République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université M'hamed Bougara de Boumerdes
Faculté des Sciences de l'Ingénieur (FSI, exINGM)

Centre de Développement des Technologies Avancées
Division Robotique et Productique - Équipe NCRM

C.D.T.A

Contribution à La Localisation et la Cartographie Simultanées (SLAM) dans un Environnement Urbain Inconnu

Abdelhak Bougouffa et Anis Hocine

Mémoire présenté pour l'obtention du diplôme
Master en Systèmes Informatiques Distribués

Soutenue le 20 juin 2017 devant le jury :

Dr.	Mohamed Amine Riahla	*Président*	UMBB
Pr.	Mhamed Hamadouche	*Examinateur*	UMBB
Dr.	Sara Bouraine	*Promotrice*	CDTA
Mme.	Samira Mechid	*co-Promotrice*	UMBB
Mme.	Hafidha Boumeridja	*Examinatrice*	UMBB

UMBB-FSI / CDTA 2017

Mémoire présenté pour l'obtention du diplôme
Master en Systèmes Informatiques Distribués

Intitulé

Contribution à La Localisation et la Cartographie Simultanées (SLAM) dans un Environnement Urbain Inconnu

Présenté par :

Abdelhak Bougouffa abougouffa@fedoraproject.org
Anis Hocine hocine.anis.1992@gmail.com

Encadré par :

Dr.	Sara Bouraine	*Promotrice*	CDTA
Mme.	Samira Mechid	*co-Promotrice*	UMBB

UMBB-FSI / CDTA 2017

بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

ٱقۡرَأۡ بِٱسۡمِ رَبِّكَ ٱلَّذِي خَلَقَ ١ خَلَقَ ٱلۡإِنسَٰنَ
مِنۡ عَلَقٍ ٢ ٱقۡرَأۡ وَرَبُّكَ ٱلۡأَكۡرَمُ ٣ ٱلَّذِي
عَلَّمَ بِٱلۡقَلَمِ ٤ عَلَّمَ ٱلۡإِنسَٰنَ مَا لَمۡ يَعۡلَمۡ ٥

Au nom d'Allah, le Tout Miséricordieux, le Très Miséricordieux

Lis, au nom de ton Seigneur qui a créé, qui a créé l'homme d'une adhérence. Lis! Ton Seigneur est le Très Noble, qui a enseigné par la plume, a enseigné à l'homme ce qu'il ne savait pas.

Surat al-Alaq, versets 1-5

ملخص

هذا العمل يعالج إشكاليات تحديد الموقع ورسم الخرائط بالنسبة لروبوت من نوع سيارة في بيئة خارجية غير معروفة مسبقًا. الروبوت مجهز بلاقط ليزر من نوع Sick LMS511 Pro ، بيانات الليزر هذه يتم استخدامها فيما بعد لإنجاز مهام تحديد الموقع ورسم الخرائط. في مثل هذه الحالة، لا يمكن فصل مشكلتيّ تحديد الموقع ورسم الخرائط عن بعض، ولهذا اخترنا مقاربة **تحديد الموقع ورسم الخرائط المتزامن** SLAM . الحل المقترح في عملنا هذا أطلقنا عليه اسم NDT-PSO ، وهو يرتكز على الـ Normal Distribution Transform (NDT) والـ Particle Swarm Optimization (PSO) ، الـ NDT هي طريقة مستعملة لتحديد الموقع ورسم الخرائط المتزامن وهي طريقة مستعملة لتحديد التحويلات الهندسية (التحويلات النقطية من انسحاب ودوران) بين مجموعتين من بيانات الليزر وذلك من خلال تقنية المحاذات Alignment الخاصة بها، تمثيلها للبيئة يرتكز على نمذجة كل النقاط ثنائية الأبعاد بمجموعة من دوال الكثافة الاحتمالية Probability Density Functions . في الطريقة المقترحة في عملنا هذا، قمنا باستخدام خوارزم PSO في مرحلة الاستمثال (التحسين) Optimization لإيجاد التحويلات النقطية الصحيحة والتي من خلالها يتم استنتاج موقع الروبوت ودرجة استدارته. قمنا ببرمجة الحل المقترح تحت نظام التشغيل ROS وباستخدام لغة البرمجة Python وقمنا بتجربته على الروبوت RobuCar في إطار مشروع للنقل الحضري ذاتي القيادة.

كلمات مفتاحية — روبوت، روبوت من نوع سيارة، تحديد الموقع، استمثال، تحسين، رسم الخرائط، تحديد الموقع ورسم الخرائط المتزامن.

Abstract

This work deals with the localization and mapping problem of a car-like mobile robot in an unknown urban environment. The robot is equipped with a *Sick LMS511 Pro* laser sensor, the laser data are exploited to accomplish the task of localization and mapping. In such a situation, the two problems can not be dissociated, therefore, we propose to adopt a Simultaneous Localization And Mapping (SLAM) approach. The developed solution is called NDT-PSO. It is based on the Normal Distribution Transformation (NDT) method and the Particulate Swarm Optimization (PSO) method. NDT is a SLAM method whose principle is to determine the geometric transformation between two successive laser scans using alignment techniques. Its representation of the environment is based on the modeling of the all 2D points reconstructed from a laser scan by a collection of local normal distributions. The PSO method is used in the transformation parameters optimization step , this parameters are used to determine the poses (positions and orientations) of the robot. The proposed algorithms are implemented on the Robot Operating System ROS using Python language and tested on the RobuCar mobile robot as part of an intelligent urban transport project.

Key words — Mobile robot, Localization, Mapping, SLAM, PSO, NDT, ROS, RobuCar, Car-like robot.

Résumé

Ce travail traite les problèmes de localisation et de cartographie pour un robot mobile de type voiture évoluant dans un environnement urbain inconnu. Le robot est équipé d'un capteur laser de type *Sick LMS511 Pro* dont les données sont exploitées pour l'accomplissement de ces tâches. Dans une telle situation, les deux problèmes ne peuvent pas être dissociés, c'est pourquoi, nous proposons d'adopter une approche de localisation et cartographie simultanées (SLAM). La solution développée dans ce manuscrit est nommée NDT-PSO. Elle est basée sur la méthode de la transformation de distribution normale (NDT) et la méthode d'optimisation par essaim particulaire (PSO). La NDT est une méthode SLAM dont le principe est de déterminer la transformation géométrique entre deux scans laser successives grâce à des techniques d'alignement. Sa représentation de l'environnement est basée sur la modélisation de tous les points 2D reconstruits à partir d'un scan laser par une collection de distributions normales locales. La méthode PSO est utilisée dans la phase d'optimisation des paramètres de transformation afin de déterminer les poses (positions et orientations) du robot. Les algorithmes proposés sont implémentés en langage Python sous le système ROS et testés sur le robot mobile RobuCar dans le cadre d'un projet de transport urbain intelligent.

Mots clés— Robot mobile, OES, Localisation, Cartographie, NDT, SLAM, ROS, RobuCar.

Dédicaces

Je dédié ce travail,

À ma mère ♡, qui a œuvré pour ma réussite, par son amour, son soutien, tous les sacrifices consentis et ses précieux conseils, pour toute son assistance et sa présence dans ma vie, reçois à travers ce travail aussi modeste soit-il, l'expression de mes sentiments et de mon éternelle gratitude.

À la mémoire de mon père, à mes chères sœurs, mes frères et à toute ma famille.

À tous mes amis des *Scouts Musulmans Algériens*, et plus particulièrement à mon prof, mon leader, mon frère, mon collègue et mon amis... Dr. Othmane Touat, que Dieu vous bénisse.

À mes deux honorables professeurs du primaire Mme. Leila Baroud et Mme. Safi, merci pour vos conseils, merci de m'avoir motivé et de m'avoir donné du courage, je ne vous oublierai jamais.

À mes chères amis, Pédro, Anes, Sifou, Mohammed^2, Amine^2, Abdellah, Hamza, Zinou, Fodil, Tabloudj... la liste est vraiment longue, je vous aime tous !

Abdelhak Bougouffa

Dédicaces

Je dédié ce travail

À mes parents,

À ma chère famille,

À mes amis,

À mes proches !

Anis Hocine

Remerciment

En préambule à ce mémoire, nous souhaitons adresser nos remerciements les plus sincères aux personnes qui nous a apporté leur aide et qui ont contribué à l'élaboration de ce travail.

Nous tenons à remercier plus particulièrement Dr. Sara Bouraine qui, en tant que promotrice, s'est toujours montrée à l'écoute. Ses conseils et le temps qu'elle a bien voulu nos consacrer ce qui nous a permis de travailler dans les meilleurs conditions possibles.

Nous tenons aussi à remercier Mme. Samira Mechid, pour sa présence, disponibilité, orientations et conseils durant tout notre cursus.

Nous remercions également tous les membres de l'équipe NCRM et plus précisément la chef d'équipe Dr. Ouahiba Azouaoui qui nous ont aidé durant la période de notre stage au sein du CDTA.

En fin, nous n'oublierons pas de remercie tous nos enseignants et plus particulièrement Pr. Mhamed Hamadouche et Mme. Hafidha Boumeridja pour leur aide et soutien durant toutes ces années.

Table des matières

Table des figures

Liste de abréviations et d'acronymes

SLAM	Simultanious Localization and Mapping
ICP	Iterative Closest Point
LF	Likelihood-field
NDT	Normal Distribition Transform
PSO	Particle Swarm Optimization
CDTA	Centre de Developpement des Technologies Avancées
KF	Kalman Filter
EKF	Extented Kalman Filter
ROS	Robot Operating System
CIR	Centre Instantané de Rotation
GPS	Global Positioning System
CML	Concurrent Mapping and Localization
LF/SoG	Likelihood-Field defined as a Sum of Gaussians
IDC	Iterative Dual Correspondences
LiDAR	Light Detection and Ranging
NCRM	Équipe de Navigation et Contrôle des Robots Mobiles autonomes au sein de CDTA

Introduction générale

"Les savants des temps passés et des nations révolues n'ont cessé de composer des livres. Ils l'ont fait pour léguer leur savoir à ceux qui les suivent. Ainsi demeurera vive la quête de la vérité"

Mohammad ibn Moussa
al-Khwarizmi

La robotique est un très bon exemple de domaine pluri-disciplinaire qui implique de nombreuses thématiques telles que la mécanique, l'infotronique, la mécatronique, l'électronique, l'automatique, l'informatique, l'intelligence artificielle... etc. Il existe donc diverses définitions du terme robot.

En général, un robot 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 la perception de son entourage.

Au cours des dernières décennies, différentes recherches ont été élaborées pour réaliser des robots mobiles qui s'étendent dans le monde humain. Plusieurs secteurs d'application ont été explorés comme l'industrie manufacturière, le secteur médical (chirurgie de précision, robots infirmiers... etc.), le secteur militaire (sauvetage, surveillance, robots démineurs, drônes... etc.), l'exploration spatiale et sous-marine, le transport... etc.

Ainsi, afin de mener à bien sa tâche, un robot doit être capable de naviguer vers son but en évitant de rentrer en collision avec les obstacles qui l'entourent, et en utilisant uniquement l'information issue de ses capteurs embarqués.

Prenons l'exemple d'un système de transport intelligent, à savoir, le domaine d'intérêt traité dans ce travail, afin que le robot puisse transporter des passagers à

partir d'un point de départ jusqu'à un point d'arrivé dans un environnement réel, il est souvent nécessaire de connaître la position du robot dans son environnement et de disposer d'une carte qui permette de planifier les déplacements pour atteindre un but précis en évitant les obstacles rencontrés sur le passage du robot.

Le processus complet qui permet à un robot de mémoriser son environnement, puis de s'y déplacer pour rejoindre un but, peut être découpé en trois phases : la cartographie, la localisation et la planification. Ces trois phases permettent de répondre aux trois questions fondamentales pour la tâche de navigation : *Où suis-je ?*, *Où sont les autres lieux par rapport à moi ?* et *Comment puis-je atteindre mon but ?*

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. Enfin la planification est la phase qui permet, en connaissant la carte de l'environnement et la position actuelle du robot, de décider des mouvements à effectuer afin de rejoindre un but fixé dans l'environnement.

Ces trois phases sont évidemment fortement interdépendantes. En générale, l'ordre dans lequel elles sont citées fait directement apparaître le fait que la seconde phase dépend de la première. En effet, estimer la position du robot au sein d'une carte de l'environnement suppose implicitement que cette carte existe et qu'elle contient la position courante du robot. De même, la troisième phase dépend des deux premières, car la planification suppose que l'on connaisse sa position et que la carte de l'environnement représente une portion de l'environnement contenant au moins un chemin reliant cette position au but qui doit être atteint.

La relation entre les deux premières phases (*La cartographie et la localisation*) est plus subtile qu'une simple relation d'antériorité, c'est le même problème que pour **l'œuf et la poule**. Chacun des deux éléments peut ; en effet, être considéré comme préalable à l'autre mais dépend aussi de l'autre pour sa réalisation.

Dans le cas de la cartographie et de la localisation, nous avons déjà vu que la localisation repose sur une phase préalable de cartographie. De l'autre côté, pour construire une carte, il est nécessaire de savoir où ajouter dans la carte partielle déjà existante toute nouvelle information recueillie par le robot. Cela requiert donc une phase préalable de localisation au sein de la carte partielle déjà existante. Pour un robot complètement autonome, il est nécessaire de traiter ces deux problèmes simultanément. Nous parlons ainsi d'un problème de *"la localisation et la cartographie simultanées"* (Simultaneous Localization and Mapping, SLAM).

Dans le chapitre d'état de l'art, nous allons présenter le problème du SLAM de manière générale, afin de mieux situer le contexte du travail.

Il existe plusieurs méthodes SLAM dans la littérature. Cependant, la majorité

d'entre elles fournissent des résultats précis en environnement d'intérieur uniquement. Dans notre cas, où le robot doit se déplacer en milieu urbain (environnement d'extérieur), il faut proposer une méthode qui prend en compte les contraintes et les incertitudes d'un tel environnement.

Dans ce travail, nous avons développé la méthode NDT-PSO. Elle est basée sur la méthode de la transformation de distribution normale (Normal Distribution Transform, NDT) proposée dans [10], que nous avons amélioré par l'optimisation par essaim particulaire (Particle Swarm Optimization, PSO) [44].

Organisation du mémoire

Ce manuscrit est organisé en trois chapitres structurés de la manière suivante :

— Le premier chapitre apporte une formulation théorique du problème de SLAM. Cet état de l'art permet de montrer la complexité dans la réalisation de la localisation et la cartographie simultanément dans des environnements inconnus, les techniques de localisation et cartographie disponibles, en suite, les différentes méthodes existantes pour résoudre ce problème.

— Le second chapitre présente en détail l'approche utilisée dans ce projet de fin d'étude, cette approche consiste à associer des scans laser 2D successives l'un par rapport à l'autre afin de construire une carte de l'environnement en fournissant simultanément l'estimation de la position dans la même carte.

— Le dernier chapitre est consacré à l'implémentation et au testes de l'algorithme SLAM basé sur la méthode NDT que nous avons amélioré. Cette approche est testée en utilisant des ensembles de données obtenus par des expériences réalisées dans des environnements intérieur et extérieur.

— Ce manuscrit se termine par une conclusion générale résumant l'essentiel des travaux menés dans ce projet.

Chapitre 1

État de l'art

"Connais ton ennemi et connais-toi toi-même ; eussiez-vous cent guerres à soutenir, cent fois vous serez victorieux"

Sun Tzu

Ces dernières années, la technologie de la robotique mobile a permis le développement de système pouvant évoluer dans des environnements réels avec une intelligence qui rend la capacité de décision du robot proche du raisonnement humain. La décomposition du problème de la mobilité pour les robots mobiles autonomes amène à définir une architecture classique en robotique, organisée suivant un fonctionnement séquentiel ***perception, décision et action***[25] ou selon [70] le cycle ***see-think-act*** (voir la *figure 1.1*). Le robot a donc besoin d'une analyse cohérente de son environnement afin de pouvoir se localiser et y évoluer en toute sécurité, cela peut être réalisé grâce à la perception qui désigne la capacité du système à recueillir, traiter et mettre en forme des informations utiles au robot pour agir et réagir.

Dans l'absence d'informations suffisantes sur la position du robot (*localisation*) et sur la nature de son environnement (*cartographie*), le robot ne peut plus être en mesure d'accomplir ses tâches correctement (génération de trajectoires, évitement d'obstacles, atteindre une cible... etc.).

Pour permettre à un robot de se déplacer de manière autonome, celui-ci doit d'une part construire une carte fiable de son environnement, et d'autre part se localisé avec précision dans cet environnement, en construisant une carte du lieu

de manière incrémentale. Le robot peut ainsi se déplacer librement dans l'espace découvert, en identifiant et évitant les objets mobiles rencontrés sur son passage.

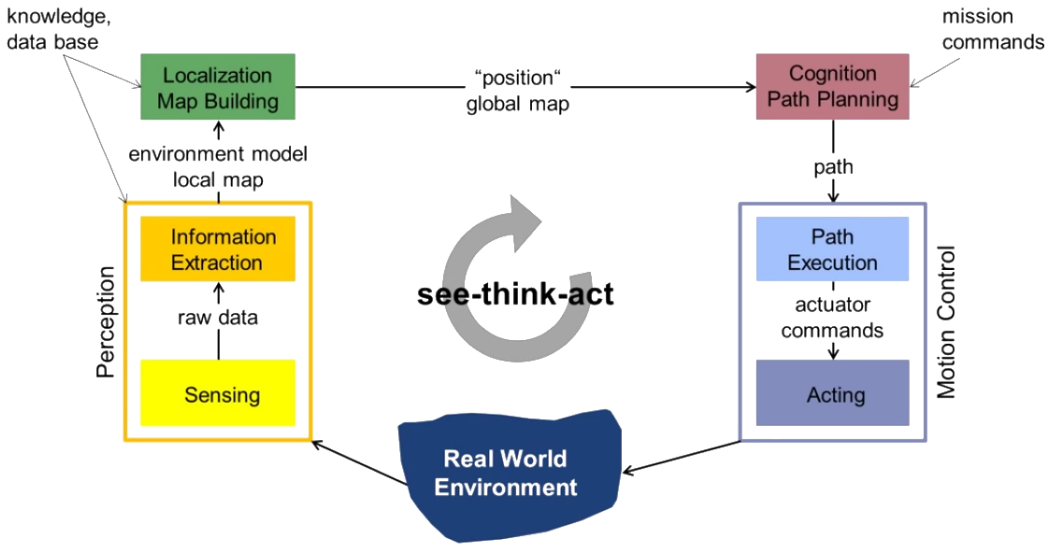


FIGURE 1.1 – Le cycle perception, décision et action ou bien see-think-act[70].

Dans ce chapitre, nous allons présenter les SLAM de manière générale, commençant par la perception en suite nous allons représenter les différentes méthodes de cartographie et localisation afin de mieux ce situe dans le contexte du travail de notre projet.

1.1 La perception

La perception consiste globalement à recueillir les informations sensorielles dans le but d'acquérir une connaissance et une compréhension du milieu d'évolution. Elle est préalable et indispensable aux tâches suivantes qui sont généralement pour un robot mobile autonome les tâches de localisation et de cartographie (voir la *figure 1.2*).

1.1.1 Les systèmes de perception

Le choix d'un système de perception est souvent dépendant du milieu d'évolution du robot mobile ainsi que du coût de l'intégration des capteurs sur le robot. La précision désirée et la fréquence d'acquisition sont autant des facteurs qui augmentent le coût d'un capteur.

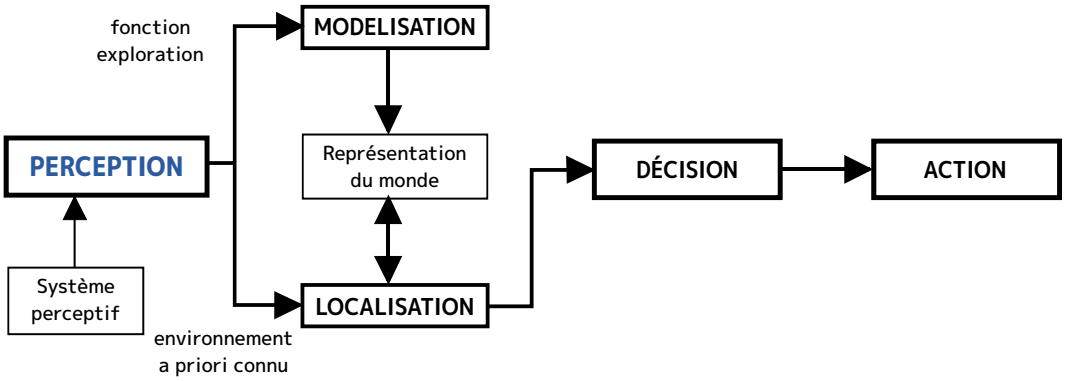


FIGURE 1.2 – Chaîne fonctionnelle d'un système de navigation [25]

La classification des capteurs est généralement faite par rapport à deux familles [25] :

— **Les capteurs proprioceptifs** : qui fournissent des informations propres au comportement interne du robot, *i.e.* déterminer son état à un instant donné.
— **Les capteurs extéroceptifs** : qui fournissent des informations sur le monde extérieur au robot.

Les capteurs proprioceptifs

Ces capteur fournissent, par intégration, des informations élémentaires sur les paramètres cinématiques ou dynamique du robot. Les informations sensorielles gérées dans ce cadre sont généralement des vitesses, des accélérations, des angles de giration, des angles d'attitude[25, 78, 36].

les capteurs proprioceptifs peuvent être regroupés en deux familles[33] :

— **Les capteurs de déplacement** qui comprennent (*les odomètres*[59], *les accéléromètres*, *les radars Doppler*[43], *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.
— **Les capteurs d'attitude** qui mesurent deux types de données : les angles de cap et les angles de roulis et de tangage. Ils sont principalement constitués par (*les gyroscopes*, *les gyromètres*, *les capteurs inertiels composites*, *les inclinomètres*, *les magnétomètres*... etc.). Ces capteurs sont en majorité de type inertiel.

Les capteurs extéroceptifs

Les capteurs extéroceptifs[84, 83] sont employés en robotique mobile pour collecter des informations sur l'environnement d'évolution du système mobile. Ils sont le complément indispensable aux capteurs proprioceptifs présentés précédemment. Ils sont utilisées pour conditionner et traiter les informations sensorielles. Ils sont notamment utilisés dans les domaines d'application tels que *l'évitement d'obstacle*, *la localisation*, *la navigation*, *la modélisation d'environnements*... etc. Les principaux capteurs utilisés en robotique mobile sont : *les capteurs télémétriques* (*les ultrasons*, *les lasers* et *les infrarouges*... etc.), *le GPS (Global Positioning System)*, *les caméras*... etc.

1.2 La localisation

La localisation est l'une des étapes les plus fondamentales des processus de navigation et de cartographie. Sa réussite conditionne l'ensemble des performances du robot. Elle est nécessaire pour les autres tâches tel que la planification, le suivi de trajectoire... etc.

Nous pouvons distinguer deux types de méthodes de localisation[25] :

— Les méthodes de localisation *relatives*, basées sur l'utilisation des capteurs *proprioceptifs*.

— Les méthodes de localisation *absolues*, basées sur l'utilisation des capteurs *extéroceptifs*.

— Les méthodes dites *hybrides* qui sont basées sur l'utilisation conjointe des *deux types de capteur*.

1.2.1 Localisation relative

La localisation relative[20, 80, 23], consiste à évaluer la position, l'orientation, et éventuellement la vitesse du robot mobile par intégration des informations fournies par ces capteurs proprioceptifs. L'intégration se faite par rapport au point de départ du robot. Ces données peuvent être des informations de déplacement (*odomètre*), de vitesse (*vélocimétre*) ou d'accélération (*accéléromètre*)[25]. Ces systèmes permettent d'obtenir un flux relativement important au niveau de l'obtention des estimations de position. Cette caractéristique intéressante a favorisé l'utilisation de ces systèmes de localisation en robotique mobile, ainsi que dans des secteurs de pointe tels que les domaines spatiaux et militaires : *fusées*, *missiles*, *avions*, *sous-marins*... etc.

1.2.2 Localisation absolue

La localisation absolue est une technique qui permet à un robot de se repérer directement dans son milieu d'évolution, que ce soit en environnement extérieur (*mer*, *espace*, *terre*), ou en environnement intérieur (*ateliers*, *immeubles*, *centrales nucléaires*... etc.). Ces méthodes de localisation sont basées sur l'utilisation des capteurs extéroceptifs[48, 24, 2, 25].

Pour estimer la position du robot, il est possible de construire une carte locale représentant l'environnement proche de la position courante. La comparaison de cette *carte locale* et de la *carte globale* de l'environnement permet alors de trouver la position comme le montre la *figure 1.3*.

