UNIVERSITE DE SAAD DAHLEB DE BLIDA

Faculté des Sciences de l'ingénieur
Département d'Electronique

MEMOIRE DE MAGISTER

Spécialité : Contrôle

CONTRIBUTION A LA LOCALISATION SIMULTANE D`UN ROBOT MOBILE DANS UN ENVIRONNEMENT DYNAMIQUE

Par

ABDELAZIZ Mohammed

Devant le jury composé de :

M.Bounekhla	Professeur, Université de Blida	Président
Z.Benselama	Maitre de conférences(A), Université de Blida	Examinateur
A.Ferdjouni	Maitre de conférences(A), Université de Blida	Examinateur
O.Azouaoui	Maitre de recherche(A), C.D.T.A, Alger	Rapporteur
N.Bennila	Charge de cours, Université de Blida	Invité

Blida, juin 2013

RESUME

Le travail présenté dans ce document s'inscrit dans le cadre de la navigation en robotique mobile. Il consiste en une implémentation du problème SLAM (Simultaneous Localization and mapping). L'algorithme est basé sur une méthode de mise en correspondance (scan matching) appelée 'Normal Distrubution Transform' (NDT), ceci en utilisant un capteur télémètre laser 2D et une représentation géométrique de l'environnement par points. L'algorithme développé (SLAM-NDT) a été implémenté et testé sur le robot mobile Robucar. Les résultats obtenus ont été présentés et expliqués dans le dernier chapitre de ce mémoire.

Mots clés : SLAM, localisation, cartographie, NDT, Robucar.

ABSTRACT

The work presented in this thesis is part of the navigation mobile robotics. It consists of an implementation of the SLAM problem (Simultaneous Localization and mapping). The algorithm is based on a scan matching method Normal Distribution Transform (NDT), using a 2D laser range finder and a geometric representation of the environment points. The developed algorithm (SLAM-NDT) has been implemented and tested on the mobile robot Robucar. The results are presented and discussed in the last chapter of this thesis.

Key words: SLAM, localization, Mapping, scan matching, NDT, Robucar.

الملخص

يندرج العمل المعروض في هده الاطروحة في اطار الملاحة في الروبوتيك المتحركة , و يتم ذلك بتطبيق تقنية SLAM

(تحديد المكان و رسم الخريطة في آن واحد)ا لخوارزمية تعتمد على طريقة تسمى NDT(التوزيع الطبيعي او الغاوسي)

و ذلك باستخدام حساس ليزري ثنائي البعد و تمثيل هندسي للنقاط.الخوارزمية قد تم تطبيقها و تجريبها على الروبوت المتنقل Robocar , النتائج المحصل عليها قد تم عرضها و شرحها في آخر فصل من هذه الأطروحة.

مفاتيح :SLAM, تسوية القياسات, رسم الخريطة, تحديد المكان,NDT,Robucar

REMERCIEMENTS

Tout d’abord je voudrais exprimer ma reconnaissance à Monsieur M.Bounekhla

pour avoir accepter la présidence du jury de ma thèse.

Je remercie également l’ensemble des membres du jury : Monsieur Z.Benselama

et Monsieur A.Ferdjouni pour avoir accepter d’examiner ce manuscrit et pour

m’avoir fait l’honneur d’évaluer mon travail.

je remercie chaleureusement Mll Azouaoui.O pour m’avoir encadré et accueilli au sein de son équipe au CDTA.et pour m’avoir proposé un sujet passionnant.

Je remercie également Monsieur Kazed.B pour son aide. Je remercie aussi tous ceux qui ont participé de près ou de loin à l’achèvement de ce travail.

DEDICACES

Je dédie ce travail à,

A mes parents

A mes frères

A mes sœurs

A tous mes amis

TABLE DES MATIERES

LISTE DES FIGURES

Glossaire

C : la matrice de covariance

H : la matrice Hessienne

I : la matrice unité

J_T : la matrice jacobienne

N : la distribution normale

P : La densité de probabilité

$\vec{P}$: Le vecteur des paramètres à estimer

$\Delta\vec{P}$: le vecteur de déplacement

X_i : les points mesurés par le capteur laser du scan i

X'_i : Le point X_i écrit dans le repère de coordonnées du premier scan

X''_i : le point transformé de X'_i

T : La fonction de transformation 2D entre deux positions du robot

f : La fonction à minimiser

g : le gradient de f

q : la moyenne

t_x : paramètre de translation sur l'axe x

t_y : paramètre de translation sur l'axe y

x : coordonnée sur l'axe x

y : coordonnée sur l'axe y

θ : l'angle de rotation

λ : un scalaire positif

INTRODUCTION

Dans les dernières décennies, beaucoup de robots ont été développés, produits et installés principalement pour des applications industrielles. Plusieurs tâches pénibles ou répétitives, réalisées par des opérateurs humains, peuvent être avantageusement confiées à des systèmes mécaniques. De telles machines sont souhaitables pour des tâches d'intervention en des lieux inaccessibles à l'homme, où l'intervention d'un robot peut être moins coûteuse en temps et argent, comme les milieux hostiles et distants : sous-marin, en terrain miné, l'exploration planétaire, la reconnaissance militaire, … Elles doivent alors être dotées d'un dispositif de locomotion et peuvent être autonomes ou contrôlées à distance par un opérateur humain. Dans ce type d'applications, le robot ne remplace pas la main-d'œuvre humaine, bien au contraire, il devient une extension humaine.

Une des principales caractéristiques des robots autonomes est de s'adapter à l'environnement qui les entoure, ceci afin de réaliser les tâches qui leurs sont confiées sans intervention d'un opérateur. Parmi ces tâches, la navigation autonome est essentielle à toute mission robotique. La réalisation d'une telle tâche requiert certaines fonctionnalités à savoir: la localisation, la cartographie, la planification, et l'exécution, etc.

La fonctionnalité de localisation est un problème fondamental de la robotique. En effet, il n'est pas possible de contrôler un robot sans estimation précise de sa position. Les données issues de ses capteurs proprioceptifs (odométrie, gyromètre, et accéléromètre, …) sont généralement insuffisantes car la position du robot ne peut être estimée que par rapport à sa position précédente. Pour compenser l'accumulation des erreurs, le processus de localisation a besoin de données externes précises. Ces données vont alors être fusionnées avec les informations proprioceptives. De nombreuses équipes utilisent un capteur télémétrique laser pour résoudre leurs problèmes de localisation, afin d'avoir la meilleure estimation globale de la position.

L'avantage principal de ce type de capteur est la qualité des données recueillies. En effet, les données d'une caméra, par exemple, doivent être extraites de l'image (contours, points d'intérêts, ...) et celles issues de capteurs infrarouges, odomètres ou ultrasons sont généralement entachées d'une grande incertitude. Au contraire, les données de télémétrie laser ont de nombreux avantages : précision des distances et des angles mesurés, grande portée et projection aisée des données dans l'espace de travail du robot.

Un autre problème qui se pose dans la robotique mobile est la construction de la carte de l'environnement dans laquelle évolue le robot. Pour ce faire, il faut que les données acquises par les différents capteurs du robot soient riches en informations, et comme nous l'avons cité dans le paragraphe précédent, la qualité des données laser et les caractéristiques qu'il possède permettent de bien cartographier l'environnement. En effet, en quelques minutes, il est possible de relever des millions de points 2D avec précision. Le post-traitement des données laser est une étape indispensable afin de rendre la carte lisible et complète.

Cette construction doit être faite simultanément avec la localisation du robot. Ce problème est connu et référencé par le terme (SLAM : Simultaneous Localization And Mapping). On le considère comme un problème complexe, parce qu'une carte précise et détaillée représente une source d'information indispensable pour réaliser une bonne localisation du robot. Mais pour élaborer une carte correcte, une localisation du robot dans son environnement est aussi importante. Les deux fonctions sont liées, ont la même priorité et doivent être achevées simultanément. Pour la plupart des travaux, la recherche s'est concentrée sur le SLAM dans les environnements statiques. Cependant, la compréhension de scènes dans l'environnement est primordiale pour rendre un robot vraiment autonome. La scène autour du robot se compose d'objets stationnaires et/ou d'objets mobiles. Dans les applications où le robot se déplace dans des environnements avec des personnes, comme la bureautique, le monde est dynamique, et contient des entités stationnaires et mobiles. En conséquence, l'apparition d'entités mobiles dans la zone de perception du robot pendant qu'il construit sa propre carte peut provoquer de graves erreurs au niveau de sa localisation, et des inexactitudes dans la carte, ce qui rend

le problème SLAM plus délicat. Ainsi, les traces des objets mobiles doivent être détectées et éliminées du modèle.

Parmi les techniques utilisées pour résoudre le problème SLAM, on trouve le filtre de Kalman [1], la mise en correspondance des scans (scan matching) [2], ….
Dans notre travail, nous nous focalisons sur cette dernière, dont l'algorithme utilisé est la NDT (Normal Distribution Transform) [3]. L'objectif de cet algorithme est de calculer le mouvement relatif d'un robot entre deux configurations consécutives de l'environnement ou bien entre une configuration actuelle et la carte construite, en maximisant la superposition entre les mesures obtenues à chaque configuration. Quant à la construction de la carte, il faut définir un repère global pour les différentes configurations dans lequel les mesures seront référencées entre les différents scans et bien regroupées et fusionnées afin d'avoir une seule carte. Pour le cas dynamique, nous utilisons le même algorithme en ajoutant des méthodes pour détecter les points mobiles et les éliminer de la carte de l'environnement.

Ce travail a été réalisé dans la Division Productique et Robotique (DPR) au sein de l'équipe NCRM (Navigation et Contrôle des Robots Mobiles) du CDTA. Il traite la technique de la mis en correspondance basée sur l'algorithme SLAM-NDT. Ce dernier permet de calculer le mouvement relatif d'un robot et de construire la carte de l'environnement en même temps. Cet algorithme a été implémenté sur le robot mobile Robucar sous l'environnement Genom dans le module localisation en parallèle avec d'autres travaux qui traitent la technique SLAM-ICP (SLAM-Iterative Closest Points). Il exploite les mesures laser (du capteur LMS 200) issues du module Lms.

Ce mémoire est divisé en quatre chapitres:

Le **chapitre 1** est consacré à l'étude bibliographique du problème énoncé ci-dessus. Une présentation de quelques types de robots mobiles existants et des différents capteurs disponibles sur le marché est introduite.
Le **chapitre 2** présente les différentes techniques de localisation et cartographie simultanées utilisées pour localiser un robot mobile.

Le **chapitre 3** décrit en détail la méthode NDT en expliquant les différentes étapes avec les démonstrations des différentes équations de l'algorithme ainsi que l'adaptation de la méthode au cas dynamique.
Le **chapitre 4** porte sur les aspects de l'implémentation du module proposé. Ainsi, il donne des détails sur la phase de développement et les outils logiciels utilisés. Il présente aussi les résultats d'expérimentation dans différents environnements.

Nous terminons ce mémoire par une conclusion générale, où nous présentons une synthèse du travail effectué au cours de ce projet, ainsi que des perspectives quant à l'amélioration éventuelle de notre travail.

CHAPITRE 1
INTRODUCTION A LA ROBOTIQUE MOBILE

1. 1. Introduction

Ce chapitre est réservé à la présentation des robots mobiles ainsi que leurs modèles cinématiques. Dans un premier temps, une présentation des différents types de robots mobiles à roues les plus utilisés en robotique mobile est abordée. Par la suite, une brève synthèse sur les capteurs utilisés dans la perception est élaborée. Enfin, les différentes méthodes de modélisation de l'environnement et de localisation permettant d'estimer la position d'un robot dans l'environnement sont présentées.

1. 2. Historique des robots mobiles

Un robot mobile est une machine équipée de capacités de perception, de décision et d'action qui lui permettent d'agir de manière autonome dans son environnement en fonction de sa perception de l'environnement.

Le terme robot apparaît pour la première fois dans une pièce de Karel Capek en 1920 (Rossum's Universal Robots) [4]. La tortue construite par Grey Walter dans les années 1950 est l'un des tous premiers robots mobiles autonomes. Ce robot est capable de se diriger vers une lumière qui marque un but et de s'arrêter face à des obstacles et de recharger ses batteries. Dans les années 60, les recherches en électronique ont conduit à des robots plus complexes. Ainsi le robot "Beast" de l'université John Hopkins peut se déplacer au centre des couloirs en utilisant des capteurs ultrasons [4]. En 1969, les premiers liens entre la recherche en intelligence artificielle et la robotique apparaissent à Stanford avec le robot Shakey [4]. Ce robot utilise des capteurs à ultrason et une caméra. Depuis, ces développements ont continué où l'arrivée sur le marché à partir des années 1990 des plates-formes intégrées a permis à de très nombreux laboratoires de travailler sur la robotique mobile et a conduit à une explosion de la diversité des thèmes de recherche [4].

1. 3. Les différentes classes de robots mobiles et leurs modèles

1. 3. 1. Disposition des roues d'un robot mobile

C'est la combinaison du choix des roues et de leur disposition qui confère à un robot son mode de locomotion propre. Sur les robots mobiles, on rencontre principalement quatre types de roues (Figure 1.1) :

• les roues fixes dont l'axe de rotation, de direction constante, passe par le centre de la roue (Figure 1.1.a).

• les roues centrées orientables dont l'axe d'orientation passe par le centre de la roue. (Figure 1.1.b).

• les roues décentrées orientables souvent appelées roues folles, pour lesquelles l'axe d'orientation ne passe pas par le centre de la roue (Figure 1.1.c).

• Les roues suédoises dont les bandes de roulement ont été remplacées par des galets inclinés par rapport à la normale au plan de la roue. C'est la combinaison de la rotation de la roue avec la rotation libre du galet en contact avec le sol qui permet un déplacement sans glissement dans toutes les directions (Figure 1.1.d).

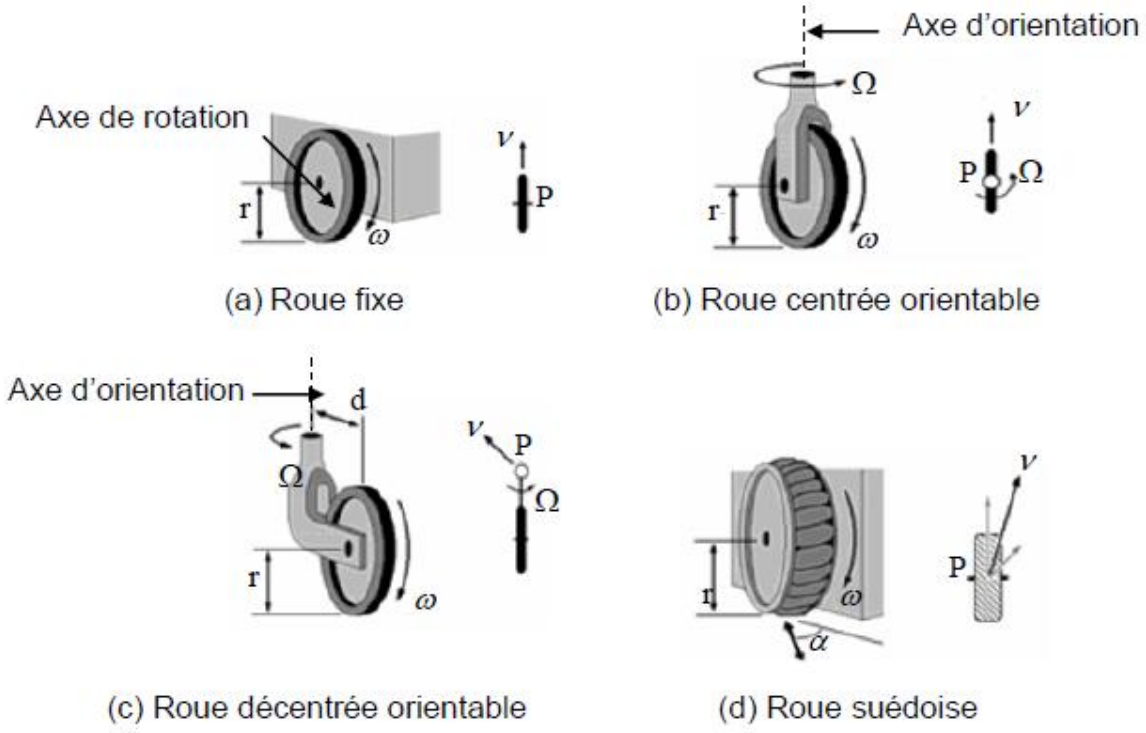


Figure 1.1 : Les principaux types de roues pour robots mobiles

Ces quatre types de roues sont les plus utilisés en robotique mobile mais d'autres types existent tels que les roues sphériques connues pour leur propriété omnidirectionnelle [5].

Bien évidemment, pour un ensemble de roues données, toute disposition ne conduit pas à une solution viable. Un mauvais choix peut limiter la mobilité du robot ou occasionner d'éventuels blocages [5]. Par exemple, un robot équipé de deux roues fixes non parallèles ne pourra pas aller en ligne droite. Pour qu'une disposition de roues soit viable et n'entraîne pas de glissement des roues sur le sol, il faut qu'il existe pour toutes ces roues un unique point de vitesse nulle autour duquel tourne le robot¸ en instantanée. Ce point, lorsqu'il existe, est appelé centre instantané de rotation (CIR). Il correspond aux points de vitesse nulle liés aux roues. Pour cette raison, il existe en pratique quatre principales catégories de robots mobiles à roues, qui sont présentées au paragraphe suivant.

1. 3. 2. Robots mobiles de type unicycle

Un robot mobile unicycle comporte deux roues fixes non orientables commandées indépendamment et possède un certain nombre de roues folles. Le schéma d'un robot de type unicycle est donné à la figure 1.2. Les roues folles n'interviennent pas dans la cinématique, dans la mesure où elles ont été judicieusement placées pour assurer l'équilibre.

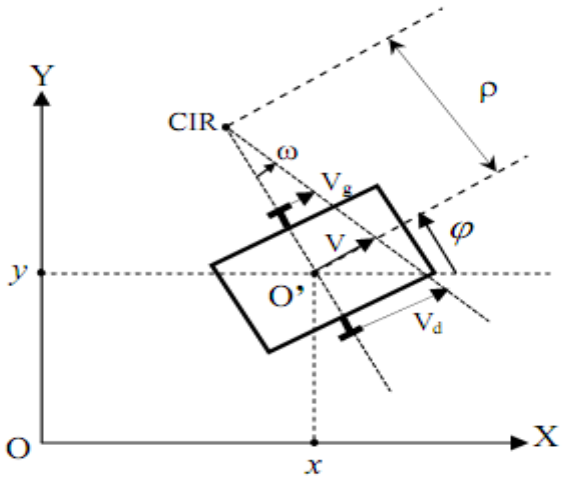


Figure 1.2 : Robot mobile unicycle

Avec :

ρ: rayon de courbure de la trajectoire du robot

x : position du robot sur l'axe X

y : position du robot sur l'axe Y

φ: angle d'orientation du robot

ω: la vitesse de rotation du robot autour du CIR

V : vitesse longitudinale

V_g: vitesse de la roue gauche

V_d: vitesse de la roue droite

Ce type de robot est très répandu en raison de sa simplicité de construction et de ses propriétés cinématiques intéressantes. Le modèle cinématique du robot unicycle est donné par les équations suivantes **[6]**:

$$\begin{cases} \dot{x} = V\ cos\varphi \\ \dot{y} = V\ sin\varphi \\ \dot{\varphi} = \omega \end{cases} \qquad (1.1)$$

1. 3. 3. Robots mobiles de type tricycle

L'architecture d'un robot mobile de type tricycle est représentée dans la figure 1.3. Ce robot est constitué de deux roues fixes de même axe et d'une roue centrée orientable placée sur l'axe longitudinal du robot. Le mouvement est conféré au robot par la vitesse longitudinale et l'orientation de la roue orientable.

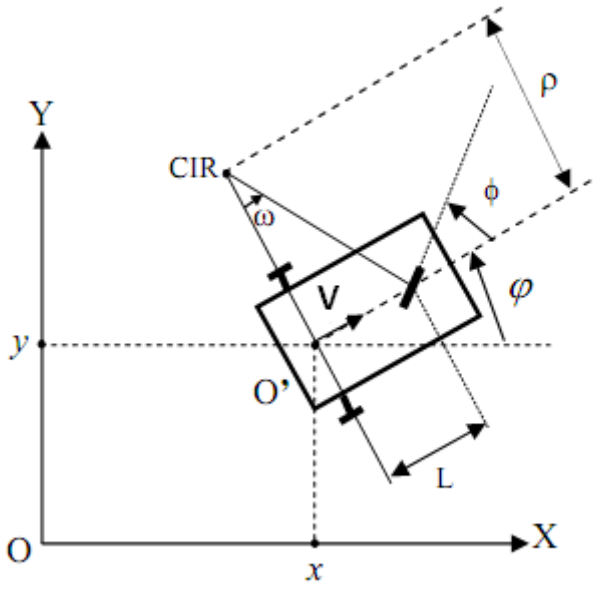


Figure 1.3 : Robot mobile tricycle

Avec :

$$\begin{cases} \rho: rayon\ de\ courbure \\ \phi: angle\ de\ braquage\ de\ la\ roue \\ L: distance\ entre\ l'axe\ des\ deux\ roues\ fixes\ et\ la\ roue\ orientable \end{cases}$$

Le modèle cinématique de ce type de robot est donné par les équations suivantes **[6]** :

$$\begin{cases} \dot{x} = V\ cos\varphi \\ \dot{y} = V\ sin\varphi \\ \dot{\varphi} = \omega = \frac{V}{L}\ tan\phi \end{cases} \qquad (1.2)$$

1.3.4. Robots mobiles de type voiture

Le cas du robot mobile de type voiture est très similaire à celui du tricycle. La différence se situe au niveau du train avant, qui comportent deux roues au lieu d'une (Figure 1.4) **[5]**. Le robot mobile Robucar, à savoir la plateforme expérimentale utilisée dans ce travail, est un exemple de ce type de robots (pour plus de détail voir le chapitre IV).Le modèle cinématique de ce type de robot est donné par les équations suivantes **[6]** :

$$\begin{cases} \dot{x} = -V_R \sin(\varphi - k\phi) \\ \dot{y} = V_R \cos(\varphi - k\phi) \\ \dot{\varphi} = \omega = \frac{V_R \sin(\phi + k\phi)}{L \cos\phi} \end{cases} \quad (1.3)$$

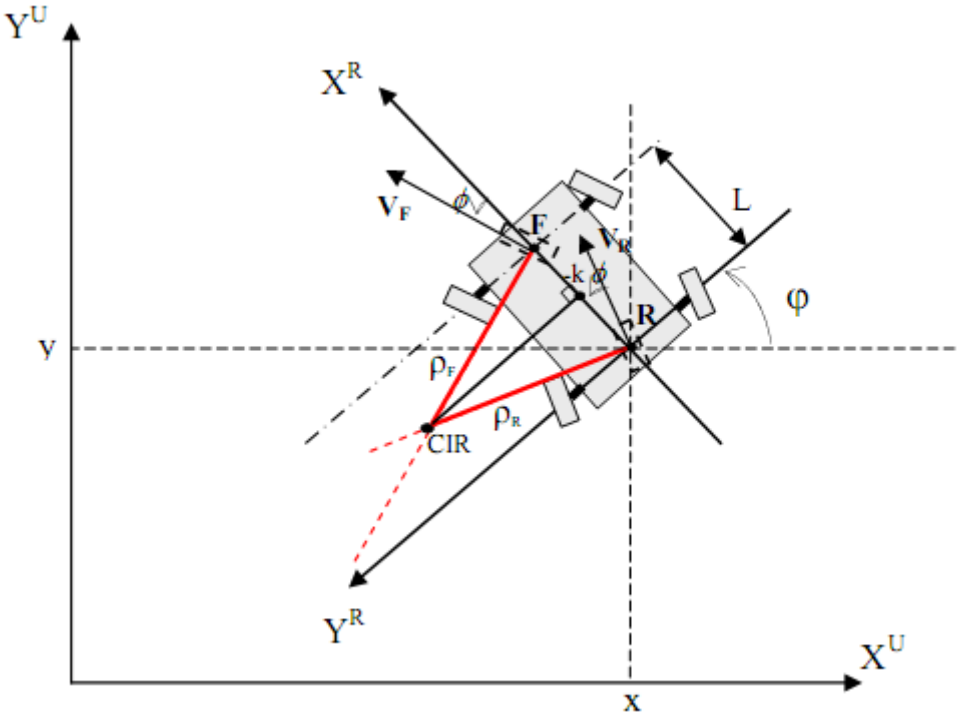


Figure 1.4 : Robot mobile de type voiture

Avec :

$$\begin{cases} V_R : vitesse\ linéaire \\ \phi : angle\ de\ braquage\ des\ roues \\ \varphi : angle\ d'orientation\ du\ robot \\ L : distance\ entre\ l'axe\ des\ roues\ avant\ et\ arrière \\ k : facteur\ de\ proportionalité\ entre\ l'angle\ de\ braquage\ des\ roues\ du\ train\ avant\ et\ les\ roues\ du\ train\ arrière \end{cases}$$

Le signe moins avant le facteur de proportionnalité c'est du au faite que les deux angles de braquage sont opposé.

