déduire le comportement du robot pour $y_I = 500$ et ainsi étendre le domaine de validité des tests effectués.

Les orientations choisies pour ce test sont des orientations comprises entre 0 et π, tous les $\pi/4$. Pour ces orientations, il n'existe pas de trajectoire de type 2.

Enfin trois abscisses de départ ont été étudiées $x_I = -500$, $x_I = 0$ et $x_I = 500$.

La vitesse est toujours de 0,3m/s. Les intervalles obtenus sont répertoriés dans le tableau 5.2.

On remarque dans ce tableau que le rayon optimal appartient à des intervalles allant de [105,110] à [135,140] en fonction de la configuration de départ. Cela signifie qu'il n'est pas possible de trouver un rayon optimal qui limite l'erreur pour toutes les configurations de départ.

Cependant on peut essayer de trouver un rayon optimal moyen qui minimise l'erreur sur l'ensemble des configurations de départ testées. En se réferrant au tableau 5.2, on remarque que les quatre rayons qui seraient susceptibles d'être utilisés comme rayon optimal moyen sont R=120mm, R=125mm, R=130mm et R=135mm. Pour en choisir un, on a comparé la valeur moyenne de S pour les 15 configurations de départ testées pour chacun des quatre rayons. Ces résultats sont présentés sur la figure 5.13.

Le rayon de courbure le plus approprié pour être utilisé comme rayon optimal moyen semble être le rayon de 125mm. Cependant il est à noter que l'utilisation de ce rayon de courbure provoque de grosses erreurs ($S > 100000mm^2$) pour les configurations initiales suivantes : (0 ;-500 ;$5\pi/4$), (500 ;-500 ;$5\pi/4$) et ((500 ;-500 ;$3\pi/2$). Le rayon de courbure de 125mm reste néanmoins le meilleur choix car les autres rayons de courbure entrainent de plus grandes erreurs pour certaines configurations.

Le rayon de courbure optimal moyen pour les trajectoires de type 1 est donc de 125mm, tout en sachant qu'il pourra engendrer de grandes erreurs de position et d'orientation pour certaines configurations initiales.

Pour ne plus avoir de configurations initiales problématiques, la solution

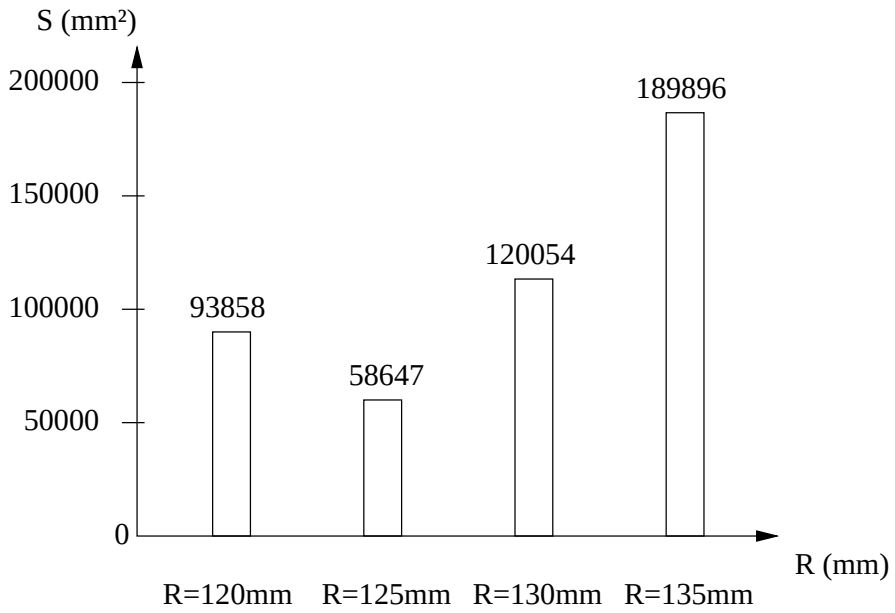


FIG. 5.13 – Valeurs moyennes de S pour les quatre rayons de courbure à comparer

serait de réaliser un tableau de correspondance entre la configuration de départ et le rayon de courbure optimal pour cette configuration. Le robot lancerait alors l'algorithme de planification avec ce rayon optimal, ce qui permettrait de toujours avoir une erreur minimale.