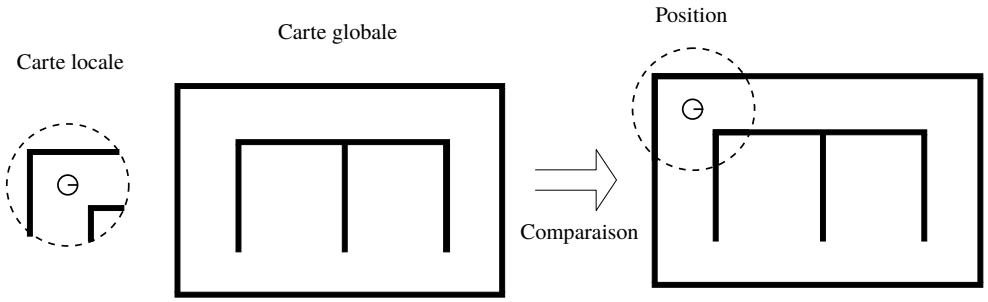


FIGURE 1.3 – Estimation de la position [31].

1.2.3 Localisation hybride

Les chercheurs se sont rendu compte de l'intérêt à utiliser conjointement la localisation relative et la localisation absolue[42, 73, 79, 14]. Il s'agit alors de recalage dynamique.

Les méthodes de localisation relatives ont pour inconvénient de générer une erreur cumulative avec la distance parcourue. Pour pallier ce problème de dérive, un système de localisation absolue peut être utilisé il sera chargé de corriger régulièrement l'estimation relative de position. Pour une optimisation en temps de l'algorithme de navigation, le recalage de la localisation relative par une estimation de position absolue ne s'effectuera que lorsque l'imprécision sur la configuration du robot devient supérieure à un seuil prédéfini.

1.3 La cartographie

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 et l'historique des positions réelles du robot[31]. Les méthodes de modélisation sont classées en deux grandes familles[25] :

— Les méthodes de modélisation métriques, qui décrivent explicitement la position ***géométrique*** des éléments de l'environnement.

— Les méthodes de modélisation topologiques basées sur des graphes représentant des informations de plus haut niveau comme certaines places caractéristiques de l'environnement (*coins*, *croisement de deux couloirs*, *jonctions en T*... etc.).

1.3.1 Les méthodes métriques

Les méthodes de modélisation métriques intègre la notion de distance au sein de la carte. L'avantage principal des cartes métriques est de permettre de représenter l'ensemble de l'environnement, et non un petit sous-ensemble de lieux comme le font les cartes topologiques. Cette représentation complète ne se limite pas aux positions physiquement explorées, mais s'étend à toutes les zones que le robot a pu percevoir depuis les lieux qu'il a visité. Cette représentation complète permet ainsi d'estimer avec précision et de manière continue la position du robot sur l'ensemble de son environnement.

Cette catégorie peut être subdivisée en deux sous familles :

— 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.

Modélisation géométrique

Le principe de cette méthode est d'utiliser une liste ou un vecteur de tous les objets de l'environnement, cette liste contient des informations sur les objets tel que leur nature, position et orientation. Il s'agit d'une représentation cartésienne de l'environnement[25] (un exemple d'une carte géométrique est illustré dans la *figure 1.4*).

En se déplaçant, le robot prend des mesures de son environnement en utilisant son système de perception, afin d'extraire des informations sur l'éventualité de la présence d'une primitive, c'est ce qui s'appelle *extraction de primitive*. Ces

primitives peuvent êtres de différentes formes géométriques tel que *des points*, *des lignes*, *des polygones*... etc.

Une fois la carte géométrique est construite, elle peut être utilisée par le robot pour sa localisation dans l'environnement. Cette méthode de modélisation est largement utilisée dans les environnements d'intérieur structurés.

Cette solution est avantageuse pour sa grande résolution en plus du fait qu'elle ne nécessite pas une grande capacité de mémoire, la position des objets peut être stockée avec une grande précision. Quoi que l'espace mémoire requis pour le stockage augmente avec la taille de l'environnement du robot[12].

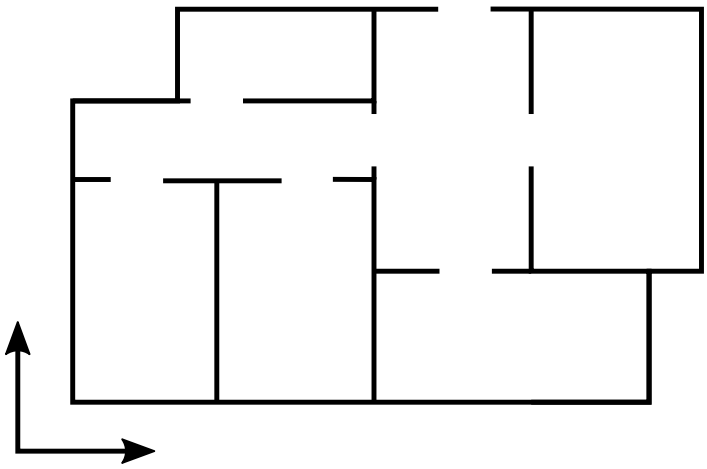


FIGURE 1.4 – Exemple de carte géométrique[39].

Grille d'occupation

La méthode de modélisation par grille d'occupation caractérise l'environnement par un ensemble de sous-régions, appelées *cellules*. Chaque cellule indique la probabilité (entre 0 et 1) de la présence d'obstacles dans la sous-région correspondante (voir la *figure 1.5*).

MORAVEC, ELFES et MATTHIES ont été parmi les premiers à utiliser le principe des grilles d'occupation[30, 55]. L'objectif de leurs travaux est de construire de manière autonome la carte de l'environnement d'un robot mobile. Pour cela, le robot évolue dans un environnement inconnu non structuré et s'y déplace en évitant les obstacles. Il doit construire une carte du lieu seulement à partir des informations données par des capteurs à ultrasons (dans leur cas) montés sur le robot.

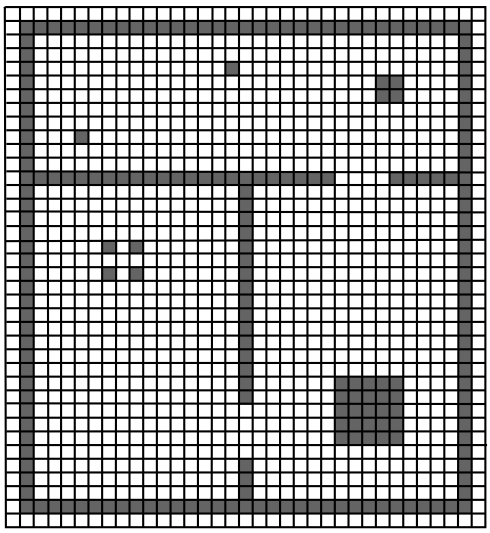


FIGURE 1.5 – Exemple de grille d'occupation binaire. Les cellules blanches correspondent à des zones de l'environnement ne contenant aucun obstacle, les cellules grises correspondent à des zones occupées par des obstacles[39].

1.3.2 Les méthodes topologiques

Les cartes topologiques[47] permettent de représenter l'environnement du robot sous forme de graphe (voir la *figure 1.6*). Les nœuds du graphe correspondent à des lieux, *i.e.* des positions que le robot peut atteindre. Les *arêtes* liant les nœuds marquent la possibilité pour le robot de passer directement d'un lieu à un autre et mémorisent en général la manière de réaliser ce passage[31].

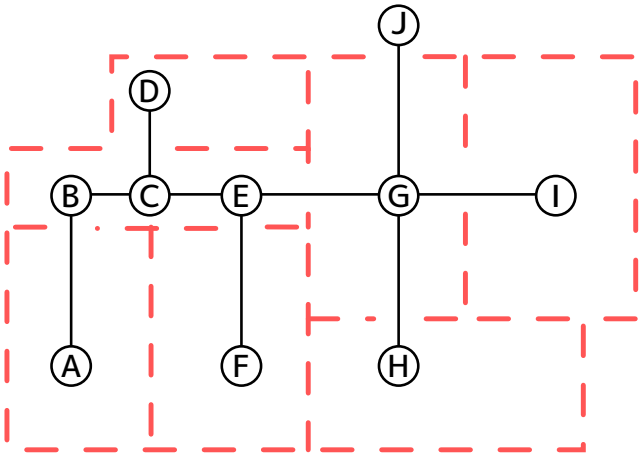


FIGURE 1.6 – Exemple de carte topologique (en noir) [39].

1.3.3 Les méthodes hybride

Les deux modélisations présentées précédemment sont complémentaires. Il est possible de considérer conjointement ces deux approches, ce qui se traduit par des modélisations hybrides. Ces dernières sont généralement des modélisations topologiques auxquelles on ajoute des données métriques. Nous pouvons par exemple noter que [68] présente une carte hybride basée sur les grilles d'occupation à laquelle il ajoute des données topologiques afin de faciliter la planification de trajectoires.

1.4 Localisation et cartographie simultanées

Un robot mobile doit savoir où il se trouve dans un environnement afin de naviguer de manière autonome et intelligente. L'auto-localisation et la connaissance de l'emplacement d'autres objets nécessitent l'existence préalable d'une carte de l'environnement[6].

Dans notre cas, l'environnement est inconnu, et sans cartographie référencée ; le robot se déplace et explore l'espace qui l'entour. Il doit, à l'aide de ses capteurs, construire au fur et à mesure de son trajet sa propre carte, et définie sa position dans cette carte. Cette opération est effectuée grâce aux algorithmes de localisation et cartographie simultanées (Simultaneous Localization and Mapping, SLAM)[67, 5, 6].

Le problème de localisation et cartographie simultanées (SLAM) traite deux questions importantes dans la robotique mobile. La première question est : *Où suis-je ?* La réponse à cette question définit ***la position*** du robot. La deuxième

question concerne les caractéristiques de l'***environnement*** du robot : *À quoi ressemble l'environnement où je me trouve ?*[28].

La localisation et la cartographie simultanées décrit le processus de la construction d'une carte d'un environnement inconnu, et de calcul de la position du robot en même temps.

Ces deux étapes (*la localisation et la cartographie*) dépendent l'une de l'autre. D'une part ; une bonne carte est nécessaire pour calculer la position du robot, et d'autre part ; une estimation précise de la position cède à une carte correcte[6].

Par conséquence le problème de localisation et de cartographie simultanés (SLAM) demande s'il est possible qu'un robot mobile soit placé à un emplacement inconnu dans un environnement inconnu et que ce robot construise progressivement une carte cohérente de cet environnement tout en déterminant son emplacement dans cette carte.

1.4.1 Bref historique du SLAM

La première formulation du SLAM fut établie par les travaux de Smith et Cheeseman [72] et de Durrant-Whyte [26]. Les bases statistiques de l'écriture mathématique reliant les amers [1] et les incertitudes de mesures sont établis. Ils montrent qu'il doit y avoir un fort degré de corrélation entre les positions estimées des amers observés, ces corrélations doivent augmenter avec les observations successives.

Les travaux de [3, 56, 4] montrent des résultats de recherche sur la navigation de robots mobiles utilisant des algorithmes exploitant le filtre de Kalman. Il en résulte la parution d'un article clef du SLAM [71]. Il montre que lorsqu'un robot se déplace dans un environnement inconnu en effectuant des acquisitions successives d'observations d'amers, les estimations des positions des amers sont corrélées entre elles.

Les conclusions de ces avancées sont qu'un fort degré de corrélation entre les amers doit être maintenu si l'on souhaite conserver une cohérence dans les résultats. Une solution est d'utiliser un vecteur d'état contenant à la fois la position du robot et celles de tous les amers qui sont mis à jour après chaque observation. Cella demande une complexité de calcul qui croît très rapidement avec le nombre d'amers.

Une étape importante de l'évolution du SLAM est franchie dans [22] où une étude théorique complète sur la convergence et la mise en œuvre du SLAM a été réalisée.

Lu et Milios [49] posent le problème comme une minimisation des erreurs sur les positions du robot par association à chacune des positions du robot une mesure

1. Cela se réfère à tous les points d'intérêt qui peuvent être observés par le robot.

extéroceptive. Cette méthode, proscrite à cause de son temps de calcul important, ne se fonde plus sur l'utilisation d'amers mais sur la mise en correspondance d'observations (nuages de points laser) permettant d'en retirer des informations de déplacement, qui est à l'origine du développement du SLAM hors-ligne qui permet de déterminer, en utilisant des techniques d'optimisation, la meilleure carte possible en connaissant toutes les observations et en utilisant les contraintes entre toutes les positions successives du robot. Thrun et al. [74] utilisent notamment une approche bayésienne très efficace lors de la localisation avec une cartographie a priori mais qui ne permet pas de fournir un SLAM incrémental.

Gutmann et Konolige [37] présentent une des premières méthodes permettant de faire la mise à jour de la carte, à chaque nouvelle observation.

Il existe une autre catégorie, les techniques qui utilisent les filtres à particules. Elles reposent sur une factorisation suivant le théorème de Rao-Blackwell [18] où chaque particule représente une trajectoire possible et une carte. Montemerlo et al. [54] utilisent cette méthode pour la construction d'une carte à base d'amers.

L'intérêt porté aux problématiques du SLAM a augmenté de manière exponentielle pendant les quinze dernières années, notamment grâce aux conférences internationales (ICRA, IROS... etc.) qui attirent de plus en plus la communauté scientifique [28]. Plus de détails sont a consulter dans [35].

1.4.2 Le principe du SLAM

De nombreuses recherches tentent de résoudre le problème de la localisation et la cartographie simultanées, les principales méthodes de SLAM sont basées sur des méthodes d'estimation statistiques. Il s'agit de filtrage statistique permettant d'estimer l'état d'un système dynamique à partir des données en provenance d'un ou plusieurs capteurs [28].

Le SLAM se décompose en plusieurs étapes :

— Une acquisition de données des divers capteurs (*Perception*).
— Une fusion des données provenant des différents capteurs (*Filtrage et analyse*).
— Une prédiction sur la nouvelle position et orientation du robot (*Estimation de l'état*).
— Une mise en correspondance de la carte en tenant compte des données capteurs et des prédictions.
— Une mise à jour de la carte globale et de la position/orientation du robot.

Cette structure du SLAM est représentée sur le schéma de la *figure 1.7*.

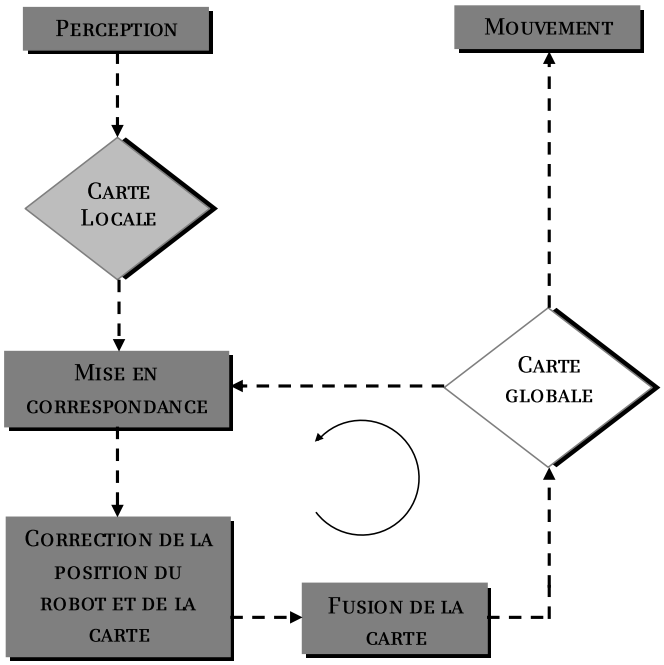


Figure 1.7 – Structure générale du SLAM.

Les algorithmes développés dans ce domaine peuvent être classés selon plusieurs critères : les types de capteurs utilisés, les méthodes de calcul adoptées, les types de cartes représentant l'environnement, les types d'environnement... etc.

1.4.3 Représentation de la carte

Le choix de la représentation de la carte de l'environnement est une étape importante dans le SLAM qui se fait selon l'environnement, La majorité des travaux de recherche se concentre sur le SLAM par des systèmes robotisés terrestres dans des environnements internes (*Indoor*) et externe (*Outdoor*)[28].

Selon [75], trois approches fondamentales de représentation de l'environnement sont utilisées dans les SLAM :

— L'approche directe[50, 60, 52, 69].
— L'approche indirect basée sur les caractéristiques géométriques (feature-based) [46, 82, 7, 57, 65].
— L'approche indirect basée sur une grille d'occupation (grid-based) [29, 30, 79, 61, 68, 66].

L'approche directe

La méthode de représentation directe de la carte de l'environnement est généralement adaptée à l'utilisation des capteurs *métrique*. Cette méthode utilise les données brutes des mesures du capteur pour représenter l'environnement sans aucune extraction d'amers ou de caractéristiques prédéfinies (lignes, coins...). Dans le cas d'un capteur laser, chaque mesure est constituée d'un ensemble de points d'impact du faisceau laser sur les objets de l'environnement. Il est possible de construire une carte simplement en superposant les différents points de mesure. On obtient ainsi une représentation en nuage de points. La *figure 1.8* montre un exemple d'une carte en nuage de points [28].

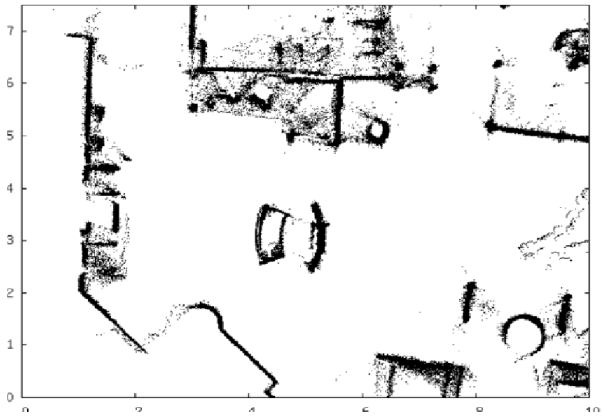


FIGURE 1.8 – L'approche directe, une carte en nuage de points[28].

Les approches indirect

Ce sont les méthodes géométrique qui réduit les données des mesures en caractéristiques prédéfinies, ils utilisent généralement des primitives géométriques, comme des lignes, des cercles ou des coins. La *figure 1.9* illustre une représentation d'un environnement par une méthode géométrique.

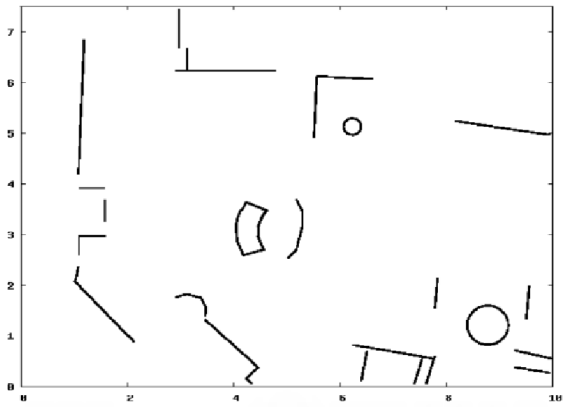


FIGURE 1.9 – L'approche géométrique (feature-based) du SLAM[28].

Et les grilles d'occupation dont la représentation de l'environnement est divisé en cellules rectangulaires.La mise à jour de l'état de chaque cellule se fait à la réception de nouvelles données (voir la *figure 1.10*), ces deux méthodes (*géométrique* ou par *grille d'occupation*) appartiennent à la famille des cartes métriques que nous l'avons détaillées dans la section 1.3.1 de ce chapitre.

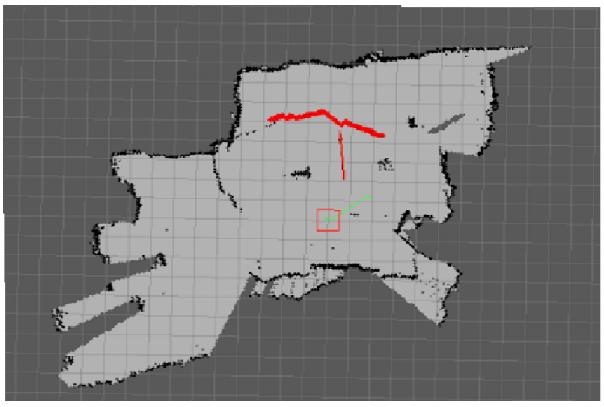


FIGURE 1.10 – L'approche grid-based[28].

1.4.4 Algorithmes pour SLAM

Filtre de Kalman

Dans [41], KALMAN propose une solution récursive au filtrage des données linéaires, Elle permet d'estimer successivement après chaque acquisition des mesures la position du robot. Lorsque la carte globale de l'environnement est construite au

fur et à mesure de l'exploration du robot, ce filtre permet aussi d'estimer la position de chaque élément géométrique introduit dans la carte. Cette méthode, améliorée ensuite par KALMAN lui-même et BUCY[15], ouvre de nouvelles pistes de recherche dans le domaine de la navigation autonome des robots mobiles. L'approche de base du filtre de KALMAN est basée sur un cycle récursif nécessitant trois hypothèses pour assurer un fonctionnement, prouvé théoriquement, optimal :

— Un modèle d'évolution linéaire du système.
— Une relation linéaire entre l'état et les mesures.
— Un bruit blanc gaussien.

Le cycle du filtre de KALMAN est constitué de deux étapes fondamentales (voir la *figure 1.11*) :

— Étape de prédiction : L'estimation de l'état du système à l'instant t en utilisant l'estimation corrigée de l'instant $(t-1)$.
— Étape de correction : durant laquelle se fait la correction l'estimation de l'état du système à l'instant t en utilisant les informations sensorielles reçu à l'instant t.

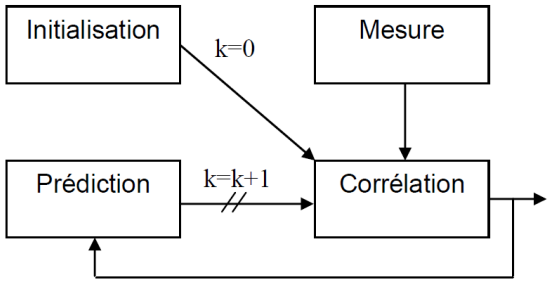


FIGURE 1.11 – Le cycle du filtre de KALMAN basé sur les deux étapes récursives : Prédiction et Correction[12]

Le filtre de KALMAN étendu[81] (Extented KALMAN Filter) (EKF) est une amélioration du filtre de KALMAN basique qui consiste à linéariser les modèles non linéaires afin que les conditions requises pour appliquer le filtre de KALMAN soient satisfaites. La première implémentation de l'EKF a été faite par STANLEY SCHMIDT. D'ou son ancien nom filtre de KALMAN-SCHMIDT.

Filtre particulaire

Un filtre particulaire est un filtre récursif qui permet d'estimer l'état a postériori en utilisant un ensemble de particules. Les origines de ce type de filtres remontent

à [53], mais le rapprochement entre le filtrage particulaire et le monde du SLAM est apparu avec les articles [11] et [62].

Contrairement aux filtres paramétriques comme le filtre de KALMAN, un filtre particulaire représente une distribution par un ensemble d'échantillons créés à partir de cette distribution. Un filtre particulaire est ainsi capable de traiter les systèmes fortement non linéaires avec un bruit non gaussien.

La complexité des calculs du filtrage particulaire augmente exponentiellement avec le nombre d'amers de l'environnement, ce qui constitue un problème majeur dans le cadre d'une application temps-réel. Afin de résoudre ce genre de problèmes, certains travaux de recherche combinent le filtrage particulaire avec d'autres méthodes. C'est le cas des travaux de MONTEMERLO dans FastSLAM [54].

L'utilisation du filtrage particulaire souffre également des difficultés rencontrées lors de la définition du nombre de particules. En effet, la qualité de l'estimation est fortement corrélée à la discrétisation de l'espace de recherche. Mais il est difficile de trouver un nombre optimal de particules.

Scan Matching

Cette approche consiste à rechercher à chaque instant la meilleure correspondance entre l'observation courante "*carte t*" (les données provenant du laser) et la "*carte* $t-1$" (combinant la connaissance de l'environnement), et à remettre à jour la carte conformément à cette mise en correspondance. Cela revient à estimer la meilleure combinaison de translation/rotation permettant aux données laser de s'ajuster avec la forme des données de la partie de la carte correspondante. L'idée générale de cette approche est présentée dans le schéma de la *figure 1.12* :

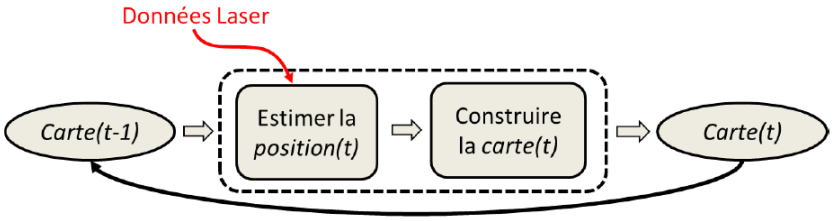


FIGURE 1.12 – Principe générale du scan-matching[28].

1.4.5 Comparaison entre les algorithmes du SLAM

L'intérêt majeur des approches basées sur le filtrage de KALMAN est la possibilité d'estimer l'état du système a posteriori en se basant sur les amers de l'environnement et les positions du robot. Leur grande faiblesse vient des contraintes et

des hypothèses fortes quelle doivent être appliquer aux différents modèles utilisés. En plus, il n'est pas toujours facile d'extraire des amers corrects et intéressants dans un environnement non-structuré ou externe.