1. 3. 5. Robots mobiles omnidirectionnels

Un robot mobile est dit omnidirectionnel si l'on peut agir indépendamment sur les vitesses : vitesse de translation selon les axes $\vec{x}$ et $\vec{y}$ et vitesse de rotation autour de $\vec{z}$. D'un point de vue cinématique, on montre que cela n'est pas possible avec des roues fixes ou des roues centrées orientables **[6]**. On peut en revanche réaliser un robot omnidirectionnel en ayant recours à un ensemble de trois roues décentrées orientables ou de trois roues suédoises disposées aux sommets d'un triangle équilatéral (Figure 1.5). Du point de vue de la transmission du mouvement, ceci ne va pas sans poser de problème.

Trois roues sont suffisantes mais dans certains cas une quatrième offre des possibilités d'optimisation et permet de rendre le système plus robuste en évitant le glissement dans le cas d'un sol qui ne serait pas parfaitement plat par exemple **[5]**.

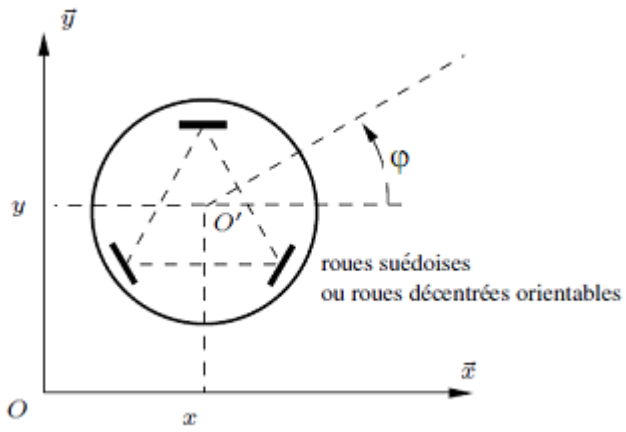


Figure 1.5 : Robot mobile de type omnidirectionnel

Dans ce cas, on peut considérer qu'il est possible d'appliquer directement la commande sur le modèle cinématique qui est défini par les équations suivantes **[5]** :

$$\begin{aligned} \dot{x} &= u_1 \\ \dot{y} &= u_2 \\ \dot{\varphi} &= u_3 \end{aligned} \qquad (1.4)$$

Où $u = (u_1, u_2, u_3)^T$ représente le vecteur de commande.

1. 4. Perception

La notion de perception en robotique mobile est relative à la capacité du système à recueillir, traiter et mettre en forme des informations utiles au robot pour agir et réagir dans le monde qui l'entoure **[6]**. Pour extraire les informations utiles à l'accomplissement de sa tâche, il est nécessaire que le robot dispose de nombreux capteurs mesurant aussi bien son état interne que l'état de l'environnement dans lequel il évolue.

Un système de perception doit être capable de distinguer le plus de lieux possibles, afin de faire la différence entre deux lieux différents mais d'apparences similaires. Or, l'augmentation de cette capacité à distinguer de petites variations dans l'environnement rend le système sensible au changement de perception au cours du temps pour un lieu donné (problème de la variabilité perceptuelle). Cette variabilité peut être due au bruit inhérent au processus de mesure où à des variations de l'environnement. Pour s'affranchir de ce problème, il faut en général mettre en place des processus de traitement des perceptions qui permettront de ne pas dépendre de ces variations et d'identifier correctement un lieu donné **[4]**.

L'ambiguïté perceptuelle, en anglais « perceptual aliasing » est un autre problème que peut rencontrer un système de perception, dans le cas où le système de perception donne une même mesure pour deux amers différents. Ce problème désigne l'incapacité d'un système de perception à distinguer de manière unique tous les lieux d'un environnement. Cette situation est très courante lorsque les robots utilisent des capteurs extéroceptifs tels que les capteurs à ultrasons **[4]**. Par exemple, dans un environnement d'intérieur ces capteurs sont capables de mesurer la distance du robot par rapport à un coin, mais ne fournissent aucune information le long d'un couloir rectiligne qui dépasse leur portée. Toutes les positions le long d'un couloir correspondent alors à des perceptions identiques. Ce problème trouve une solution par l'utilisation d'un capteur extéroceptif plus précis tel que le capteur laser ou une caméra **[4]**. Nous pouvons définir deux catégories de capteurs couramment utilisées en robotique mobile : ceux qui interviennent dans le fonctionnement du système robotique et ceux qui délivrent une information caractérisant l'environnement. Les premiers sont appelés capteurs *proprioceptifs* qui fournissent des informations propres au comportement interne du robot

mobile, et les seconds sont les capteurs *extéroceptifs* qui fournissent des informations sur le monde extérieur au robot mobile. Le choix des capteurs dépend bien évidemment de l'application envisagée.

1. 4. 1. Les capteurs proprioceptifs

Les capteurs proprioceptifs fournissent par intégration des informations élémentaires sur les paramètres cinématiques du robot mobile. Les informations sensorielles gérées dans ce cadre sont généralement des vitesses, des accélérations, des angles de rotation ou des angles d'attitude. Ce type de capteurs peut être regroupé en deux familles ; les capteurs de déplacement et les capteurs d'attitude **[7]**.

1. 4. 1. 1. Capteurs de déplacement

Les capteurs de déplacement comprennent les odomètres, les accéléromètres, les radars doppler, les mesureurs optiques, etc. Cette catégorie permet de mesurer des déplacements élémentaires, des variations de vitesse ou d'accélération sur des trajectoires rectilignes ou curvilignes.

1.4.1.1.1. Les odomètres :

Les systèmes odométriques fournissent la position du robot mobile pendant son mouvement, par intégration des rotations élémentaires de ses roues. Le principe de l'odométrie consiste à mesurer la rotation des roues qui sont effectuées dans la plupart des cas par des codeurs optiques incrémentaux montés sur les roues du robot mobile. Cette information est utilisée pour déduire une mesure de la vitesse de rotation des roues. L'odométrie est l'un des systèmes les plus utilisés en robotique mobile car ce système présente beaucoup d'avantages **[8]**:

• fonctionne indépendamment de l'environnement,
• système peu coûteux,
• fréquence d'acquisition élevée,
• très bonne précision à court terme,
• très grande facilité de mise en œuvre,
• système disponible en tout temps,
• système autonome et fiable.

Cependant, l'odométrie présente également des inconvénients :

• mauvaise précision à long terme : l'erreur en orientation induit d'importantes erreurs en position. Les erreurs peuvent être regroupées en deux catégories selon leur source. Les erreurs systématiques résultent des imperfections du modèle géométrique du robot (diamètres des roues différents, incertitude sur les dimensions des axes de la base,…).

Les erreurs non systématiques résultent de l'interaction entre le robot et son environnement telle que les glissements ou les chocs qui ne sont pas pris en compte dans la mesure du mouvement effectué.

• sensibilité au terrain : puisque l'odométrie ne détecte pas les irrégularités du terrain; les erreurs de position peuvent être considérables si la surface d'évolution du robot n'est pas lisse.

• localisation relative : l'odométrie ne fournit qu'une localisation relative, elle a donc besoin d'être initialisée pour obtenir un positionnement absolue.

1.4.1.1.2. Les accéléromètres

L'accéléromètre est un capteur qui mesure l'accélération linéaire en un point donné. La mesure à l'aide d'une jauge de contrainte de la force $\vec{F}$ appliquée à un mobile de masse m permet de déterminer l'accélération m γ = $\frac{F}{m}$ que subit ce mobile et donc le robot mobile (Figure 1.6). Les mesures fournies par ces capteurs doivent être intégrées deux fois pour donner une information de position ce qui induit une faible immunité au bruit pour les faibles accélérations **[9]**.

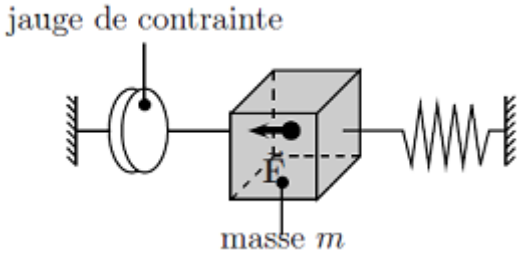


Figure 1.6 : Principe de fonctionnement d'un accéléromètre.

1. 4. 1. 2. Capteurs d'attitude

Les capteurs d'attitude mesurent deux types de données : les angles de cap et les angles de roulis et de tangage. Ces capteurs sont principalement de type inertiel.

Ces capteurs ont pour point commun d'être généralement coûteux et sensibles au bruit, d'où une intégration moins fréquente dans les robots mobiles que les odomètres. Ils sont principalement les gyroscopes, les gyromètres, les inclinomètres et les magnétomètres.

1. 4. 2. Les capteurs extéroceptifs

Les capteurs extéroceptifs sont employés en robotique mobile pour collecter des informations sur le milieu d'évolution d'un robot mobile. Ils sont le complément indispensable aux capteurs proprioceptifs présentés précédemment. Les capteurs proprioceptifs fournissent des informations sur le déplacement du robot mobile, alors que les capteurs extéroceptifs fournissent des informations sur la position du robot mobile dans l'environnement. Les informations fournies par un capteur extéroceptif permettent de choisir des perceptions qui peuvent être utilisées comme points de repère (des amers). Ces points de repère sont indépendants des déplacements du robot mobile et pourront être reconnus quelle que soit l'erreur cumulée par les données proprioceptives. La reconnaissance de ces points est évidemment soumise à une incertitude, mais pas à une erreur cumulative, ce qui les rend utilisables comme référence à long terme.
Deux familles de capteurs extéroceptifs embarqués peuvent être identifiées : les capteurs télémétriques et les systèmes de vision **[10]**.

1. 4. 2. 1. Les capteurs télémétriques

On appelle télémétrie toute technique de mesure de distance par des procédés acoustiques, optiques ou radioélectriques. L'appareil permettant de mesurer les distances est appelé *télémètre*. Il existe différentes technologies pour réaliser un télémètre.
Nous présentons dans ce qui suit les télémètres les plus utilisés dans les systèmes de robotique mobile à savoir les capteurs à ultrasons et les capteurs laser.

1.4.2.1.1. Les capteurs à ultrasons

Les capteurs à ultrasons, appelés également sonar, sont les premiers à avoir été employés dans les systèmes de robotique mobile. Ils ont été originellement utilisés dans les applications sous-marines et dans les systèmes autofocus d'appareils photographiques. L'idée de base de la mesure de distance par sonar est la suivante. L'émetteur à ultrasons produit un court train d'ondes ultrasonores, et un récepteur reçoit l'écho de ce train d'ondes. Le signal de retour est utilisé pour la mesure du temps de vol (time of flight en anglais (TOF)), c'est à dire la mesure du temps écoulé entre le début de l'émission et la fin de la réception du signal acoustique. Connaissant la célérité du son dans l'air (≈ 340m/s) et le temps de vol, il est facile de calculer la distance parcourue par l'onde. Par contre, il est plus délicat de connaître les paramètres comme la température, l'humidité et la pression atmosphérique, alors que ceux-ci modifient la vitesse du son. Ces paramètres n'étant pas connus exactement, ils contribueront à l'incertitude sur la mesure de la distance. En environnement d'intérieur, ces paramètres sont plus ou moins constants. La distance à l'objet ayant renvoyé l'onde est donnée par la relation suivante : $D = \frac{1}{2}$ cT

où c est la célérité du son, et T le temps d'aller-retour mesuré. Il existe plusieurs facteurs qui ont un effet sur la propagation de l'énergie acoustique **[9] :**

- L'intensité décroît proportionnellement à l'inverse de la distance parcourue au carré car l'émission se fait dans un cône et non en ligne droite.
- La réflectivité de la cible conditionne la quantité énergétique renvoyée.
- Le vent et la température affectent la propagation de l'onde dans l'air.
- L'onde acoustique est diffractée par le capteur et se propage à l'intérieur d'un cône (Figure 1.7). Le capteur le plus fréquemment utilisé, l'émetteur/ récepteur Polaroid, possède une ouverture à 3dB de 25°et une plage de mesure de 30 cm à 10m **[9]**. En raison du phénomène de diffraction, il n'est pas possible de déterminer la direction de détection de l'objet à partir d'une seule position du robot. L'objet est situé quelque part sur la surface sphérique terminale du cône, notée S sur la figure 1.7.

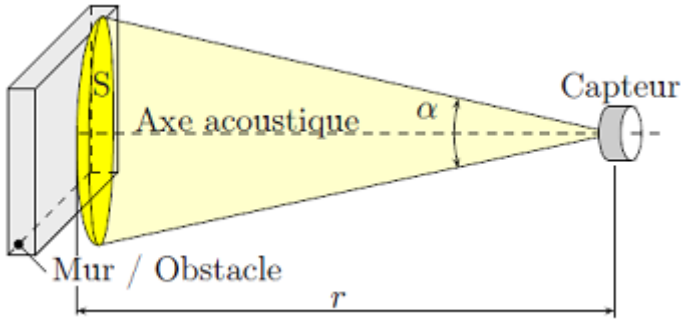


Figure 1.7 – Cône d'émission de l'énergie acoustique.

La mesure de la distance à un objet (distance « r » sur la figure 1.7) est déterminée par la mesure du temps de vol, mais elle ne permet pas de déterminer la direction de détection de l'objet à partir d'une seule position du capteur.

Les capteurs à ultrasons sont simples et peu coûteux, mais possèdent de nombreux inconvénients induisant une utilisation de moins en moins fréquente **[11]**:

- ✓ Une très faible directivité qui est liée au cône d'émission de l'onde dont l'angle d'ouverture est important.
- ✓ Un angle d'incidence relativement faible qui n'excède généralement pas 30 à 40° suivant le matériau de la paroi.
- ✓ Une forte sensibilité aux conditions d'utilisation telles que ; l'écho parasite, la température et l'humidité.
- ✓ Une forte influence aux problèmes de réflexions multiples. Ce phénomène se produit lorsque l'onde ultrasonore heurte plusieurs parois avant de revenir sur le capteur.

1.4.2.1.2. Les capteurs laser

Les capteurs laser, les plus utilisés à l'heure actuelle pour les applications de la robotique mobile, sont les capteurs laser à balayage qui présentent une haute précision et vitesse de mesure **[10]**. Ces capteurs émettent en rotation un faisceau laser pour balayer l'environnement. Le faisceau laser émis est très concentré, ce qui permet d'avoir un cône d'émission très étroit et donc une bonne précision de mesure.

Un capteur laser est constitué d'un corps et d'une tête (Figure 1.8). Son principe de fonctionnement consiste à émettre une impulsion luminescente par une diode laser, à ce moment précis une horloge est démarrée. L'impulsion lumineuse est renvoyée par le premier obstacle rencontré sur son chemin et retourne (au capteur laser) sur un récepteur qui déclenche l'arrêt de l'horloge. La distance séparant le capteur et l'obstacle est déduite en mesurant le temps de vol du faisceau laser (temps d'aller + temps de retour)/2).

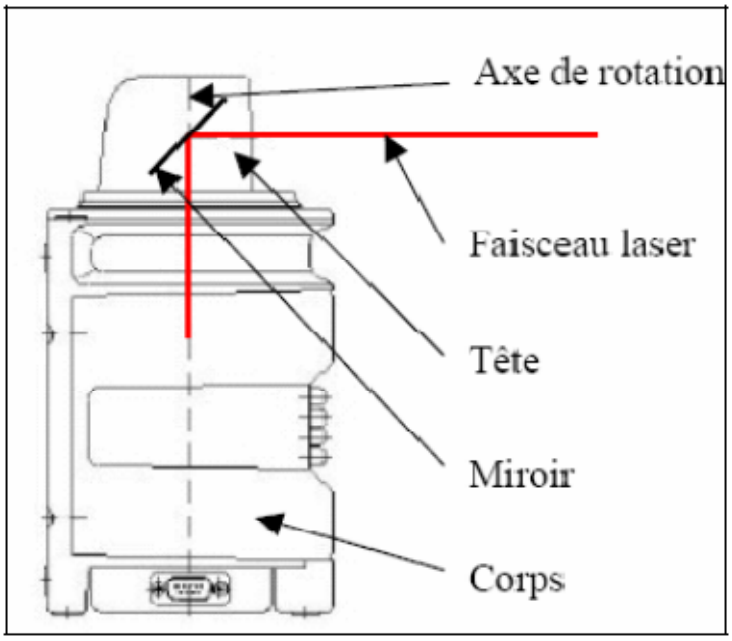


Figure 1.8 : Description d'un capteur laser

1.4.2.1.2.1. Le capteur laser LMS 200

Le capteur laser LMS 200 (Figure 1.9) est un scanner de mesure à balayage à deux dimensions, il scrute son environnement et récupère les coordonnées polaires de celui-ci **[10]**. C'est une combinaison d'un télémètre à temps de vol avec un système de rotation du faisceau de mesure (à des fréquences allant de 5 à 20 Hz). Grâce à cette technique nous obtenons une vision sous forme de radar.

Figure 1.9 : Le capteur laser LMS 200

1.4.2.1.2.2. Caractéristiques du capteur laser LMS 200

Le capteur laser LMS 200 possède les caractéristiques et avantages suivants **[10]**:

• Mesure optique sans contact, même à grande distance.
• Temps de scrutation rapide permettant la mesure d'objets en mouvement.
• Mesure des objets dans n'importe quelle position.
• Système actif ne demandant pas d'éclairage spécifique.
• Utilisation à l'extérieur (versions étanches).
• Auto test intégré.

Ils possèdent toutefois un certain nombre d'inconvénients. En premier lieu, leur zone de perception est restreinte à un plan et ne permet donc pas de détecter les obstacles situés hors de ce plan. Ils ne peuvent pas non plus détecter les objets ne réfléchissant pas correctement la lumière du laser (en premier lieu les vitres, mais aussi certains objets très réfléchissants, tels que les objets chromés).

1. 4. 2. 2. Les systèmes de vision

L'élément de base d'un système de vision est la caméra. Une caméra est caractérisée par deux groupes de paramètres ; les paramètres intrinsèques et les paramètres extrinsèques. Alors que les premiers définissent la géométrie interne

de la caméra, les seconds permettent de définir la localisation (position et orientation) tridimensionnelle de la caméra.

La caméra permet la projection d'un espace à trois dimensions vers une image en deux dimensions. Quel que soit le modèle de caméra, les éléments suivants sont toujours présents (Figure 1.10) :

- le centre de la caméra C (également nommé centre optique),
- le point principal P confondu avec le centre de l'image,
- l'axe optique : droite passant par les points C et P,
- la distance focale : distance entre le point C et le point P,
- le plan image : plan sur lequel se projettent les points 3D de l'espace, il est normal à l'axe optique et est situé en P.

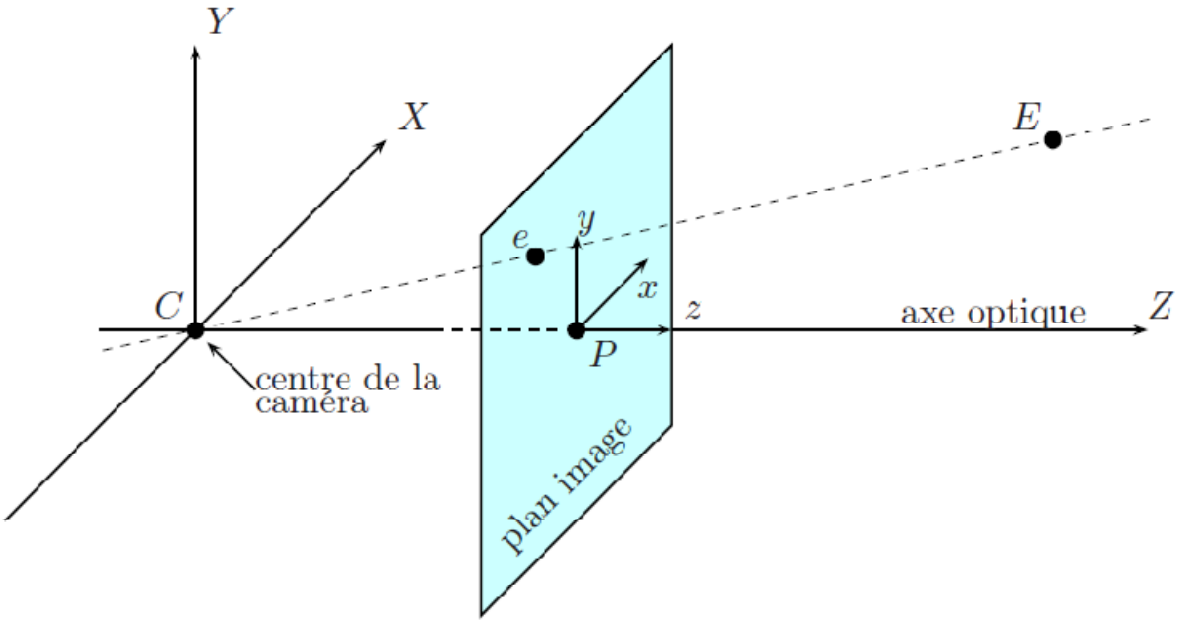


Figure 1.10 – Géométrie du modèle de caméra sténopé

Les systèmes de vision les plus utilisés en robotique mobile sont basés sur l'utilisation d'une caméra CCD (Charge Coupled Device). La rapidité d'acquisition, la robustesse et la miniaturisation sont les plus importants avantages qui ont facilité leur intégration dans un système de vision d'un robot mobile autonome **[11]**. Les systèmes de vision sont très performants en termes de portée, précision et quantité d'informations exploitables. Ils sont de plus les seuls capables de restituer une image sensorielle de l'environnement la plus proche de celle perçue par l'être humain. En revanche, l'inconvénient majeur de tels systèmes de

perception se situe au niveau du traitement d'une image qui est une opération délicate et surtout coûteuse en temps de calcul **[11]**.