L'erreur d'orientation n'a pas été utilisée pour cette comparaison car on peut la corriger plus facilement que l'erreur de position représentée par S. En effet l'erreur d'orientation peut être corrigée par un simple pivotement, après une localisation par exemple.

5.6 Problème de l'accélération

Comme nous l'avons vu au paragraphe 4.1, l'accélération n'est pas réglable si on utilise la commande en vitesse. L'accélération maximale est utilisée pour les deux roues. Dans les virages, cela va introduire une erreur.

Influence de l'accélération identique des roues

L'accélération identique des roues intérieure et extérieure va introduire, en début de virage, un écart par rapport au rayon de courbure voulu. Pour se rendre compte de ce phénomène, comparons notre cas à un cas pour lequel l'accélération serait réglable de manière assymétrique (cf. figure 5.14). Sur cette figure a_{ext} et a_{int} sont respectivement les accélérations des roues

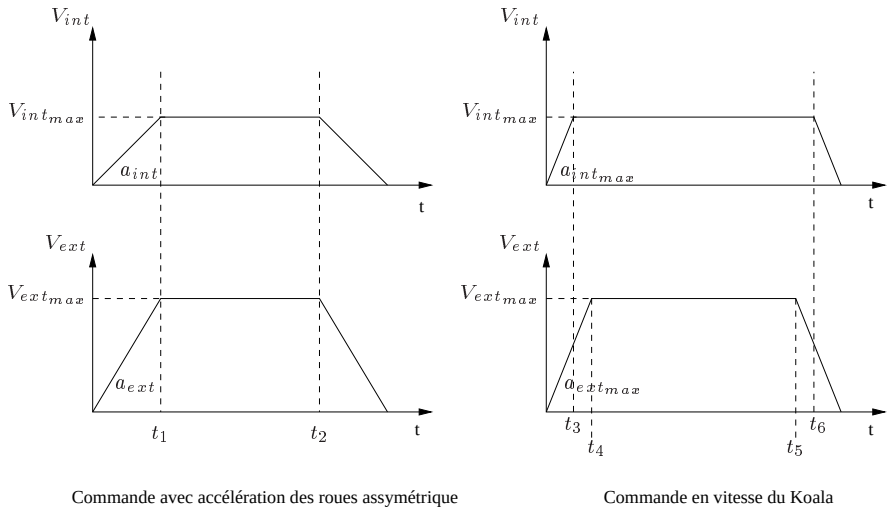


FIG. 5.14 – Comparaison entre une commande avec accélération assymétrique, et la commande en vitesse du Koala

extérieure et intérieure, et $V_{ext_{max}}$ et $V_{int_{max}}$ sont respectivement les valeurs des vitesses de consigne des roues extérieure et intérieure.

Pendant la phase d'accélération on a :

$$V_{ext} = a_{ext} \cdot t \text{ et } V_{int} = a_{int} \cdot t \tag{5.13}$$

Or on a vu au paragraphe précédent que la relation à vérifier pour décrire une trajectoire de rayon de courbure R est :

$$\frac{V_{ext}}{V_{int}} = \frac{2R+b}{2R-b} \tag{5.14}$$

En remplaçant dans l'équation 5.13, on obtient la condition que doivent vérifier les accélérations des roues extérieure et intérieure pour avoir un rayon de courbure R pendant la phase d'accélération :

$$\frac{a_{ext}}{a_{int}} = \frac{2R+b}{2R-b} \tag{5.15}$$

Dans le cas d'une accélération assymétrique, on règle a_{ext} et a_{int} pour quelles vérifient la relation 5.15. Sur la figure 5.14, on remarque que les accélérations des roues extérieure et intérieure sont proportionnelles car les roues atteignent leurs vitesses de consigne $V_{ext_{max}}$ et $V_{int_{max}}$ au même instant t_1. Durant la phase d'accélération, la courbure est constante et égale à R.