L'utilisation d'un filtre particulaire Rao-Blackwellisé permet de maintenir une estimation a posteriori de l'état du système d'une manière plus rapide que le KF. Le filtrage particulaire peut également être utilisé avec des cartes *grid-based* ou *feature-based*. Cette méthode souffre néanmoins de plusieurs problèmes à cause de sa complexité grandissantes et ses difficultés de paramétrage.

Le scan matching reste une solution intéressante grâce à sa simplicité et son efficacité en temps de calcul. En plus, il peut être appliqué à tous les types de cartes. Pour cela nous avons choisi d'utiliser le scan matching dans notre algorithme. Le tableau *figure 1.1* montre une comparaison entre ces différentes approches.

Table 1.1 – Comparaison entre les algorithmes du SLAM[28].

	Avantages	Inconvénients
Filtre de Kalman	- Prise en compte des incertitudes. - Convergence prouvée. - Fermeture de boucle.	- Hypothèse forte. - Complexité. - Problème d'association des données. - Utilisation possible sur un seul type de carte.
Filtre particulaire	- Elimination des hypothèses du KF. - Fermeture de boucle.	- Complexité. - Paramétrage difficile.
Scan matching	- Concept simple et rapide. - Tous les types de cartes.	- Divergence par construction. - Pas de fermeture de boucle.

La majorités des approches de scan matching sont basées sur l'utilisation des capteurs laser qui fournissent un ensemble de points spatiaux à partir d'une surface. On outre, plusieurs algorithme utilises les données brutes du capteur qui sont des nuages de point.

Cependant, l'utilisation de nuages de points pour représenter les environnements présente de nombreuses limitations. Par exemple, les nuages de points ne contiennent pas d'informations explicites sur les caractéristiques de surface telles que l'orientation, la douceur, les trous... etc.

Selon la configuration du capteur, les nuages de points peuvent être inefficaces, nécessitant une quantité de stockage inutilement importante. Pour obtenir une résolution d'échantillon suffisamment loin de l'emplacement du capteur, il est généralement nécessaire de configurer le capteur de manière à produire une grande quantité de données redondantes à partir des surfaces proches du capteur.

1.4.6 Différentes méthodes basées sur le Scan Matching

Le *scan matching* est un processus important dans plusieurs domaines. Il est utilisé pour la construction de modèles à partir d'analyses partielles dans des disciplines aussi diverses que l'imagerie médicale, l'archéologie, la robotique... etc. Il est également utile pour permettre la localisation du robot mobile. Plusieurs algorithme dans ce domaine sont basés sur le scan matching parmi les plus connus, l'algorithme itératif du point proche (Iterative Closest Point, ICP), la double correspondance itérative (Iterative Dual Correspondences, IDC), la transformation de distribution normale (Normal Distribution Transform, NDT) et la méthode de champ de vraisemblance (Likelihood-field matching, LF).

Itératif du point proche (ICP)

L'ICP est largement utilisé pour l'alignement des nuages de points. Il a été introduit en 1992 par Besl et McKay[8].

Cette technique de SLAM est basée sur le scan matching, cet algorithme permet le recalage entre deux données géométriques (dans ce cas des nuages points), les étapes de l'algorithme sont :

- La sélection des points à aligner.
- L'association des points.
- La pondération des paires de points obtenues pour estimer la transformation de repère.
- Le rejet des mauvaises associations (points aberrants).

Généralement la sélection de points se fait aléatoirement et pour l'association la distance euclidienne entre chaque point est calculée et la plus petite distance permet de les associer[8].

Puis vienne l'étape de la pondération où chaque paire de points est attaché à un poids qui permet de déterminer si l'association est juste, la valeur 1 correspond donc à une association correcte et 0 à une fausse association. Ces points seront rejetés pour éviter une fausse estimation, l'organigramme de la *figure 1.13* représente les grandes étapes de l'ICP.

Pour pouvoir faire l'association de points, on doit disposer initialement de deux scans consécutifs et chaque scan sera projeté sur le précédent pour pouvoir les comparer, une matrice de transformation homogène est nécessaire pour cela, et pour finir le seuil de sortie de la boucle est un seuil de nombre d'itérations d'après les tests effectuée (25 itérations) permettent de déduire une position correcte, la méthode détaillées de l'ICP est expliquée sur [49].

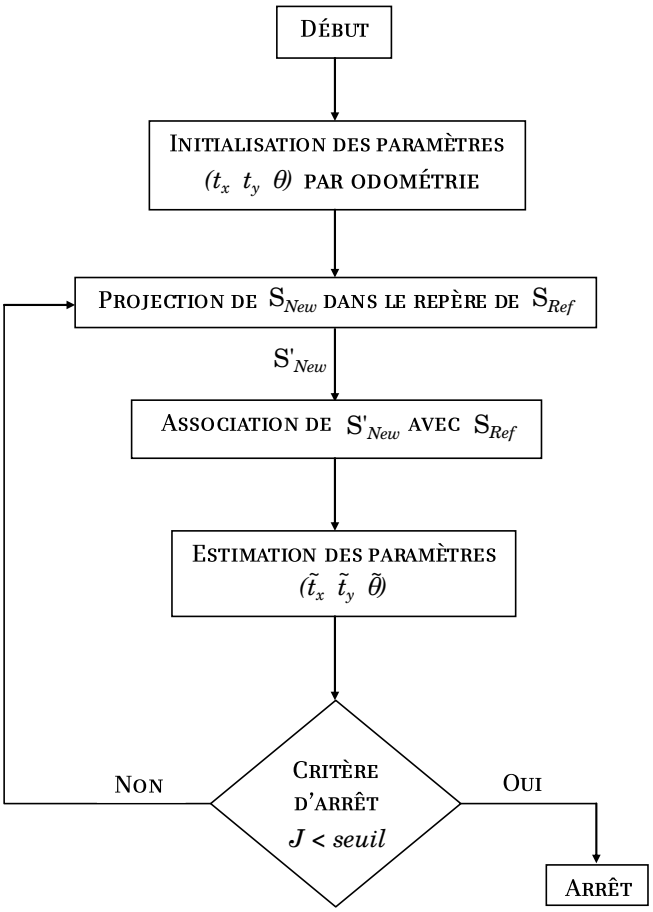


FIGURE 1.13 – L'organigramme général d'ICP.

La transformation de distribution normale (NDT)

La NDT est une méthode qui consiste à associer deux scan laser 2D l'un par rapport à l'autre. Il s'agit également de faire correspondre plusieurs scans lasers l'un par rapport à l'autre, en utilisant uniquement les résultats de la correspondance par paire dans une étape d'optimisation globale. Nous pouvons appliquer cet algorithme au suivi de position (Position Tracking) (*i.e.* estimer le mouvement d'un objet mobile) et au problème de localisation et de cartographie simultanée (SLAM) [10]. Semblable à une grille d'occupation, la NDT établit une subdivision régulière de plan. Mais là où la grille d'occupation représente la probabilité d'une cellule sont occupés, la NDT représente la probabilité de mesure d'un échantillon pour chaque position dans la cellule.

FIGURE 1.14 – NDT des points du scan laser.

Correspondance par champ de vraisemblance (LF)

En 2008, une autre méthode d'alignement semblable à NDT, nommée correspondance par champ de vraisemblance (Likelihood-field matching, LF) a été présentée par BURGUERA, GONZÁLEZ et OLIVER[16].

BURGUERA et al. sont principalement concernés par l'alignement des scans sonar 2D. La faible résolution angulaire des sonars par rapport aux LiDARs est un problème lors de l'obtention des correspondances point à point pour l'alignement des scans. Par conséquent, un certain nombre de scans consécutifs sont agrégés à l'aide de l'odométrie du robot afin de générer des scans avec plus de points. Cette addition peut être appliquée à n'importe quel algorithme d'alignement des scans, et BURGUERA et al. Ont introduit une nouvelle famille d'algorithmes avec cette addition : sNDT, sICP, et ainsi de suite[17].

Leur algorithme, appelé LF/SoG (Likelihood-Field defined as a Sum of Gaussians), est tout à fait semblable à NDT en ce que les points d'un scan sont adaptés aux distributions normales en fonction des points de l'autre scan. La différence principale est que la matrice d'identité est utilisée comme estimation de covariance au lieu de la matrice de covariance réelle des points voisins. Au lieu d'une discrétisation explicite de la grille, BURGUERA et al. rééchantillonnons le scan de référence en déplaçant une fenêtre circulaire sur l'analyse, en substituant les lectures à l'intérieur de la fenêtre par leur centre de gravité. Un Gaussien est placé à chaque point du scan rééchantillonné avec le vecteur moyen en position des points et la matrice d'identité comme estimation de la covariance. Au cours de l'alignement, tous les Gaussiens à une certaine distance sont utilisés lors de l'évaluation de la fonction de conditionnement physique en un point, et pas seulement le plus proche.

MAGNUSSON[51] dit qu'il n'est pas clair pourquoi il serait préférable d'utiliser la matrice d'identité au lieu d'une matrice de covariance calculée à partir de la distribution des points voisins. Son hypothèse est que le résultat s'explique par les différentes échelles de discrétisation de NDT et LF/SoG utilisées dans le document,

mais il reste à tester[51].

1.5 Avantages de la NDT par rapport aux autres approches

L'objectif de faire correspondre deux scans est de trouver la transformation relative entre les deux positions, où les scans ont été pris. La base des algorithmes les plus réussis est l'établissement de correspondances entre les *primitives des deux scans*. En dehors de cela, une mesure d'erreur peut être dérivée et minimisée. Cox a utilisé des points comme primitives et les a adaptés à des lignes fournies dans un modèle a priori[21]. Dans le projet Amos[38], ces lignes ont été extraites à partir des scans.

Une approche plus générale qui consiste à faire correspondre deux ensembles de points, a été introduite dans [49]. Il s'agit essentiellement de l'algorithme ICP (Iterated Closest Point)[8] appliqué à la mise en correspondance des scans laser.

La NDT partage avec plusieurs travaux (Lu et Milios [50]) la stratégie de cartographie, au lieu de construire une carte explicite, une collection de scans sélectionnées avec leur positions est utilisé comme carte implicite.

Toutes fois il existe également d'autres approches, qui évitent de résoudre le problème de la correspondance. Elle sont basées sur l'histogramme. Weiss et Puttkammer [77] ont utilisé des histogrammes angulaires pour récupérer la rotation entre deux poses. Par la suite, les histogrammes x et y, après avoir trouvé la meilleur direction sont été utilisés pour récupérer la translation.

1.6 Conclusion

Dans ce chapitre, nous avons étudié une partie de l'état de l'art du SLAM. Nous avons commencé par définir la cartographie et la localisation. Ensuite nous avons donné une présentation de la problématique générale de la localisation et la cartographie simultanées. Nous avons commencé par la section qui décrit les représentations de l'environnement les plus utilisées dans les SLAM. Puis une section pour la présentation des trois algorithmes les plus utilisés pour la résolution du SLAM. Le filtre de Kalman, le filtrage particulaire et enfin le scan matching qu'on va utilisé pour résoudre notre problématique. Ensuite une section pour présenter quelques algorithmes qui sont basés sur le scan matching. Il parait qu'aucune approche n'arrive à réaliser des cartes de SLAM exactes pour de grands espaces ou pour la navigation externe (outdoor), principalement à cause du coût de calcul élevé et des incertitudes. Il s'agit donc d'une problématique qui nécessite de l'amélioration. Certaines approches cherchent à résoudre cette problématiques en

combinant entre les algorithme et en utilisant plusieurs cartes de l'environnement, ou des cartes locales qu'on regroupe ensuite pour créer une carte globale. Nous avons choisi la NDT-SLAM (Normal Distributions Transform) que nous avons améliorée pour réaliser cette étape qu'on va étudier en détails dans le chapitre suivant.

Chapitre 2

Normal Distribution Transform

"Bien que cela puisse paraître paradoxal, toute science exacte est dominée par la notion d'approximation"

Bertrand Russell

La transformation de distribution normale (Normal Distribution Transform, NDT) peut être décrite comme une méthode pour représenter de façon compacte une surface. Cette méthode à été proposé par Biber et Strasser en 2003 [10] comme une méthode pour l'alignement des scans 2D. Biber et Strasser ont ensuite développé la méthode dans un document commun avec Sven Fleck [9], également dans le contexte de l'alignement et de la cartographie. Dans leurs travail ; le nuage de points est transformé en une représentation de surface lisse, décrite comme un ensemble de fonctions locales de densité de probabilité (PDF), chacune décrivant la forme d'une section de surface[51].

2.1 Représentation par des densités de probabilités

Cette section décrit la transformation de distribution normale (NDT) d'un seul scan laser. Ce même principe peut être utilisé lors du suivi de position (Position Tracking) et du SLAM.

La NDT modélise la distribution de tous les points 2D reconstruits à partir d'un scan laser par une collection des distributions normales locales. Tout d'abord, l'espace 2D autour du robot est subdivisé régulièrement en cellules avec une taille constante (carte locale). Ensuite, pour chaque cellule, qui contient au moins trois points, Nous effectuons les opérations suivantes :

2.1.1 Collection des points 2D

Cette phase consiste à collecter tous les points 2D $X_{i=1...n}$ contenus dans cette cellule (n points).

Chaque point X est représenté par ses coordonnées cartésienne, soit :

$$X_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix}_{i=1...n}$$

Calcule de la moyenne

La moyenne des points présents à l'intérieur de la cellule est exprimée par :

$$Q = \begin{pmatrix} q_x \\ q_y \end{pmatrix} = \frac{1}{n}\sum_{i=1}^{n} X_i \tag{2.1}$$

Calcule de la covariance

Calculer la matrice de covariance

$$\Omega = \frac{1}{n}\sum_{i=1}^{n}(X_i - Q)(X_i - Q)^t \tag{2.2}$$

Dans le cas 2D, la covariance est une matrice 2×2 :

$$\begin{aligned}
\Omega &= \frac{1}{n}\sum_{i=1}^{n}(X_i - Q)(X_i - Q)^t \\
&= \frac{1}{n}\sum_{i=1}^{n}\begin{pmatrix} x_i - q_x \\ y_i - q_y \end{pmatrix}\begin{pmatrix} x_i - q_x & y_i - q_y \end{pmatrix} \\
&= \frac{1}{n}\sum_{i=1}^{n}\begin{pmatrix} (x_i - q_x)^2 & (x_i - q_x)(y_i - q_y) \\ (y_i - q_y)(x_i - q_x) & (y_i - q_y)^2 \end{pmatrix}
\end{aligned}$$

Après avoir calculé les moyennes et les covariances des cellules, la probabilité de mesurer un échantillon X (un point 2D) contenu dans cette cellule est modélisée par la distribution normale Π :

$$\Pi(X) = \exp\left(-\frac{(X - Q)^t\Omega^{-1}(X - Q)}{2}\right) \tag{2.3}$$

Semblable à une grille d'occupation, la NDT établit une subdivision régulière du plan de l'espace de travail. Cependant là où la grille d'occupation représente la probabilité d'occupation d'une cellule, la NDT représente la probabilité de mesure pour chaque position d'échantillon dans la cellule. Nous utilisons une taille de cellule de $1m$ sur $1m$ (plus de détails sont fournies dans le troisième chapitre.

A ce niveau la question qui se pose est : *Quel est l'utilité de cette représentation ?* Nous disposons désormais d'une description continue et différentielle du plan 2D par morceaux !

2.2 Limitations numériques

La fonction (2.3) nécessite l'inverse de la matrice de covariance Ω^{-1}. Dans le cas où les points dans une cellule sont parfaitement alignés, la matrice de covariance est singulière et ne peut pas être inversée. Dans le cas 2D, une matrice de covariance calculée à partir de deux points ou moins sera donc toujours singulière. Pour cette raison, les PDFs sont calculés uniquement pour les cellules contenant plus de trois points.

En outre, pour remédier a ces problèmes numériques, Ω est légèrement gonflée chaque fois qu'il se révèle être presque singulière.

Si la plus grande valeur propre λ_2 de Ω est supérieure plus que 1000 fois à la valeur propre inférieure λ_1, alors λ_1 est remplacée par λ_1' tel que [51] :

$$\lambda_1' = \frac{\lambda_2}{1000}.$$

La matrice $\Omega' = V\Lambda' V$ est utilisée à la place de Ω, avec V contenant les vecteurs propres de Ω et

$$\Lambda' = \begin{pmatrix} \lambda_1' & 0 \\ 0 & \lambda_2 \end{pmatrix}$$

La *figure 2.1* contient quelques exemples des cellules NDT individuelles. Les points 2D et la densité de probabilité sont représentés. La visualisation est créée en évaluant la densité de probabilité pour chaque point de la cellule, les zones lumineuses indiquant des densités de probabilité élevées. La *figure 2.2* montre l'exemple d'un scan laser entier à un instant donné. La section suivante montre comment cette transformation est utilisée pour aligner deux scans laser.

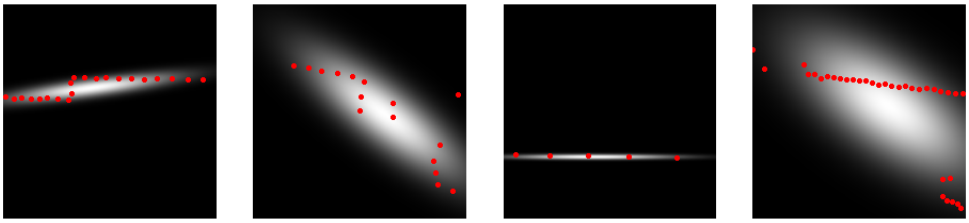


FIGURE 2.1 – Représentation des cellules par des densités de probabilité (Représenté par le nœud `ndt_slam` que nous avons réalisé).

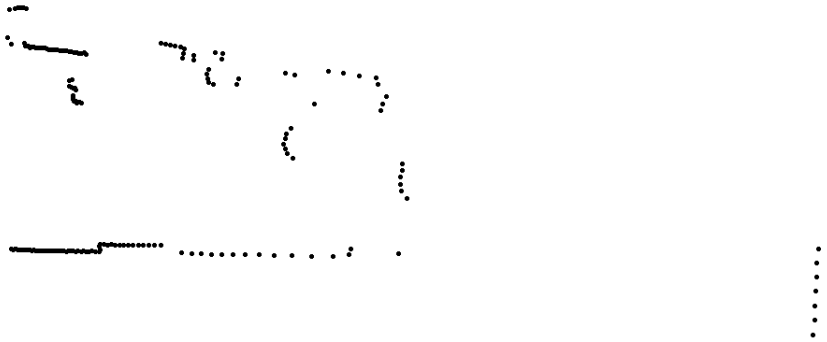
2.3 L'alignement avec NDT

La transformation géométrique T entre deux repères du robot est une rotation et translation, sa formule est donnée par :

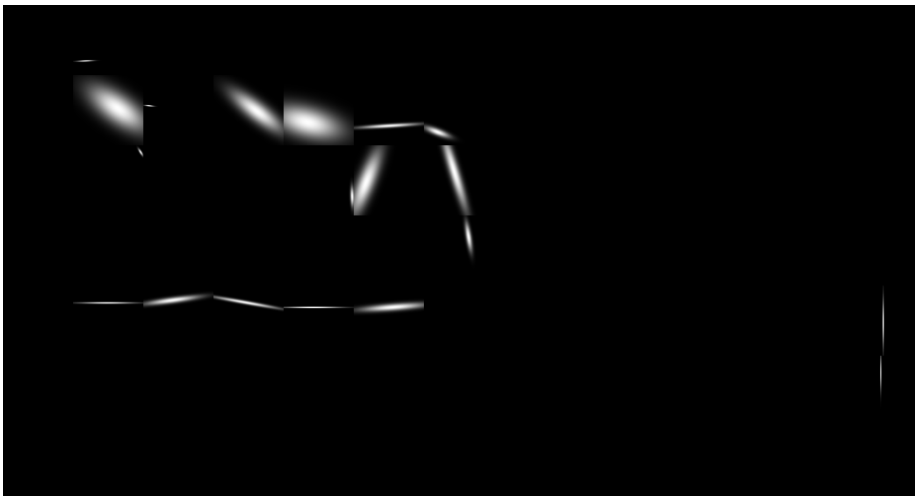
$$T: \begin{pmatrix} x' \\ y' \end{pmatrix} \longmapsto \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix} \tag{2.4}$$

Où $\begin{pmatrix} t_x & t_y \end{pmatrix}^t$ décrit la translation et ϕ la rotation entre les deux scans correspond à chaque repère. L'objectif de l'alignement d'un scan (*Scan alignment/Registration*) est de récupérer ces paramètres (t_x, t_y et ϕ) à l'aide des scans laser effectués à deux positions différente (deux scans successives). Le processus de l'approche proposée dans la NDT (compte tenu de deux scans ; le premier considéré comme scan de référence et le deuxième) est le suivant :

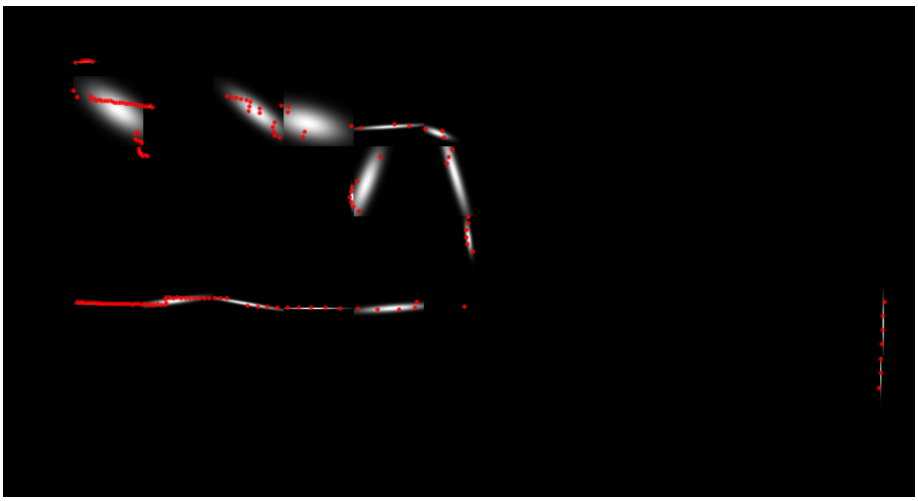
1. Construire la transformation de distribution normale du première scan.
2. Initialiser l'estimation des paramètres (t_x, t_y et ϕ) (par zéro ou en utilisant les données de l'odométrie).
3. Pour chaque point du deuxième scan :
   — Transformer le point 2D au repère de scan de référence en fonction des paramètres (t_x, t_y et ϕ) en utilisant l'équation 1.4.
   — Déterminer les distributions normales correspondantes pour chaque point transformé en utilisant l'équation 2.3.
4. Le *score* pour les paramètres (t_x, t_y et ϕ) est déterminé en évaluant la distribution pour chaque point transformé et en additionnant le résultat.



(a) Les points récupérés à partir d'un scan laser à un instant donné



(b) Les densités de probabilité pour chaque cellule de scan contenant plusieurs points



(c) La superposition des deux (points et densités)

FIGURE 2.2 – Représentation d'un scan laser par des densités de probabilité (Représenté par le nœud `ndt_slam` que nous avons réalisé).

5. Calculez une nouvelle estimation des paramètres en essayant d'optimiser le score. Cela se fait en effectuant une étape de l'algorithme de NEWTON.
6. Revenir à l'étape 3 jusqu'à ce qu'un critère de convergence soit satisfait.

Les trois premières étapes sont simples : la construction du NDT a été décrite dans les équations 2.1 et 2.2. Les données d'odométrie pourraient être utilisées pour initialiser l'estimation. La transformation du deuxième scan est effectuée en utilisant T (équation 2.4) et la distribution normale correspondante est une recherche en utilisant l'équation 2.3 dans la grille NDT construite.

Le reste est maintenant décrit en détail en utilisant la notation suivante :

— $p = (p_k)_{k=1...3}^t = \begin{pmatrix} t_x & t_y & \phi \end{pmatrix}^t$: Le vecteur des paramètres à estimer.
— X_i : Le point 2D de l'échantillon i du nouveau scan laser (correspondant a l'instant actuel).
— X_i' : Le point X_i transformé selon le vecteur de paramètre p au repère de scan précédent (scan de référence), *i.e.* $X_i' = T(X_i, p)$.
— On fait correspondre chaque point du nouveau scan X_i' à sa cellule dans le scan de référence.
— Ω_i et Q_i : La matrice de covariance et la moyenne de la cellule dans la quelle se trouve le point X_i' dans le scan de référence.

La transformation selon le vecteur p pourrait être considérée comme optimale, si la somme évaluant les distributions normales de tous les point X_i' avec les paramètres Ω_i et Q_i est un **maximum**. Nous appelons cette somme le "*score*" de p. Il est défini comme suit :

$$\textbf{score}(p) = \sum_{i=1}^{n} \Pi(X_i') = \sum_{i=1}^{n} \exp\left(-\frac{(X_i' - Q_i)^t \Omega_i^{-1} (X_i' - Q_i)}{2}\right) \tag{2.5}$$

Ce "*score*" sera optimisé dans la section suivante.