1. 4. 3. Modélisation de l'environnement

Le processus permettant à un robot de mémoriser son environnement, puis de s'y déplacer vers un but, est constitué de trois phases : la cartographie, la localisation et la planification **[12]**. Ces trois phases sont fortement interdépendantes. Elles permettent de répondre aux trois questions fondamentales pour un robot mobile : Où suis-je ? Où sont les autres lieux par rapport à moi ? Et comment puis-je atteindre mon but ?[10]

• La *cartographie* est la phase qui permet la construction d'une carte reflétant la structure spatiale de l'environnement à partir des différentes informations recueillies par le robot.

• Une telle carte étant disponible, la *localisation* permet alors de déterminer la position du robot dans la carte qui correspond à sa position dans son environnement réel.

• La *planification* est la phase qui permet de prévoir les mouvements à effectuer afin de rejoindre un but fixé dans l'environnement, tout en connaissant la carte de l'environnement et la position actuelle du robot.

L'ordre dans lequel ces tâches sont citées montre bien que la localisation dépend de la cartographie. En effet, estimer sa position dans une carte de l'environnement suppose implicitement que cette carte existe et qu'elle contient la position courante du robot. De même, la planification dépend des deux premières, car elle suppose que l'on connaisse la position du robot et que la carte de l'environnement représente une portion de l'environnement contenant au moins un chemin reliant cette position au but à atteindre. Un robot qui ne possède aucune information sur l'environnement où il se déplace doit être capable de modéliser son environnement grâce à l'ensemble de ses capteurs proprioceptifs et extéroceptifs.

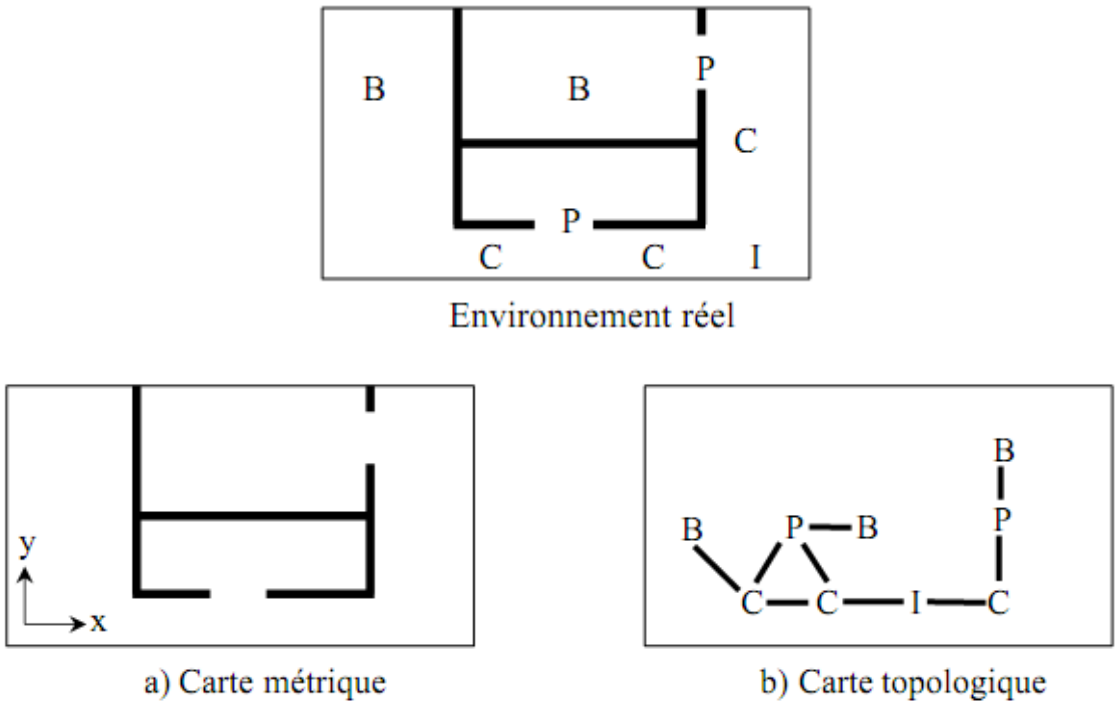
Il est possible que l'on fournisse au robot une carte construite au préalable et qu'on ne s'intéresse qu'à l'estimation de la position du robot mobile au sein de cette carte. La carte peut être obtenue en utilisant un plan d'architecte d'un environnement qui sera transformée en une carte utilisable par un robot mobile. Il est également possible que le robot réalise sa carte, en procédant par l'exploration de l'environnement.

La difficulté du problème de la modélisation de l'environnement (cartographie) vient généralement de plusieurs raisons parmi lesquelles **[13] :**

- ✓ La taille de la carte construite : plus l'environnement est grand, plus les traitements seront lourds et l'espace mémoire de stockage devient important.
- ✓ Le bruit : la présence du bruit, sur les mesures des capteurs, nécessite un traitement qu'il faut prendre en considération.
- ✓ Ambiguïté perceptuelle : plus les différents endroits ont les mêmes aspects, plus il est difficile d'établir des bonnes correspondances entre ces endroits.

Les méthodes de modélisation sont classées en deux grandes familles (Figure 1.11).

1) Les méthodes de modélisation métriques (cartes métriques) (Figure 1.11.a) : Elles décrivent explicitement la position «géométrique» des éléments de l'environnement **[14]**.
2) Les méthodes de modélisation topologiques (Figure 1.11.b) : Elles sont basées sur des graphes représentant des informations de certaines places caractéristiques de l'environnement (coins, croisement de deux couloirs, jonction en T, etc.). Donc, ces méthodes mémorisent un ensemble de lieux, ainsi que les manières de se déplacer de l'un à l'autre.



(Dans cet exemple, B=Bureau, C=Couloir, P=Porte et I=Intersection)

Figure 1.11: Les méthodes de modélisation de l'environnement.

1. 4. 3. 1. Les cartes métriques

Dans une carte métrique, l'environnement est représenté par un ensemble d'objets auxquels sont associées des positions dans un espace métrique, généralement en deux dimensions. Ces méthodes permettent de mémoriser un ensemble d'objets perçus (des murs, pilier, table, etc.). Ils sont subdivisés en deux sous familles **[11]**:

- ✓ Les méthodes de modélisation purement géométrique qui gèrent explicitement les positions cartésiennes des primitives cartographiques.
- ✓ Les méthodes probabilistes, appelées encore méthodes de modélisation par grille d'occupation, qui décrivent les propriétés métriques par discrétisation de l'environnement en y ajoutant des informations d'incertitude.

1.4.3.1.1. Modélisation géométrique

Les méthodes de modélisation géométrique visent à produire, sous forme de carte, une représentation géométrique de l'environnement à partir des données perceptuelles. Ces cartes sont souvent plus faciles à comprendre par l'homme car

elles offrent une relation bien définie avec le monde réel. Il s'agit là d'une représentation cartésienne de l'environnement qui est représentée par un ensemble d'objets auxquels sont associées des positions dans un espace métrique. Ces objets sont décrits à l'aide de primitives géométriques qui peuvent être de différentes natures (droites, points, arcs de cercle, ...) et représentent des informations différentes (limites de l'espace libre ou caractéristiques de l'environnement, ...). Les données perceptuelles permettent de détecter ces objets et d'estimer leur position par rapport au robot.
La position de ces objets dans l'environnement est alors calculée en utilisant la position estimée du robot.

La construction des cartes métriques nécessite la gestion de la cohérence géométrique de l'ensemble de la représentation **[15]**.

1.4.3.1.2. Grille d'occupation

Une autre façon de représenter l'environnement est de le découper sous forme de grille où chaque cellule représente une portion de l'espace. A chaque cellule est associée une valeur qui représente sa probabilité d'occupation. La grille d'occupation utilise une matrice pour stocker les différentes informations concernant les cellules de la carte. La figure 1.12 donne un exemple de grille d'occupation construite. Cette figure, représente les contours réels d'un environnement où les cellules occupées sont en bleu et les cellules vides sont en blanc.

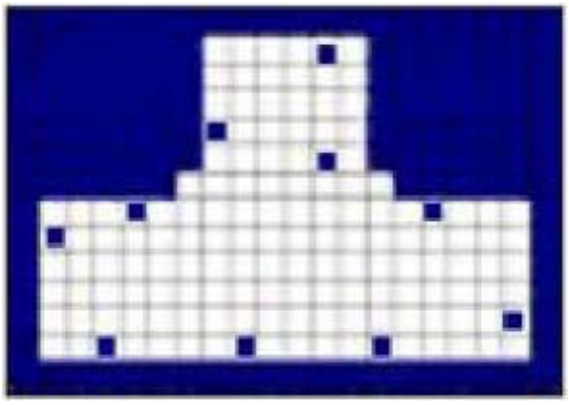


Figure 1.12 : Un exemple de modélisation par grille d'occupation

La taille de la carte est définie par la taille de la matrice. Chaque élément de la grille est une cellule qui fournie :

• une information sur la probabilité d'existence de l'obstacle (elle porte l'information d'être occupée ou non par un obstacle),
• la position d'un obstacle dans l'environnement.

Les premiers travaux à avoir intégré ce type de représentation sont ceux de Moravec et Elfes **[5]**. Dans ces travaux, des capteurs à ultrasons sont utilisés permettant de déterminer l'état d'une cellule (occupé, indéterminé, libre).

L'avantage de cette représentation est le fait que c'est une méthode facile à implémenter et extrêmement robuste **[5]**, mais possède l'inconvénient d'être relativement lourde d'un point de vue mémoire. La précision est évidemment limitée à la résolution de la grille. Précisons que cette résolution conditionne le temps de calcul : plus cette résolution est forte, plus le temps de calcul sera important [3].

1. 4. 3. 2. Les méthodes de modélisation topologiques (Cartes topologiques)

Une carte topologique est une représentation plus abstraite décrivant les relations entre les éléments de l'environnement, sans utiliser un repère de référence absolu. Elles se présentent sous forme de graphes, dont les sommets (noeuds) correspondent à des lieux, souvent associés à des informations perceptuelles (images, données télémétriques, etc.), et dont les arêtes indiquent l'existence d'un chemin traversable par le robot, reliant les lieux associés aux deux sommets de l'arête (Figure 1.13).

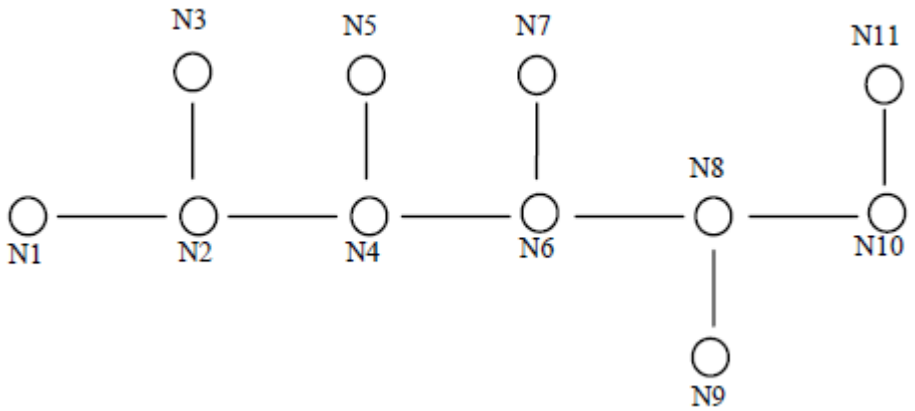


Figure 1.13 : Carte topologique

La détection et la mémorisation des lieux utilisent les perceptions et reposent en général sur deux procédures :

- La première permet de comparer deux perceptions et donc de reconnaître un lieu de la carte ou de détecter un lieu nouveau en faisant appel aux données proprioceptives.
- La seconde procédure permet de mémoriser un nouveau lieu ou d'adapter la définition d'un lieu lors des passages successifs du robot en ce lieu.

L'avantage principal des cartes topologiques est l'absence de l'incertitude dans le mouvement des robots : les incertitudes ne s'accumulent pas globalement car le robot se contente de naviguer localement, entre endroits **[15]**. Ces cartes peuvent présenter un découpage de l'espace qui facilite l'interaction avec l'homme, notamment si les lieux correspondent à des pièces ou à des couloirs : par exemple, on peut imaginer de donner l'ordre au robot mobile d'aller à la salle de séjour plutôt qu'aux coordonnées cartésiennes (x, y).

En revanche, le principal inconvénient des modèles topologiques est l'absence d'informations géométriques. Cette lacune peut empêcher le robot mobile de réaliser des raisonnements spatiaux sur l'ensemble de son environnement. En particulier, le système robotisé peut avoir des difficultés à sélectionner le chemin optimal entre deux lieux : d'une part, le manque d'information sur la longueur des chemins (sur les arcs) peut l'empêcher de choisir entre deux branches du graphe menant au même endroit, et d'autre part, il n'est pas possible de trouver un chemin plus direct dans l'espace métrique 2D que ceux qui sont implicitement codés dans les arêtes du graphe. Si les sommets sont impossibles à distinguer, la construction d'une carte purement topologique nécessite de mettre en oeuvre un processus très coûteux et peu efficace lors de la création d'un nouveau sommet **[15]**.

1.5. Localisation

Le problème de localisation d'un robot mobile consiste à répondre à la question « où suis-je? » posée en permanence par le robot mobile. La localisation permet de calculer et maintenir à jour la connaissance de la position (x, y) et de l'orientation (φ) du robot mobile.

Un grand effort a été noté dans le développement des méthodes de localisation. Plusieurs techniques et méthodes ont été développées pour assurer la connaissance exacte et de façon autonome de la position d'un robot mobile dans son environnement. Ces techniques peuvent être regroupées en trois catégories principales:

1. les méthodes de *localisation relatives* ou à l'estime, basées sur l'utilisation des capteurs proprioceptifs,
2. les méthodes de *localisation absolue*, basées sur l'utilisation des capteurs extéroceptifs,
3. les méthodes de *localisation hybrides* qui sont basées sur l'utilisation conjointe des capteurs proprioceptifs et extéroceptifs.

1. 5. 1. Localisation relative

La localisation relative consiste à déterminer la position et l'orientation du robot mobile par intégration des informations fournies par ses capteurs proprioceptifs depuis un point de départ. Ces données peuvent être des informations de déplacement (odomètre), de vitesse (vélocimètre) ou d'accélération (accéléromètre) **[14]**.

Les capteurs proprioceptifs mesurent en effet le déplacement ou le changement d'états du robot entre deux instants mais ne donnent aucune information sur sa localisation absolue. La précision de la localisation relative se dégrade avec la distance parcourue. Les méthodes de localisation relative couramment utilisées sont basées principalement sur l'odométrie et sur les mesures inertielles.

1. 5. 1. 1. Localisation par odométrie

La localisation par odométrie est l'un des systèmes les plus utilisés en robotique mobile. L'idée fondamentale de ce système est l'intégration de l'incrément de la position, calculé par des codeurs incrémentaux, par rapport au temps. Par exemple pour un robot mobile de type voiture **[16]**, les déplacements élémentaires Δdd et Δdg des roues droite et gauche permettent de calculer la variation de l'orientation ainsi que la variation de la position Δd entre l'état n et l'état n+1 (Figure 1.14).

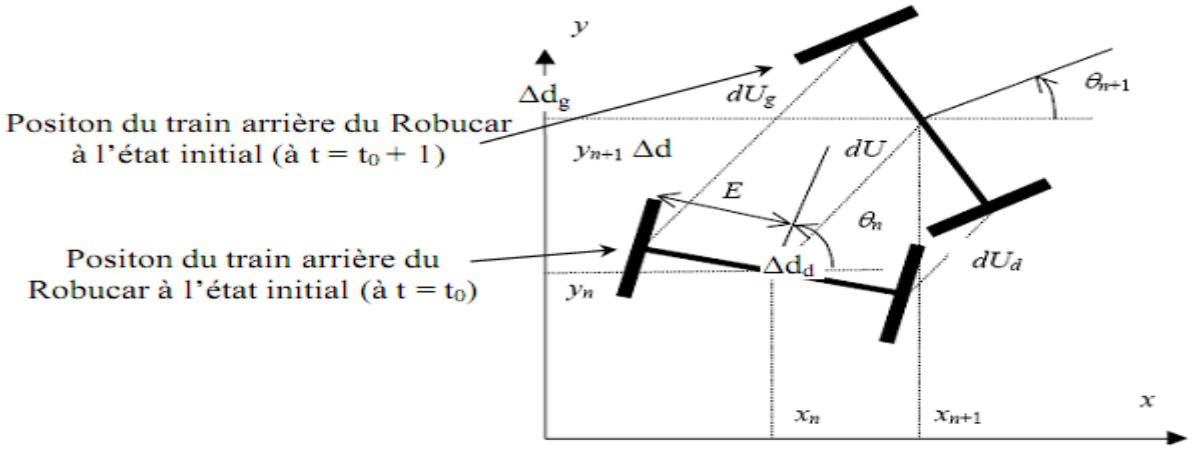


Figure 1.14 : Calcul de la position grâce à l'odométrie

Les avantages de cette technique résident dans sa simplicité de mise en oeuvre et dans son faible coût. Ces caractéristiques en font la technique de localisation la plus couramment utilisée en robotique mobile. En outre, l'utilisation d'odomètres permet d'obtenir une estimation de la position et l'orientation à une cadence relativement élevée avec une précision qui reste bonne sur de faibles distances.
Cependant, dès que les distances à parcourir augmentent, l'imprécision sur la position et l'orientation du robot mobile augmente considérablement, imprécision due à l'erreur cumulative générée lors de l'intégration des déplacements élémentaires **[14]**.

Exemple : Calcul de la position odométrique du **Robucar**

Le système odométrique du Robucar fournit deux paramètres, qui sont : la vitesse linéaire vR et l'angle de braquage des roues ϕ (Figure 1.15). Ces deux paramètres sont fournis par le Robucar à chaque période d'échantillonnage. Ils permettent de localiser le Robucar en déterminant sa position (x, y, φ) par rapport au repère univers (Ru) où (x, y) représentent les coordonnées cartésiennes du

Robucar dans (Ru) et φ est l'orientation de l'axe longitudinal du Robucar dans ce même repère.

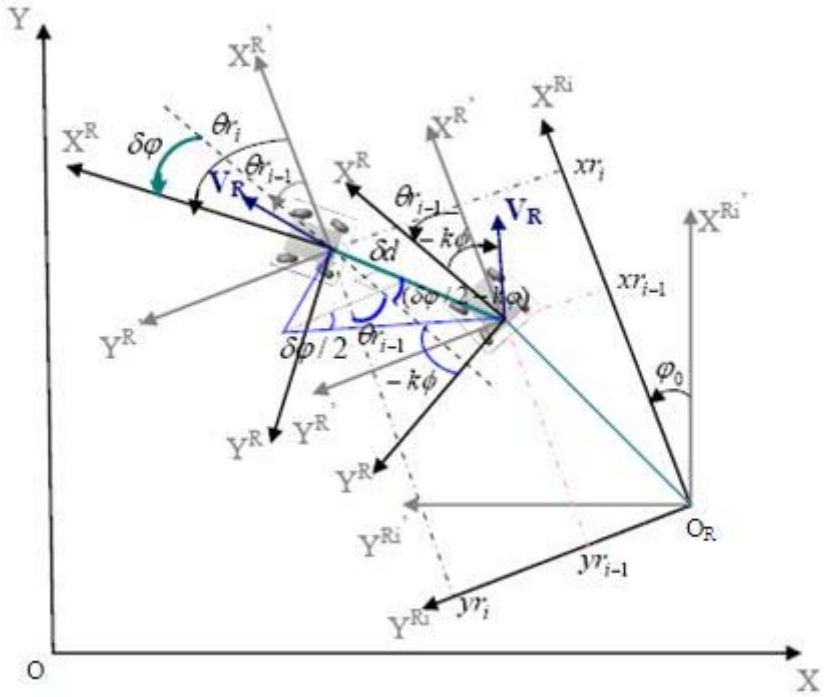


Figure 1.15 : Translations et rotations élémentaires du Robucar

Cette position est obtenue après l'intégration des translations et rotations élémentaires du Robucar (δd, δφ) entre deux instants d'échantillonnages (*i-1* et *i*). Les translations et rotations élémentaires s'expriment par les équations suivantes :

- Translation élémentaire du robot Robucar : **δd =ΔT** $\boldsymbol{V_R}$ (1.5)
- Déplacement angulaire du robot : **δφ = ΔT Ω = ΔT** $\frac{V_R \sin(\phi+k\phi)}{L\cos\phi}$ **(1.6)**

La représentation géométrique de la Figure 1.15 montre qu'à chaque période d'échantillonnage **ΔT ,** la position odométrique actuelle (x_r, y_r, θ_r) à l'état i peut être exprimée en fonction de $(\boldsymbol{V_R}, \phi)$ et de la position odométrique précédente à l'état *i-1.*

Cette position est obtenue grâce aux relations suivantes :

$$\begin{cases} x_{r_i} = x_{r_{i-1}} + \delta d \cos(\theta_{r_{i-1}} - K\phi + \frac{\delta\varphi}{2}) \\ y_{r_i} = y_{r_{i-1}} + \delta d \sin(\theta_{r_{i-1}} - K\phi + \frac{\delta\varphi}{2}) \\ \theta_{r_i} = \theta_{r_{i-1}} + \delta\varphi \end{cases} \quad (1.7)$$

Avec :
$R^u(O, Xu, Yu)$: Repère univers (lié à l'environnement)
$R^i(O_R, X_{Ri}, Y_{Ri})$: Repère mobile, lié à la position initiale
$R^R(O_R, X_{Ri}', Y_{Ri}')$: Repère mobile (lié au robot)
$R^{R'}(O_R, X_R', Y_R')$ et $R^{Ri'}(O_{Ri}, X_{Ri}', Y_{Ri}')$: Repère transitoire.

1. 5. 1. 2. Localisation par les capteurs inertiels

Cette technique utilise des accéléromètres et des gyromètres pour mesurer l'accélération et la vitesse angulaire du robot mobile **[17]**. L'intégration de cette mesure (ou la double intégration dans le cas de l'accéléromètre) permet de calculer la variation de la position. Les capteurs utilisés dans ce type de localisation présentent l'avantage d'avoir des mesures qui ne dépendent pas de l'environnement.
Cependant, l'inconvénient est que l'erreur même minime est amplifiée par l'intégration et entraîne une accumulation d'erreurs qui constitue une dérive d'estimation dans le temps.

1. 5. 2. Localisation absolue

La localisation absolue est une technique de localisation qui permet à un robot mobile de se repérer directement dans un repère lié à l'environnement, que ce soit en environnement d'extérieur (route, parking, etc.), ou en environnement d'intérieur (couloir, salle, etc.). Cette technique est basée sur l'utilisation de capteurs extéroceptifs et nécessite toujours une modélisation de l'environnement. Le robot possède donc une banque de données regroupant les éléments caractéristiques de son milieu d'évolution (balises par exemple), et pour sa localisation absolue, il doit déduire de la perception de ces éléments caractéristiques sa position dans son environnement **[5]**.