Pour la commande en vitesse du Koala, le rapport $\frac{a_{ext}}{a_{int}}$ est fixe et égale à 1 (aux différences de performances des moteurs près). Or quelque soit R, le rapport $\frac{2R+b}{2R-b}$ est différent de 1. Le rayon de courbure varie donc de la façon suivante (cf. figure 5.14) :

- il est infini entre 0 et t_3, le robot se déplace donc en ligne droite ;
- il est inférieur à R entre t_3 et t_4 ;
- il est égale à R après t_4 jusqu'à la décélération.

Le phénomène inverse se produit lors de l'arrêt du robot : le rayon de courbure diminue entre t_5 et t_6, et le robot se déplace en ligne droite après t_6 jusqu'à son arrêt. L'équation de la variation du rayon de courbure en fonction du temps est donnée par la formule déduite de l'équation 5.5 :

$$R = \frac{1}{2} b \frac{(V_{ext} + V_{int})}{(V_{ext} - V_{int})} \tag{5.16}$$

Pour l'intervalle de temps $[t_3, t_4]$, cette équation devient :

$$\forall\, t \in [t_3, t_4]\ R = \frac{1}{2} b \frac{(a_{max} \cdot t + V_{int_{max}})}{(a_{max} \cdot t - V_{int_{max}})} \tag{5.17}$$

$$\text{Avec } a_{max} = a_{ext} = a_{int}$$

Cette équation est de la forme :

$$R = k_1 \left(\frac{t + k_2}{t - k_2} \right) \tag{5.18}$$

$$\text{Avec } k_1 = \frac{b}{2} \text{ et } k_2 = \frac{V_{int_{max}}}{a_{max}}$$

On reconnaît l'équation d'une hyperbole. Cependant l'accélération maximale n'étant pas donnée dans le manuel du Koala, il est difficile de connaître avec précision la variation du rayon de courbure. La courbe de la figure 5.15, qui représente la variation du rayon de courbure pendant l'intervalle de temps $[t_3, t_4]$ lors d'un virage, a été tracée en utilisant une valeur de a_{max} égale à $0,6 m/s^2$, une vitesse $V_{int_{max}}$ de 0,1m/s et un rayon de courbure R de 200mm.

Il est à noter que le phénomène présenté ci-dessus est plus influant dans le cas d'une trajectoire de type 1, et en particulier dans le premier virage d'une trajectoire de type 1. En effet pour ces trajectoires, le robot commence son mouvement en virage. Sa vitesse initiale est nulle et l'accélération a une forte influence sur le mouvement. Pour le second virage ou pour une trajectoire de type 2, le robot n'atteindra pas le virage avec une vitesse nulle, il est déjà en mouvement. L'accélération ou la décélération des roues sera plus courte que pour une trajectoire de type 1, et donc moins influente.

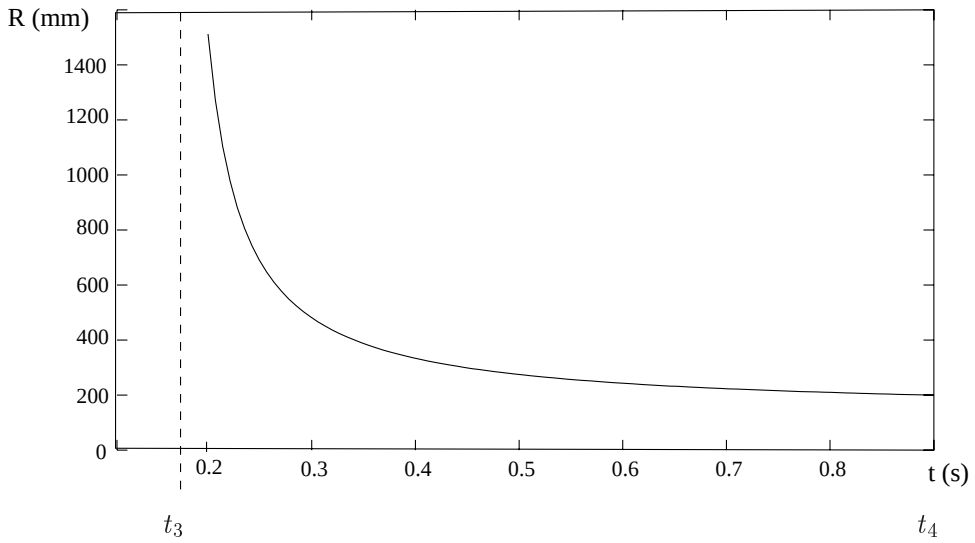


FIG. 5.15 – Variation du rayon de courbure en fonction du temps lors d'un virage avec une commande en vitesse