2.4 Le processus d'optimisation avec la méthode de Newton

Le fonctionnement de la NDT dépend de la maximisation du "*score*" (fonction 2.5). La méthode d'optimisation utilisée dans la NDT d'origine proposée dans [10] est la methode de NEWTON.

Étant donné que les problèmes d'optimisation sont généralement décrits comme des problèmes de minimisation, nous adopterons notre notation à cette convention. Ainsi, la fonction à minimiser dans cette section est la $-$**score**. L'algorithme de NEWTON cherche itérativement les paramètres $p = (p_k)_{k\in[1,3]}^t$ qui minimisent une

fonction f. Chaque itération résout l'équation suivante :

$$H\Delta p = -g \tag{2.6}$$

Où g est le gradient transposé de f avec les entrées

$$g_k = \frac{\partial f}{\partial p_k} \quad \text{avec } k \in [1,3] \tag{2.7}$$

Et H est la matrice Hessienne de f avec des entrées

$$H_{kj} = \frac{\partial f}{\partial p_k \partial p_j} \quad \text{avec } (k,j) \in [1,3] \times [1,3] \tag{2.8}$$

La solution de ce système linéaire est une différence Δp, ajoutée à l'estimation actuelle :

$$p \leftarrow p + \Delta p \tag{2.9}$$

Cette étape est une étape dans une direction de descente, à condition que la matrice Hessienne H soit définit positive. Si ce n'est pas le cas, l'approche du modèle de région de confiance propose de remplacer H par $H' = H + \lambda I$, avec λ choisis de sorte que H' soit définit positive.

Cet algorithme est maintenant appliqué à la fonction $-$**score** (équation 2.5). Le gradient et la matrice Hessienne sont construits en collectant les dérivées partielles de tous les sommets de l'équation 2.5.

Pour une notation plus courte, l'indice de l'échantillon du scan laser i est supprimé, nous écrivons alors :

$$Q = X'_i - Q_i$$

Nous pouvons vérifier facilement que les dérivées partielles de Q par rapport à p sont égales aux dérivées partielles de X'_i par rapport à p.

$$\frac{\partial Q}{\partial p_k} = \frac{\partial X'_i - Q_i}{\partial p_k} \tag{2.10}$$

$$= \frac{\partial X'_i}{\partial p_k} - \frac{\partial Q_i}{\partial p_k} \tag{2.11}$$

Et comme la moyenne est constante pour la cellule pendant toutes les itérations, sa dérivée est donc nulle

$$\frac{\partial Q_i}{\partial p_k} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Et alors l'équation 2.11 donne

$$\frac{\partial Q}{\partial p_k} = \frac{\partial X_i'}{\partial p_k} \tag{2.12}$$

Un élément de la somme de la fonction $-$**score** (2.5) est alors donné par :

$$s = -\exp\left(-\frac{(X'-Q)^t\Omega^{-1}(X'-Q)}{2}\right) \tag{2.13}$$

$$= -\exp\frac{-Q^t\Omega^{-1}Q}{2} \tag{2.14}$$

Les composants du gradient g sont alors :

$$\begin{aligned} g_k &= \frac{\partial s}{\partial p_k} \quad \text{avec } k \in [1,3] \\ &= \frac{\partial s}{\partial Q}\frac{\partial Q}{\partial p_k} \\ &= Q^t\Omega^{-1}\frac{\partial Q}{\partial p_k}\exp\frac{-Q^t\Omega^{-1}Q}{2} \end{aligned}$$

Les dérivées partielles de Q par rapport à p sont données par la matrice Jacobienne J_T de T (voir équation 2.4) :

$$J_T = \begin{pmatrix} 1 & 0 & -x\sin\phi - y\cos\phi \\ 0 & 1 & x\cos\phi - y\sin\phi \end{pmatrix} \tag{2.15}$$

Les entrées de la matrice Hessienne H sont données par :

$$\begin{aligned} H_{kj} &= \frac{\partial f}{\partial p_k \partial p_j} \\ &= -\exp\frac{-Q^t\Omega^{-1}Q}{2}((Q^t\Omega^{-1}\frac{\partial Q}{\partial p_k})(-Q^t\Omega^{-1}\frac{\partial Q}{\partial p_j}) + Q^t\Omega^{-1}\frac{\partial^2 Q}{\partial p_k \partial p_j} + (\frac{\partial Q}{\partial p_j})^t\Omega^{-1}\frac{\partial Q}{\partial p_k}) \end{aligned}$$

Les deuxièmes dérivées partielles de Q sont (voir l'équation 2.15) :

$$\frac{\partial^2 Q}{\partial p_k \partial p_j} = \begin{cases} \begin{pmatrix} -x\cos\phi + y\sin\phi \\ -x\sin\phi - y\cos\phi \end{pmatrix} & \text{si } \; k = j = 3 \\ \begin{pmatrix} 0 \\ 0 \end{pmatrix} & \text{sinon} \end{cases} \tag{2.16}$$

L'illustration de la *figure 2.3* représente l'organigramme de la méthode NDT.

2.4.1 Problèmes de méthode de Newton

L'inconvénient majeur de la méthode est sa sensibilité au choix de point de départ. Si ce point est mal choisi ("loin" de la solution) la méthode peut soit diverger, soit converger vers une autre solution (minima local).

Pour éviter ce problème, nous avons proposé de remplacer la méthode de Newton par l'optimisation par essaim particulaire, qui est une méthode évolutionnaire méta-heuristique très efficace[27, 19].

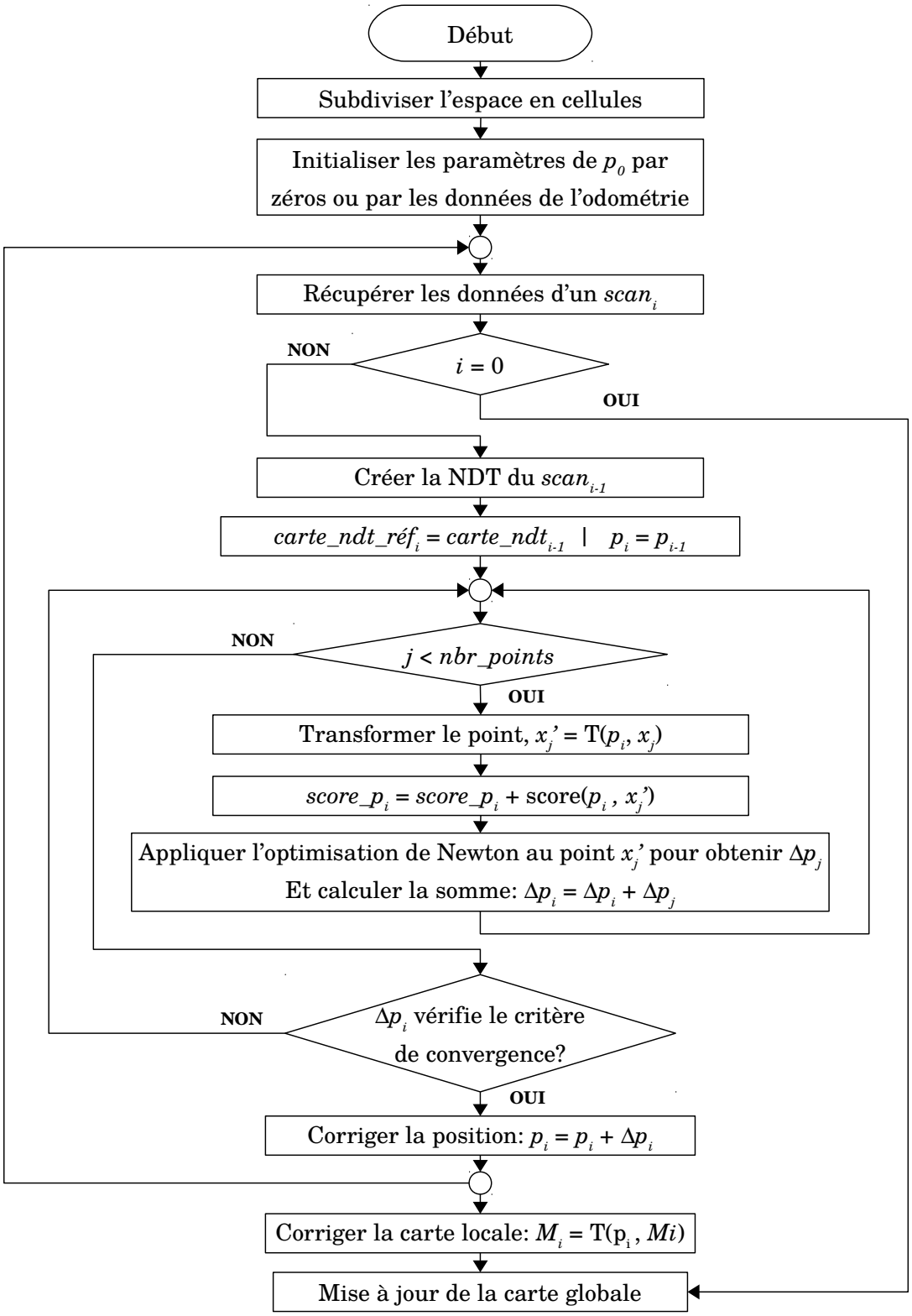


FIGURE 2.3 – L'organigramme de la méthode NDT.

2.5 Optimisation par essaim particulaire

L'optimisation par essaim particulaire (Particle Swarm Optimization, PSO), est un algorithme évolutionnaire qui utilise une population de solutions candidates pour développer une solution optimale au problème. Cet algorithme a été proposé par Russel Eberhart (ingénieur en électricité) et James Kennedy (socio-psychologue) en 1995 [44]. Il s'inspire à l'origine du monde des vivants, plus précisément du comportement social des animaux évoluant en essaim, tels que les bancs de poissons et les vols groupés d'oiseaux (voir la *figure 2.4*).

Figure 2.4 – La méthode PSO est inspirée du comportement social des animaux évoluant en essaim.

En effet, nous pouvons observer chez ces animaux des dynamiques de déplacement relativement complexes, alors qu'individuellement chaque individu a une *"intelligence"* limitée, et ne dispose que d'une connaissance locale de sa situation dans l'essaim. L'information locale et la mémoire de chaque individu sont utilisées pour décider de son déplacement. Des règles simples, telles que *"rester proche des autres individus"*, *"aller dans une même direction"* ou *"aller à la même vitesse"*, suffisent pour maintenir la cohésion de l'essaim, et permettent la mise en œuvre de comportements collectifs complexes et adaptatifs.

2.6 L'idée de PSO

L'essaim de particules correspond à une population d'agents simples, appelés particules. Chaque particule est considérée comme une solution du problème, où elle possède une **position** (le vecteur solution) et une **vitesse**.

De plus, chaque particule possède une mémoire lui permettant de se souvenir de **sa meilleure performance** (en position et en vitesse) et de la meilleure performance atteinte par les particules *"voisines"* (informatrices). Chaque particule dispose en effet d'un groupe d'informatrices, historiquement appelé son voisinage[27].

Un essaim de particules, qui sont des solutions potentielles au problème d'optimisation, *survole* l'espace de recherche, à la recherche de l'*optimum global*. Le déplacement d'une particule est influencé par les trois composantes suivantes :

1. **Une composante d'inertie :** la particule tend à suivre sa direction courante de déplacement.
2. **Une composante cognitive :** la particule tend à se diriger vers le meilleur site par lequel elle est déjà passée.
3. **Une composante sociale :** la particule tend à se fier à l'expérience de ses congénères et, ainsi, à se diriger vers le meilleur site déjà atteint par ses voisins.

La stratégie de déplacement d'une particule est illustrée dans la *figure 2.5*.

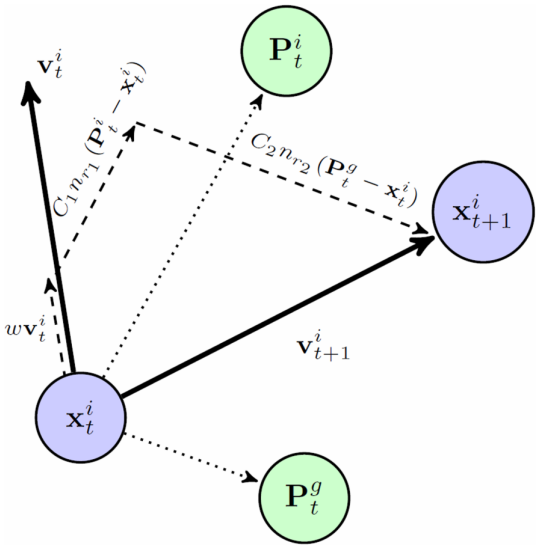


FIGURE 2.5 – La stratégie PSO de déplacement d'une particule.

2.7 Formalisation

Dans un espace de recherche de dimension $\mathcal{D}$, la particule i de l'essaim de $\mathcal{N}$ particules est modélisée par son vecteur position

$$\overrightarrow{x^i} = \begin{pmatrix} x^{i,1} & x^{i,2} & \dots & x^{i,\mathcal{D}} \end{pmatrix}_{i \in [1,\mathcal{N}]} \tag{2.17}$$

et par son vecteur vitesse

$$\overrightarrow{v^i} = \begin{pmatrix} v^{i,1} & v^{i,2} & \dots & v^{i,\mathcal{D}} \end{pmatrix}_{i \in [1,\mathcal{N}]} \tag{2.18}$$

avec D est la dimension du vecteur.

La qualité de sa position est déterminée par la valeur de la fonction objectif en ce point.

Chaque particule garde en mémoire la meilleure position par laquelle elle est déjà passée, qu'on note

$$\overrightarrow{P^i_{best}} = \begin{pmatrix} P^{i,1}_{best} & P^{i,2}_{best} & \dots & P^{i,\mathcal{D}}_{best} \end{pmatrix}_{i[1,\mathcal{N}]} \tag{2.19}$$

La meilleure position atteinte par les particules de l'essaim est notée

$$\overrightarrow{G_{best}} = \begin{pmatrix} G^1_{best} & G^2_{best} & \dots & G^{\mathcal{D}}_{best} \end{pmatrix} \tag{2.20}$$

Nous nous référons à la version globale de PSO, où toutes les particules de l'essaim sont considérées comme voisines de la particule i, d'où la notation $\overrightarrow{G_{best}}$ (*Global best*).

on fait la remarque que le terme de "*vitesse*" est ici abusif, car les vecteurs $\overrightarrow{P^i_{best}}$ ne sont pas homogènes à une vitesse. Il serait plus approprié de parler de "*direction de déplacement*". Cependant, pour respecter l'analogie avec le monde animal, les auteurs ont préféré utiliser le terme de "*vitesse*".

Au début de l'algorithme, les $\mathcal{N}$ particules de l'essaim sont initialisées de manière aléatoire/régulière dans l'espace de recherche du problème de dimension $\mathcal{D}$. Ensuite, à chaque itération, chaque particule se déplace, en combinant linéairement les trois composantes citées ci-dessus. En effet, à l'itération $t+1$, le vecteur vitesse et le vecteur position sont calculés à partir de l'équation 2.21 et de l'équation 2.22, respectivement.

$$v^{i,j}_{t+1} = w v^{i,j}_t + c_1 r^{i,j}_{1^t}(P^{i,j}_{best^t} - x^{i,j}_t) + c_2 r^{i,j}_{2^t}(G^j_{best^t} - x^{i,j}_t) \tag{2.21}$$

$$x^{i,j}_{t+1} = x^{i,j}_t + v^{i,j}_{t+1} \quad \backslash \quad i \in [1,\mathcal{N}], j \in [1,\mathcal{D}] \tag{2.22}$$

Où w est une constante, appelée *coefficient d'inertie* ; c_1 et c_2 sont deux constantes, appelées *coefficients d'accélération* ; $r^{i,j}_{1^t}$ et $r^{i,j}_{2^t}$ sont deux nombres aléatoires tirés

uniformément dans l'intervalle $[0, 1]$, à chaque itération t et pour chaque dimension j. Les trois composantes mentionnées ci-dessus (*i.e.* d'inertie, cognitive et sociale) sont représentées dans l'équation 2.21 par les termes suivants :

1. $wv_t^{i,j}$ correspond à la composante d'*inertie du déplacement*, où le paramètre w contrôle l'influence de la direction de déplacement sur le déplacement futur ;
2. $c_1 r_{1^t}^{i,j}(P_{best^t}^{i,j} - x_t^{i,j})$ correspond à la *composante cognitive* du déplacement, où le paramètre c_1 contrôle le comportement cognitif de la particule ;
3. $c_2 r_{2^t}^{i,j}(G_{best^t}^{j} - x_t^{i,j})$ correspond à la *composante sociale* du déplacement, où le paramètre c_2 contrôle l'aptitude sociale de la particule.

Une fois le déplacement des particules effectué, les nouvelles positions sont évaluées et les deux vecteurs $\overrightarrow{P_{best}^i}$ et $\overrightarrow{G_{best}}$ sont mis à jour, à l'itération $t+1$, suivant les deux équations 2.23 (dans le cas d'une minimisation) et 2.24 (dans une version globale de PSO), respectivement. Cette procédure est présentée dans l'Algorithme 1, où $\mathcal{N}$ est le nombre de particules de l'essaim, la modification des positions et vitesses est illustrée dans la *figure 2.5*.

$$\overrightarrow{P_{best^{t+1}}^i} = \begin{cases} \overrightarrow{x_{t+1}^i} & \text{si } f(\overrightarrow{x_{t+1}^i}) < f(\overrightarrow{P_{best^t}^i}) \\ \overrightarrow{P_{best^t}^i} & \text{sinon} \end{cases} \tag{2.23}$$

$$\overrightarrow{G_{best^{t+1}}} = \begin{cases} \overrightarrow{P_{best^{t+1}}^i} & \text{si } f(\overrightarrow{P_{best^{t+1}}^i}) < f(\overrightarrow{G_{best^t}}) \\ \overrightarrow{G_{best^t}} & \text{sinon} \end{cases} \tag{2.24}$$

Algorithme 1 : Optimisation par essaim particulaire

initialiser les $\mathcal{N}$ particules aléatoirement (position $\overrightarrow{x_{t=0}^i}$ et vitesse $\overrightarrow{v_{t=0}^i}$).
pour *chaque particule* i **faire**
    $\overrightarrow{P_{best^{t=0}}^i} := \overrightarrow{x_{t=0}^i}$;
calculer $\overrightarrow{G_{best^{t=1}}}$ selon l'équation 2.24;
tant que *le critère d'arrêt n'est pas satisfait* **faire**
    pour *chaque particule* i **faire**
        estimer la vitesse particule $\overrightarrow{v_{t+1}^i}$ selon l'équation 2.21;
        calculer la nouvelle position de particule $\overrightarrow{x_{t+1}^i}$ selon l'équation 2.22;
        mettre à jour le $\overrightarrow{P_{best^{t+1}}^i}$ selon l'équation 2.23;
        mettre à jour le $\overrightarrow{G_{best^{t+1}}}$ selon l'équation 2.24;
retourner *la solution optimale* $\overrightarrow{G_{best^{t=t_{final}}}}$

2.8 NDT-PSO

Au début, nous avons implémenté la NDT optimisée avec la méthode de NEWTON, mais cette implémentation n'a pas donnée des bonnes résultats. Nous avons remarqué que la méthode de NEWTON ne converge que si nous l'avons initialisée par une estimation très proche de la solution désirée[76], si non, la méthode de NEWTON donne des résultats erronées ou tombe sur un minima local sans toucher le minima global, ce problème est très rencontré dans le domaine de scan-matching[58] (voir la *figure 2.6*).

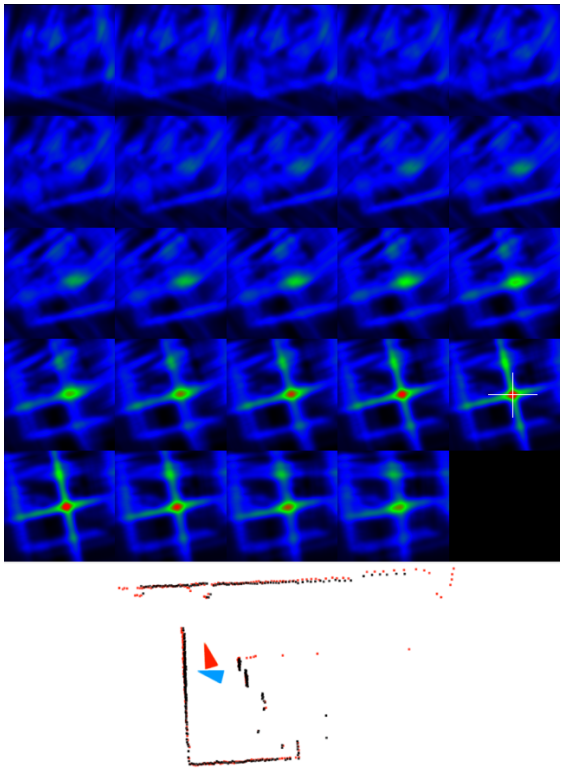


FIGURE 2.6 – La mise en correspondance de deux scans, plusieurs valeurs des paramètres de transformation vérifies la minimalité locale, mais seulement la combinaison marqué en (+) qui vérifie la minimalité globale (*prise de* [58]).

Après une recherche intensive dans la littérature, nous avons constaté que la méthode PSO a plus de chance de tomber sur le minima global sans d'être initialisée[27, 63, 45, 34].

Dans notre cas de la NDT, les particules de PSO sont des paramètres de trans-

formation (*translation et rotation*) entre deux scans successives.

$$\overrightarrow{x^i} = \begin{pmatrix} t_x & t_y & \phi \end{pmatrix} \tag{2.25}$$

Dans la première itération de PSO, les particules $\overrightarrow{x^i}$ sont initialisés aléatoirement dans un domaine précis, après les expériences, nous avons choisi les domaines suivants :

$$\begin{pmatrix} t_x & t_y & \phi \end{pmatrix} \in [-1m, 1m] \times [-\frac{1}{2}m, \frac{1}{2}m] \times [-\frac{\pi}{8}rad, \frac{\pi}{8}rad] \tag{2.26}$$

Cela signifie qu'en terme d'initialisation, le déplacement entre deux scans successives (*intervalle de temps d'ordre de trois seconds*) du robot est considéré comme $\pm 1m$ selon l'axe des X (*mouvement en avant*), $\pm\frac{1}{2}m$ selon l'axe des Y (*mouvement latéral*) et $\pm\frac{\pi}{8}rad \approx 23^{\circ}$ d'orientation.

Étant donner un scan de n points, nous évaluons le "*score*" de chaque particule initialisée en utilisant la fonction 2.5 :

$$score_i = -\mathbf{score}(\overrightarrow{x^i}) \tag{2.27}$$

Après, nous pouvons procéder les étapes de l'algorithme PSO comme indiqué dans l'algorithme 1.

Une fois le minima global $\overrightarrow{G_{best}}$ est déterminé, nous l'utilisant pour mettre à jour la position de robot en faisant à chaque alignement

$$\overrightarrow{pose_t} = \overrightarrow{pose_{t-1}} + \overrightarrow{G_{best^t}} \tag{2.28}$$

Et la carte globale sera mettre à jour en transforment tous les points de scan actuel par le paramètre $\overrightarrow{pose}$, donc pour tout point X_i de ce scan, le point X_i'' est ajouté à la carte globale tel que :

$$X_i'' = T(X_i, \overrightarrow{pose_t}) \tag{2.29}$$

2.9 Conclusion

Dans ce chapitre, nous avons présenté la méthode NDT pour l'alignement des scans laser, détaillé sa principe de fonctionnement et présenté les différentes équations qu'elle utilises, puis avons présenté l'optimisation des paramètres de transformation par la méthode du NEWTON tel qu'elle est proposée dans [10], puis nous avons présenté le problème rencontré lors de l'optimisation par la méthode de NEWTON, et à la fin, nous avons proposé la méthode d'optimisation par essaim

particulaire (PSO) comme remplaçant de la méthode du Newton dans la phase d'optimisation. Par conséquence, nous avons détaillé les techniques utilisées par la PSO afin de délivré des solutions optimales.

Et à la fin de chapitre, nous avons présenté notre algorithme nommé NDT-PSO qui se base sur la NDT pour l'alignement et exploite la méthode PSO dans la phase d'optimisation.

Plus de détails sur le fonctionnement du NDT-PSO se trouve dans le chapitre suivant.

Chapitre 3

Implémentation et résultats

"Le savant restera ignorant de ce qu'il a appris jusqu'à ce qu'il œuvre sur celle-ci. Ainsi une fois qu'il aura œuvré, il deviendra savant"

AL-KHATIB AL-BAGHDADI

Ce chapitre est consacré à la présentation et l'analyse des différents résultats obtenus durant nos expérimentations, Nous allons d'abord présenter le Robot RobuCar et son modèle cinématique, Ensuite, nous allons présenter les tests et les resultats de NDT-PSO.