Pour répondre à la problématique qu'est la localisation d'un robot dans son environnement, deux types de stratégies sont utilisables :

- ✓ la première utilise des points de repère ; balises artificiels ou naturels,
- ✓ la deuxième utilise la mise en correspondance de modèles.

1.5.2.1. Localisation utilisant les balises

1.5.2.1.1. Les repères artificiels

Les repères artificiels sont des balises caractéristiques qui sont ajoutées au milieu d'évolution du robot et dont les positions sont connues. L'inconvénient de ce type de techniques réside essentiellement dans son manque de souplesse et dans sa lourdeur d'utilisation. En effet, un domaine d'évolution vaste nécessitera un investissement lourd en équipement. En outre, tout changement de configuration de l'environnement impliquera une remise en cause du réseau de balises. En revanche, cette technique a le gros avantage d'être précise, robuste et surtout de satisfaire la contrainte temps réel. Les balises artificielles peuvent être de deux types :

- ✓ les balises actives : elles émettent des signaux,
- ✓ les balises passives : elles ne peuvent pas émettre.

1.5.2.1.2. Les repères naturels

Cette technique utilise les éléments caractéristiques de l'environnement (des amers naturels) pour estimer la position du robot. L'intérêt de ces méthodes est donc sa souplesse d'utilisation puisqu'elles ne nécessitent pas d'aménager le milieu d'évolution du robot mobile. Pour la problématique de localisation, une connaissance de l'environnement est nécessaire. Il s'agira d'une représentation cartographique qui intégrera la position des amers qui serviront à localiser le robot. Suivant le niveau sémantique adopté pour décrire l'environnement, plusieurs types de représentations cartographiques pourront être utilisés (voir paragraphe 1.4.3).

1.6. Conclusion

Le paradigme de localisation et modélisation incrémentale de l'environnement nécessite de s'intéresser obligatoirement aux éléments de la chaîne de perception que sont les capteurs, les méthodes de localisation et celles de modélisation. C'est ce que nous avons fait de façon la plus synthétique possible dans cet état de l'art. Les constats pouvant être dégagés quant à ces trois modules sont multiples.

Par rapport à notre plateforme disponible (Robucar) on va utiliser le capteur LMS 200 pour la modélisation de l'environnement en utilisant le modèle géométrique qui nous donne une carte plus détaillée et plus proche à l'environnement réel.

Dans le chapitre suivant on va présenter les différentes méthodes pour résoudre le problème de SLAM (« Simultaneous Localization And Mapping »).

CHAPITRE 2

LOCALISATION ET CARTOGRAHIE SIMULTANEES

2.1. Le problème de SLAM

Si la fonction de localisation bénéficie grandement des capacités de cartographie, la modélisation de l'environnement nécessite à l'inverse une estimation de la position du robot. En effet, pour savoir où se positionner dans la carte il faut être capable de situer le système robotisé dans la carte en cours de construction. Or en général, le champ de perception du robot n'embrasse pas l'ensemble de l'environnement, du fait des occultations et des limites de portée des capteurs, le robot doit donc se déplacer pour réaliser une cartographie plus exhaustive et sa position doit être régulièrement recalculée. Il faut donc gérer le bouclage entre localisation et modélisation de l'environnement. Ce problème de l'œuf et de la poule est bien connu des roboticiens, qui le désignent sous le nom de « localisation et cartographie simultanées » ou SLAM en anglais (« Simultaneous Localization And Mapping »), voire plus rarement CML (« Concurrent Mapping and Localization ») [4]. Ce chapitre présente les différentes techniques de résolution du problème SLAM.

2.2. Taxonomie du problème de SLAM

On peut classer les différentes approches dans trois grands groupes, dont le plus connu est celui des méthodes probabilistes. On trouve aussi les méthodes Visuelles ainsi que les méthodes de mise en correspondance (scan matching) (Figure 2.1).

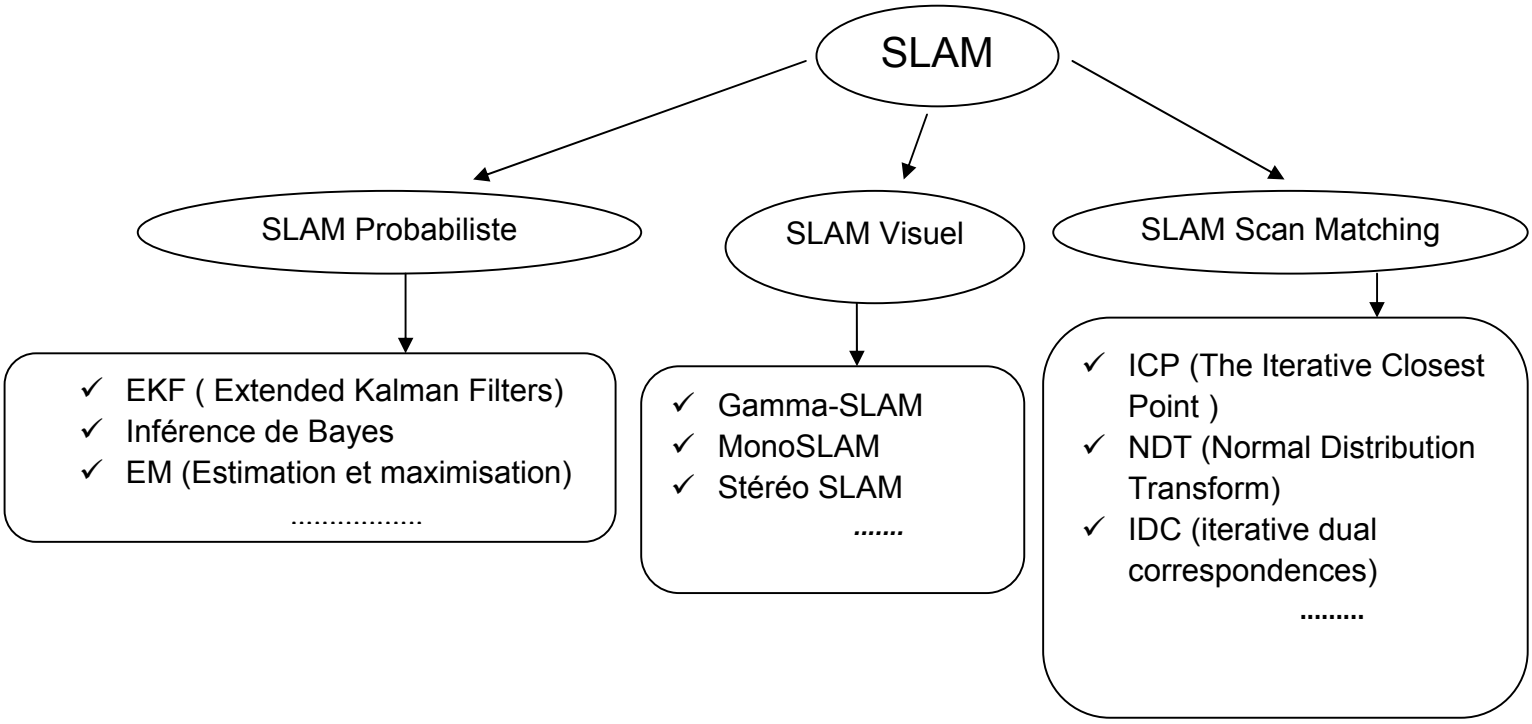


Figure 2.1. Classification des approches SLAM

2.3. L'état de l'art

Le problème SLAM est considéré comme un problème important de la robotique.

La problématique du SLAM implique en outre la gestion de multiples sous-problèmes.

Il faut d'abord gérer les erreurs de perception et de localisation (dérive des capteurs proprioceptifs, imprécision des mesures extéroceptives, présence de bruit, erreurs d'interprétation, etc.). A partir de ces mesures sensorielles imparfaites, il s'agit ensuite de sélectionner la meilleure carte dans un espace de recherche de très grande dimension. Il faut aussi pouvoir mettre en correspondance les observations locales entachées d'erreur avec les éléments de la carte globale en cours de construction. Dans certains cas, il faut également ajouter le traitement des environnements dynamiques.

2.3.1. Des approches pour le problème SLAM

Pour résoudre le problème SLAM, plusieurs méthodes sont présentées. La méthode la plus connue est la méthode probabiliste.

2.3.1.1. Les méthodes probabilistes

2.3.1.1.1. EKF - Filtre de Kalman Etendu

L'utilisation du filtre de Kalman pour la cartographie a été proposée par Cheeseman et al.[18]. Ils proposent de cartographier l'environnement en utilisant la localisation moyenne des points de repère (Appelés amers) ainsi que la covariance les reliant. La carte est construite de manière incrémentale en insérant au fur et à mesure les amers découverts tout au long du trajet du robot. Ce travail a été étendu par Moutarlier et Chatila [19] en incluant la notion de propagation de l'incertitude de localisation du robot dans le temps. Ils utilisent un modèle dynamique qui prédit la localisation du mobile et un modèle d'observation qui utilise les mesures des capteurs extéroceptifs.

Cependant, les améliorations apportées sur la qualité des résultats de l'estimation sont associées à la dégradation du temps d'exécution : plus la carte est grande, plus le temps d'estimation et important.

2.3.1.1.2. Inférence de Bayes

Une autre approche consiste à considérer le problème SLAM comme un modèle de Markov et utiliser l'inférence de Bayes pour résoudre le problème [20]. Cependant en pratique, il y a des cartes qui contiennent un nombre important de points de repères ce qui rend cette méthode impraticable.

2.3.1.1.3. EM (Estimation et maximisation)

Cet algorithme est appliqué au problème SLAM lorsque les modèles d'´etat et d'observations sont non linéaires et non gaussiens [21], [22]. Pour ce faire, la carte est modélisée comme un paramètre du modèle de Chaines de Markov cachées « Hidden Markov Models » (HMM). L'originalité de cette contribution à la résolution du SLAM réside à la fois dans la formulation du problème de cartographie comme un problème d'estimation de paramètres dans un HMM; et dans l'approche algorithmique retenue, basée sur l'algorithme EM.

2.3.1.1.4. relaxation et «relaxation multi niveau»

Une autre méthode de SLAM relative à l'estimation de maximum de vraisemblance est la relaxation [23], [24]. On utilise « Relaxation de Gauss-Seidel» pour résoudre un système d'équations du problème SLAM. L'idée de cette méthode est de « déplacer un nœud à la position où ses voisinages pensent qu'il est».

Cette méthode est très appliquée dans des algorithmes récents. Le «relaxation *multi niveau*» [25] est basé sur une méthode multi-grille (multigrid method) pour résoudre un système d'équations différentielles partielles. Cet algorithme augmente la convergence de Relaxation par l'optimisation de la carte de chaque couche.

2.3.1.1.5. Filtre particulaire

Cette méthode [26], [27] est basée sur l'utilisation de filtres particulaires. Le principe est de suivre un grand nombre d'hypothèses en parallèle qui sont autant de trajectoires possibles. Ces différentes hypothèses correspondent à un échantillonnage de la distribution de probabilité des trajectoires. Pour chacune d'elles, on construit la carte en fonction des perceptions du robot à l'aide d'un filtre de Kalman.

Cependant dans ce cas le traitement est simplifié puisque la trajectoire est connue: les perceptions successives des différents amers ne sont plus corrélées et la matrice de covariance se simplifie puisqu'on ne mémorise plus que les variances individuelles des amers.

Le problème principal de cette technique est que la représentation de la carte et de la trajectoire du robot devient vite très lourde. En effet, il faut que le nombre de particules soit suffisant pour échantillonner correctement la distribution de probabilité des trajectoires.

2.3.1.2. SLAM visuel

Le SLAM visuel consiste à estimer simultanément le mouvement d'une caméra et l'environnement dans lequel elle se déplace. Dans la communauté de vision par ordinateur, ce problème est également connu sous le nom de *Structure and Motion*. Cette tâche est traditionnellement divisée en trois étapes principales. D'abord, des primitives géométriques sont extraites de l'image, par exemple grâce à un détecteur d'Harris ou des points SIFT (Scale Invariant Feature Transform), puis suivies (ou appariées) entre les images successives de la séquence. Une fois ce problème d'association de données réalisé, seules les coordonnées des points retenus sont utilisées dans les traitements ultérieurs.

2.3.1.2.1 Odométrie visuelle

L'odométrie visuelle [28] repose bien souvent sur des techniques de suivi dans une séquence d'images. Pour que ce suivi soit possible, il faut qu'il soit possible de détecter des caractéristiques particulières dans les images, qui sont alors plus ou moins texturées. On entend par là que l'image contient des motifs ou du moins, l'image n'est pas totalement uniforme, bref que sur certaines zones, au moins, il y ait des variations de contraste dans le cas des images en niveau de gris. Tout ceci dans le but de pouvoir détecter des primitives visuelles pour les suivre d'une image à l'autre.

2.3.1.2.2. Gamma-SLAM

On utilise l'odométrie visuelle et un filtre de particules Raoblackwellized (Rao-Blackwellized Particle Filter - RBPF) [26]. On utilise une distribution gamma pour représenter les hauteurs des objets de l'environnement. Cet algorithme s'appelle

Gamma-SLAM [29]. On maintient une distribution postérieure sur la variance d'hauteur dans chaque cellule.

On peut représenter la carte par une grille. Chaque cellule représente une pose de robot. Dans cet algorithme, on peut considérer une carte comme une grille de la probabilité conditionnelle qui confirme la correction de la carte.

Le résultat obtenu on utilisant la méthode gamma-SLAM est meilleur comparé à l'algorithme qui utilise seulement l'Odométrie Visuelle.

2.3.1.2.3. Mini SLAM

Le mini SLAM est une approche pour le problème de SLAM dans un environnement large avec des exigences de perception et calcul minimal [30]. Cette approche est basée sur deux principes. Premièrement l'odométrie est assez précise si la distance parcourue est courte. Deuxièmement, à l'aide de correspondance visuelle, la correspondance entre les poses du robot peuvent être détectées de manière fiable même si la covariance de l'estimation de position actuelle (la zone de recherche) est grande.

Nous avons donc deux types de relations, Relation basée sur l'odométrie et la relation basée sur les similarités visuelles.

2.3.1.2.4. MonoSLAM

Dans la méthode MonoSLAM [31] une caméra constitue le seul et unique capteur du système. C'est un cas particulier du *SLAM visuel*. Par rapport à des techniques plus conventionnelles, il n'y a ici pas d'odométrie ni de données inertielles, mais seulement des données 2D. Et pourtant, à partir de cela uniquement, il faut remonter à des informations 3D comme la position et l'orientation de la caméra, ainsi que la position des amers (la carte est constituée de l'ensemble de ces amers).

La profondeur (distance entre la caméra et l'amer) doit être estimée grâce au mouvement de la caméra. Cependant, utiliser le mouvement propre ("ego motion")

pour caractériser complètement les amers rend le MonoSLAM plus difficile que les méthodes traditionnelles de SLAM. En effet, l'estimation de la position d'un nouvel amer est étroitement liée à l'estimation du mouvement de la caméra : il s'agit d'un exemple de ce qu'on appelle *"structure from motion"*.

2.3.1.2.5. Stéréo SLAM

Le principal avantage de l'utilisation de la stéréo vision [32] par rapport à un télémètre laser par exemple est la possibilité de détecter des obstacles à des hauteurs différentes (le télémètre laser retourne des mesures de distance sur un plan 2D à une hauteur fixe). Avec la vision stéréoscopique, nous pouvons marquer l'obstacle le plus proche quelle que soit la hauteur. Par exemple, quand un télémètre laser voit quatre pieds de table, il ne marque que les pieds de la table comme espace occupé sur la carte. Cela pourrait induire en erreur le robot qui pourrait essayer de passer sous la table alors qu'il doit l'éviter. L'approche stéréo vision serait capable de détecter la table entière et éviter les collisions.

2.3.1.3. Mise en correspondance des scans (SLAM Scan Matching)

Dans les applications de modélisation de l'environnement, on dispose généralement de plusieurs données 2D de la même scène prises de points de vue différents. L'alignement de ces vues peut être défini comme étant le processus d'estimation des transformations rigides (rotations et translation) qui permettent de ramener ces différentes vues dans un référentiel commun. La mise en correspondance des scans de deux vues est souvent considérée comme un problème d'optimisation, dont la fonction de coût est basée sur la mesure d'une distance entre les parties communes de ces deux vues. Les différentes méthodes proposées pour résoudre ce problème diffèrent essentiellement selon la métrique utilisée pour formuler cette distance et selon la technique de minimisation mise en œuvre.

La mise en correspondance des scans vise à fournir un ensemble de données exploitables pour la reconstruction du modèle numérique d'un objet ou d'un environnement de telle sorte que l'on puisse, d'une part le visualiser sur un écran, et

d'autre part traiter ces données (simplification, échantillonnage sélectif,…) pour obtenir une représentation adaptée à l'application envisagée.

Mais ce n'est pas le seul domaine d'application de la mise en correspondance des Scans. Elle peut également servir pour la reconnaissance de formes 2D ou pour résoudre des problèmes de localisation en robotique mobile en maximisant la superposition de deux configurations consécutives.

Il est possible de classer les techniques de mise en correspondance des scans selon leur méthode d'association, telle que :

- **figure à figure**: elles traitent les données brutes afin d'acquérir un ensemble de caractéristiques géométriques claires de l'environnement qui sera apparié dans une deuxième étape.
- **Point à figure** : les points d'un ensemble de données sont associés aux caractéristiques géométriques de la scène. L'approche la plus populaire dans ce type de méthodes a été développée par cox [33].
- **Point à point** : traitant directement des données brutes. Diosi et al. [34] ont proposé une méthode basée sur les coordonnées polaires de données laser acquises. cette méthode est nommée mise en correspondance des Scans polaire (PSM).

Différents algorithmes existent à cet effet :

2.3.1.3.1. Le plus proche point Itératif (ICP)

L'ICP est basé sur la recherche du point le plus proche, il calcule de façon itérative la matrice de transformation permettant d'effectuer l'alignement. Le principe de l'algorithme est d'itérer les deux étapes d'alignement: la mise en correspondance des données et l'estimation de la transformation de repères entre les données à recaler. Au bout de chaque itération l'algorithme fournit une liste de points associés et une estimation de la transformation de repère entre les données. Cette transformation est utilisée à l'itération suivante pour la mise à jour de la liste des points associés. Ces derniers serviront, à leur tour, pour calculer une nouvelle

estimation de la transformation. Ces étapes sont répétées jusqu'à la convergence de l'algorithme. Cependant, il existe plusieurs variantes de l'algorithme ICP, en effet de nombreuses améliorations y ont été apportées par la suite. Rusinkiewitcz et Levoy [42] proposent une étude comparative de plusieurs variantes de ICP, en les appliquant à trois exemples de données de synthèse qui sont bien représentatives des différents types de formes 2D rencontrées. A partir du bilan de cette étude, ils proposent une version optimisée de l'ICP dans le but d'améliorer la vitesse de convergence. L'étude concerne les améliorations apportées par chaque auteur à chaque étape de l'algorithme original. Ces étapes sont:

1. la sélection des points à aligner (échantillonnage des données, utilisation de points de contrôle)
2. la technique d'association
3. la pondération des paires de points obtenues pour estimer la transformation de repère
4. le rejet des mauvaises associations (points aberrants)

De très nombreux travaux ont été menés pour l'amélioration de l'algorithme de base au niveau de chaque point cité ci dessus. Un des objectifs les plus importants à atteindre est l'amélioration de la robustesse de l'algorithme vis à vis des mauvaises associations (erreurs de mesure générées par une imperfection du capteur utilisé) en utilisant des méthodes d'alignement robustes aux points aberrants. D'autres travaux ont été effectués dans le but d'améliorer la vitesse de convergence de l'algorithme. En effet, dans le cas de l'approche globale de la mise en correspondance des scans, l'alignement concerne souvent des données qui occupent un espace mémoire important et qui ralentissent considérablement l'algorithme ICP classique pendant la phase de recherche du plus proche voisin.

2.3.1.3.2 Double correspondances Itératif (IDC)

Le processus itératif double correspondances (IDC) est une extension de l'ICP qui vise essentiellement à accélérer la convergence de la partie rotative de la position estimée [35]. Avec l'ICP classique, la composante de translation converge généralement assez rapidement, tandis que la rotation prend plus d'itérations.

Cependant, l'IDC utilise deux règles pour trouver des correspondances. A chaque itération, l'ensemble rotation - translation $\overrightarrow{\tau_1}$=(R_1,$\overrightarrow{t_1}$) est déterminé en utilisant les points les plus proches, comme pour l'ICP normal. Sans appliquer la transformation, un nouvel ensemble de points correspondants sont sélectionnés en utilisant un autre critère: le critère de correspondance-distance- point. Celui-ci utilise les coordonnées polaires des points, et cherche à correspondre les points dans un intervalle angulaire, qui est formulé en deux dimensions avec : [θ-b,θ+b], θ est l'angle du point $\vec{P}$ et b est la bande qui limite la zone de recherche du point le plus proche. Une deuxième transformation $\overrightarrow{\tau_2}$=(R_2,$\overrightarrow{t_2}$) est calculée en utilisant les correspondances trouvées avec cette méthode et la transformation qui a été appliquée avant la prochaine itération $\overrightarrow{\tau_3}$=(R_3,$\overrightarrow{t_3}$).

2.3.1.3.3 Algorithme d'enregistrement probabiliste à base de points

Cet algorithme est présenté par Hahnal et Burgard [36], il modélise le scan comme des fonctions de probabilité, dont chacune est une combinaison entre la distribution normale et la distribution uniforme.

Pour calculer la vraisemblance entre les points du scan source, un rayon est tracé à partir de la position courante estimée du capteur de position jusqu'à la surface la plus proche dans le scan ciblé. La longueur de ce rayon est prise comme la distance estimée de cette mesure. La vraisemblance de la mesure courante est calculée à partir d'une combinaison d'une distribution normale qui est centrée sur la distance estimée, avec une variance réglée sur les caractéristiques du scanner laser et une distribution uniforme réglée également à la précision du scanner.

2.3.1.3.4 Algorithme d'enregistrement (branchement et séparation)

Certains chercheurs [37], [38] ont utilisé une stratégie « branch-and-bound » pour la mise en correspondance des scans, la partie translation de l'espace de position est discrétisée en plusieurs résolutions, avec des niveaux ordonnés à partir de la plus grossière à la plus fine. Un certain nombre de positions est considéré comme un certain niveau d'une pyramide. Les meilleures correspondances sont considérées au

niveau inférieur suivant de la pyramide (branching), et les autres, ainsi que tous leurs sous-nœuds dans la pyramide, sont mis au début (bounding).

2.3.1.3.5 Algorithme d'enregistrement de champ gaussien

Boughorbel et al. [39] ont développé un critère d'enregistrement basé sur des champs gaussiens, semblable à la distribution de la NDT. L'idée fondamentale de cette approche est d'utiliser un champ gaussien pour mesurer la distance spatiale et la similitude visuelle de deux points de deux scans.

2.3.1.3.6 La transformé de distribution normale (NDT)

La transformé de distribution normale (NDT) algorithme [3] utilise une autre représentation de l'analyse. Au lieu d'utiliser des points individuels des données du capteur recueillies, il convertit les données en une combinaison des distributions normales, décrivant la probabilité de trouver un obstacle en tout point de la zone balayée. Les distributions normales donnent une représentation continue des données discrètes avec une dérivée continue de première et second ordre.
En utilisant cette représentation, il est possible d'appliquer la méthode standard d'optimisation numérique pour l'enregistrement.