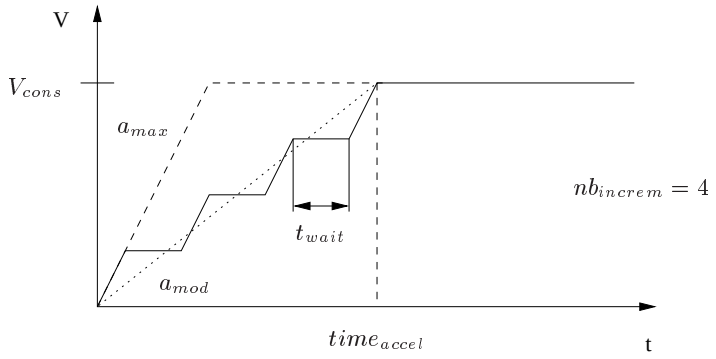


FIG. 5.16 – Discrétisation de l'accélération

Discrétisation de l'accélération

La discrétisation de l'accélération est une méthode qui permettrait de contrôler l'accélération en incrémentant la vitesse d'une roue (cf figure 5.16) et donc d'obtenir une accélération différente sur les deux roues. On appelera la nouvelle accélération obtenue *accélération modifiée*, a_{mod}. On a :

$$a_{mod} = \frac{V_{cons}}{time_{accel}} \tag{5.19}$$

Avec $V_{cons} = V_{ext}$ ou $V_{cons} = V_{int}$ en fonction de la roue commandée.

En modifiant le paramètre $time_{accel}$, qui est le temps au bout duquel la roue aura atteint sa vitesse de consigne, on peut régler la valeur de *l'accélération modifiée* de la roue.

Le second paramètre à choisir est le nombre d'incrémentation de la vitesse, nb_{increm}. Plus celui-ci sera élévé, et plus les consignes de vitesse envoyées à la roue seront nombreuses.

Les consignes de vitesse durant l'accélération seront données par la formule :

$$\forall\, i \in [1, nb_{increm}]\ V_i = \frac{i}{nb_{increm}} V_{cons} \tag{5.20}$$

Le temps que doit rester la roue à cette vitesse V_i est constant, et donné par la relation :

$$t_{wait} = \frac{time_{accel}}{nb_{increm} - 1} - \frac{V_{cons}}{nb_{increm} a_{max}} \tag{5.21}$$

On voit que la valeur t_{wait}, qui va servir à commander le robot, dépend de l'accélération maximale a_{max}. Comme on ne connaît pas sa valeur, on ne peut pas calculer t_{wait} avec précision. Dans le programme de commande des roues, le second terme de l'expression 5.21 a donc été négligé. Cela n'est théoriquement valable que si la valeur de a_{max} est grande devant celle de V_{cons}.

Il faudra donc vérifier expérimentalement si l'utilisation de l'accélération discrétisée apporte un gain de précision par rapport à une commande en vitesse classique.

Influence réelle de l'accélération discrétisée

La validité de la méthode de commande en vitesse avec accélération discrétisée a été vérifiée expérimentalement, en la comparant à la commande en vitesse classique.

En essayant de régler les paramètres de l'accélération $time_{accel}$ et nb_{increm}, on se rend compte que le facteur limitant est la bande passante de 38400 bauds de la liaison série qui relie le Koala et le serveur.

En effet la durée de certains virages étant si courte, il faut réduire la durée de l'accélération pour que les roues atteignent leur vitesse de consigne avant la fin du virage. La fréquence des requêtes envoyées au robot pour qu'il modifie sa vitesse (une requête est envoyée à chaque incrémentation) est alors trop importante. Si on limite le nombre d'incrémentations pour réduire la fréquence des requêtes, on réduit par là même l'intérêt de l'utilisation de l'accélération discrétisée.