3.1 Le robot mobile RobuCar

Le RobuCar (*figure 3.1*) est un prototype de petit véhicule électrique construit sur la base d'un châssis tubulaire des petites voitures utilisées dans les terrains de golf. Il comporte une architecture parallèle permettant le développement d'applications temps réel. La *figure 3.2* illustre in dessin descriptif du RobuCar, et ses caractéristiques générales et techniques principales sont les suivantes :

- Longueur totale : $1.836m$, largeur totale : $1.306m$, hauteur : $0.616m$.
- Poids total avec batteries : $310kg$.
- Motorisation : 4 Roues motrices et directrices, chacune liée à un moteur électrique de $1.2kW$.

— Vitesse maximale : $18km/h$ ($5m/s$)
— 8 batteries de $12V$, $60Ah$ avec un gestionnaire automatique de charge.
— Capacité d'accueil : 2 personnes avec bagages.
— Conduite automatique ou manuelle.
— Deux capteurs laser, un Sick LMS200 en arrière et un Sick LMS511 PRO en avant.

FIGURE 3.1 – Le RobuCar du CDTA.

Le RobuCar est constitué d'un ordinateur embarqué et de deux cartes RSMPC555 équipées d'un micro-contrôleur MOTOROLA MPC555. Ces cartes contrôlent respectivement les moteurs de tractions avant et arrière ainsi que ceux de direction. L'ensemble de PC embarqué et cartes RSMPC555 communique via un bus de terrain Controller Area Network (CAN) [64].

L'architecture logicielle du RobuCar a été modifiée par l'équipe NCRM du CDTA ; ils ont dédié l'ordinateur embarqué pour les fonctionnalités du niveau bas (communication avec les micro-contrôleurs, contrôle des roues, des capteurs... etc.), et ils ont développé une interface de communication entre le RobuCar et un ordinateur distant (équipé de système d'exploitation ROS) pour les opérations d'analyse des données, prise de décision... etc.

Le robot est muni de deux capteurs laser ; un Sick LMS511 PRO placé en avant et un Sick LMS200 placé en arrière, Il comporte aussi des encodeurs (liés aux quatre

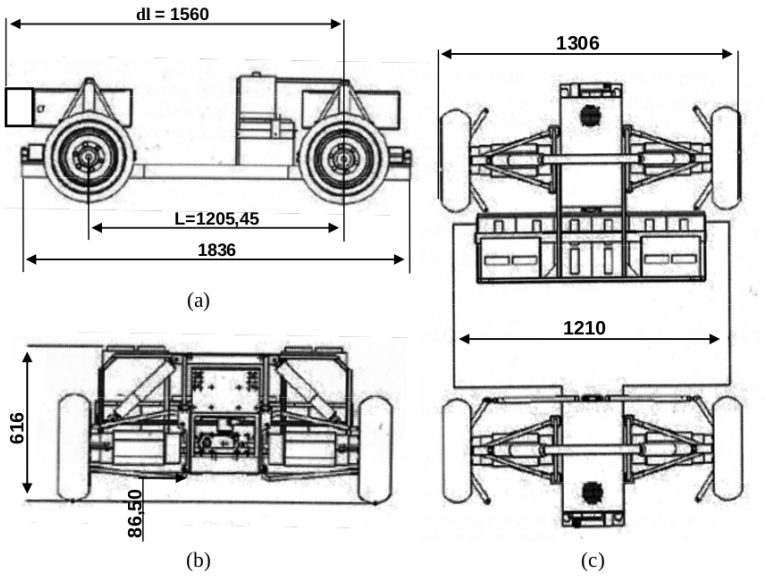


FIGURE 3.2 – Dessin descriptif du RobuCar

moteurs de traction et aux deux vérins) qui servent à mesurer le déplacement et l'orientation effectués par le robot. Les données provenant de ces capteurs sont présentées de façon à donner la vitesse linéaire et l'angle de braquage du robot à partir desquelles peut être calculée la position du robot.

Le robot mobile RobuCar dispose de trois modes de guidage voir *figure 3.3* :

— **Mode simple braquage** : Le robot est dirigé uniquement par les roues du train avant. Les roues du train arrière sont fixes.

— **Mode double braquage** : Les deux trains avant et arrière sont utilisés pour contrôler sa direction, en tournant dans deux sens différents.

— **Mode parking** : Tout comme le mode double braquage, les deux trains avant et arrière sont utilisés pour contrôler l'orientation du robot. Seulement dans ce cas les deux trains tournent dans le même sens[13].

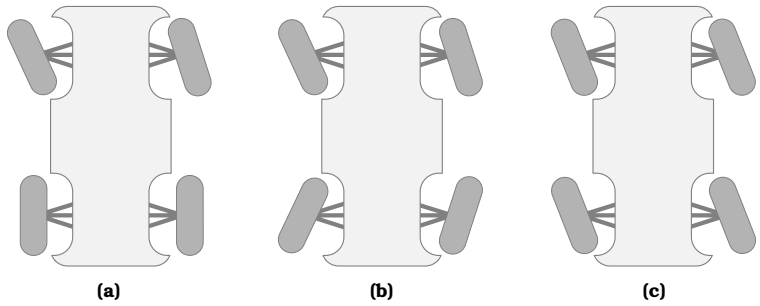


FIGURE 3.3 – Les modes de fonctionnement du robot mobile Robucar : (a) mode simple braquage, (d) mode double braquage, (c) mode parking.

3.2 Cinématique du robot mobile RobuCar

Le RobuCar est constitué de 4 roues motrices liées chacune à un moteur, le mouvement du robot se fait par deux actions : une vitesse longitudinale et l'orientation des roues.

Pour établir le modèle cinématique d'un robot mobile de type voiture, nous acceptons la présence de contraintes non-holonomiques dues au roulement sans glissement entre les roues et le sol. Notre système a donc deux contraintes, contrainte de roulement et contrainte de non-glissement[40].

$$\begin{cases} \text{Contrainte de roulement} : -\sin(\phi+\varphi)\dot{x} + \cos(\phi+\varphi)\dot{y} + L\sin(\phi)\dot{\theta} - V & = 0 \\ \text{Contrainte de non-glissement} : \cos(\phi+\varphi)\dot{x} + \sin(\phi+\varphi)\dot{y} - L\cos(\phi)\dot{\theta} & = 0 \end{cases} \tag{3.1}$$

Avec :
— ϕ est l'angle de braquage,
— φ est l'orientation du robot,
— $\dot{x}$, $\dot{y}$ sont les vitesses linières selon les axes X et Y respectivement,
— et $\dot{\theta}$ est la vitesse angulaire.

3.2.1 Cinématique du RobuCar en mode simple braquage

Le RobuCar est équipé de quatre roues : deux roues fixes à l'arrière et deux roues orientables à l'avant.

Le mécanisme d'ACKERMANN [1] assure un différentiel d'orientation des roues avant qui permet que les axes de rotation de toutes les roues se croisent en un même

1. Pour plus d'informations sur le mécanisme d'ACKERMANN voir l'annexe A.

point CIR, le centre instantané de rotation du véhicule (voir la *figure 3.4*). De ce fait, un robot de type voiture devient équivalent d'un point de vue cinématique à un vélo *bicycle*, *i.e.* un système avec une roue arrière fixe et une roue avant orientable[32].

En général, parmi les robots mobiles du type voiture, nous pouvons distinguer deux configurations :

— Les roues arrières motrices : le moteur est lié au roues arrières. Le braquage se fait par les roues avants au moyen du mécanisme d'Ackermann.

— Les roues avants motrices : le moteur est lié au roues avants et le braquage se fait toujours par les roues avants.

Les modèles cinématiques des robots de type voiture peuvent être établis à partir d'un modèle de vélo, nous réduisons le système à 4 roues à un système de deux roues en ajoutant des roues imaginaires placés au milieu des essieus avant et arrière comme l'illustre la *figure 3.4*.

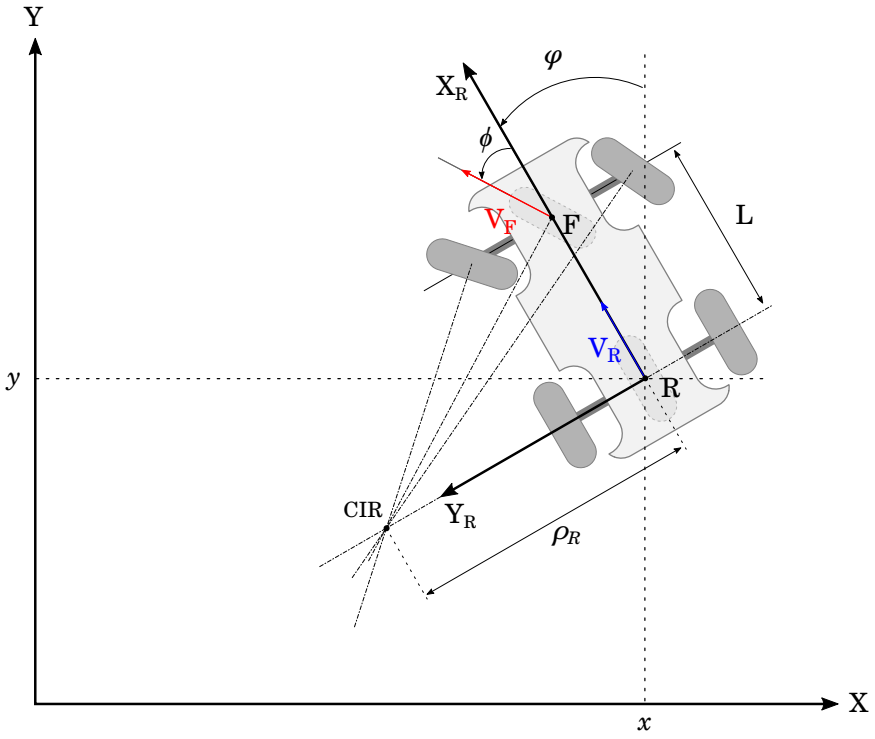


FIGURE 3.4 – Le mode simple braquage
L : La distance entre l'essieu avant et l'essieu arrière,
F et R sont les centres de rotation des roues avant et arrière respectivement,
(R, X_R, Y_R) : Repère lié au robot,
ρ_R : Le rayon de braquage des roues arrières (la distance entre R et CIR),
V_F et V_R : Les vitesses linéaires aux points F et R respectivement

À l'aide des équations de (3.1), nous pouvons établir pour les deux roues les équations suivantes :

$$\begin{cases} \text{Contrainte de roulement de R} : -\sin(\varphi)\dot{x} + \cos(\varphi)\dot{y} - V_R & = 0 \\ \text{Contrainte de non-glissement de R} : -\cos(\varphi)\dot{x} - \sin(\varphi)\dot{y} & = 0 \\ \text{Contrainte de roulement de F} : -\cos(\phi+\varphi)\dot{x} - \sin(\phi+\varphi)\dot{y} + L\cos(\phi)\dot{\theta} & = 0 \end{cases} \tag{3.2}$$

En résoudront les équations de (3.2) nous trouvons :

$$\boxed{\begin{cases} \dot{x} & = -V_R \sin\varphi \\ \dot{y} & = V_R \cos\varphi \\ \dot{\varphi} & = V_R \dfrac{\tan\phi}{L} \end{cases}} \tag{3.3}$$

3.2.2 Cinématique du RobuCar en mode double braquage

Le RobuCar a la possibilité de braquer l'ensemble des roues avant et arrière avec des vitesses et des angles différents, l'angle de braquage des roues arrière dépend de celle des roues avant avec un facteur k.

Comme dans le mode simple braquage, deux roues imaginaires sont considérées au centre des essieus des roues avants et arrières, les axes de rotations de ces deux roues imaginaires s'intersectant dans le point CIR (Centre Instantané de Rotation), les détails sont présentés dans la *figure 3.5*.

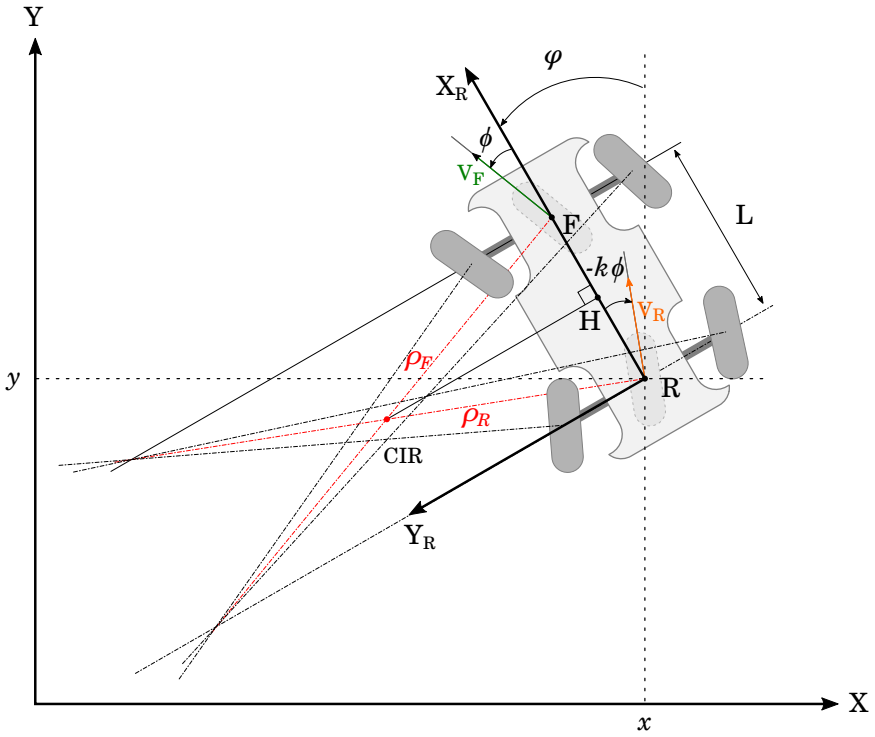


FIGURE 3.5 – Le mode double braquage

Considérant le centre de gravité du robot au milieu de l'essieu des roues arrière, le modèle cinématique du robot est donnée par les équations suivantes[13] :

$$\begin{cases} \dot{x} &= V_R \sin(k\phi - \varphi) = -v_R \sin(\varphi - k\phi) \\ \dot{y} &= V_R \cos(k\phi - \varphi) = v_R \cos(\varphi - k\phi) \\ \dot{\varphi} &= \dfrac{V_R}{\rho_R} \end{cases} \tag{3.4}$$

À partir du schèma de la *figure 3.5* on obtient :

$$\rho_R = \frac{RH}{\sin(k\phi)} \tag{3.5}$$

$$\begin{cases} RH &= GH \tan(k\phi) \\ GH &= \dfrac{HF}{\tan\phi} \end{cases} \tag{3.6}$$

En remplaçant la 2^{eme} équation de (3.6) dans la 1^{ere} on obtient :

$$\begin{cases} RH &= \dfrac{HF \tan(k\phi)}{\tan\phi} \\ RH + HF &= L \end{cases} \tag{3.7}$$

Les équations du (3.7) donnant :

$$RH + \frac{RH \tan\phi}{\tan(k\phi)} = L \Longleftrightarrow RH = \frac{L}{1 + \dfrac{\tan\phi}{\tan(k\phi)}}$$

$$\begin{aligned} RH &= \frac{L\tan(k\phi)}{\tan(k\phi) + \tan\phi} = \frac{L\tan(k\phi)}{\dfrac{\sin(k\phi)}{\cos(k\phi)} + \dfrac{\sin(\phi)}{\cos(\phi)}} \\ &= \frac{L\tan(k\phi)}{\dfrac{\sin(k\phi)\cos\phi + \sin\phi\cos(k\phi)}{\cos(k\phi)\cos(\phi)}} = \frac{L\tan(k\phi)\cos(k\phi)\cos\phi}{\sin(\phi + k\phi)} \end{aligned}$$

Donc on déduit l'expression de ρ_R :

$$\rho_R = \frac{L\cos\phi}{\sin(\phi + k\phi)} \tag{3.8}$$

Et enfin, on obtient le modèle cinématique de robot mobile RobuCar :

$$\boxed{\begin{cases} \dot{x} & = -V_R \sin(\varphi - k\phi) \\ \dot{y} & = V_R \cos(\varphi - k\phi) \\ \dot{\varphi} & = \dfrac{V_R \sin(\phi + k\phi)}{L \cos\phi} \end{cases}} \tag{3.9}$$

3.3 Le capteur laser Sick LMS511-10100 PRO

Le scanner laser Sick LMS511-10100 PRO (*figure 3.6*) est un capteur de mesure laser de moyenne portée, il fonctionne même dans des conditions climatiques difficiles grâce à la technologie multi-écho.

Les principaux avantages du scanner Laser Sick LMS511 sont sa faible puissance absorbée et le traitement rapide du signal [2].

FIGURE 3.6 – Le capteur Sick LMS511-10100 Pro

3.3.1 Caractéristiques techniques du LMS511

Les caractéristiques techniques du scanner laser LMS511-10100 PRO sont les suivantes :

- Domaine d'application : Externe (Outdoor)
- Angle d'ouverture : 190
- Fréquence de balayage : $25Hz$, $35Hz$, $50Hz$, $75Hz$ ou $100Hz$
- Résolution angulaire (en degrés) : 0.167, 0.25, 0.333, 0.5, 0.667 ou 1.0
- Source lumineuse Infrarouge : $905nm$

2. https://www.sick.com/fr/fr/solutions-de-mesure-et-de-detection/capteurs-2d-lidar/lms5xx/lms511-10100-pro/p/p215941, Consulté le 19 juillet 2017.

— Zone de fonctionnement : de $0m$ à $80m$
— Correction de brouillard : Oui
— Temps de réponse : $\geq 10ms$

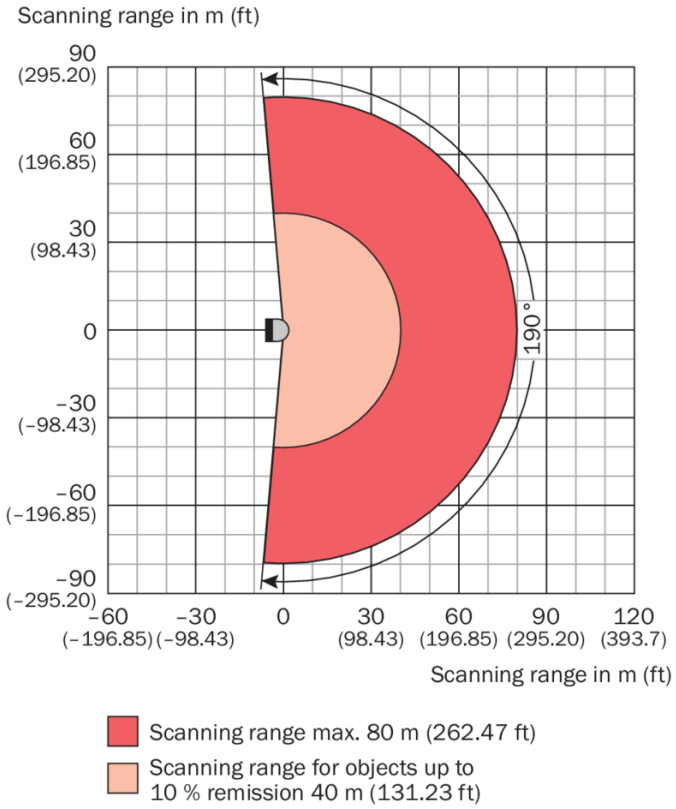


FIGURE 3.7 – Angle d'ouverture et la portée du laser.

3.4 Outils utilisés

Pendant les différentes étapes de développement de notre solution proposée, nous avons utilisé quelques outils :

3.4.1 Environnement de développement

Ubuntu 16.04.2 LTS

Nous avons choisi le système d'exploitation Ubuntu [3], pas juste pour sa stabilité et sa large communauté qui offre un support de très bonne qualité, mais parce-qu'il

3. https://www.ubuntu.com, Consulté le 19 juillet 2017.

est le seul système d'exploitation officiellement supporté par ROS.

Ubuntu est un système d'exploitation libre et open source développé par la société Canonical [4] sur la base de la distribution Linux Debian [5]. Son nom provient d'un ancien mot africain *"bantou"* qui signifie *"je suis ce que je suis grâce à ce que nous sommes tous"*[6]. Dans le même ordre d'esprit, les utilisateurs sont encouragés à étudier son fonctionnement, le modifier, l'améliorer et enfin de le redistribuer [7]. En 2011, une estimation donne plus de 25 millions d'utilisateurs des différentes versions pour ordinateurs [8].

Robot Operating System (ROS)

Dans le cadre de son projet de fin d'études[1], l'étudiant Ilyas Massinissa Abbas et l'équipe NCRM du CDTA ont développé une solution pour rendre possible au RobuCar de communiquer avec le système ROS, ce travail a facilité le développement des applications pour RobuCar en bénéficiant des fonctionnalités du ROS.

Le système d'exploitation des robots (Robot Operating System ROS), est un ensemble d'outils informatiques libre et open source permettant de développer des logiciels pour la robotique. Son développement est mené par l'*Open Source Robotics Foundation (OSRF)*. ROS est officiellement supporté par plus de 75 robots.

3.4.2 Paquets ROS

Sous ROS, nous avons utilisé le paquet `robucar_driver` de [1] pour communiquer avec le RobuCar, et les paquets `sicktoolbox2` [9] et `sicktoolbox_wrapper2` pour accéder au données fournis par le capteur laser.

Au cours de nos tests, nous avons exploité les outils du ROS `rqt_graph` et `rviz`.

3.5 Implémentation de NDT-PSO

Nous avons implémenté la solution proposée dans le deuxième chapitre sur le robot mobile RobuCar, l'implémentation est faite sous le système d'exploitation

4. https://www.canonical.com, Consulté le 19 juillet 2017.
5. https://www.debian.org, Consulté le 19 juillet 2017.
6. http://www.metronews.fr/info/discours-d-obama-pour-mandela-ca-veut-dire-quoi-ubuntu/mmlj!uURda8g9FNtA/, Consulté le 19 juillet 2017.
7. http://www.ubuntu.com/about, Consulté le 19 juillet 2017.
8. http://www.20minutes.fr/article/725116/ubuntu-projet-communautaire, Consulté le 19 juillet 2017.
9. http://sicktoolbox.sourceforge.net, Consulté le 19 juillet 2017.

pour les robots ROS (Robot Operating System) (voir l'annexe B), et en utilisant le langage de programmation *Python*, ainsi que la bibliothèque *NumPy* qui apporte les fonctionnalités mathématique au langage Python.

Le schéma de communication de notre nœud ndt-SLAM est illustré dans la *figure 3.8*.

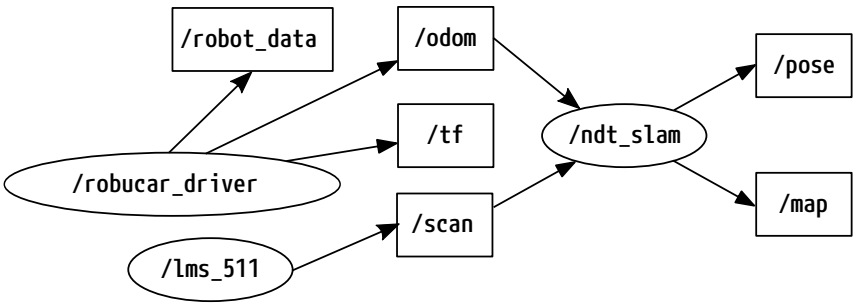


FIGURE 3.8 – Schéma de nœud `ndt_slam`, généré par l'outil `rqt_graph` de ROS.

L'organigramme de fonctionnement général du nœud `ndt_slam` est illustré dans *figure 3.9*.