2.3.2 Comparaison entre les différentes approches

Les approches basées sur les techniques probabilistes (les techniques d'estimation récursive), (le filtre de Kalman, Kalman étendu, Filtre particulaire, …) on été largement utilisées. Aujourd'hui, le filtre de Kalman étendu reste l'implémentation de référence à laquelle les nouveaux algorithmes sont souvent comparés. Cependant, il présente des inconvénients :

- La complexité en temps et en espace est liée au nombre d'amers de la carte ce qui rend l'algorithme inutilisable dans des applications où le robot doit naviguer dans un environnement comportant de nombreux amers.
- L'algorithme n'est pas robuste face aux mauvaises associations, une seule association d'amer erronée peut faire diverger le filtre.

Par contre l'approche de la mise en correspondance des scans a été proposée et a suscité beaucoup d'attention dans les applications de SLAM. Notre travail se

focalise sur cette approche et surtout sur les deux plus populaires algorithmes ICP et NDT.

Une comparaison est faite entre ces deux algorithmes en donnant les avantages et les inconvénients de chaque approche.

2.3.1. Comparaison entre l'algorithme ICP et NDT

- le temps d'exécution de l'algorithme NDT est très faible en comparaison à l'algorithme ICP qui passe la plupart de son temps à chercher le point voisin le plus proche (jusqu' à 1.5 s pour l'ICP contre 0.05 s pour la NDT).
- pour les cas où la position du robot est précise, l'ICP donne de meilleurs résultats que la NDT. Mais ce n'est pas toujours le cas en raison d'erreurs du capteur.
- Pour les cas a l'intérieur où le robot est confronté à des caractéristiques similaires telles que les couloirs, l'algorithme de la NDT est meilleur que l'algorithme ICP qui manque beaucoup de détails comme les coins, car ils seront mélangés avec des murs.
- les méthodes basées sur les points, comme l'ICP ne tiennent pas compte de la forme locale de la surface autour de chaque point.

2.4. SLAM Dynamique

Dans le cas des environnements dynamiques, le problème SLAM est certainement plus compliqué car l'apparition d'objets mobiles dans la zone de perception du robot, pendant le processus de construction de la carte, provoque des erreurs au niveau de la localisation ainsi que des inexactitudes au niveau de la carte. Un algorithme qui différencie entre la partie dynamique et statique de l'environnement peut considérablement contribuer à améliorer le SLAM dans des environnements réels.

Toutes les techniques utilisées pour résoudre le problème SLAM, que ce soient celles basées sur le principe d'alignement des mesures ou celles utilisant le filtrage de Kalman, peuvent échouer en présence des entités dynamiques.

2.4.1. Effet des objets dynamiques sur le SLAM

Afin d'élaborer une stratégie efficace pour détecter les objets dynamiques et par la suite les filtrer du modèle de l'environnement, la compréhension et l'interprétation de leurs effets sur le modèle de l'environnement restent essentiels.
La Figure 2.2 montre l'effet des objets dynamiques sur le modèle de l'environnement. On remarque l'apparition d'obstacles inhabituels dans les espaces traversés par ces objets dynamiques. L'objectif du SLAM dynamique est d'avoir une carte propre de l'environnement représentant uniquement ses éléments statiques. D'où, la nécessité de détecter et d'effacer les objets présumés mobiles. Comme le montre la figure 2.2, la carte résultante ne devrait contenir que les éléments représentés en noir. Par contre, ceux en rouge doivent être détectés et désignés comme étant des objets dynamiques.

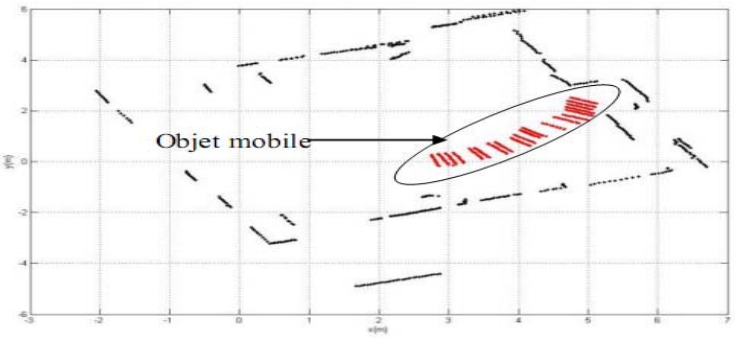


Figure 2.2: Effet d'un objet dynamique sur le modèle de l'environnement

2.4.2. Résolution du problème

Afin de construire un modèle correct de l'environnement, les traces des objets dynamiques doivent être détectées et éliminées de la carte et le robot doit être capable de distinguer entre deux catégories d'objets :

1. Objets statiques :

Ils représentent les parties statiques de l'environnement. Ces objets doivent être représentés dans la carte.

2. Objets dynamiques :

Ils représentent les parties mobiles de l'environnement telles que des personnes, des robots, ou bien des objets à état variable (portes, fenêtre, etc.).

La modélisation des environnements dynamiques a été étudiée ces dernières années. Le travail le plus célèbre est celui de Wang, C.-C [40]. Il a étudié le problème des environnements urbains dynamiques parcourus par un véhicule équipé d'un capteur laser.

Pour résoudre le problème, il a proposé l'utilisation de deux cartes séparées, une carte pour les objets statiques et une autre pour les objets dynamiques. La détection des objets dynamiques est faite sur la base d'une technique de classification simple suivie par une méthode de mise en correspondance. Le groupe de points qui change entre deux mesures appartient à un objet dynamique. Les résultats de ce travail montrent deux étapes fondamentales pour résoudre le SLAM dynamique: premièrement, la nécessité d'utiliser deux cartes pour modéliser l'environnement dynamique. Deuxièmement, la localisation du robot est basée sur la partie statique pour détecter la partie dynamique.

L'idée d'utiliser deux cartes est ensuite relatée pratiquement dans tous les travaux qui traitent ce sujet. A. Baba [41] a utilisé la grille d'occupation pour modéliser l'environnement. Concernant la détection, il a utilisé la notion de changement drastique dans l'état des cellules entre deux mesures successives. Ces cellules sont groupées et mémorisées dans une deuxième carte afin de modéliser la partie dynamique de l'environnement.

Bien que ces travaux aient contribué à la résolution et la compréhension du problème, des questions demeurent toujours. Celles-ci incluent, principalement, comment différencier les parties statiques et dynamiques de l'environnement « détection » et comment représenter une telle information dans la carte. Avant de répondre à ces questions, il est intéressant de clarifier la notion des objets mobiles dans un environnement dynamique. Il y a deux types différents d'objets mobiles à considérer. Les objets qui sont en mouvement permanent comme les piétons et ceux mobiles de temps en temps. De même que pour les objets statiques, il y a les objets statiques en permanence comme les murs et les objets statiques qui peuvent bouger comme les portes.

En plus des sous problèmes du SLAM statique, dans le cas du SLAM dynamique s'ajoute le problème de détection des objets mobiles qui est la partie la plus délicate dans le SLAM dynamique.

2.5. Conclusion

Dans ce chapitre, nous avons passé en revue les approches proposées dans la littérature afin de résoudre le problème SLAM comme les méthodes probabilistes qui sont les méthodes les plus anciennes et les plus connues en robotique avec l'inconvénient majeur de leur temps d'exécution qui augmente avec l'augmentation de nombre d'amers au contraire aux méthodes de mise en correspondance des scans qui ne nécessite aucune connaissance préalable de l'environnement, et on trouve aussi les méthodes visuels avec l'inconvénient classique du bruit d'image dû aux conditions d'éclairage.

Nous avons opté pour la technique de mise en correspondance des scans basée sur la NDT qui nous arrange surtout en termes de temps d'exécution si on veut que notre implémentation s'exécute en temps réel (prenant en considération le temps de prétraitement au niveau de capteur LMS et les encodeurs).

Le prochain chapitre explique en détail l'algorithme SLAM-NDT.

CHAPITRE 3

LA TRANSFORMEE DISTRIBUTION NORMALE

3.1 Principe

La Transformée Distribution Normale « The Normal Distribution Transform » (NDT) est une méthode de mise en correspondance des scans introduite par Biber and Strasser en 2003 [**3**]. Elle est essentiellement destinée à la modélisation 2D et 3D. Dans notre cas d'étude, la méthode NDT est utilisée à la fois pour construire la carte et estimer la position du robot (Cas 2D). L'élément clé de cet algorithme est la nouvelle représentation des points des scans. Au lieu de les représenter comme des nuages, la NDT transforme le scan brut en une matrice de probabilités, c'est la probabilité de trouver un obstacle à une certaine position et il est modélisé par une combinaison linéaire des distributions normales. Cela donne une représentation lisse des données numérisées, avec dérivées continues du premier et du second ordre.

En utilisant cette représentation, il est possible d'appliquer les méthodes classiques d'optimisation numérique telles que la méthode de Newton, pour l'enregistrement. L'optimisation numérique est un problème qui a été étudié pendant plusieurs années. Des méthodes rapides et fiables pour l'optimisation des fonctions ont été développées et testées au cours du temps.

la première étape de l'algorithme est de subdiviser l'espace 2D autour du robot en cellules de taille régulière, puis, pour chaque cellule, qui contient plus d'un certain nombre minimal de points (seuil), la moyenne q des points à l'intérieur de la cellule et sa matrice de covariance C sont calculées comme suit:

$$q = \frac{1}{n}\sum_{k=1}^{n} X_k \qquad (3.1)$$

$$C = \frac{1}{n}\sum_{k=1}^{n} (X_k - q)\,(X_k - q)^T \qquad (3.2)$$

Avec X_k= 1 ... n sont les points contenus dans la cellule.

La probabilité de mesurer un échantillon au point 2D est modulée par la distribution normale N (q, C).

La densité de probabilité « Probability *Density* Function »(PDF) est formulée par :

$$P(X) = exp\left(-\frac{(X-q)^T C^{-1}(X-q)}{2}\right) \quad (3.3)$$

Où q et C sont la moyenne et la covariance respectives pour la cellule qui contient le point X. Un scan laser 2D et la distribution normale correspondante sont montrés sur la Figure 3.1.

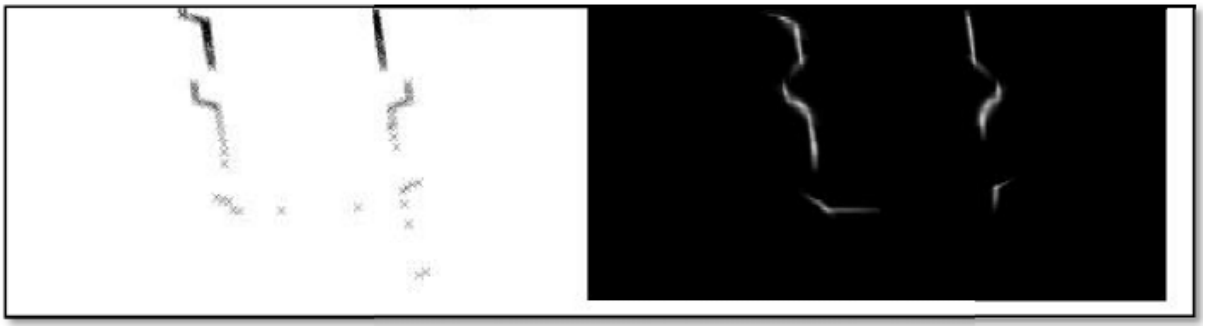


Figure 3.1 : Un scan laser 2D d'une partie d'un couloir (représenté en tant que points sur la gauche) et la distribution normale (à droite). Chaque cellule est un carré de 0,5 m de côté, la zone la plus blanche représente une plus forte probabilité

De la même manière que la grille d'occupation, la NDT établit une subdivision régulière du plan, mais la grille d'occupation représente la probabilité d'occupation de la cellule entière, tandis que la NDT représente la probabilité d'occupation de chaque position dans la cellule.

C'est donc une description continue et dérivable du plan 2D sous la forme d'une densité de probabilité. Cette transformation est en fait utilisée, dans ce travail, pour aligner deux scans laser.

3.2 Alignement de Scans

Les paramètres à optimiser, à savoir la rotation et la translation de l'estimation de position courante, peuvent être enregistrées dans un vecteur $\vec{P}$. Pour l'enregistrement 2D, il y a trois paramètres de transformation à optimiser. Posons $\vec{P} = (t_x, t_y, \theta)$ le vecteur des paramètres.

Avec :

$t_x \; et \; t_y$ sont des paramètres de translation et θ l'angle de rotation.

$X_{i=}(x_i, y_i, \theta_i)$: les points mesurés par le capteur laser du scan i.

$X'_{i=}(x'_i, y'_i, \theta_i)$: les points mesurés par le capteur laser du scan i écrit dan le repère du scan i -1.

La fonction de transformation 2D entre deux positions du robot est donnée par:

$$T(\vec{P}, X): \quad X' = \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix} \qquad (3.4)$$

L'objectif de l'alignement de scans est de récupérer ces paramètres en utilisant le scan laser pris dans deux positions différentes.

L'algorithme de la technique, étant donné deux scans successifs, se présente comme suit:

1. Construire La NDT du premier scan.
2. Initialiser l'estimation des paramètres en utilisant les données de l'odométrie.
3. Pour chaque point du deuxième scan: Ecrire les coordonnées de ce point dans le repère du premier scan selon le paramètre $\overrightarrow{P}$.
4. Déterminer les distributions normales correspondantes pour chaque point.
5. le score pour les paramètres est déterminé en évaluant la distribution pour chaque point et en sommant le résultat.

6. Calculer une nouvelle estimation des paramètres en essayant d'optimiser le score, en exécutant une itération de l'algorithme de Newton.
7. Aller à 3 jusqu'à ce qu'un critère de convergence soit atteint (le score ne change plus).

Le reste est maintenant décrit en détail en utilisant les notations suivantes:

- $\vec{P} = (t_x, t_y, \theta)^T$: Le vecteur des paramètres à estimer.
- $\mathrm{X_i}$: Le point 2D reconstruit du scan laser i.
- $\mathrm{X'_i}$: Le point $\mathrm{X_i}$ écrit dans le repère de coordonnées du premier scan selon le paramètre $\overrightarrow{P}$, $\mathrm{X'_i} = \mathrm{T}(\vec{P}, \mathrm{X_i})$.
- C_i, q_i: La matrice de covariance et la moyenne de la distribution normale correspondante du point $\mathrm{X'_i}$.

La cartographie, en utilisant le paramètre $\overrightarrow{P}$, est considérée comme optimale si la somme évaluée de la distribution normale de tous les points $\mathrm{X'_i}$, avec les paramètres $C_i \; et \; q_i$, est maximale. Nous appelons cette somme score de $\vec{P}$ et est définie par :

$$Score(P) = \sum i \exp\left(-\frac{(X_i' - q)^T C^{-1} (X_i' - q)}{2}\right) \qquad (3.5)$$

Ce score est optimisé dans le paragraphe suivant.

3.3 Optimisation à l'aide de l'algorithme de Newton

Étant donné que les problèmes d'optimisation sont décrits comme des problèmes de minimisation, nous avons alors adopté cette notation. La fonction à minimiser dans ce paragraphe est donc notée f =-Score.

L'algorithme de Newton trouve itérativement les paramètres $\vec{P}$ qui minimisent la fonction f. Chaque itération résout l'équation suivante:

$$H\Delta P = -\overrightarrow{g} \qquad (3.6)$$

Où g est le gradient de f transposée

Avec :

$$g_i = \frac{\partial f}{\partial p_i} \qquad (3.7)$$

Et H est la matrice Hessienne de f Avec

$$H_{ij} = \frac{\partial^2 f}{\partial P_i \partial p_j} \qquad (3.8)$$

Avec P_i et P_j sont les éléments de $\overrightarrow{P} = P(i)^T{}_{i=1..3} = (t_x, t_y, \theta)^T$

Pour démontrer l'équation (3.6), on doit calculer le développement de Taylor du second ordre de la fonction f :

$$T(\vec{P} + \Delta\vec{P}) = f(\vec{P}) + g^T \Delta\vec{P} + \frac{1}{2}\Delta\vec{P}^T H_{f(\vec{P})} \Delta\vec{P} \qquad (3.9)$$

Trouver le minimum de $T(\vec{P} + \Delta\vec{P})$, avec $\Delta\vec{P}$ est le vecteur de déplacement.

Le minimum de l'équation 3.9 représente la solution de l'expression $\nabla T(\vec{P}+\Delta\vec{P}) = 0$.

Donc, nous voulons résoudre le système d'équations linéaires :

$$\nabla T(\vec{P}+\Delta\vec{P}) = g + H_{f(p)} \Delta\vec{P} = 0 \qquad (3.10)$$

Avec $\Delta\vec{P} = -H^{-1}\vec{g}$ (3.11)

La solution est un incrément $\Delta\vec{P}$, qui est ajouté à l'estimation actuelle:

$$\vec{P} \leftarrow \vec{P} + \Delta\vec{P} \qquad (3.12)$$

Comme nous l'avons dit auparavant, si la matrice Hessienne H est définie positive f($\vec{P}$) diminue dans la direction de $\Delta\vec{P}$. Si ce n'est pas le cas, H est remplacée par $H' = H + \lambda I$, Avec λ un scalaire positif plus grand que la valeur absolue du plus grand élément diagonale de la Hessienne et I la matrice unité.

Le gradient g et la Hessienne H sont construits en recueillant les dérivées partielles de tous les termes de la somme de l'équation (3.5).

Pour éviter de confondre le numéro du paramètre i avec l'indice de l'échantillon du balayage laser i, l'indice *i* pour le numéro d'échantillon est supprimé. Donc, nous pouvons écrire:

Posons $X^{''} = X^{'} - q$ (3.13)

Il est tout à fait remarquable à partir de (3.13) que les dérivées partielles de $X^{''}$ par rapport à $\vec{P}$ sont égales aux dérivées partielles de $X^{'}$.

Les entrées pour le gradient de la fonction de score peuvent être écrites comme suit : $g_i = \frac{\partial f}{\partial P_i}$

Avec $f = -\sum_k \exp\left(-\frac{(X^{'}-q)^T C^{-1}(X^{'}-q)}{2}\right)$

$$g_i = \frac{\partial}{\partial p_i}\left(-\sum_{k=1}^{n} \exp\left(-\frac{X^{''t}{}_k C^{-1} X^{''}_k}{2}\right)\right)$$

$$g_i = -\sum_{k=1}^{n} \frac{\partial}{\partial p_i}\left(-\frac{X^{''t}{}_k C^{-1} X^{''}_k}{2}\right) \exp\left(-\frac{X^{''t}{}_k C^{-1} X^{''}_k}{2}\right)$$

$$= -\sum_{k=1}^{n}\left[-\frac{1}{2}\left(\frac{\partial X^{''t}{}_k}{\partial p_i} C^{-1} X^{''}{}_k\right) - \left(\frac{1}{2} X^{''t}{}_k C^{-1} \frac{\partial X^{''}_k}{\partial p_i}\right)\right] \exp\left(-\frac{X^{''t}{}_k C^{-1} X^{''}_k}{2}\right) \quad (3.14)$$

Soit la matrice de covariance C:

$$C = \begin{pmatrix} j & k \\ l & m \end{pmatrix} = \begin{pmatrix} \frac{1}{n}\sum_{k=1}^{n}\left(x^{'}_k - q_x\right)^2 & \frac{1}{n}\sum_{k=1}^{n}\left(x^{'}_k - q_x\right)\left(y^{'}_k - q_y\right) \\ \frac{1}{n}\sum_{k=1}^{n}\left(x^{'}_k - q_x\right)\left(y^{'}_k - q_y\right) & \frac{1}{n}\sum_{k=1}^{n}\left(y^{'}_k - q_y\right)^2 \end{pmatrix}$$

On remarque que l=k donc $C = \begin{pmatrix} j & l \\ l & m \end{pmatrix}$ et $C^{-1} = \frac{1}{jm-l^2}\begin{pmatrix} m & -l \\ -l & j \end{pmatrix} = \begin{pmatrix} m^{'} & l^{'} \\ l^{'} & j^{'} \end{pmatrix}$

Nous avons $X^{''}_k = \begin{pmatrix} x^{''}_k = x_k cos\theta - y_k sin\theta + t_x - q_x \\ y^{''}_k = x_k sin\theta + y_k cos\theta + t_y - q_y \end{pmatrix} = \begin{pmatrix} A \\ B \end{pmatrix}$

$\frac{\partial X^{''}_k}{\partial t_x} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ et $\frac{\partial X^{''}_k}{\partial t_y} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ et $\frac{\partial X^{''}_k}{\partial \theta} = \begin{pmatrix} -x_k sin\theta - y_k cos\theta \\ x_k cos\theta - y_k sin\theta \end{pmatrix} = \begin{pmatrix} a \\ b \end{pmatrix}$

$\left(\frac{\partial X''_k{}^t}{\partial t_x}\ C^{-1}X''_k = m'A + l'B\right)$ et $\left(X''_k{}^t\ C^{-1}\frac{\partial X''_k}{\partial t_x} = m'A + l'B\right)$ .....(α)

$\left(\frac{\partial X''_k{}^t}{\partial t_y}\ C^{-1}X''_k = l'A + j'B\right)$ et $\left(X''_k{}^t\ C^{-1}\frac{\partial X''_k}{\partial t_y} = l'A + j'B\right)$ .....(β)

$\frac{\partial X''_k{}^t}{\partial \theta}\ C^{-1}X''_k = A(m'a + l'b) + B(l'a + j'b)$ et

$X''_k{}^t\ C^{-1}\frac{\partial X''_k}{\partial \theta} = A(m'a + l'b) + B(l'a + j'b)$.....(γ)

De α, β et γ, nous concluons que : $\left(\frac{\partial X''_k{}^t}{\partial P_i}\ C^{-1}X''_k\right) = \left(X''_k{}^t\ C^{-1}\frac{\partial X''_k}{\partial P_i}\right)$

Alors l'équation (3.14) devient:

$$g_i = \frac{\partial f}{\partial P_i} = \sum_{k=1}^{n} X''_k{}^t\ C^{-1}\frac{\partial X''_k}{\partial P_i}\ exp\left(-\frac{X''_k{}^t C^{-1}X''_k}{2}\right) \quad (3.15)$$

Les entrées de la Hessienne sont les suivantes:

$$H_{ij} = \frac{\partial^2}{\partial P_i \partial P_j}\left(-\sum_{k=1}^{n} exp\left(-\frac{X''_k{}^t\ C^{-1}X''_k}{2}\right)\right)$$

$$= -\sum_{k=1}^{n}\frac{\partial^2}{\partial P_i \partial P_j} exp\left(-\frac{X''_k{}^t C^{-1}X''_k}{2}\right)$$

$$= -\sum_{k=1}^{n}\frac{\partial}{\partial P_j}\left(\frac{\partial}{\partial P_i} exp\left(-\frac{X''_k{}^t C^{-1}X''_k}{2}\right)\right)$$

$$= -\sum_{k=1}^{n}\frac{\partial}{\partial P_j}\left(X''_k{}^t\ C^{-1}\frac{\partial X''_k}{\partial P_i}\ exp\left(-\frac{X''_k{}^t C^{-1}X''_k}{2}\right)\right)$$ Voir (3.15)

$$= \sum_{k=1}^{n}\left[\frac{\partial}{\partial P_j}\left(X''_k{}^t\ C^{-1}\frac{\partial X''_k}{\partial P_i}\right) * exp\left(-\frac{X''_k{}^t C^{-1}X''_k}{2}\right) + \left(X''_k{}^t\ C^{-1}\frac{\partial X''_k}{\partial P_i}\right) * \frac{\partial}{\partial P_j} exp\left(-\frac{X''_k{}^t C^{-1}X''_k}{2}\right)\right]$$

$$= \sum_{k=1}^{n}\left[\frac{\partial}{\partial P_j}\left(X''_k{}^t\ C^{-1}\frac{\partial X''_k}{\partial P_i}\right) * exp\left(-\frac{X''_k{}^t\ C^{-1}X''_k}{2}\right) + \left(X''_k{}^t\ C^{-1}\frac{\partial X''_k}{\partial P_i}\right) * \left[\left(X''_k{}^t\ C^{-1}\frac{\partial X''_k}{\partial P_j}\right) * exp\left(-\frac{X''_k{}^t\ C^{-1}X''_k}{2}\right)\right]\right] \quad (3.16)$$

Alors :

$$H_{ij} = \frac{\partial^2 f}{\partial P_i \partial P_j} = \sum_{k=1}^{n} exp\left(-\frac{{X''_k}^t\, C^{-1} X''_k}{2}\right)\left(\left({X''_k}^t\, C^{-1}\frac{\partial X''_k}{\partial P_i}\right)\left(-{X''_k}^t\, C^{-1}\frac{\partial X''_k}{\partial P_j}\right)\right.$$
$$\left. + {X''_k}^t\, C^{-1}\frac{\partial^2 X''_k}{\partial P_i \partial P_j} + \frac{\partial {X''_k}^t}{\partial P_j} C^{-1}\frac{\partial X''_k}{\partial P_i}\right) (3.17)$$

Les dérivées partielles de X''_k par rapport à P_i sont données par la matrice jacobienne J_T de la fonction de transformation 2D $T(\vec{P}, X)$ donnée par l'équation(3.4).