Enfin, la durée des virages variant en fonction de la configuration de départ, il faudrait modifier les paramètres pour chaque configuration.

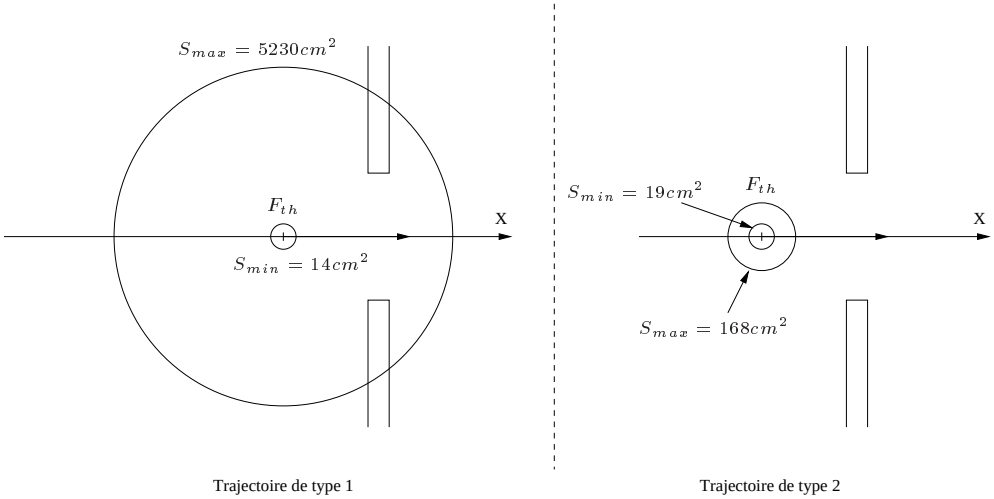


FIG. 5.17 – Représentation des valeurs maximales et minimales de S pour chacune des trajectoires

L'utilisation de l'accélération discrétisée, qui semble intéressante en théorie, se révèle être, dans la pratique, d'un intérêt limité. Elle pourrait être envisagée avec une liaison de plus haut débit entre le Koala et le poste serveur.

5.7 Conclusion : Modifications à apporter à l'algorithme de planification

Après cette phase d'expérimentations, deux modifications sont à apporter à l'algorithme de planification. La première concerne le choix entre les trajectoires de type 1 et de type 2. Les tests ont montré que les trajectoires de type 2 engendrent moins d'erreur de position à l'arrivée, elles sont donc à choisir à chaque fois qu'elles existent.

La seconde modification à apporter est la différenciation des rayons de courbure pour les trajectoires de type 1 et 2. En effet le rayon optimal moyen choisi pour les trajectoires de type 1 est de 125mm, alors que celui des trajectoires de type 2 est de 122mm.

Après avoir fait ces modifications, on peut représenter les valeurs maximales et minimales de S pour chacune des trajectoires par le schéma de la figure 5.17.

Chapitre 6

Conclusion

6.1 Bilan des résultats obtenus

Ce travail a permis d'arriver à un fonctionnement complet du robot du point de vue de la navigation, même si certains problèmes subsistent :

- Les *algorithmes de construction et de choix* de trajectoire fonctionnent et permettent au robot d'arriver devant la porte. Cependant il n'existe pas de solution pour toutes les configurations de départ. La zone des positions de départ n'admettant pas de trajectoire pour se rendre au but correspond à une bande de 250mm de large située sur le pourtour de l'environnement.
- L'*erreur de position* à l'arrivée a été minimisée en adaptant les paramètres de la trajectoire, notamment le rayon de courbure. Le rayon de courbure optimal pour les trajectoires de type 1 a été fixé à 125mm, alors que celui des trajectoires de type 2 est de 122mm. Pour les trajectoires de type 1, l'erreur S est comprise entre $14cm^2$ et $5230cm^2$, si on considère toutes les configurations testées ; elle est comprise entre $14cm^2$ et $334cm^2$, si on ne tient pas compte des configurations problématiques. Pour les trajectoires de type 2, S est compris entre $19cm^2$ et $168cm^2$. Suite aux résultats des expérimentations, l'algorithme de départ a été modifié pour toujours choisir la trajectoire de type 2 si elle existe, et pour construire les deux types de trajectoire avec des rayons de courbure différents. Cependant, même avec le rayon de courbure optimal, certaines configurations de départ induisent beaucoup d'erreur à l'arrivée pour les trajectoires de type 1.