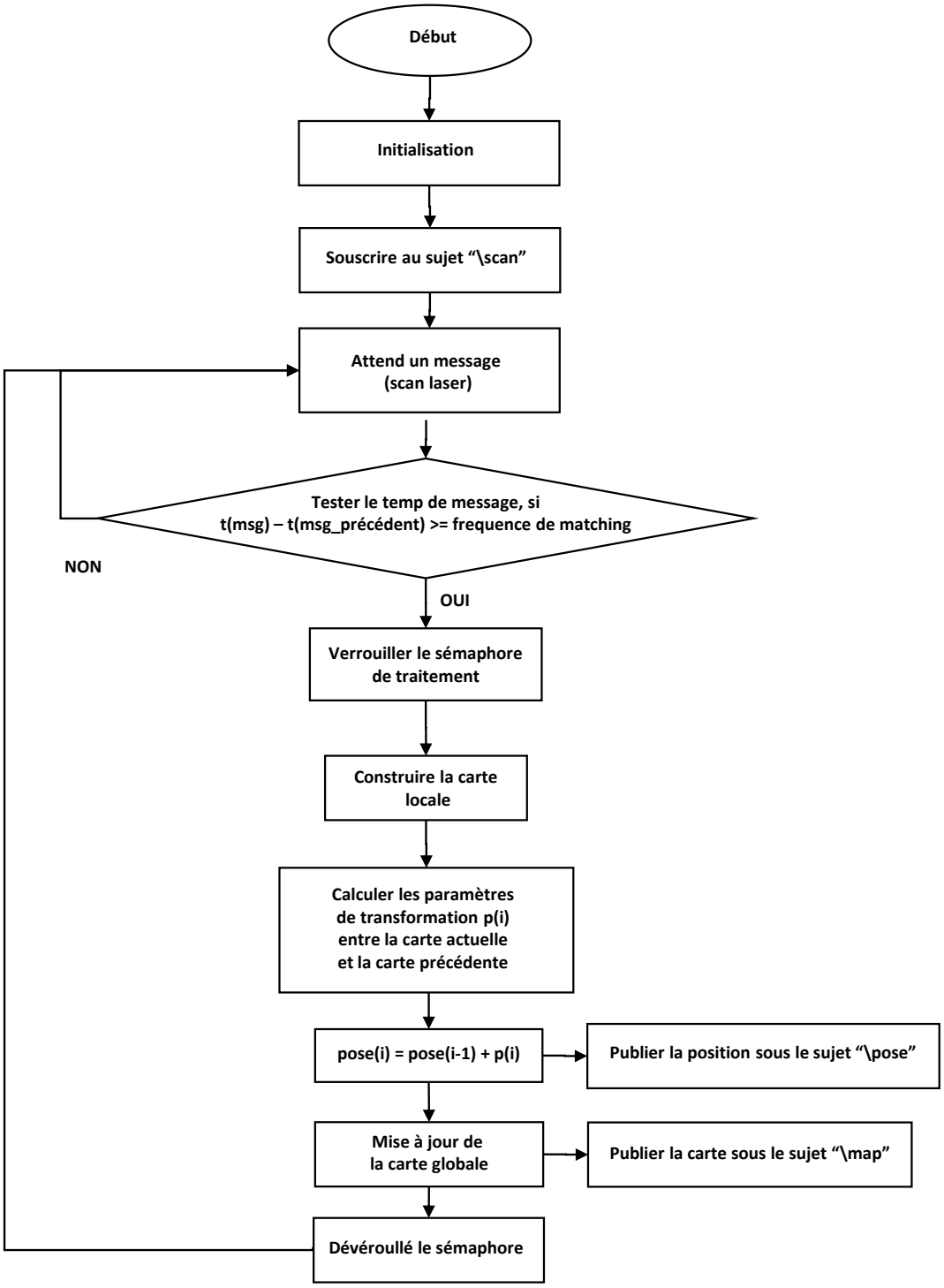


FIGURE 3.9 – L'organigramme général de nœud `ndt_slam`.

3.6 NDT-PSO en action

Dans cette section, nous présentons les résultats obtenus à la suite de l'implémentation de la NDT avec PSO, sur le robot mobile RobuCar dans des environnements d'intérieur et d'extérieur.

Premièrement, nous commençons par illustrer les étapes de l'alignement dans l'algorithme NDT-PSO qui sont les suivantes :

— Récupération des données laser.
— Représentation de l'environnement.
— L'alignement.

3.6.1 Récupération des données laser

Premièrement, le nœud `ndt_slam` récupère les données laser publiées par le nœud `lms_511` sur le sujet "*topic*" `/scan`. Sous ROS, les données laser sont de type `sensor_msgs.msg.LaserScan`, un exemple de contenu d'un objet `LaserScan` est :

```
header:
    seq: 5188
    stamp:
    secs: 1496065499
    nsecs: 777696723
    frame_id: laser
angle_min: -1.65806281567
angle_max: 1.65806281567
angle_increment: 0.0184229202569
time_increment: 0.0
scan_time: 0.0
range_min: 0.0
range_max: 81.0
ranges: [2.870000123977661,  2.872000217437744,
         2.888000249862671,  2.9040000438690186,
         2.9000000953674316, 2.9140000343322754,
         2.9180002212524414, 2.936000108718872,
         2.948000192642212,  2.954000234603882,
         2.9700000286102295, 2.9880001544952393, ...]
intensities: []
```

Le membre `ranges` contient une liste des distances r entre le capteur et les obstacles détectés, l'angle correspondant à chaque élément (d'indice $i \in [0, N]$) de

la liste est calculé par la formule suivante :

$$\alpha_i = \alpha_{min} + \frac{i \times (\alpha_{max} - \alpha_{min})}{N - 1} \tag{3.10}$$

Chaque couple de coordonnées polaires (r_i, α_i) sera converti en coordonnées cartésiennes (x_i, y_i) pour être prêt à utiliser. La conversion se fait par l'équation :

$$\begin{pmatrix} x_i \\ y_i \end{pmatrix} = \begin{pmatrix} r_i \cos \alpha_i + L \\ r_i \sin \alpha_i \end{pmatrix} \tag{3.11}$$

Ou L est la distance entre le centre de gravité de robot et le centre de capteur laser.

3.6.2 Représentation de l'environnement

Pour la représentation de l'environnement chaque point correspond à un objet détecté par le capteur laser, l'ensemble de ces points est représentés dans une carte locale dont les dimension de cette dernière sont fixée à $20m \times 20m$. Un exemple de la représentation d'un scan laser de l'environnement *figure 3.10* est illustré dans la *figure 3.11*.

FIGURE 3.10 – L'environnement de teste.

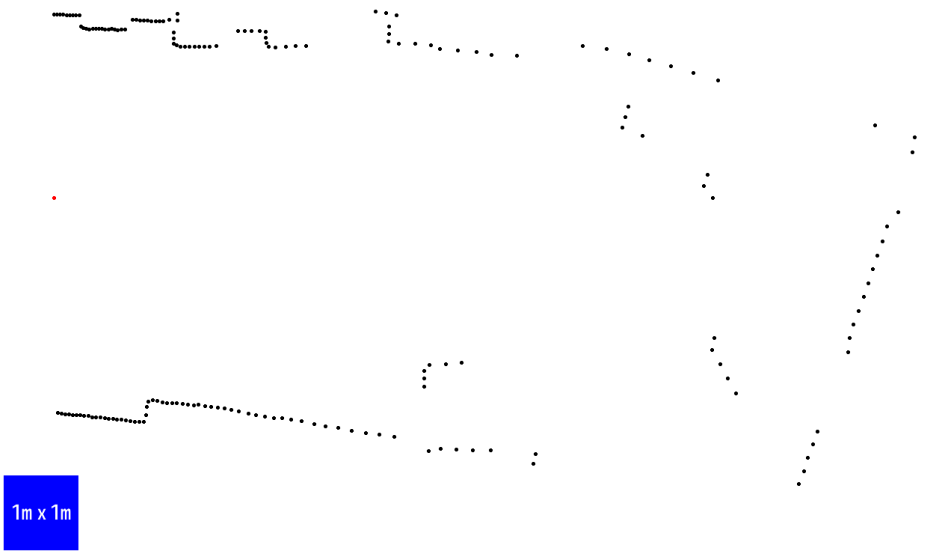


FIGURE 3.11 – Les données laser représentés.

Ensuite, l'espace est subdivisé en cellules de même taille (de $1m \times 1m$ dans ce cas), voir *figure 3.12*.

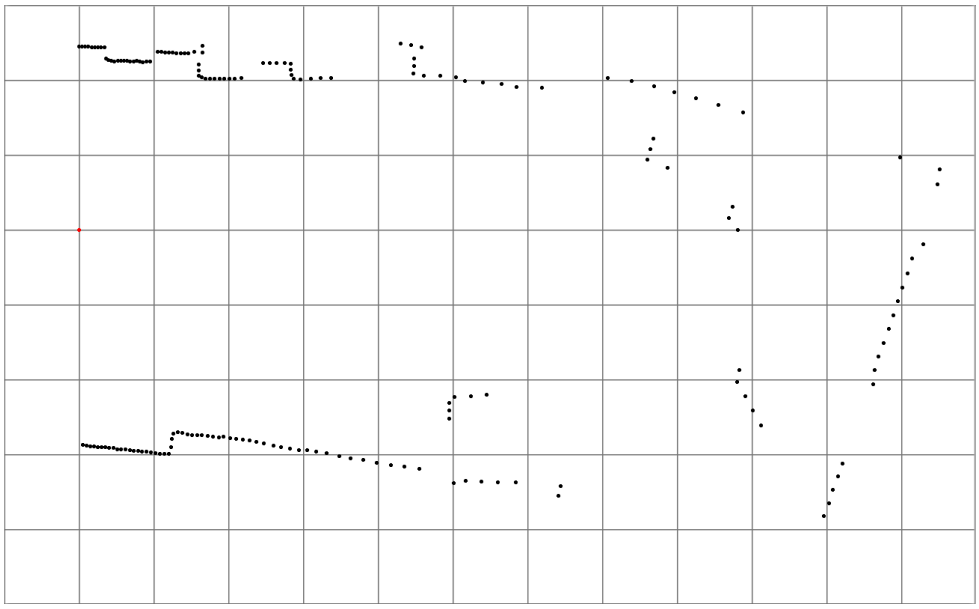


FIGURE 3.12 – La subdivision de la carte locale en cellules de $1m \times 1m$.

Pour chaque cellule qui contient plus de trois points, les paramètres (Q, Ω) de la NDT doivent être calculés selon les équations 2.1 et 2.2. Les fonctions de distribution des probabilités sont représentés dans la *figure 3.13*.

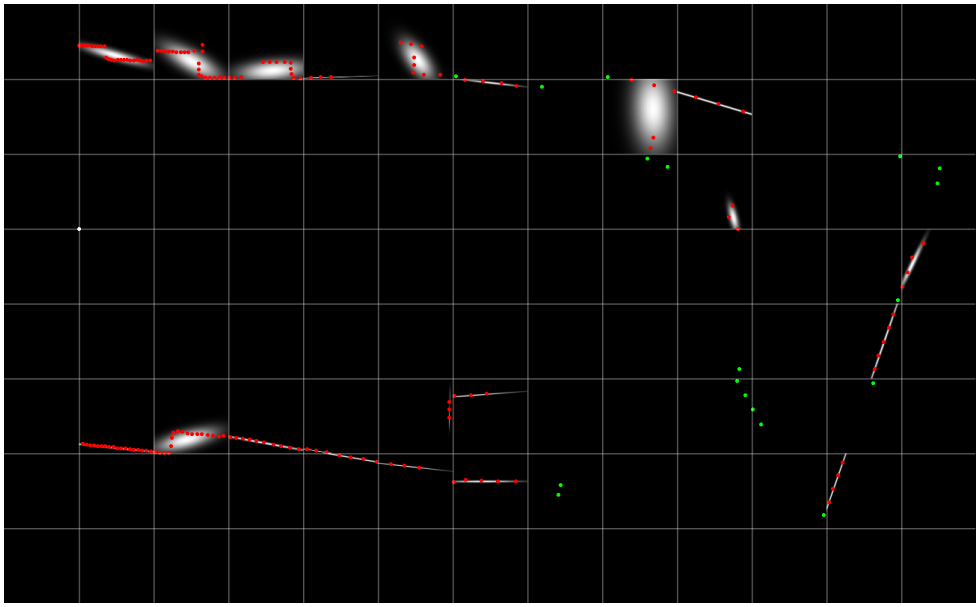


FIGURE 3.13 – Les fonctions de distribution des probabilités de chaque cellule, une zone blanche indique une probabilité de 1, les zone noires indique des probabilité de 0, les grises sont entre [0, 1].

Enfin, cette représentation de l'environnement autour du robot sera considérée comme référence lors de l'alignement avec le prochain scan.

3.6.3 L'alignement

Après avoir eu la représentation de référence, le robot continuera sont parcours et le nœud récupère les données des nouveaux scans successivement (par exemple, il récupère des données d'un scan, qui sont représentées dans la *figure 3.14*).

FIGURE 3.14 – Les données laser du nouveau scan.

Avec un scan de référence et un nouveau scan, l'optimisation de la fonction "*score*" doit fournir le paramètre $p = (t_x, t_y, \phi)$ qui représentante la transformation (*translation et rotation*) relative entre les deux scans.

Les figures de 3.15 à 3.20, représentent des différentes valeurs pour la transformation $p_i = (t_x, t_y, \phi)$ jusqu'à la convergence du PSO, cette convergence a couté dans cet exemple 15 itérations, le scan représenté en noir est le scan de référence, et le scan rouge est le nouveau scan, le point en rouge est l'origine du scan référence, et le point bleu entouré est l'origine de nouveau scan (*i.e.* la position estimée dans cette itération). L'algorithme a été appliqué sans une estimation initiale *i.e.* $p_0 = (0, 0, 0)$.

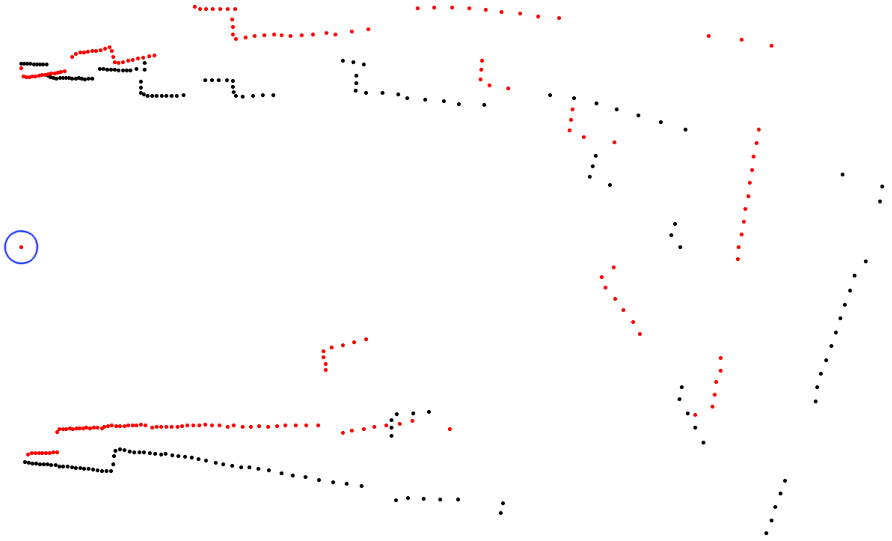


FIGURE 3.15 – Itération 0, $p_0 = (0.00000, 0.00000, 0.00000)$

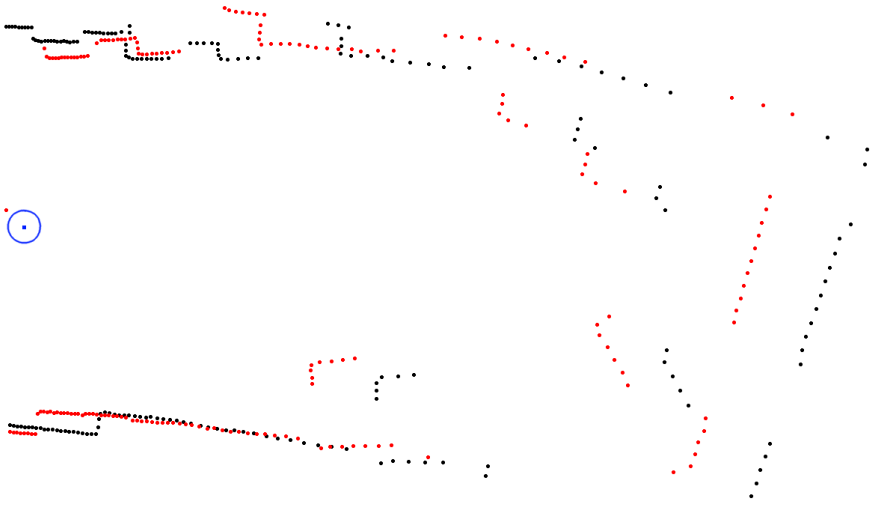


FIGURE 3.16 – Itération 1, $p_1 = (0.23543, -0.21046, -0.11834)$

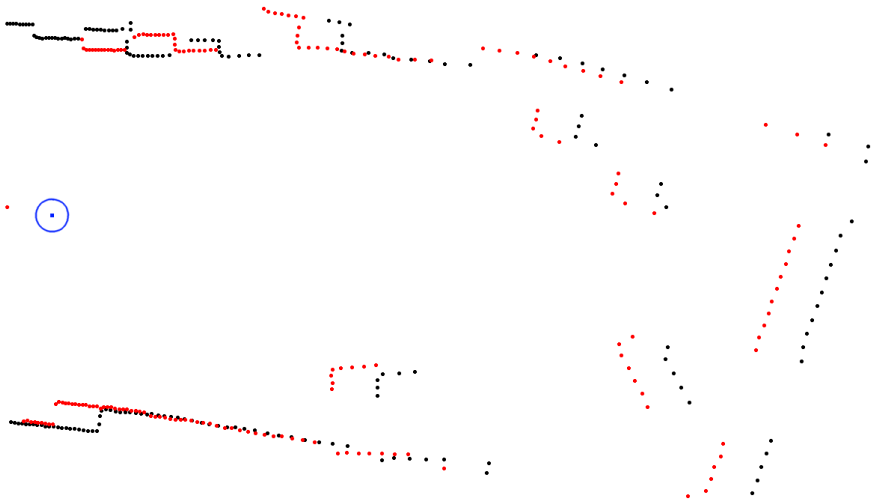


FIGURE 3.17 – Itération 4, $p_4 = (0.60109, -0.11892, -0.17062)$

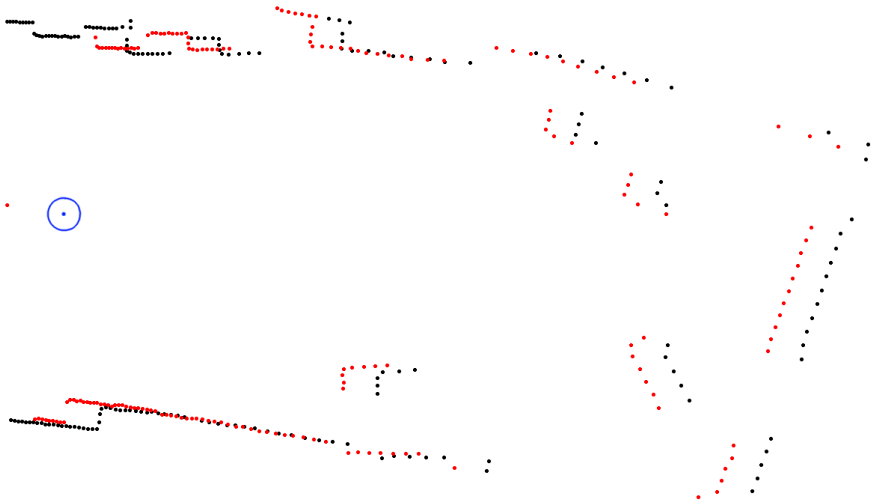


FIGURE 3.18 – Itération 6, $p_6 = (0.76586, -0.11687, -0.17606)$

FIGURE 3.19 – Itération 9, $p_9 = (1.19760, -0.15199, -0.19087)$

FIGURE 3.20 – Itération 15, $p_{15} = (1.25337, -0.15880, -0.18095)$

Après les étapes d'optimisation, le paramètre p est ajouté a la position précé-

dente de robot (lors de scan de référence) afin de trouver la nouvelle position. La carte (en noir), la position précédente du robot (en rouge) et la nouvelle position (en blue) sont représentés dans la *figure 3.21*.

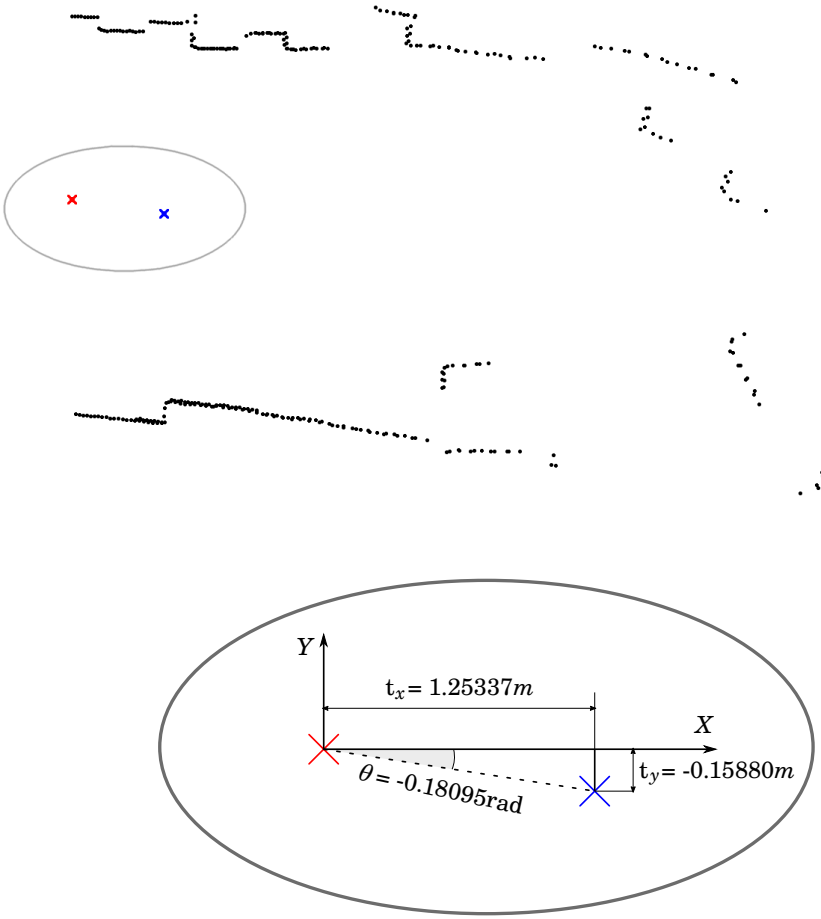


FIGURE 3.21 – Le résultat de l'optimisation, une carte et une nouvelle position (en blue), l'ancienne position est celle en rouge, l'illustration montre les positions zoomées, et les paramètres résultats d'optimisation.

Une carte complète de l'environnement de la *figure 3.10* a été délivrée par le nœud `ndt_slam`, cette carte est représentée dans la *figure 3.22*.

FIGURE 3.22 – La carte délivrée pour l'environnement de la *figure 3.10*.

3.7 L'influence de la taille de la cellule sur la représentation de la carte

Dans ce test nous avons varié la taille de la cellule de $0.25m \times 0.25m$ à $2m \times 2m$.

Automatiquement, quand la taille de la cellule diminue, le nombre de points par cellule diminue également. D'un autre coté, dans le cas où le nombre de points par cellule est inférieur à 3 points, la cellule ne sera pas prise en considération [10],

10. Nous avons expliqué la cause du choix de cette valeur d'au moins trois points par cellule

donc y'aura une **perte de données**, l'exemple de la *figure 3.23* nous montre que la taille de $0.25m \times 0.25m$ a causée une importante perte de données. Dans ces figures, les points représentés en verre sont ignorés lors de la construction de la carte NDT.

Quand la taille de la cellule dépasse un seuil donné, le nombre de points par cellule augmente, mais dans une telle situation, la densité de la probabilité de la cellule *(représentée en nuage blanc dans les figures)* occupe même des espaces libres dans cette dernière, contrairement par rapport à la diminution de la taille de la cellule, y'aura pas de perte de données mais de **détails**, voir la *figure 3.26*.

Durant ces tests, nous avons obtenus avec la taille de $1m \times 1m$ les meilleurs résultats, voir la *figure 3.25*.

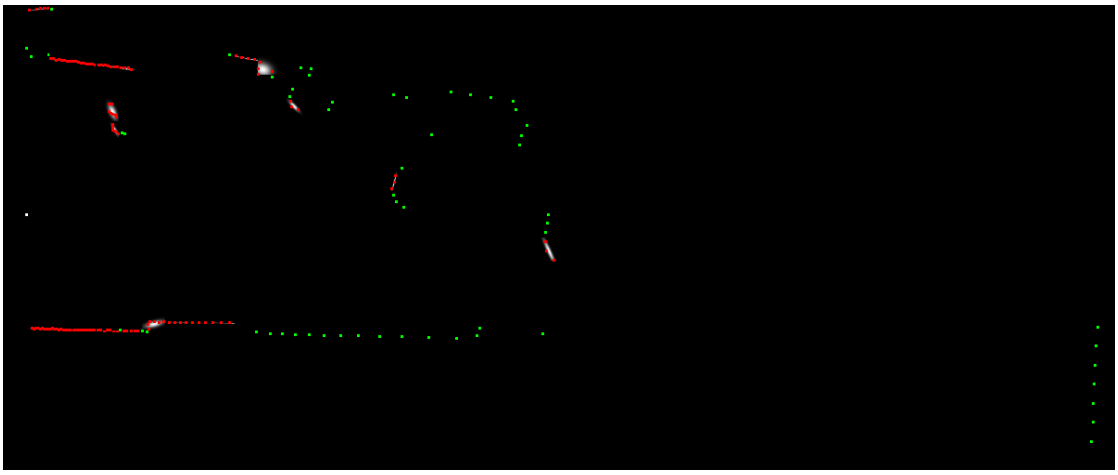


FIGURE 3.23 – Les densités de probabilités pour une carte locale avec des cellules de $0.25m \times 0.25m$.

dans la section 2.2 du chapitre précédent.