Les dérivées du premier et du second ordre dans les équations partielles (3.15) et (3.16) dépendent de la fonction de transformation. En utilisant la fonction de transformation $T(\vec{P}, X)$ de l'équation(3.4), on trouve les dérivées du premier ordre $\frac{\partial X''_k}{\partial P_i}$ qui sont données par la colonne i de la matrice jacobienne J_T de $T(\vec{P}, X)$

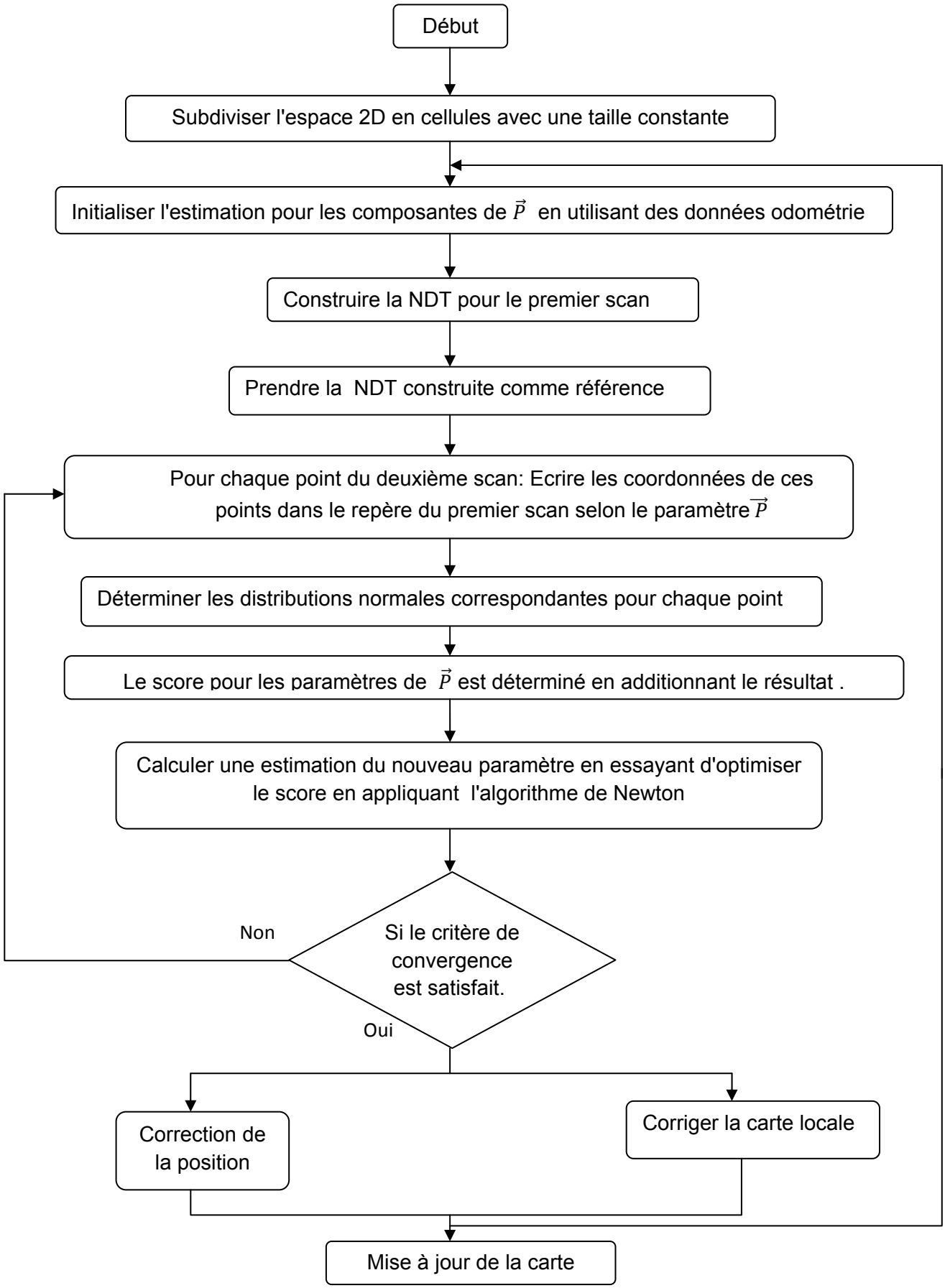
$$J_T = \begin{bmatrix} 1 & 0 & -\mathrm{x_k}\sin\theta - \mathrm{y_k}\cos\theta \\ 0 & 1 & \mathrm{x_k}\cos\theta - \mathrm{y_k}\sin\theta \end{bmatrix} \qquad (3.17)$$

Avec $\mathrm{x_k}$ et $\mathrm{y_k}$ les deux éléments de $X_k = \begin{pmatrix} \mathrm{x_k} \\ \mathrm{y_k} \end{pmatrix}$ et les dérivées du second ordre sont données par :

$$\frac{\partial^2 X''_k}{\partial P_i \partial P_j} = \begin{cases} \begin{pmatrix} -\mathrm{x_k}\cos\theta + \mathrm{y_k}\sin\theta \\ -\mathrm{x_k}\sin\theta - \mathrm{y_k}\cos\theta \end{pmatrix} & Si\ i = j = 3 \\ \begin{pmatrix} 0 \\ 0 \end{pmatrix} & Sinon \end{cases} \qquad (3.18)$$

3.4 L'algorithme

Notre algorithme NDT est donné dans l'organigramme suivant :

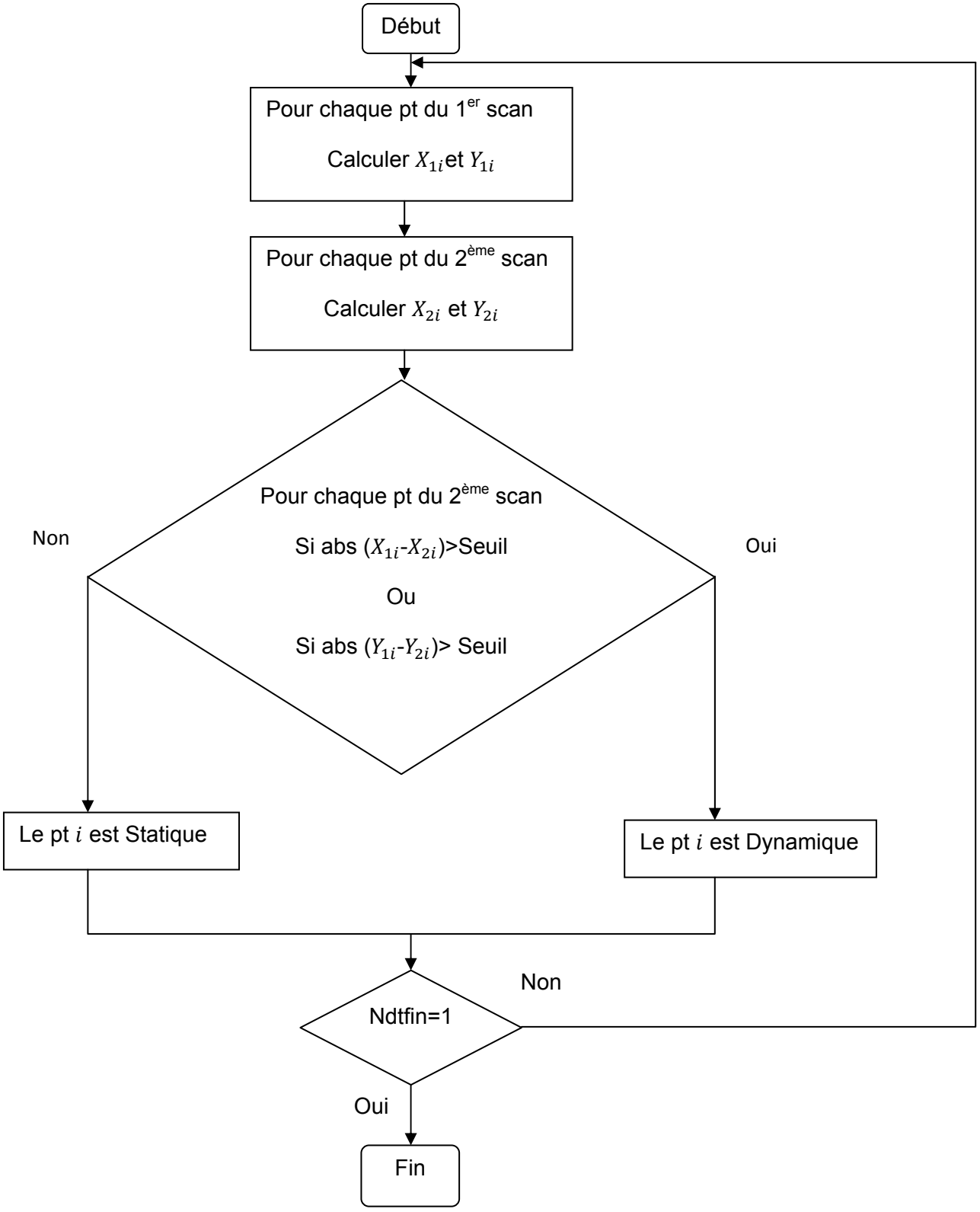


3.5 Implémentation de La NDT dans un environnement Dynamique

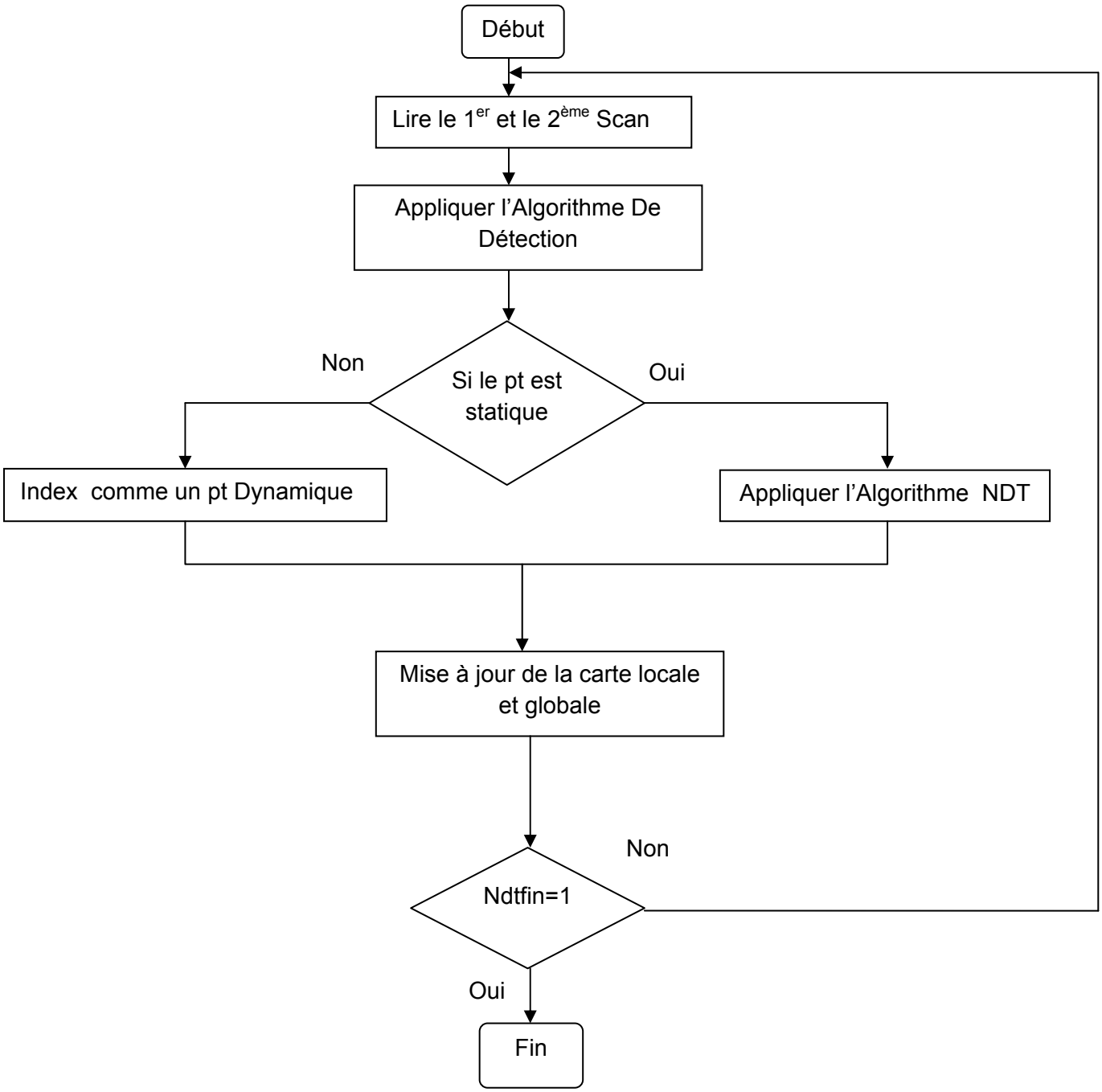
Afin de résoudre le problème de détection de mouvement qui est la partie la plus délicate dans le SLAM dynamique on propose cet algorithme. L'idée c'est de prendre deux scans successifs et de comparer point par point les distances parcoures entre les deux scans. Si cette distance et supérieure à un certain seuil, on classe donc ce point comme un point dynamique sinon il est statique. Ensuite, on va adapter notre algorithme de façon à ne traiter que les points statiques et marquer les points dynamiques afin des les visionner plus tard par une couleur différente dans la carte globale.

3.5.1 Algorithme de détection

On note les points mesurés par le laser pour le 1er scan par $P_{1i} = (X_{1i}, Y_{1i})$ et $P_{2i} = (X_{2i}, Y_{2i})$ pour le deuxième scan. Ndtfin est un paramètre pour arrêter l'application.



3.5.2 l'Algorithme NDT Dynamique



3.6 Conclusion

Dans ce chapitre, on a expliqué en détail les différentes étapes de la méthode NDT, et on a proposé une solution pour résoudre le problème de la détection des objets mobiles qui est la partie critique dans le SLAM dynamique.

Le chapitre suivant présente les résultats expérimentaux obtenus suite à l'implémentation des algorithmes développés dans ce chapitre sur le robot mobile Robucar.

CHAPITRE 4

IMPLEMENTATION ET RESULTATS

4.1 Introduction

Notre implémentation nécessite deux informations de deux types de capteurs différents, le scanner laser et les encodeurs du Robucar. Ces deux informations doivent être reçues en parallèle et en temps réel. Le logiciel Genom nous donne cette possibilité. Dans ce chapitre, on présente nos résultats avec plus de détail pour les parties software et hardware utilisées pour arriver à ces résultats.

4.2 Description du Robucar

Le Robucar est un robot mobile autonome de type voiture (Figure 4.1). Il est considéré comme un prototype de véhicule électrique servant de plateforme de recherches de la DPR (*Division Productique et Robotique*) du CDTA. Il a été conçu par la société Robosoft [43] qui est spécialisée dans les solutions de robotique, commercialise plusieurs types de bases électriques autonomes à roues, sans conducteur et filoguidées destinées à évoluer dans des milieux sains ou hostiles, ou à accomplir des missions spécifiques (transports dangereux, déminage, explorations, …). Les caractéristiques techniques du Robucar sont les suivantes **:**

- Longueur totale : 1836mm.
- Largeur totale : 1306mm.
- Hauteur : 616mm.
- Poids total avec batteries : environ 310kg.
- Capacité d'accueil : 2 personnes avec bagages.
- Vitesse maximale : 18km/h (5m/s).
- Autonomie : 2 heures d'utilisation continue.
- Conduite automatique ou manuelle.
- Alimentation 48v.
- 8 batteries de 12volts 60Ah avec un gestionnaire automatique de charge.
- Motorisation : 4 moteurs électriques de 1200watts.

- 4 roues motrices et directrices.
- 2 vérins électriques de direction (avant et arrière).
- Un frein à commande électrique par moteur.

Figure 4.1 : Le Robucar

1)PC embarqué.
2) Axe avant (2 roues motrices, vérin direction).
3) Arrêt d'urgence.
4) Joystick.
5) Châssis nid d'abeille.
6) Télémètre laser LMS 200.
7) Caméra CCD.
8) Batteries.
9) Ultrasons.
10) Axe arrière (2 roues motrices, vérin direction).

Le Robucar est équipé par quatre moteurs de traction, avec un encodeur incrémental pour chacune et deux vérins électriques pour le braquage (Figure 4.2), avec un encodeur absolu pour chaque vérin. La commande est assurée par deux microcontrôleurs MPC 555 de Motorola reliés à un PC embarqué via un bus CAN. Ce PC et relié via un câble croisé à un PC portable (Figure 4.3).

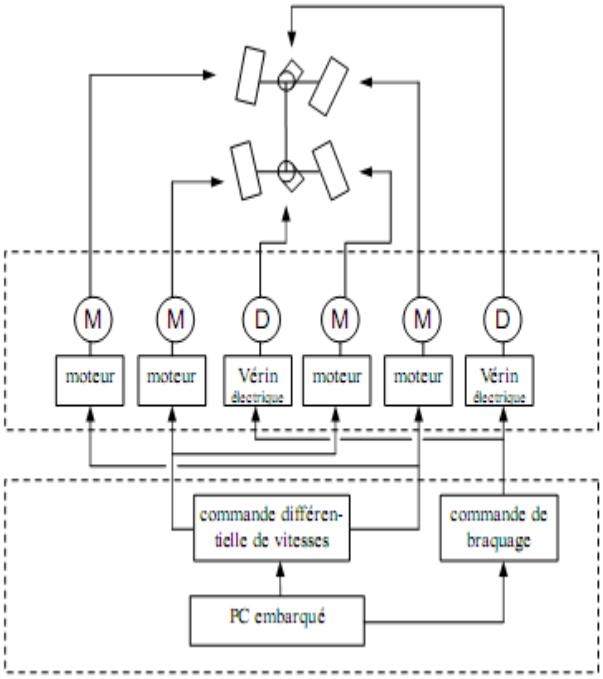


Figure 4.2: Schéma synoptique de commande

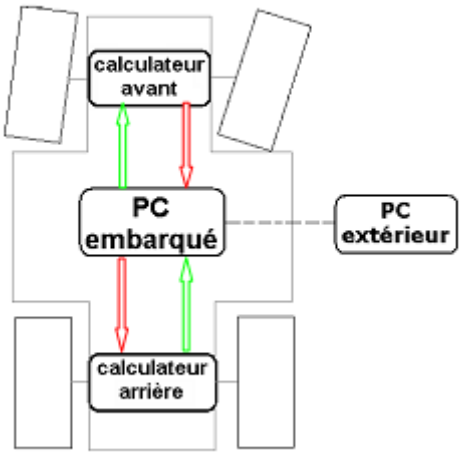


Figure 4.3: Organisation de la communication

4.2.1 Architecture logicielle du Robucar

Le robot mobile Robucar est piloté soit manuellement, par un joystick ou bien automatiquement. Le contrôle du Robucar est assuré par sa vitesse instantanée et son angle de braquage (V_R, ϕ). Ces deux variables sont envoyées vers une application temps réel développée sous un logiciel appelé Syndex v5.1

(Synchronized distribution executive) utilisant linux comme système d'exploitation (redhat v6.2).

SynDEx [44] est un logiciel de CAO (Conception Assistée par Ordinateur) niveau système fondé sur la méthodologie adéquation algorithme-architecture(AAA) pour aider à la réalisation d'applications distribuées temps-réel embarquées complexes comme on en trouve dans les domaines de l'avionique, de l'automobile, des télécommunications, de la robotique mobile, etc. Ces applications sont, avant tout, réactives c'est-à-dire qu'elles interagissent en permanence avec l'environnement auquel elles sont connectées. Elles doivent de plus respecter des contraintes temps réel sous peine de conséquences catastrophiques (temps réel critique).

4.2.2 Les modes de fonctionnement du Robucar

Le Robucar possède 3 modes de fonctionnement :

- ***Mode simple braquage (Single) :*** la traction se fait sur les quatre roues mais la direction se fait seulement sur le train avant. Les roues du train arrière sont fixes (**Figure 4.4-a)**
- ***Mode double braquage (Dual) :*** la traction se fait sur les quatre roues. La direction se fait à l'avant comme à l'arrière. Dans ce cas les deux trains tournent dans un sens opposé (**Figure 4.4-b**).
- ***Mode parking (Park) :*** la traction se fait sur les quatre roues. La direction se fait à l'avant comme à l'arrière de façon à faciliter le stationnement du véhicule. Dans ce cas les deux trains tournent dans le même sens (**Figure 4.4-c**).