6.2 Perspectives et améliorations possibles

Réduction de la zone des positions de départ n'admettant pas de solution

Pour réduire cette zone, on peut envisager que le robot, quand il est dans une position problématique, réalise un premier mouvement pour atteindre une position admettant une trajectoire pour aller jusqu'à la porte. Une fois à cette position, il lance l'algorithme de calcul et de choix.

Réduction de l'erreur de position à l'arrivée

Quelques améliorations sont possibles pour encore réduire l'erreur de position à l'arrivé :

- On a vu qu'il subsiste toujours une erreur d'orientation à l'arrivée. Cette erreur est corrigeable par simple rotation du Koala sur lui-même. Il serait donc intéressant de lancer une localisation à l'arrivée avec le programme de Cyrille Roussel pour savoir de combien il faut corriger l'orientation.
- Comme on a vu que les trajectoires de type 2 sont plus précises, le robot pourrait effectuer un mouvement supplémentaire au départ pour se retrouver dans une configuration qui admet une trajectoire de type 2.
- Pour l'utilisation des trajectoires de type 1, une correspondance entre configuration de départ et rayon de courbure optimal serait intéressante à établir. Le robot pourrait alors toujours calculer sa trajectoire en utilisant le rayon de courbure optimal.
- Enfin l'utilisation du programme de localisation pourrait aussi permettre d'effectuer une correction de position au milieu de la trajectoire : le robot part de son point de départ, il s'arrête au milieu de sa trajectoire, lance le programme de localisation et recalcule une nouvelle trajectoire à partir de la configuration localisée.

Annexe A

Notice d'utilisation du programme Matlab

Les différentes fonctions uitilisées sont décrites dans le tableau A.1.

Quand on est sous Matlab, lancer le programme *main*. Le programme demande alors de taper 'c' pour tracer une courbe seule, ou de taper 'm' pour calculer plusieurs trajectoires pour une orientation donnée.

Si on a choisi de tracer une courbe seule, il faut ensuite saisir la configuration de départ (x_I en mm, y_I en mm et θ_I en radians). La courbe choisie par l'algorithme de choix est ensuite tracée dans le fenêtre graphique.

Si 'm' est entré au clavier et qu'on désire réaliser un maillage de l'environnement et calculer les trajectoires pour plusieurs positions de départ, le programme demande la valeur de l'orientation initiale (en radians) pour laquelle on veut effectuer le maillage. Il faut ensuite entrer le pas du maillage désiré (il correspond à l'intervalle entre deux positions consécutives de départ). On a enfin la possibilité de tracer toutes les trajectoires trouvées, ou de n'avoir que les positions de départ qui n'admettent pas de solutions pour se rendre au point final.

Le réglage des paramètres se fait en éditant le fichier 'main.m'.

NOM DE LA FONCTION	DESCRIPTIF
constructraj2	calcule les coordonnées des points de passage et des points de construction de la trajectoire 2
appart2	vérifie l'appartenance à l'environnement de la trajectoire 2
longtraj2	calcule la longueur de la trajectoire 2
trajectoire2	lance la fonction constructraj2, puis si la trajectoire existe, lance la fonction appart2, et enfin si la trajectoire appartient à l'environnement, lance longtraj2.
choixcercles1 et choixcercles2	choisissent respectivement le cercle de départ et le cercle d'arrivée de la trajectoire 1
tangentes	calcule les coordonnées des centres des cercles choisis et des points de tangence entre ces deux cercles
nonpivote	détermine les deux points de tangence qui seront les points de passage de la trajectoire 1
appart1	vérifie l'appartenance à l'environnement de la trajectoire 1
longtraj1	calcule ma mongueur de la trajectoire 1
constructraj1	lance les fonctions tangentes puis nonpivote, et, si le robot ne se trouve pas dans la zone interdite, lance la fonction appart1
trajectoire1	lance les fonctions choixcercles1 et choixcercles2, puis la fonction constructraj1. si aucune trajectoire n'est trouvée, relance constructraj1 en utilisant l'autre cercle de départ. si finalement une trajectoire est trouvée, lance longtraj1.
choixtrajectoire	lance les fonctions trajectoire1 et trajectoire2, et choisit une des trajectoires comme expliqué dans l'algorithme de choix de trajectoire
main	fait l'interface avec l'utilisateur du programme, lance choixtrajectoire pour une configuration ou pour un maillage de l'environnement. trace la (les) solution(s) si elle(s) existe(nt), trace le point initial seulement si elle(s) n'existe(nt) pas.