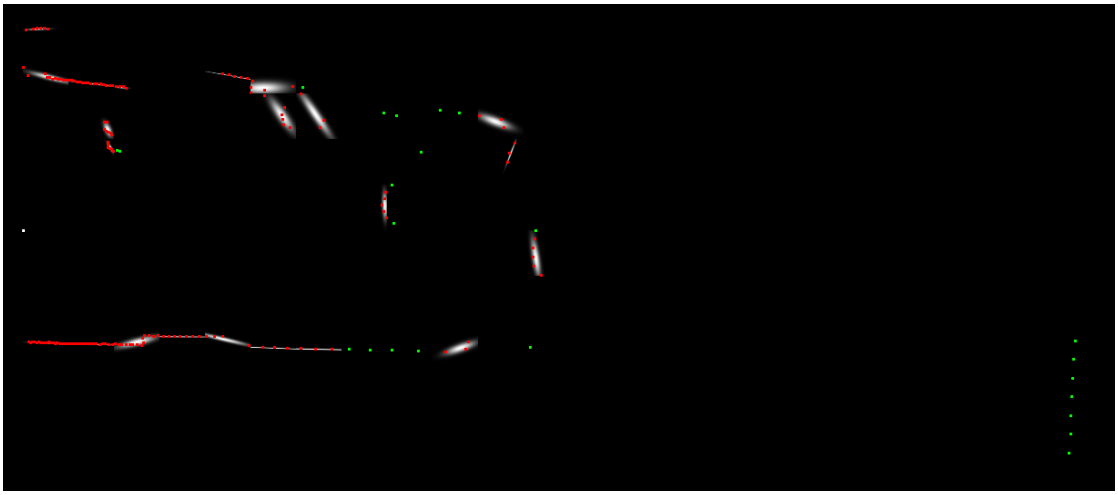


FIGURE 3.24 – Les densités de probabilités pour une carte locale avec des cellules de $0.5m \times 0.5m$.

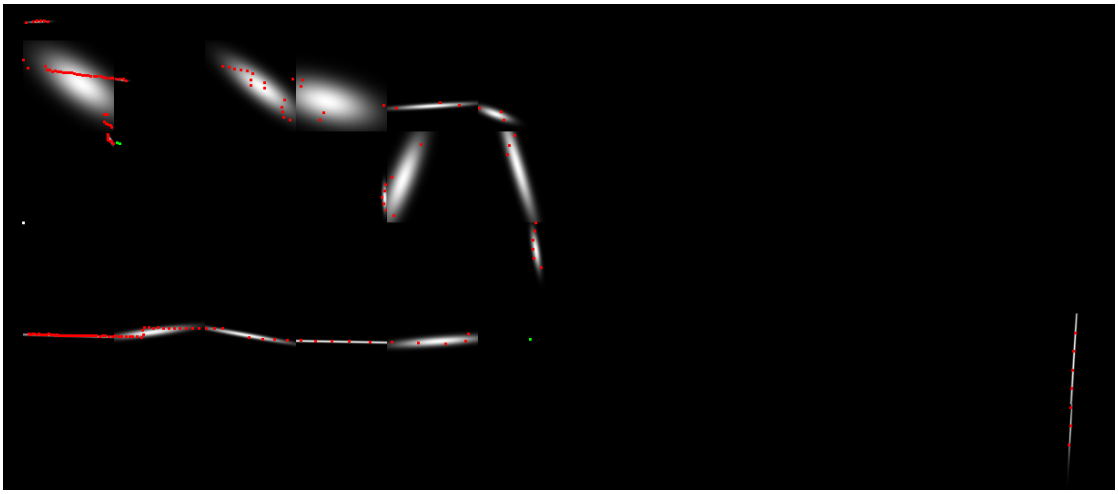


FIGURE 3.25 – Les densités de probabilités pour une carte locale avec des cellules de $1m \times 1m$.

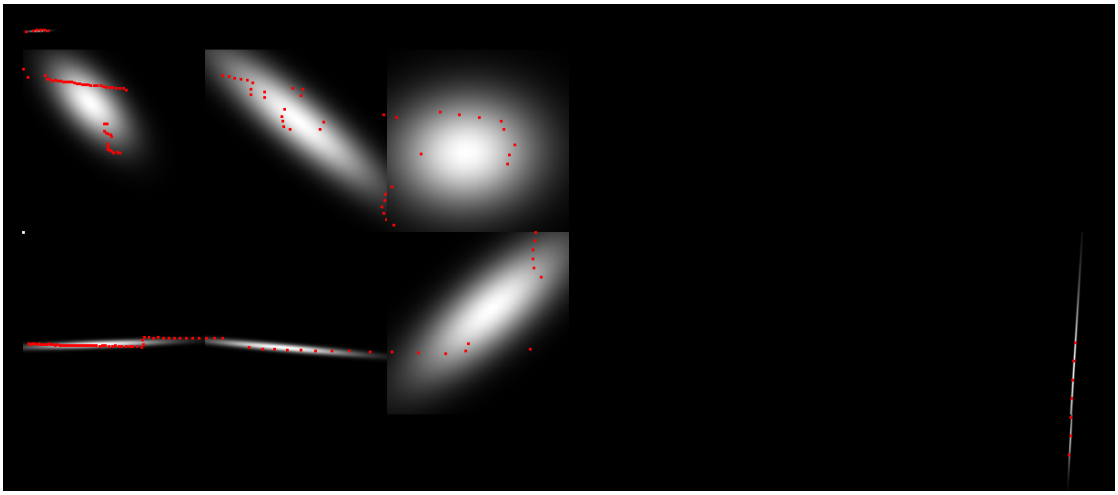


FIGURE 3.26 – Les densités de probabilités pour une carte locale avec des cellules de $2m \times 2m$.

3.8 Testes de fonctionnement

Dans cette section, nous allons présenter quelques résultats de l'implémentation de notre solution proposée dans un environnement externe.

Le teste a été déroulé dans la cours de CDTA (voir la *figure 3.27*).

FIGURE 3.27 – La scène de teste, environnement externe.

Les données laser brutes récupérés du capteur lors de sa trajectoire sont représentés dans la *figure 3.28*.

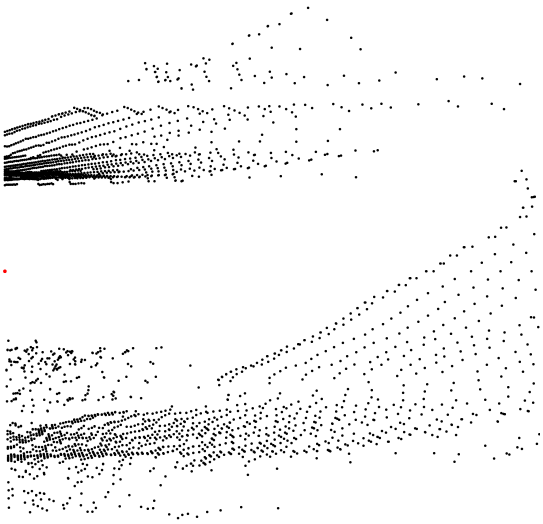


FIGURE 3.28 – Les données laser brutes.

Le résultat de la cartographie et la localisation fournis par le nœud `ndt_slam` dans le cas des cellules de taille $1m \times 1m$ sont représentés dans la *figure 3.29*.

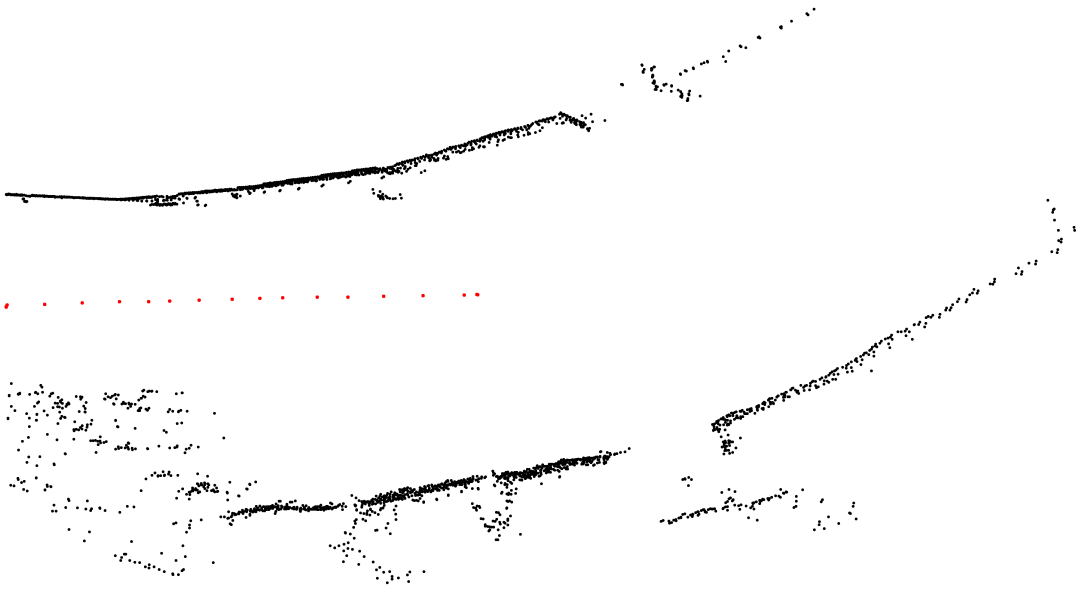


FIGURE 3.29 – Résultat du nœud, la carte globale (en noir) ainsi que les positions du robot (en rouge).

Un autre test réalisé au couloir du CDTA (*figure 3.30*) a donné une bonne représentation de la carte et une bonne estimation de la position pendant le trajectoire, le test est illustré dans la *figure 3.31*.

FIGURE 3.30 – Environnement du test, couloir du CDTA.

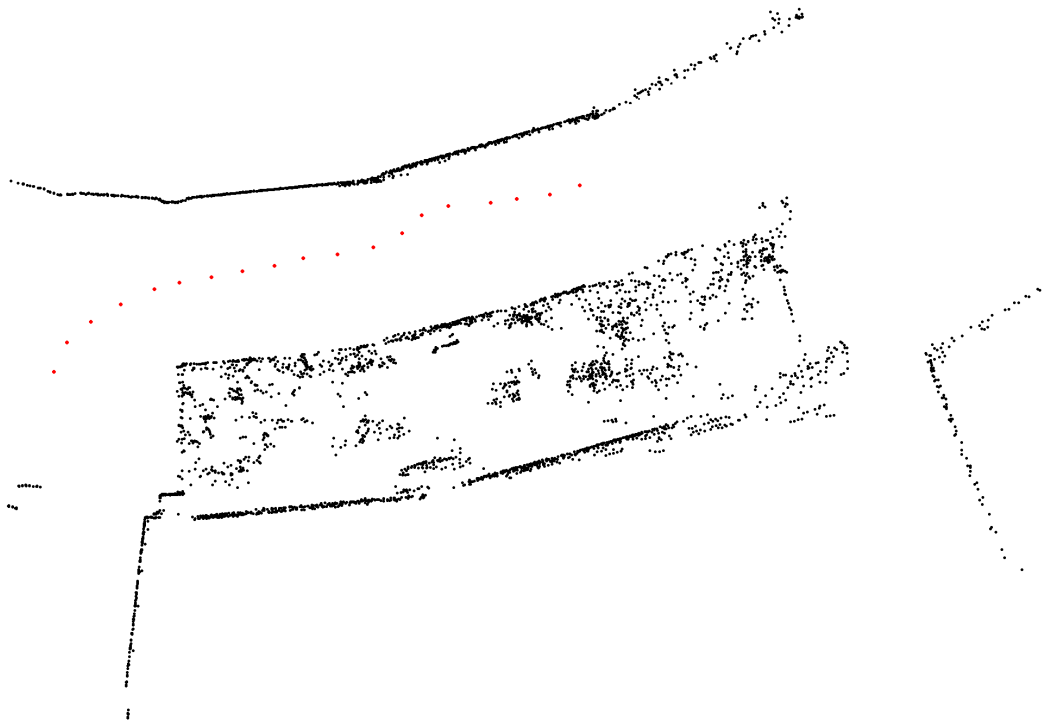


FIGURE 3.31 – Résultat du nœud, la carte globale (en noir) et les positions du robot (en rouge).

3.9 Conclusion

Nous avons présenté dans ce chapitre l'implémentation du NDT-PSO ; notre solution proposée pour accomplir la tâche de la localisation et la cartographie simultanées, et nous avons détaillé le principe de fonctionnement de cette méthode.

Nous avons présenté une discussion sur le chois de la taille de cellule et nous avons comparé des différentes valeur de cette dernière.

Nous avons aussi présenté quelques tests sur le RobuCar ainsi que la discussion des résultats obtenues.

Conclusion générale

"Les machines un jour pourront résoudre tous les problèmes, mais jamais aucune d'entre elles ne pourra en poser un !"

Albert Einstein

Les tâches de localisation et de cartographie, regroupées sous le terme de SLAM, constituent un pilier fondamental du fonctionnement autonome d'un Robot mobile. Bien que plusieurs algorithmes ont été développés dans le but de résoudre le problème du SLAM, durant notre projet de fin d'étude nous avons travaillé sur le développement et l'amélioration d'un algorithme de SLAM qui est NDT-PSO. Mettant en pratique les connaissances acquises tout au long de nos études.

D'abord, nous avons introduit la problématique du SLAM en commençant par parcourir quelques techniques de localisation et cartographie disponibles, puis nous avons donnés une présentation et une comparaison entre trois approches du SLAM qui sont le filtre de Kalman, les filtres particulaires et le scan matching. Nous avons utilisé scan matching pour son simplicité et son efficacité.

Dans le cadre des méthodes basées sur le scan matching, plusieurs approches ont été développées, parmi ces méthodes nous avons choisi la méthode NDT proposé par [10] qui donne un cadre mathématique qui permet à associer des scans lasers par alignement. La représentation de l'environnement dans NDT est basée sur la modélisation de la distribution de tous les points 2D reconstruits à partir d'un scan laser par une collection de distributions normales locales.

Pour l'estimation de la position au lieu d'utiliser la méthode d'optimisation de Newton qui semble d'être inefficace dans nos expérimentations, nous avons la remplacé par la méthode d'optimisation par essaim particulaire (Particle Swarm Optimization, PSO).

Finalement, nous avons procédé à l'implémentation et les tests du notre algorithme pour SLAM nommé NDT-PSO, basé sur la méthode d'alignement NDT et la méthode d'optimisation PSO. Nous avons implémenté notre solution sous le système d'exploitation ROS en utilisant le langage Python, et nous avons la testée sur le robot mobile RobuCar.

Pendant nos tests, nous avons rencontrés quelques problèmes tel que le problème d'optimisation connu par *"le problème des minimas locaux"*, au cours d'optimisation des paramètres, l'algorithme d'optimisation peut tomber dans une combinaison des paramètres qui vérifie la minimalité locale mais pas la minimalité globale, et dans un cas pareil les paramètres fournis par l'algorithme affectant le processus du SLAM.

Un autre problème était due au traversement des vitres par le laser, ce qui affecte la précision des données laser retournées et donc touche à la précision de l'estimation de la position et de la construction de la carte.

Nous avons rencontré aussi des problèmes dans les environnements intérieurs dus au fait que le robot donnait de faux résultats quand les scans lasers successifs se ressemblent, comme dans l'avancement dans un couloir ou l'environnement reste similaire.

Une évaluation a été faite sur plusieurs paramètres pour la validation de notre solution, les résultats obtenus de notre tests sur Robucar démontre bien l'efficacité de l'algorithme.

Annexe A

Concepts fondamentales

A.1 Les points, les positions et les poses

Dans ce qui suit, les points d'un scan 2D sont souvent désignés par un vecteur $X = \begin{pmatrix} x & y \end{pmatrix}^t$ représentant leur **position** dans l'espace. Un point d'un scan peut également avoir d'autres propriétés, telles que la *couleur* et les informations sur l'*orientation* de la *surface*, mais la propriété la plus intéressante dans ce contexte est habituellement sa *position*, donc X et le terme "point" seront souvent utilisés de manière interchangeable pour un point de scan et sa position. Le concept de **pose** est essentiel à l'*enregistrement* et à la *localisation*. Une pose dans ce contexte est une *combinaison d'une position et d'une orientation*. Plus précisément, une pose est représentée par une rotation autour de l'origine du système de coordonnées suivie d'une translation.

A.2 Représentation des poses

La représentation des poses des scans est essentielle à l'enregistrement par scan. En deux dimensions, la translation peut être représentée directement comme un vecteur 2D $\Gamma = \begin{pmatrix} t_x & t_y \end{pmatrix}^t$ et une rotation en tant que scalaire représentant l'angle de rotation dans le sens direct, dans ce qui suit, nous allons représentés la transformation géométrique par un vecteur 3D qui contient les trois paramètres de la transformation $P = \begin{pmatrix} t_x & t_y & \phi \end{pmatrix}^t$.

A.3 Géométrie directionnelle d'Ackermann

La géométrie de direction de Rodolphe Ackermann est une dispositif géométrique des liaisons dans la direction d'une voiture ou d'un autre véhicule conçu

pour résoudre le problème des roues à l'intérieur et à l'extérieur d'un virage devant tracer des cercles de différents rayons.

Une approximation simple de la géométrie de direction d'Ackermann peut être générée en déplaçant les points de pivot de direction vers l'intérieur de manière à se trouver sur une ligne tracée entre les pivots de direction et le centre de l'essieu arrière. Les points de pivotement de direction sont reliés par une barre rigide appelée *"tirant"* qui peut également faire partie du mécanisme de direction, par exemple sous forme de crémaillère et de pignon. Avec la géométrie d'Ackermann, à n'importe quel angle de direction, le point central de tous les cercles tracés par toutes les roues se trouveront à un point commun. Notez que cela peut être difficile à mettre en pratique avec des liaisons simples, et les concepteurs sont invités à dessiner ou à analyser leurs systèmes de direction sur toute la gamme des angles de direction (voir la *figure A.1*).

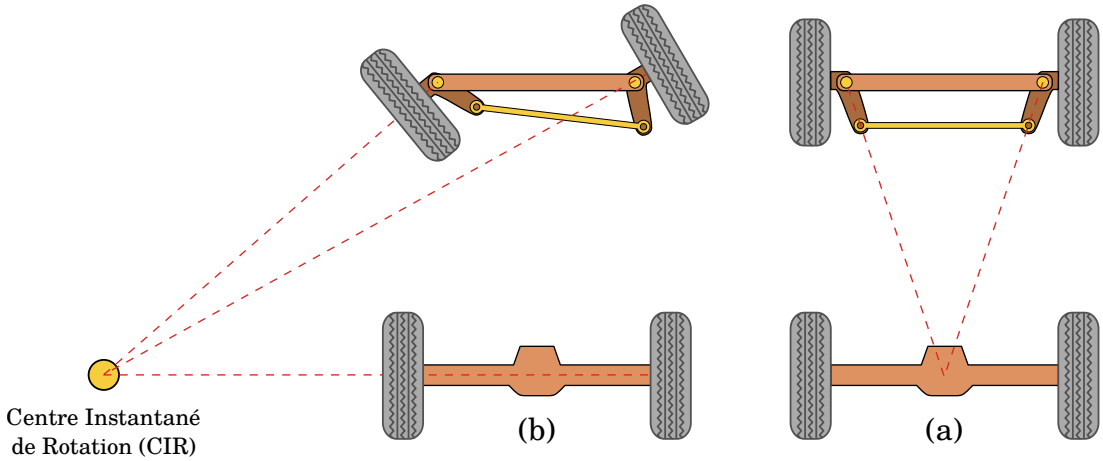


Figure A.1 – Illustration d'un véhicule muni d'un mécanisme d'Ackermann, (a) le système en état normale, (b) le système en état du braquage.

Les voitures modernes n'utilisent pas exactement la direction d'Ackermann, en partie parce qu'elle ignore les effets dynamiques et compliants importants, mais le principe est sain pour les manœuvres à basse vitesse. Certaines voitures de course utilisent la géométrie inverse d'Ackermann pour compenser la grande différence d'angle de glissement entre les pneus avant interne et externe tout en virage à grande vitesse. L'utilisation d'une telle géométrie contribue à réduire la température des pneus lors des virages à grande vitesse, mais compromet les performances à basse vitesse.

A.4 Les transformations géométriques

En robotique, nous nous intéresse au transformations géométriques de type *"translation"* et *"rotation"*, les deux types sont détaillés ci-dessus :

A.4.1 Les translations

En géométrie, une translation est une transformation géométrique qui correspond à l'idée intuitive de *"glissement"* d'un objet, sans rotation, retournement ni déformation de cet objet.

En géométrie classique, la notion de translation est très fortement liée à celle de vecteur, qu'elle suit ou précède. Ainsi trouve-t-on la translation de vecteur $\vec{u}$ définie comme une transformation qui, à tout point M, associe le point M' tel que :

$$\overrightarrow{MM'} = \vec{u} \tag{A.1}$$

On dit alors que M' est le translaté de M. C'est l'image de M par cette translation.

Soit disant vecteur $\vec{u}$ et un point M les deux de dimension $\mathcal{N}$, la translation de vecteur $\vec{u} = \begin{pmatrix} u_1 & u_2 & \dots & u_\mathcal{N} \end{pmatrix}^t$, transforme le point $M = \begin{pmatrix} M_1 & M_2 & \dots & M_\mathcal{N} \end{pmatrix}^t$ en $M' = \begin{pmatrix} M'_1 & M'_2 & \dots & M'_\mathcal{N} \end{pmatrix}^t$ tel que :

$$M' = \vec{u} + M \tag{A.2}$$

$$= \begin{pmatrix} M'_1 \\ M'_2 \\ : \\ M'_\mathcal{N} \end{pmatrix} = \begin{pmatrix} u_1 \\ u_2 \\ : \\ u_\mathcal{N} \end{pmatrix} + \begin{pmatrix} M_1 \\ M_2 \\ : \\ M_\mathcal{N} \end{pmatrix} \tag{A.3}$$

A.4.2 Les rotations

Dans le plan vectoriel euclidien orienté, une rotation vectorielle est simplement définie par son angle φ. Sa matrice dans une base orthonormée directe est :

$$\begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix} \tag{A.4}$$

Autrement dit, un vecteur $\vec{U}$ de composantes $\begin{pmatrix} x & y \end{pmatrix}^t$ a pour image le vecteur $\vec{V}$ de composantes $\begin{pmatrix} x' & y' \end{pmatrix}^t$ que l'on peut calculer avec l'égalité matricielle :

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{A.5}$$

C'est-à-dire que l'on a :

$$x' = x\cos\varphi - y\sin\varphi \tag{A.6}$$
$$y' = x\sin\varphi + y\cos\varphi \tag{A.7}$$

A.5 Coordonnées polaires

Les coordonnées polaires sont un système de coordonnées à deux dimensions, dans lequel chaque point du plan est entièrement déterminé par un angle φ et une distance r. Ce système est particulièrement utile dans les situations où la relation entre deux points est plus facile à exprimer en termes d'angle et de distance.

Les deux coordonnées polaires r et φ peuvent être converties en coordonnées cartésiennes x et y en utilisant les fonctions trigonométriques sinus et cosinus (voir la *figure A.2*) :

$$x = r\cos\theta, \quad y = r\sin\theta \tag{A.8}$$

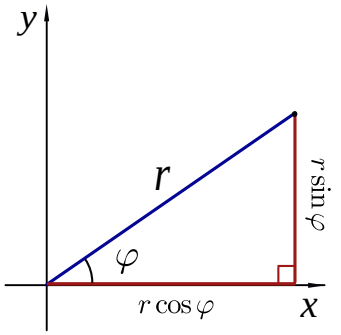


FIGURE A.2 – Conversion des coordonnées polaires en coordonnées cartésiennes.

Annexe B

Robot Operating System (ROS)

ROS a été développé à l'origine en 2007 sous le nom de *switchyard* par le Stanford Artificial Intelligence Laboratory dans le cadre du projet *Stanford AI Robot* STAIR (STanford AI Robot). La description de STAIR sur sa page d'accueil dite :

> *Notre plate-forme unique pour la robotique intégrera les méthodes tirées de tous les domaines de l'intelligence artificielle, y compris l'apprentissage par machine, la vision, la navigation, la planification, le raisonnement et le traitement de la parole / du langage naturel.*
>
> *Cela contraste nettement avec la tendance de 30 ans sur les sous-domaines fragmentés de l'IA et sera un moyen de conduire la recherche vers une véritable IA intégrée.*

FIGURE B.1 – Logo officiel du projet ROS.

ROS est un système d'exploitation complet pour la robotique de service. En effet, ROS est un ***méta-système d'exploitation***, quelque chose entre le système d'exploitation et le middleware. Il fournit des services proches d'un système d'exploitation (abstraction du matériel, gestion de la concurrence, des processus...) mais aussi des fonctionnalités de haut niveau (appels asynchrones, appels synchrones, base centralisée de données, système de paramétrage du robot...).