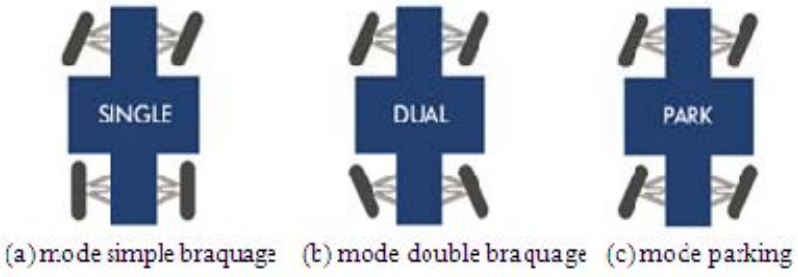


Figure 4.4: Les modes de fonctionnement du robot mobile Robucar

4.3 Description de Genom

GenoM (Génération of Module) est un environnement de développement qui permet de définir et de produire des modules qui encapsulent des algorithmes. Ainsi, un module est une entité logicielle standardisée qui offre des services et gère les traitements (i.e. nos algorithmes) associés à ces services : démarrage/arrêt du traitement, passage de paramètres, récupération des résultats, etc.

Ce logiciel est installé sous linux fédora en utilisant un PC Portable « Hp dv4 ». Ce portable est relié par un câble réseau croisé au PC embarqué. Le protocole TCP/IP est utilisé pour la communication entre les deux PCs.

Trois modules doivent être lancés en même temps pour notre application.

- ✓ **Le module Lms :** ce module communique avec le capteur Lms 200 et met les distances mesurées d'un balayage de 180 degré dans un vecteur.
- ✓ **Le module communication :** sert à communiquer avec le PC embarqué pour récupérer les quatre vitesses des quatre roues et la vitesse moyenne du Robucar et en plus les deux angles de braquage (avant et arrière).
- ✓ **Le module Ndt:** c'est le module où on a implémenté l'algorithme Ndt qui utilise les données des deux modules précédents et donne en sortie les coordonnées du Robucar (x_ndt,y_ndt,teta_ndt) ainsi que la carte de l'environnement.

La Figure suivante montre les trois modules lancés par Genom via le serveur tclServ.

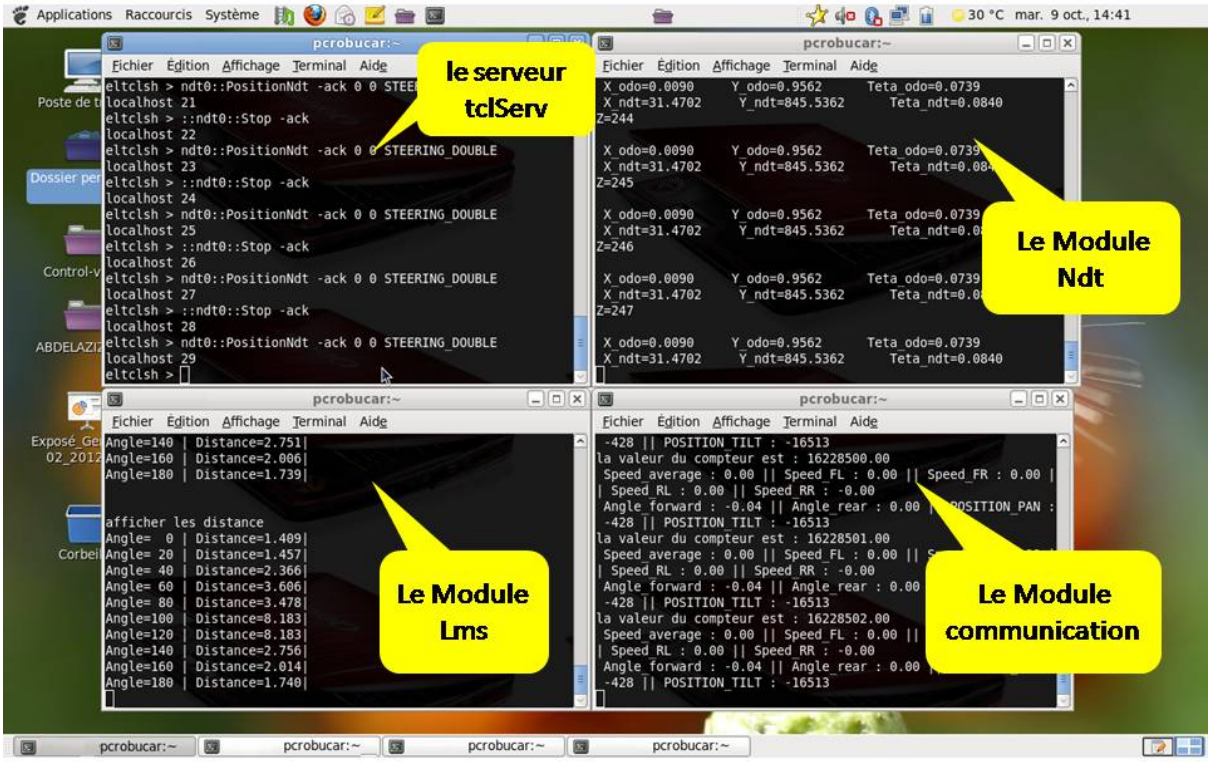


Figure 4.5: L'application lancée par Genom

Notre plateforme finale (Robucar et PC portable) utilisé pour implémenter nos algorithmes est montré sur la figure 4.6.

Figure 4.6: Plateforme Finale

4.4. Résultats d'implémentation de La NDT dans environnement Statique

Dans ce paragraphe, on présente les résultats d'implémentation de l'algorithme NDT dans différents environnements statiques.

4.4.1. Environnement 1

Dans cet environnement le robot démarre du couloir de la DPR pour atteindre le hall de la direction.

Figure 4.7: Photos du premier environnement d'expérimentation

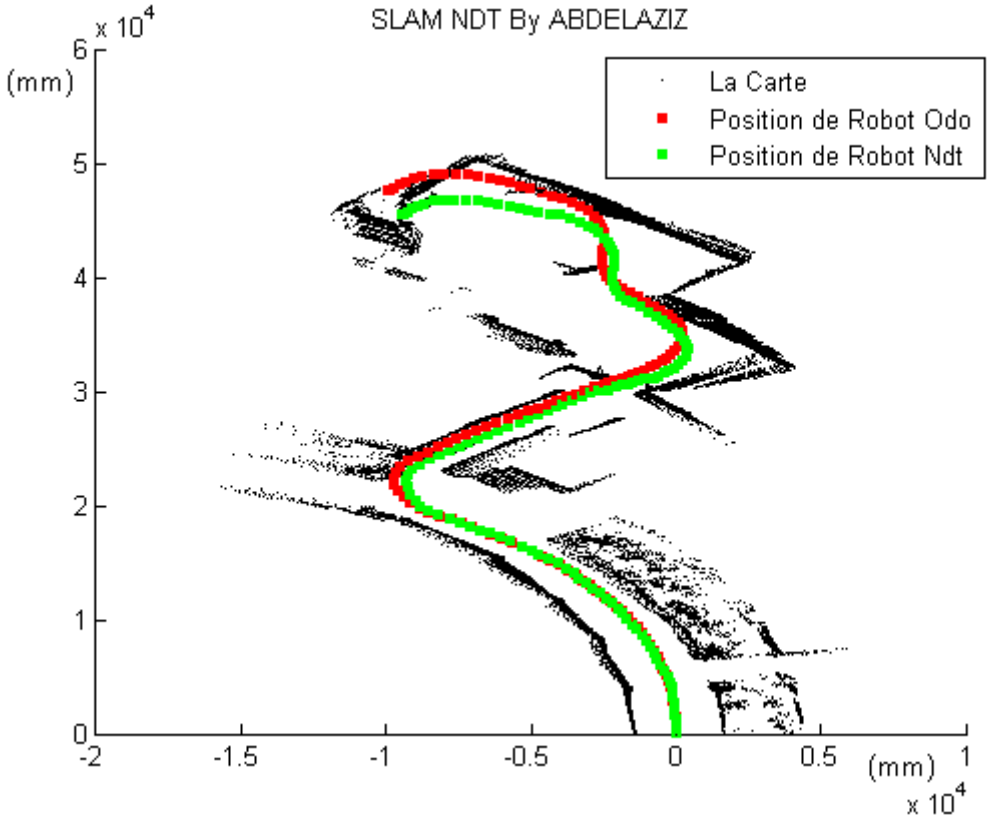


Figure 4.8: carte obtenue du premier environnement d'expérimentation

4.4.2. Environnement 2

Le robot démarre de la division jusqu'à la deuxième entrée de la direction (figure 4.9).

Figure 4.9: Photos du deuxième environnement d'expérimentation

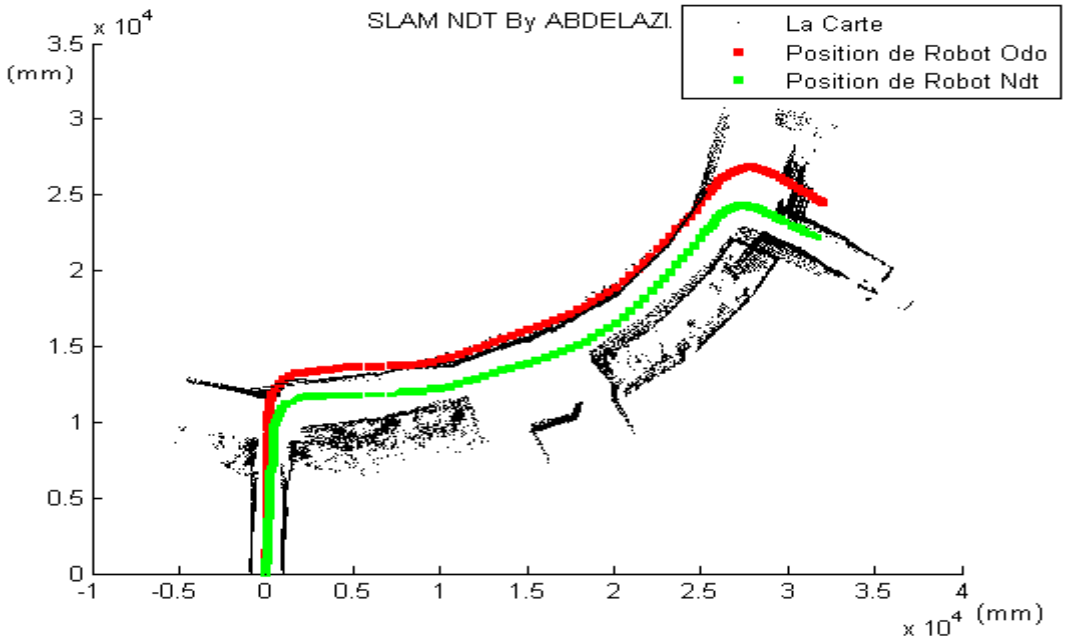


Figure 4.10 : carte obtenue du deuxième environnement d'expérimentation

4.4.3 Environnement 3

Dans cet environnement, tout le couloir a été traversé par le robot (figure 4.11).

Figure 4.11: Photos du troisième environnement d'expérimentation

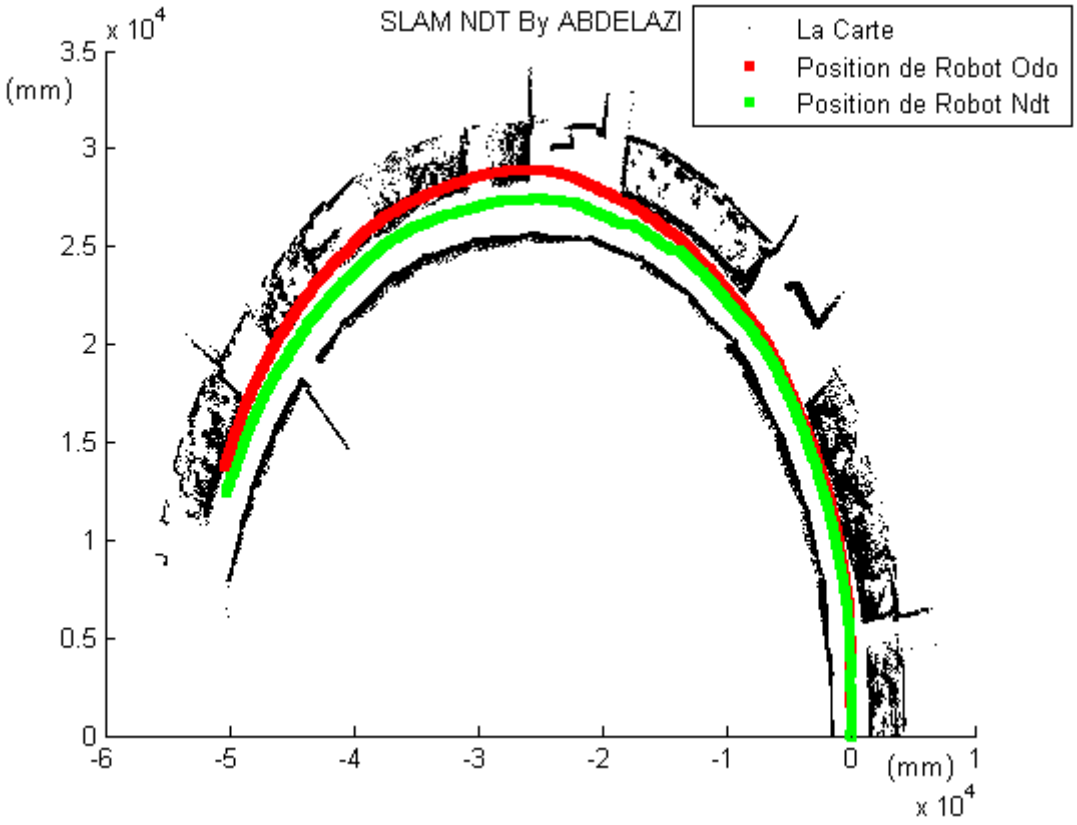


Figure 4.12: carte obtenue du troisième environnement d'expérimentation

4.4.4 Environnement 4

Cet environnement (le jardin de CDTA) est un environnement extérieur (outdoor) (voir figure 4.13).

Figure 4.13 : Photos du quatrième environnement d'expérimentation

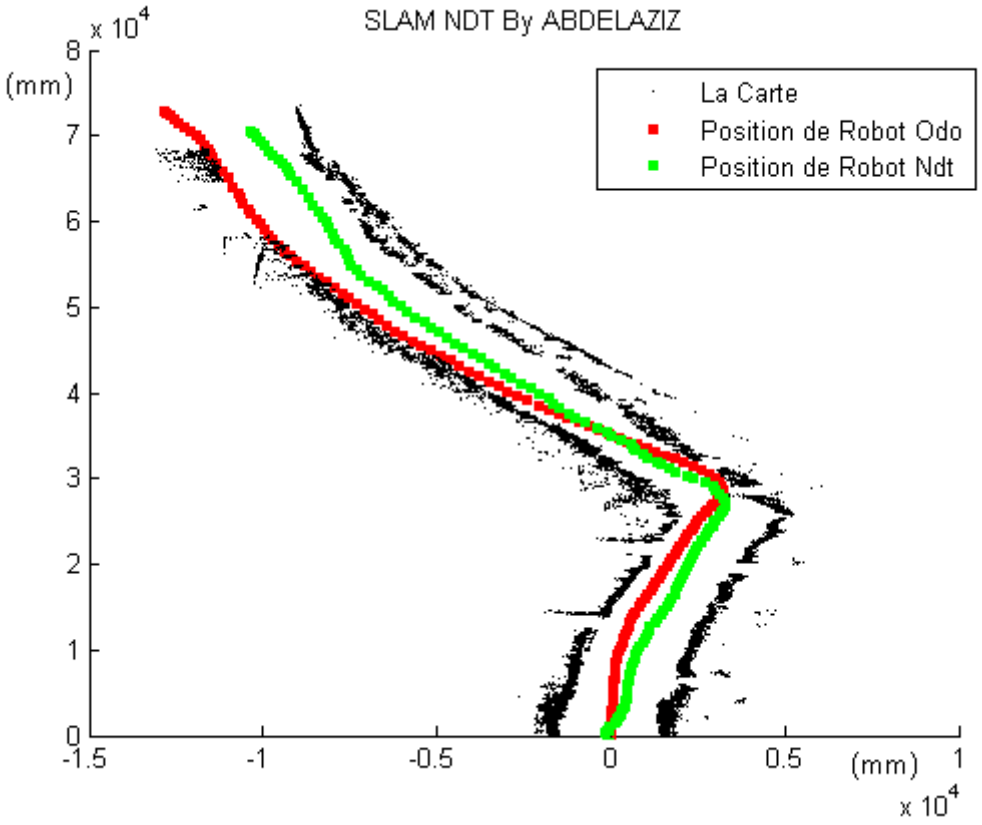


Figure 4.14: carte obtenue du quatrième environnement d'expérimentation

4.4.5 Discussion sur les résultats dans les environnements statiques

On remarque bien que l'algorithme NDT corrige bien les erreurs de l'odométrie. Ces erreurs qui croient proportionnellement avec la distance traversée par le robot, la Figure 4.15 montre les graphes d'erreurs pour le quatrième environnement. L'algorithme nous donne une carte fiable et claire de l'environnement statique qu'il soit intérieur ou extérieur.

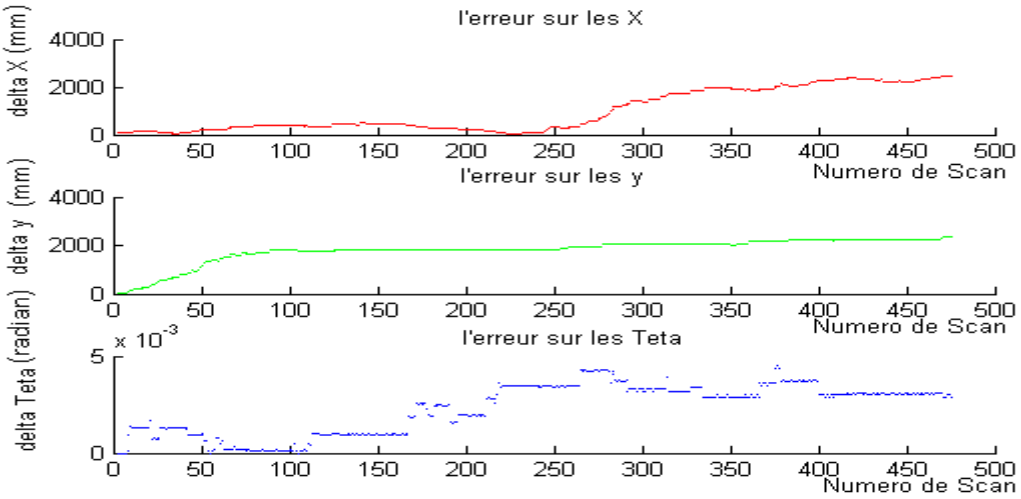


Figure 4.15: Les graphes d'erreurs

4.4.6 Temps d'exécution

Pour arriver à un temps d'exécution le plus petit possible, En plus de traitement parallèle des données par Genom on a décidé de ne pas utiliser les fonctions précompilées pour calculer par exemple les multiplications matricielles et les matrices inverse et transposé.

La Figure 4.16 montre le temps d'exécution en fonction du nombre d'itérations pour le deuxième environnement. La variation entre une itération et une autre est due au fait que la NDT traite d'une part un nombre de cellules différent d'un scan à un autre et d'autre part des cellules qui contiennent un nombre variable de points. De ce fait, le temps d'exécution varie avec la variation du nombre des cellules traité et du nombre des points traités dans chaque cellule.

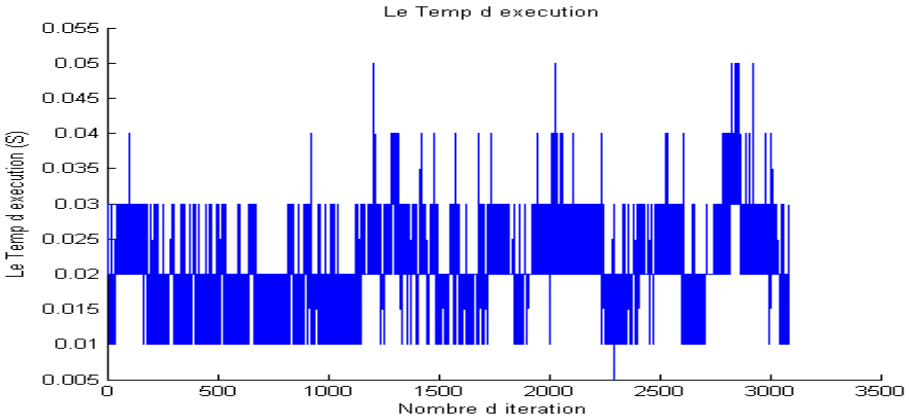


Figure 4.16: Temps d'exécution

4.5 Résultats d'Implémentation de La NDT dans un environnement dynamique

4.5.1 l'environnement Dynamique

Pour tester l'efficacité de notre algorithme de détection, on a décidé de prendre un piéton comme un obstacle mobile qui est souvent le cas pour un véhicule automatisé. Ce piéton va prendre des trajectoires aléatoires. La Figure 4.17 montre l'environnement d'expérimentation dynamique.

Figure 4.17 : Photos d’environnement d’expérimentation dynamique

4.5.2 les résultats pour l'environnement dynamique

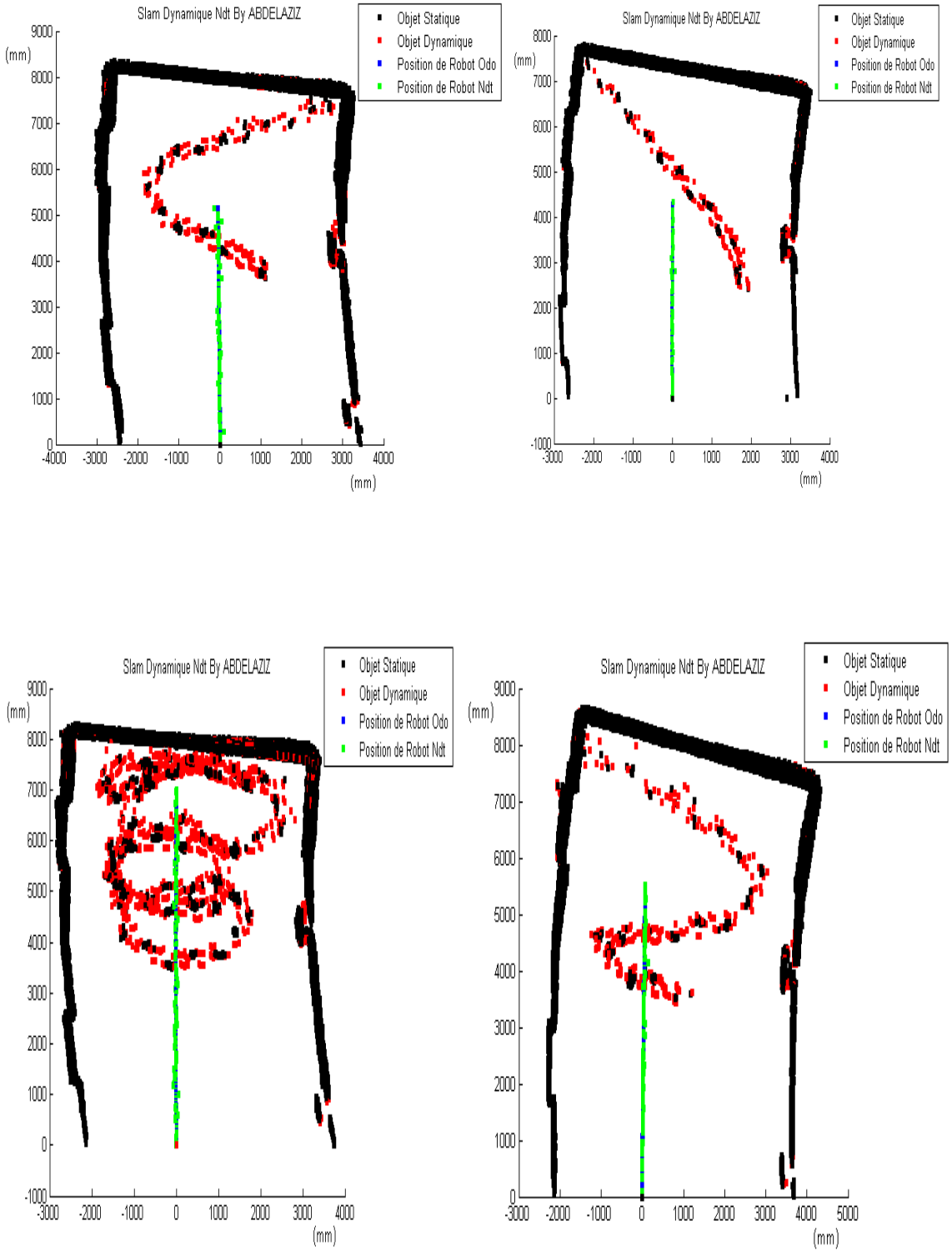


Figure 4.18: Cartes obtenues pour quatre trajectoires différentes

4.5.5. Discussion sur les résultats dans l'environnement dynamique

Les résultats d'implémentation de notre algorithme de détection d'objets mobiles proposé dans le chapitre précédent montre bien son efficacité, même si l'algorithme confond des fois entre un point dynamique et un point statique et ceci est dû essentiellement au bruit du capteur laser et aussi de la nature d'objet dynamique (un piéton) qui reste plus difficile a le détecter car la capteur laser ne capte que les jambes de piéton.