TAB. A.1 – Tableau descriptif des fonctions utilisées dans le programme Matlab

Annexe B

Notice d'utilisation du programme C

Avant d'utiliser le programme de planification, il faut s'assurer que le fichier texte contenant les valeurs du rayon de courbure, de la vitesse en virage, de la vitesse en ligne droite et du coéfficient de choix k se trouve bien dans le même répertoire que le fichier exécutable. Si ce fichier texte n'est pas présent, le message d'erreur "Fichier non ouvert ! ! !" apparaitra.

Pour lancer le programme, il faut préciser après le nom du programme respectivement le nom du serveur (*odie* dans l'état actuel des choses), le port qui a été ouvert et qui servira pour la communication entre le Koala et le poste serveur, et enfin le nom complet du fichier texte. Au lancement de l'exécutable, les valeurs lues dans le fichier texte sont affichées à l'écran. Le programme demande ensuite d'entrer respectivement x_I, y_I et θ_I. Le Koala décrit alors la trajectoire choisie par l'algorithme et les coordonnées des points de passage s'affichent, ainsi que les valeurs des angles au sommet des arcs de cercle décrits. Si le port utilisé pour la liaison entre le Koala et le poste serveur n'a pas pu être trouvé, un message d'erreur apparaît à l'écran.

Les paramètres autres que ceux présents dans le fichier texte, tels que les dimensions du robot, les dimensions de l'environnement, certaines constantes mathématiques et mécaniques, ..., sont définis dans le fichier 'param.h'.

Bibliographie

[1] Cyrille Roussel *Conception d'un système de navigation pour un robot mobile basé sur la détection des marques p-similaires.* Stage de DEA ENSAM-SUPELEC, 2001.

[2] Vincent Rieger . Studienarbeit ENSAM-SUPELEC, 2000.

[3] Alain Pruski *Robotique mobile, la planification de trajectoire.* Hermès, Paris, 1996.

[4] Alain Pruski *Robots mobiles autonomes.* Article R7850, Volume S, publié sur le site www.techniques-ingenieur.fr, 1998.

[5] E. W. Dijkstra *A note on two Problems in connexion with graphs.* Numerische Mathematik 1 pp. 269-271 1959.

[6] L. E. Dubins *On curves of minimal lenght with a constraint on average curvature, and with prescribed initial and terminal positions and tangents.* American Journal of Mathematics 79. pp. 497-516 1957.

[7] J.A. Reeds et L.A. Shepp *Optimal path for a car that goes both forwards and backwards.* Pacific Journal of Mathematics Vol. 145 n° 2 1990.

[8] *Programmation, génération de trajectoires et recalages pour le robot mobile autonome SARAH* Thèse de l'Université de Picardie, 1991.

[9] Hans-Joachim von der Hardt *Contribution au pilotage et à la localisation d'un robot mobile.* Thèse du CRAN, 1997.

[10] Y. Kanayama, Y. Kimura, F. Miyazaki, T. Nogushi *A stable tracking control method for an autonomous mobile robot.* In Proceedings of the 1990 IEEE International Conference on Robotics and Automation. pp. 384-389 Cincinnati, Ohio, Etats-Unis, 1990.

[11] K-Team S.A. *Koala User Manual* Version 1.1 Lausanne, 10 novembre 1999.

Table des figures