On remarque aussi qu'il n'y a pas un écart entre l'odométrie et la Ndt ce qui est dû au fait qu'on a pris une trajectoire droite et de petite longueur.

4.6. Conclusion

Ce chapitre a été consacré aux résultats d'expérimentation. L'algorithme a été implémenté sur le robot mobile Robucar et validé expérimentalement dans divers environnements statiques et dynamiques intérieur et extérieur, et montre bien la robustesse de notre implémentation.

CONCLUSION

Le travail présenté dans ce mémoire s'inscrit dans le cadre de la navigation en robotique mobile. Il traite le problème de la localisation et la cartographie simultanées dans un environnement statique et dynamique. L'idée principale est de réaliser une carte suffisamment claire décrivant l'environnement afin d'être utilisée pour la navigation d'un robot.

En premier lieu, nous avons présenté des généralités sur la robotique mobile et ses différents capteurs. Notre attention s'est surtout portée sur le télémètre laser SICK LMS 200, utilisé pour extraire les données de l'environnement.

Ces données seront par la suite utilisées par l'approche NDT.

Ensuite, nous avons introduit les concepts de localisation et cartographie qui sont les fonctions de base composant le problème SLAM. Puis, le problème SLAM a été mis en évidence ainsi que les différentes approches et techniques proposées dans la littérature pour résoudre ce problème.

Après cela, nous avons introduit la technique d'alignement des mesures utilisée pour résoudre le problème. Notre choix s'est focalisé sur l'algorithme NDT qui répond à nos contraintes comme le temps d'exécution. Les différentes étapes de l'algorithme ont été illustrées.

Nous avons, par la suite, implémenté un algorithme SLAM basé sur la méthode NDT. Les résultats obtenus en utilisant la plateforme Robucar dans des environnements statiques et dynamiques que ce soit intérieur ou extérieur démontrent bien l'efficacité et la fiabilité de l'algorithme.

Perspectives

Comme perspectives, nous proposons :

• L'utilisation d'un capteur LMS plus précis (grande résolution) et avec une portée plus longue.

• L'intégration de GPS pour la localisation dans les environnements à grande échelle.

ANNEXE A

FONCTONNEMENT DE GENOM

1. Implémentation sous l'environnement GenoM (Generator of Module)

Principe de génération d'un module

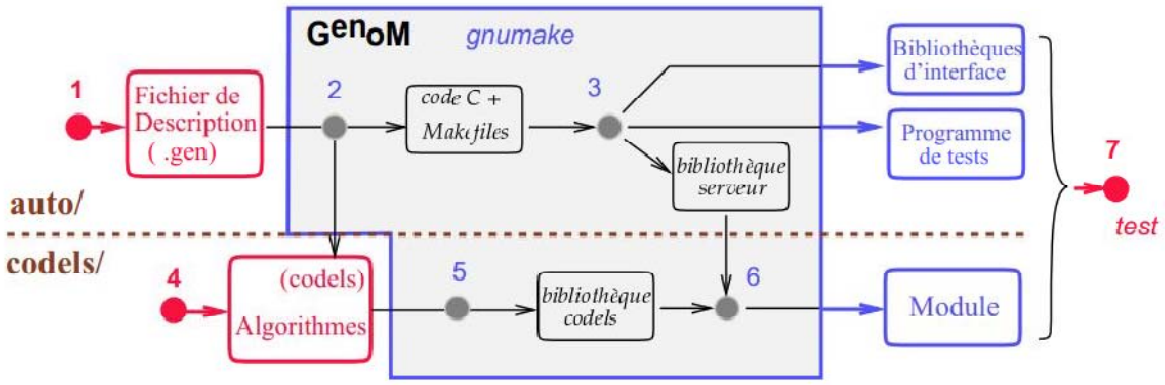
On distingue deux parties dans un module :

- le serveur, c'est à dire la partie d'encapsulation générée par GenoM à partir du cahier de description du module,
- l'ensemble des algorithmes (les codels) que nous avons développés et qui seront associés au serveur.

L'association de ces deux parties compose le module.

Dans un premier temps GenoM va produire ces codels mais ce ne seront que des fonctions vides : à nous de les remplir en y insérant nos algorithmes.

Pour bien distinguer ces deux parties les fichiers correspondants vont être répartis dans deux sous-répertoires : le répertoire **auto/** pour ce qui est généré par GenoM, et le répertoire **codels/** pour nos algorithmes (**Figure I).**



1. **décrire le module** : édition du fichier .gen,
2. et 3. Générer le module, puis le compiler,

4. **écrire les algorithmes** (les codels),
5. et 6. Compiler les codels, puis procéder à l'édition de liens avec le module,
7. **tester le module**.

Cycle de développement et séparation serveur/codels

Le développement d'un module se fait en trois étapes consécutives :

A. La première étape : La déclaration du module

Consiste à affecter un nom et un identifiant décrivant ce module, à donner un nom au type de sa structure de données interne et à mentionner les modules dont il aura besoin comme la lecture de leurs posters (données exportées à la lecture des autres modules) ou l'importation de leurs structures de données ou afin de leurs envoyer desrequêtes. Ainsi la déclaration ce fait de la manière suivante

```
module <<nom-module>> {
        number:         << identifiant-module>>;
        internal_data:  <<type-SDI >>;
        requires:       << module-name>>; ….;

};
```

1.2. La déclaration de la structure de données interne (fonctionnelle) SDI/f

Qui comporte toutes les variables et données manipulées par le module ou les traitements encapsulés par ce dernier. La description de la SDI /f revient à décrire le type déjà mentionné dans la déclaration du module, cette structure de données peut comporter des structures importées des autres modules et se décrit de la manière suivante :

```
/* importer une structure de données d'un autre module*/
Import from <module-name> {
#include ''decla-struct-de-module-source-de-la-structure.h'' /* inclusion de la
structure
                                                        prédéfinies d'un autre
module
                                                         */
……
                                    }
/* déclarer la structure interne du module */
#include "decla-struct-module.h" /* inclusion des types structurés des variables
                                        constituant la structure interne du module */
Typedef struct identificateur {
                                            /* Définition de la SDI/f */
…(c-like declaration) }
```

1.3. La déclaration des posters

Les posters représentent les données et les variables de la structure de données interne qui sont exposés à la lecture des autres module ou operateur, ou autrement dit : que l'operateur ou les autre module puissent les lire, les posters sont appelés aussi données exportées. Ils sont mis à jour par une tâche d'exécution qui étend le changement de valeurs des données interne sur le poster similaire.

```
Poster <<poster-name>> {
update:          <<update-type>>; // généralement mis à auto
data:            <<nom>> :: <<sdi-ref>>, ...; // les données à exportées
exec_task:       << nom-tâche-Exécution >>; // la tâche d'exécution qui le mis à jour
                        };
```

Où <<sdi-ref>> représente la données de la structure interne à exportée et <<nom>> représente la donnée similaire dans la zone poster.

1.4. Déclaration des requêtes

La requête est un élément fondamentale d'un module, elle peut être soit de contrôle ou elle modifie les paramètres du module et /ou vérifie leurs validité, soit d'exécution ou elle réalise toutes les tâches à effectuées par le module, ces tâches sont assurées par des algorithmes encapsulés dans des fonctions appelées codels appartenant à la requête, Elle est déclenchée par l'élément « tâche d'exécution ».

Une requête d'exécution à des paramètres d'entrée et des paramètres de sortie.

1.4.1. Requête de contrôle

```
request<<nom_requête>> {
doc:                "<some document about the request purpose>";
type:               control;                    //type requête de contrôle
input:              <nom>::<sdi-ref>;           // paramètres d'entrée de la requête
input_info:         <val-par-défaut>::"<nom>", ...;
output:             <nom>::<sdi-ref>;           // paramètres résultat de la requête
c_control_func:     <codel>;                    // codel de contrôle de la requête
fail_msg: <nom-msg >, ...;                      //message d'erreur
incompatible_with: <exec-rqst-name>,...;        /*la requête incompatible avec celle-ci qui sera
                        interrompue et arrêtée lors de lancement de celle-ci */        } ;
```

1.4.2. Requête d'exécution

```
request<<request-name>>{
doc:                    "<some document about the request purpose>";
type:                   exec; // type requête
exec_task:               <<nom-tâche-Exécution>>;
input:                  <nom>::<sdi-ref>; // paramètres d'entrée de la requête
input_info:              <val-par défaut>::"<name>", …;
output:                 <nom>::<sdi-ref>; // paramètres résultat de la requête
c_control_func:         <codel>;
c_exec_func_start:      <codel>; // codel de démarrage de la requête
c_exec_func:            <codel>; // codel principal de la requête
c_exec_func_end:        <codel>; // codel de terminaison de la requête
c_exec_func_inter:      <codel>; // codel de terminaison en cas d'interruption
c_exec_func_fail:       <codel>; //codel de terminaison de la requête en cas d'échec
fail_msg:               <nom-msg>, ...; //message d'erreur
incompatible_with:      <nom-req-exéc>, ...; /*la requête incompatible avec celle-ci qui
                        seraInterompue et arrêtée lors de lancement de celle-ci */ } ;
```

1.5. La déclaration de la tâche d'exécution

La tâche d'exécution est responsable du déclanchement des requêtes ou plus précisément de déclanchement de ses codels avec un codel supplémentaire faisant son initialisation. Le traitement se déclenche avec une priorité, une période s'il est périodique et avec une taille de pile attribuée. La tâche d'exécution s'occupe aussi des mises à jourdes posters.

Sa déclaration se fait de la manière suivante :

```
exec_task <<nom-tâche-exéc >> {

        period:         <nombre>;
        delay:          <nombre>;
        priority:       <<nombre>>;
        stack_size:     <<nombre>>;
        c_init_func:    <codel>; //codel d'initialisation
        fail_msg:       <nom-msg>,...; //message d'erreur


                          };
```

1.6. La tâche de contrôle

Elle est générée automatiquement par GenoM. Pour cela on n'a pas besoin de la décrire. Elle s'occupe du déclanchement de la requête de contrôle demandée par l'operateur et de la tâche d'exécution responsable du déclanchement de la requête d'exécution si l'operateur la désigne, ainsi que de récupérer les bilant d'erreurs.

B. La deuxième étape : Introduction des algorithmes et compilation

- On introduit nos algorithmes dans les zones vides des codels en spécifiant :

• Les traitements d'initialisation des paramètres (codel Init).

• Les premiers traitements à effectuer (codel Start).

• Les traitements principaux (codel Exec).
• Les traitements de terminaison (codel End).

C. **La troisième étape :** Après avoir remplir les codels avec nos algorithmes, on recompile et
on exécute le module pour commencer à l'utiliser.

- **Compilation :**

  On génère le module et ses éléments (.gen, Const & Struct) avec:

  1) genom –i –t … module.gen (le –i pour la première génération du module, le –t si le module existait déjà)

  2) Créer manuellement le build : **mkdir build**

  3) ../configure - -prefix=/home/genom/openrobots

  On exécute **Make regen** (si on modifie les structures du module) :

  4) Make (compiler le contenu du build)

  5) Make install (installer à l'intérieur du build)

- **Exécution:**

  a)Au moyen du programme interactif de test : Essay

Le programme interactif de test est un client du module. On peut en lancer plusieurs à condition de leur attribuer un numéro différent. Ils se présentent sous la forme d'un menu qui associe un numéro à chaque commande :

1. initialisation : **Lancer h2 init**
2. premiére MANIP : Lancer le module : **<nom du module > -b**
Lancer un programme de test : **<nom du module >Test 1**
3. manips suivantes (pour relancer un module) :

- ✓ Tuer l'ancien module : killmodule **<nom du module >**, ou -99 depuis **<nom du module >Test**)
- ✓ Recommencer l'opération 2.

4. FIN : Tuer la communication avec **h2 end.**

b) Au moyen de tclServ via un shell tcl :

tclServ permet de s'adresser depuis une interface unique à un ensemble de modules (envois de
requêtes et lecture de posters). Il offre un langage de programmation complet (tcl) et interactif
(interpréteur TCL, macros) pour contrôler les requêtes et données.

- **Utilisation du tclserv :**

1. Lancer le module avec l'option : **-b**
2. Lancer à bord de la machine cible le serveur : **tclserv**
3. Lancer sur la machine de tclserv un interpréteur tcl avec le package genom qui va permettre interactivement de s'adresser aux modules :

eltclsh -package genom

4. Commencer la session :

(a) Se connecter au serveur tclServ avec : connect **<nom du pc>**
(b) Charger les bibliothèques des modules qu'on veut contrôler : **lm <nom du module>**

3. L'importation des données via le module communication et le module lms :

3.1. Via le module communication : Connexion avec le pc du robot:

Le pc embarqué du robot doit être connecté avec le pc portable sur lequel on va travailler via un câble réseaux Ethernet en configurant les adresses IP pour obtenir la connexion.

3.2. Via le module Lms2 : Connexion du télémètre

Le télémètre laser doit être lié à un port série de l'ordinateur portable sur le quel on va travailler pour tester les modules sous GenOM, pour cela nous avons utilisé une carte (serial EXPRESS CARD RS-232), c'est une carte série qui contient un seul port RS-232 pour bus PCIuniversel. Elle répond aux nouvelles normes des slots pour carte d'extension, et fonctionne aussi bien sur les slots PCI à 3.3V ou à 5V.

3.3. Installation de la carte

Pour faire fonctionner la carte sous linux nous avons suivi la procédure suivante :

1. Dans un terminal on doit passer sur « root » en tapant l'instruction suivante :
    - **sudo su**
2. Pour initialiser la carte on introduit les instructions suivantes :
    - **modprobe -r pl2303**
    - **modeprob pl2303**
    - **chown utilisateur : groupe /dev/ttyUSB0**

Après cela la carte sera prête à fonctionner.

REFERENCES

1. S. Thrun, « Probabilistic algorithms in robotics» AI Magazine, vol. 21, no. 4, pp. 93–109, 2000
2. A. Diosi,Lindsay Kleeman, « Laser Scan Matching in Polar Coordinates With Application to SLAM» ARC Centre for Perceptive and Intelligent Machines in Complex Environments Departement of Electrical and Computer Systems Engineering Monash University, Victoria 3800,Australia.
3. P.Biber and W.Straber, «The normal distributions transform: A new approach to laser scan matching». In IEEE/RSJ International Conference onIntelligent Robots and Systems, pages 2743–2748, 2003.
4. D. Filliat, « cours de la robotique mobile », Ecole Nationale Supérieure de Techniques Avancées, http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
5. S. Bouraine, « Contribution à la localisation dynamique du robot mobile d'intérieur B21r en utilisant la plateforme multi sensorielle », mémoire de magister, Université de Saad Dahleb de Blida, Novembre 2007.
6. B. Bayle, « cours de la robotique mobile », Ecole Nationale Supérieur de Physique de Strasbourg Université de Strasbourg, année 2010–2011
7. G. Frappier « Système inertiels de navigation pour robots mobiles », Séminaire sur les robots mobiles, EC2, Paris, 1990
8. C. Cappelle, « Localisation de véhicules et détection d'obstacles Apport d'un modèle virtuel 3D urbain », Thèse de doctorat, Université des Sciences et Technologies de Lille, Décembre 2008.
9. M. K. Drocourt, « Perception Multisensorielle Pour le Positionnement d'un Véhicule Autonome dédié aux Personnes Handicapés Moteurs », Thèse de doctorat, Université de Metz, 21 Septembre 2004.
10. Quick Manual for LMS communication setup, « Hardware setup and measurement mode configuration ». Document technique, version 1.1 March 2002 (HW/MV)
11. C. Drocourt, « Localisation et modélisation de l'environnement d'un robot mobile par coopération de deux capteurs omnidirectionnels », Thèse de doctorat, Université de Technologie de Compiègne, 22 Février 2002.
12. T. S. Levitt et D. T. Lawton, « Qualitative navigation for mobile robots ». Artificial Intelligence, 44:305–360, 1990.

13. S. Thrun « Robotic mapping : A survey» éditeurs: Exploring Artificial Intelligence in the New Millenium. Morgan Kaufmann. 2.4.1, 2.4.2, 2.5.2(2002).
14. A. O. Djekoune, « Localisation et guidage du robot mobile Atrv2 dans un environnement naturel », Thèse de doctorat, Université des Sciences et de la Technologie Houari Boumediene, 15 Décembre 2010.
15. A. Zureiki, « Fusion de Données Multi-Capteurs pour la Construction Incrémentale du Modèle Tridimensionnel Texturé d'un Environnement Intérieur par un Robot Mobile », Thèse de doctorat, Université Toulouse III - Paul Sabatier, 16 septembre 2008.
16. P. Hoppenot, « Contribution de la robotique mobile à l'assistance aux personnes handicapées », Thèse de doctorat, Université d'Evry Val d'Essonne (EVE), 27 novembre 1997
17. B. Barshan et H.F. Durrant-White, « Inertial navigation system for a mobile robot », IEEE Trans on Robotics and Automation, Vol 12, N° 5, pp. 328-342, 1995.
18. R.S.M.S.P Cheeseman, R.Smith et M. Self, « A stochastic map for uncertain spatial relationships». In international symposium on Robotics Research, 1987.
19. P.Moutarlier et R.Chatila, «Stochastic multisensory data fusion for mobile robot location and environmental modelling». In international symposium on Robotics Research, 1990.
20. S. Thrun, « Probabilistic algorithms in robotics » AI Magazine, vol. 21, no. 4, pp. 93–109, 2000
21. P. Dempster, N. M. Laird, and D. B. Rubin, « Maximum likelihood from incomplete data via the EM algorithm » J. Royal Statistical Society, Series B, vol. 39, no. 1, pp. 1–38, 1977.
22. G. J. McLachlan and T. Krishnan, « The EM Algorithm and Extensions». New York: Wiley, 1997.
23. T. Duckett, S. Marsland, and J. Shapiro, « Learning globally consistent maps by relaxation,» in Proceedings of the IEEE International Conference on Robotics and Automation, San Francisco, 2000, pp. 3841–3846.
24. T. Duckett, S. Marsland, and J. Shapiro, « Fast, on-line learning of globally consistent maps,» Autonomous Robots, vol. 12, no. 3, pp. 287– 300, 2002.

25. U. Frese, P. Larsson, and T. Duckett, « A multilevel relaxation algorithm for simultaneous localisation and mapping,» IEEE Transactions on Robotics, vol. 21, no. 2, pp. 196–207, April 2005.
26. M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, « FastSLAM 2.0: An improved particle ltering algorithm for simultaneous localization and mapping that provably converges,» in Proceedings of the Eighteenth Int. Joint Conf. on Articial Intelligence (IJCAI-03). San Francisco.
27. Michael Montemerlo and Sebastian Thrun, «Simultaneous localization and mapping with unknown data association using FastSLAM ». In Proc. ICRA, 2003.
28. Levin and R. Szeliski, « Visual odometry and map correlation » in Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2004), Washington, DC, USA, 2004.
29. Tim K. Marks, Andrew Howard, Max Bajracharya, Garrison W. Cottrell, and Larry Matthies. « Gamma-SLAM: Stereo Visual SLAM in Unstructured Environments Using Variance Grid Maps». 2007.
30. Henrik Andreasson, Tom Duckett and Achim Lilienthal, « Mini-SLAM: Minimalistic Visual SLAM in Large-Scale Environments Based on a New Interpretation of Image Similarity» in Robotics and Automation, 2007 IEEE International Conference. Apr, 2007
31. R. Smith, M. Self, and P. Cheeseman, « A Stochastic Map for Uncertain Spatial Relationships » Proc. Fourth Int'l Symp. Robotics Research, 1987.
32. Pantelis Elinas, Robert Sim, and James J. Little, «σSLAM: Stereo vision SLAM using the Rao-Blackwellised particle filter and a novel mixture proposal distribution». In Proc. 2006 IEEE ICRA, 2006.
33. I.J Cox,»Blanche, «An Experiment in Guidance and Navigation Of Autonomous Robot Vehicle»,in IEEE Transacions On Robotics and Automation,1991
34. Paul J. Besl and Neil D. McKay, « A method for registration of 3-d shapes». IEEE Tran.on Pattern Analysis and Machine Intelligence, 14(2):239–256, 1992.
35. F.Lu and E.Milios, «Robot pose estimation in unknown environments by matching 2D range scans». Journal of Intelligent and Robotic Systems, 18(3):249–275, March 1997.

36. D.Hahnel and W.Burgard, «Probabilistic matching for 3d scan registration». InProc. of the VDI-Conference Robotik 2002 (Robotik), 2002
37. P.Forsman and A.Halme, «Feature based registration of range images for mapping of natural outdoor environments». In Proceedings of the Internationa Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT), 2004.
38. C.F. Olson and D. P. Huttenlocher, «Automatic target recognition by matching oriented edge pixels ». IEEE Transactions on Image Processing, 6(1):103–113, January 1997.
39. F.Boughorbel, A.Koschan, B.Abidi, and M.Abidi, «Gaussian fields: a new criterion for 3D rigid registration». Pattern Recognition, pp 1567–1571, 2004.
40. C.Wang and C. Thorpe, « A hierarchical object-based representation for simultaneous localization and mapping », In International Conference on Intelligent Robots and Systems, (IROS'04), Sendai (Japon), septembre - octobre 2004.
41. B.Abdellatif, «Cartographie de l'Environnement et Suivi Simultané de Cibles Dynamiques Par un Robot Mobile», thèse doctorat de l'université de Toulouse délivré par l'Université Toulouse III- Paul Sabatier,18 Décembre 2007
42. S.Rusinkiewicz and M.Levoy, «Efficient Variant of the ICP Algorithm», 2001.
43. www.robosoft.com
44. http://www.syndex.org