Une vue globales des composantes du système d'exploitation ROS sont schématisés dans la *figure B.2*.

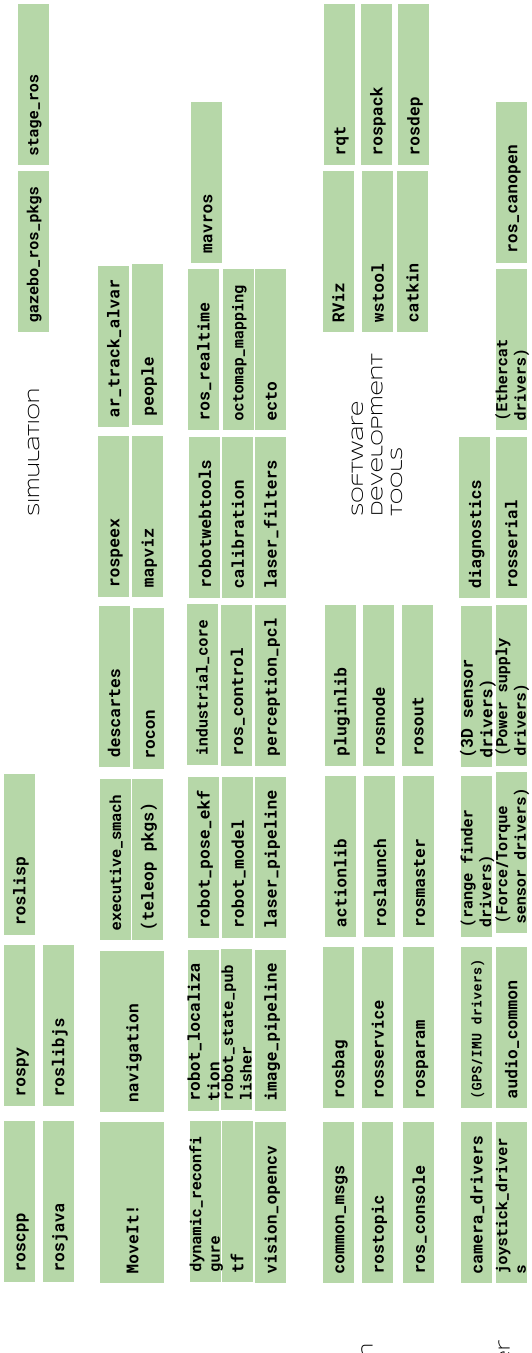


FIGURE B.2 – Les composants de méta-système d'exploitation ROS.

B.1 Bibliothèques du client ROS

Une bibliothèque du client ROS (ROS Client Library) est une bibliothèque qui facilite la tâche du programmeur ROS. Elle prend beaucoup de concepts ROS et les rend accessibles via le code.

En général, ces bibliothèques vous permettent d'écrire des nœuds ROS, de publier (`publish`) et d'abonner (`subscribe`) aux `topics`, d'écrire et d'appeler des services, et d'utiliser le serveur de paramètres. Une telle bibliothèque peut être implémentée dans n'importe quel langage de programmation, bien que ROS fournit un support officiel robuste des langages de programmation C++ (`roscpp` [1]), Python (`rospy` [2]) et Lisp (`roslisp` [3]), d'autres langages sont supportés par la communauté de ROS, à travers les bibliothèques de client comme `rosjava`, `rosr`, `rosnodejs`, `roscs`, `rosgo`, `rosruby`, `roslua`... et d'autres bibliothèques [4].

B.2 Philosophie du ROS

ROS a trois niveaux de concepts : le niveau du *système de fichiers (ROS File-system Level)*, le niveau du *graphe de calcul (ROS Computation Graph Level)* et le niveau *communautaire (ROS Community Level)*. Ces niveaux et ces concepts sont résumés ci-dessous.

B.3 Niveau du système de fichiers ROS

Les concepts de niveau de système de fichiers couvrent principalement les ressources ROS que vous rencontrez sur le disque, telles que :

Les paquets : (*Packages*) sont l'unité principale pour l'organisation de logiciels, ils sont l'élément le plus atomique dans ROS.

Un paquet peut contenir des processus d'exécution ROS (nœuds - `nodes`), une bibliothèque dépendant de ROS, des données, des fichiers de configuration, ou tout autre élément organisé de manière ultime.

Les méta-paquets : (*Metapackages*) sont des paquets spécialisés qui ne servent qu'à représenter un groupe d'autres paquets liés.

1. http://wiki.ros.org/roscpp, Consulté le 19 juillet 2017.
2. http://wiki.ros.org/rospy, Consulté le 19 juillet 2017.
3. http://wiki.ros.org/roslisp, Consulté le 19 juillet 2017.
4. http://wiki.ros.org/ClientLibrarie, Consulté le 19 juillet 2017.

Le manifeste du paquet : (*Manifest*, le fichier `package.xml` dans un paquet) fournit des méta-données sur un paquet, y compris son nom, sa version, sa description, ses informations de licence, ses dépendances et d'autres méta-informations.

La structure de fichier manifeste `package.xml` est définie dans `REP-0127` [5].

Les dépôts : un dépôt (*Repository*) est une collection de paquets qui partagent un système de contrôle de version (VCS) commun. Les paquets qui partagent un VCS partagent la même version et peuvent être diffusés ensemble à l'aide de l'outil d'automatisation `bloom` [6] de `catkin`.

Les types des messages (msg) : contient la description des messages, définissent les structures de données pour les messages envoyés dans ROS, stockés dans :
`my_package/msg/MyMessageType.msg`

Les type des services (srv) : contient la descriptions de service, définissent les structures de données de demande et de réponse pour les services dans ROS, stockés dans :
`my_package/srv/MyServiceType.srv`

B.4 Niveau de traitement du ROS

Sous ROS, le *Computation Graph* est le réseau peer-to-peer des processus qui traitent ensemble les données. Les concepts de *Computation Graph* de base sont des nœuds *nodes*, le nœud principale (*Master*), les serveurs de paramètres (*Parameter Server*), les messages (*Messages*), les services (*Services*), les *topics* et les *bags*, qui fournissent tous des données au Graphique de différentes façons.

Ces concepts sont implémentés dans le dépôt (repository) `ros_comm`.

B.4.1 Les nœuds

Les nœuds (`nodes`) sont des processus qui effectuent le traitement. ROS est conçu pour être modulaire à une échelle fine ; Un système de contrôle de robot comprend généralement de nombreux nœuds. Par exemple, un nœud contrôle le capteur laser, un autre nœud contrôle les moteurs des roues, un nœud effectue la localisation, un nœud effectue la planification du chemin, un nœud fournit une

5. http://www.ros.org/reps/rep-0127.html, Consulté le 19 juillet 2017.
6. http://wiki.ros.org/bloom, Consulté le 19 juillet 2017.

vue graphique du système, etc. Un nœud ROS est écrit avec l'utilisation d'une bibliothèque client ROS, comme `roscpp` ou `rospy`.

Le lancement d'un nœud sous ROS se fait par la commande `rosrun` ou bien `roslaunch`.

B.4.2 ROS Master

L'un des objectifs de base de ROS est de permettre aux roboticiens de concevoir des logiciels comme une collection de petits programmes, principalement indépendants (les nœuds), qui fonctionnent tous en même temps. Pour que cela fonctionne, ces nœuds doivent pouvoir communiquer entre eux. La partie de ROS qui facilite cette communication s'appelle le *ROS Master*.

Le lancement de *ROS Master* se fait par la commande `roscore`.

B.4.3 Le serveur de paramètres

Le serveur de paramètres (*Parameter Server*) permet de stocker des données par clé dans un emplacement central. Il fait actuellement partie du *Master*.

B.4.4 Les messages

Les nœuds communiquent entre eux en transmettant des messages. Un message est simplement une structure de données, comprenant des champs saisis.

Les types primitifs standard (les entiers `int`, les nombres en virgule flottante `float`, les boolean `bool`, ... etc.) sont pris en charge, de même que des tableaux de types primitifs. Les messages peuvent inclure des structures et des tableaux arbitrairement imbriqués (tout comme les structures du langage C).

B.4.5 Les sujets

Les messages sont acheminés avec la sémantique de publication / souscrire (`publish` / `subscribe`). Un nœud envoie un message en le publiant sur un sujet (`topic`) donné. Le sujet est un nom utilisé pour identifier le contenu du message. Un nœud qui s'intéresse à un certain type de données s'abonnera (`subscribe`) au sujet approprié. Il peut y avoir plusieurs éditeurs et abonnés simultanés pour un seul sujet, et un seul nœud peut publier et/ou s'abonner à plusieurs sujets.

Logiquement, on peut penser à un sujet comme un bus de message fortement typé. Chaque bus a un nom et n'importe qui peut se connecter au bus pour envoyer ou recevoir des messages tant qu'ils sont du bon type.

B.4.6 Les sacs

Les sacs (`bags`) sont un format pour sauvegarder et lire les données du message ROS. Les sacs sont un mécanisme important pour stocker des données, telles que des données de capteurs, qui peuvent être difficiles à collecter, mais nécessaires pour développer et tester des algorithmes.

B.5 Niveau communautaire de ROS

Les concepts ROS Community Level sont des ressources ROS qui permettent aux communautés séparées d'échanger des logiciels et des connaissances. Les principaux concepts du niveau communautaire sont :

B.5.1 Les distributions

Les distributions du ROS sont des collections d'applications versionnées que vous pouvez installer. Les distributions jouent un rôle similaire aux distributions Linux : elles facilitent l'installation d'une collection de logiciels, et elles maintiennent également des versions cohérentes sur un ensemble de logiciels.

B.5.2 Les dépôts

ROS repose sur un réseau fédéré de dépôts de code, où différentes institutions peuvent développer et publier leurs propres composants de logiciel de robot.

B.5.3 Le Wiki ROS

Le Wiki ROS est le principal forum pour documenter des informations sur ROS. Toute personne peut s'inscrire à un compte et apporter sa propre documentation, fournir des corrections ou des mises à jour, écrire des tutoriels et plus encore.

B.6 Python

Comme c'est déjà cité, le ROS supporte le langage de programmation Python officiellement a travers la bibliothèque du client ROS `rospy`.

Python est un langage de programmation objet, multi-paradigme et multi-plateformes. Il favorise la programmation impérative structurée, fonctionnelle et orientée objet. Il est doté d'un typage dynamique fort, d'une gestion automatique de la mémoire par ramasse-miettes et d'un système de gestion d'exceptions ; il est ainsi similaire à Perl, Ruby, Scheme, Smalltalk et Tcl.

Le langage Python est placé sous une licence libre proche de la licence BSD [7] et fonctionne sur la plupart des plates-formes informatiques, des supercalculateurs aux ordinateurs centraux [8], de Windows à Unix avec notamment GNU/Linux en passant par macOS, ou encore Android, iOS, et aussi avec Java ou encore .NET. Il est conçu pour optimiser la productivité des programmeurs en offrant des outils de haut niveau et une syntaxe simple à utiliser.

FIGURE B.3 – Logo officiel de l'implémentation référentielle du langage de programmation Python (CPython).

B.7 La bibliotheque NumPy

NumPy est une bibliothèque logicielle open source qui offre une extension du langage de programmation Python, destinée à manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux.

Plus précisément, cette bibliothèque fournit de multiples fonctions permettant notamment de créer directement un tableau depuis un fichier ou au contraire de sauvegarder un tableau dans un fichier, et manipuler des vecteurs, matrices et polynômes.

7. http://www.python.org/psf/license/, Consulté le 19 juillet 2017.
8. https://www.python.org/download/other/, Consulté le 19 juillet 2017.

Bibliographie

[1] Abbas, I. M. Contribution au développement d'une application de télérobotique. Master's thesis, Faculté des sciences, Université M'hamed Bougara de Boumerdes, 2015.

[2] An, X., Hong, W., and Xia, H. Research on binocular vision absolute localization method for indoor robots based on natural landmarks. In *Chinese Automation Congress (CAC), 2015* (2015), IEEE, pp. 604–609.

[3] Ayache, N., and Faugeras, O. D. Building, registrating, and fusing noisy visual maps. *The International Journal of Robotics Research 7*, 6 (1988), 45–65.

[4] Ayache, N., and Faugeras, O. D. Maintaining representations of the environment of a mobile robot. *IEEE transactions on Robotics and Automation 5*, 6 (1989), 804–819.

[5] Bailey, T., and Durrant-Whyte, H. Simultaneous localization and mapping : part i. *IEEE Robotics Automation Magazine 13*, 2 (June 2006), 99–110.

[6] Bailey, T., and Durrant-Whyte, H. Simultaneous localization and mapping (slam) : Part ii. *IEEE Robotics Automation Magazine 13*, 3 (Sept 2006), 108–117.

[7] Barrios, P., Adams, M., Leung, K., Inostroza, F., Naqvi, G., and Orchard, M. E. Metrics for evaluating feature-based mapping performance. *IEEE Transactions on Robotics 33*, 1 (2017), 198–213.

[8] Besl, P. J., and McKay, N. D. Method for registration of 3-d shapes. In *Robotics-DL tentative* (1992), International Society for Optics and Photonics, pp. 586–606.

[9] Biber, P., Fleck, S., and Strasser, W. A probabilistic framework for robust and accurate matching of point clouds. In *Joint Pattern Recognition Symposium* (2004), Springer, pp. 480–487.

[10] Biber, P., and Strasser, W. The normal distributions transform : A new approach to laser scan matching. In *Intelligent Robots and Systems,*

2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on (2003), vol. 3, IEEE, pp. 2743–2748.

[11] Blackwell, D. Conditional expectation and unbiased sequential estimation. *The Annals of Mathematical Statistics* (1947), 105–110.

[12] Bouraine, S. Contribution à la localisation dynamique du robot mobile d'intérieur b21r en utilisant la plateforme multi sensorielle. Master's thesis, Université de Saad Dahleb de Blida, Faculté des sciences d'ingénieur, 2007.

[13] Bouraine, S. Localisation du robot mobile robucar basée sur la méthode des grilles d'occupation et le filtre de kalman étendu. Tech. rep., Centre de Développement des Technologies Avancées, Algérie, 2009.

[14] Bouraine, S., Djekoune, A. O., and Azouaoui, O. Hybrid localization approach of a bi-steerable mobile robot based on grids matching and extended kalman filter. In *2008 11th International IEEE Conference on Intelligent Transportation Systems* (Oct 2008), pp. 1136–1141.

[15] Bücken, S. T. A. Learning maps for indoor mobile robot navigation. Tech. rep., 1996.

[16] Burguera, A., González, Y., and Oliver, G. The likelihood field approach to sonar scan matching. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on* (2008), IEEE, pp. 2977–2982.

[17] Burguera, A., González, Y., and Oliver, G. A probabilistic framework for sonar scan matching localization. *Advanced Robotics 22*, 11 (2008), 1223–1241.

[18] Casella, G., and Robert, C. P. Rao-blackwellisation of sampling schemes. *Biometrika 83*, 1 (1996), 81–94.

[19] Clerc, M., and Siarry, P. Une nouvelle métaheuristique pour l'optimisation difficile : la méthode des essaims particulaires. *J3eA 3* (2004), 007.

[20] Cornejo, A., and Nagpal, R. Distributed range-based relative localization of robot swarms. In *Algorithmic Foundations of Robotics XI*. Springer, 2015, pp. 91–107.

[21] Cox, I. J. Blanche-an experiment in guidance and navigation of an autonomous robot vehicle. *IEEE Transactions on robotics and automation 7*, 2 (1991), 193–204.

[22] Csorba, M., Uhlmann, J. K., and Durrant-Whyte, H. F. New approach to simultaneous localization and dynamic map building. In *Aerospace/Defense Sensing and Controls* (1996), International Society for Optics and Photonics, pp. 26–36.

[23] De Palma, D., Indiveri, G., and Parlangeli, G. Multi-vehicle relative localization based on single range measurements. *IFAC-PapersOnLine 48*, 5 (2015), 17–22.

[24] Delobel, L., Aufrere, R., Chapuis, R., and Chateau, T. Efficient fleet absolute localization and environment re-mapping. *IFAC-PapersOnLine 49*, 15 (2016), 236–241.

[25] Drocourt, C. *Localisation et modélisation de l'environnement d'un robot mobile par coopération de deux capteurs omnidirectionnels*. PhD thesis, Université de Technologie de Compiègne, Centre de Robotique, d'Electrotechnique et d'Automatique, 2002.

[26] Durrant-Whyte, H. F. Uncertain geometry in robotics. *IEEE Journal on Robotics and Automation 4*, 1 (Feb 1988), 23–31.

[27] El Dor, A. *Improvement of particle swarm optimization algorithms : applications in image segmentation and electronics*. Theses, Université Paris-Est, Dec. 2012.

[28] El Hamzaoui, O. *Simultaneous Localization and Mapping for a mobile robot with a laser scanner : CoreSLAM*. PhD thesis, Ecole Nationale Supérieure des Mines de Paris, Sept. 2012.

[29] Elfes, A. *Occupancy Grids : A Probabilistic Framework for Robot Perception and Navigation*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 1989. AAI9006205.

[30] Elfes, A. Using occupancy grids for mobile robot perception and navigation. *Computer 22*, 6 (June 1989), 46–57.

[31] Filliat, D. *Robotique Mobile*. École Nationale Supérieure de Techniques Avancées ParisTech, 2012.

[32] Fraichard, T. Contributions à la planification de mouvement. Mémoire d'Habilitation à Diriger des Recherches, Mar. 2006.

[33] Frappier, G. Système inertiels de navigation pour robots mobiles. In *Séminaire "Les robots mobiles" EC2, Paris* (1990).

[34] Gacôgne, L. Comparaison entre pso et autres heuristiques d'optimisation avec opérateurs implicites. *Séminaire Optimisation par Essaim Particulaire* (2003), 1–12.

[35] Gérossier, F. *Localisation et cartographie simultanées en environnement extérieur à partir de données issues d'un radar panoramique hyperfréquence*. PhD thesis, Université Blaise Pascal - Clermont-Ferrand II, June 2012.

[36] Guertel, I., Schillaci, G., and Hafner, V. V. Using proprioceptive information for the development of robot body representations. In *Development and Learning and Epigenetic Robotics (ICDL-EpiRob), 2016 Joint IEEE International Conference on* (2016), IEEE, pp. 172–173.

[37] Gutmann, J.-S., and Konolige, K. Incremental mapping of large cyclic environments. In *Computational Intelligence in Robotics and Automation, 1999. CIRA'99. Proceedings. 1999 IEEE International Symposium on* (1999), IEEE, pp. 318–325.

[38] Gutmann, J.-S., and Schlegel, C. Amos : Comparison of scan matching approaches for self-localization in indoor environments. In *Advanced Mobile Robot, 1996., Proceedings of the First Euromicro Workshop on* (1996), IEEE, pp. 61–67.

[39] Guyonneau, R. *Set-Membership Approaches for Mobile Robot Localization.* PhD thesis, Université d'Angers, Nov. 2013.

[40] Habumuremyi, J.-C. Models of a wheeled robot named robudem and design of a state feedback controller for its posture tracking. In *Proceedings of the 15th International Symposium on Measurement and Control in Robotics* (2005).

[41] Kalman, R. E., et al. A new approach to linear filtering and prediction problems. *Journal of basic Engineering 82*, 1 (1960), 35–45.

[42] Karakaya, S., Ocak, H., Küçükyildiz, G., and Kilinç, O. A hybrid indoor localization system based on infra-red imaging and odometry. In *Proceedings of the International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV)* (2015), The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), p. 224.

[43] Kellner, D., Barjenbruch, M., Dietmayer, K., Klappstein, J., and Dickmann, J. Joint radar alignment and odometry calibration. In *Information Fusion (Fusion), 2015 18th International Conference on* (2015), IEEE, pp. 366–374.

[44] Kennedy, J., and Eberhart, R. Particle swarm optimization. Proceedings Of Ieee International Conference On Neural Networks.

[45] Lemouari, A., and Benmohamed, M. Une méthode d'optimisation par essaimes particulaire pour le problème de collectes et de livraisons (pcl). In *CIIA* (2009).

[46] Leonard, J. J., and Durrant-Whyte, H. F. Simultaneous map building and localization for an autonomous mobile robot. In *Intelligent Robots and Systems' 91.'Intelligence for Mechanical Systems, Proceedings IROS'91. IEEE/RSJ International Workshop on* (1991), Ieee, pp. 1442–1447.

[47] Li, X., and Qiu, H. An effective laser-based approach to build topological map of unknown environment. In *Robotics and Biomimetics (ROBIO), 2015 IEEE International Conference on* (2015), IEEE, pp. 200–205.

[48] Lourakis, M., and Hourdakis, E. Planetary rover absolute localization by combining visual odometry with orbital image measurements. In *Proc. of the 13th Symposium on Advanced Space Technologies in Automation and Robotics (ASTRA'15). European Space Agency* (2015).

[49] Lu, F., et al. Robot pose estimation in unknown environments by matching 2d range scans. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on* (1994), IEEE, pp. 935–938.

[50] Lu, F., and Milios, E. Globally consistent range scan alignment for environment mapping. *Autonomous robots 4*, 4 (1997), 333–349.

[51] Magnusson, M. *The three-dimensional normal-distributions transform : an efficient representation for registration, surface analysis, and loop detection.* PhD thesis, Örebro universitet, 2009.

[52] Maiseli, B., Gu, Y., and Gao, H. Recent developments and trends in point set registration methods. *Journal of Visual Communication and Image Representation* (2017).

[53] Metropolis, N., and Ulam, S. The monte carlo method. *Journal of the American statistical association 44*, 247 (1949), 335–341.

[54] Montemerlo, M., Thrun, S., Koller, D., Wegbreit, B., et al. Fastslam : A factored solution to the simultaneous localization and mapping problem. In *Aaai/iaai* (2002), pp. 593–598.

[55] Moravec, H. P. Sensor fusion in certainty grids for mobile robots. *AI magazine 9*, 2 (1988), 61.

[56] Moutarlier, P., and Chatila, R. Stochastic multisensory data fusion for mobile robot location and environment modeling. In *5th Int. Symposium on Robotics Research* (1989), vol. 1, Tokyo.

[57] Mur-Artal, R., and Tardós, J. D. Probabilistic semi-dense mapping from highly accurate feature-based monocular slam. In *Robotics : Science and Systems* (2015).

[58] Olson, E. B. Real-time correlative scan matching. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on* (2009), IEEE, pp. 4387–4393.

[59] Pei, Y., and Kleeman, L. Online robot odometry calibration over multiple regions classified by floor colour. In *Mechatronics and Automation (ICMA), 2015 IEEE International Conference on* (2015), IEEE, pp. 2589–2596.

[60] Pomerleau, F., Colas, F., Siegwart, R., et al. A review of point cloud registration algorithms for mobile robotics. *Foundations and Trends® in Robotics 4*, 1 (2015), 1–104.

[61] Ramos, F., and Ott, L. Hilbert maps : scalable continuous occupancy mapping with stochastic gradient descent. *The International Journal of Robotics Research 35*, 14 (2016), 1717–1730.

[62] Rao, C. R. Information and accuracy attainable in the estimation of statistical parameters. *Bull. Calcutta Math. Soc 37*, 3 (1945), 81–91.

[63] Rao, R. V., and Savsani, V. J. *Advanced Optimization Techniques*. Springer London, London, 2012, pp. 5–34.

[64] RoboSoft. *Document technique du RobuCar*.

[65] Rosen, D. M., Mason, J., and Leonard, J. J. Towards lifelong feature-based mapping in semi-static environments. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on* (2016), IEEE, pp. 1063–1070.

[66] Saeedi, S., Paull, L., Trentini, M., and Li, H. Occupancy grid map merging for multiple robot simultaneous localization and mapping. *International Journal of Robotics and Automation 30*, 2 (2015), 149–157.

[67] Santos, J. M., Portugal, D., and Rocha, R. P. An evaluation of 2d slam techniques available in robot operating system. In *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)* (Oct 2013), pp. 1–6.

[68] Schmuck, P., Scherer, S. A., and Zell, A. Hybrid metric-topological 3d occupancy grid maps for large-scale mapping. *IFAC-PapersOnLine 49*, 15 (2016), 230–235.

[69] Serafin, J., and Grisetti, G. Using extended measurements and scene merging for efficient and robust point cloud registration. *Robotics and Autonomous Systems 92* (2017), 91–106.

[70] Siegwart, R., Nourbakhsh, I. R., and Scaramuzza, D. *Introduction to autonomous mobile robots*, 2nd ed. Intelligent Robotics and Autonomous Agents. MIT Press, 2011.

[71] Smith, R., Self, M., and Cheeseman, P. Estimating uncertain spatial relationships in robotics. *arXiv preprint arXiv :1304.3111* (2013).

[72] Smith, R. C., and Cheeseman, P. On the representation and estimation of spatial uncertainty. *The international journal of Robotics Research 5*, 4 (1986), 56–68.

[73] Teimouri, M., Salehi, M. E., and Meybodi, M. R. A hybrid localization method for a soccer playing robot. In *Artificial Intelligence and Robotics (IRANOPEN), 2016* (2016), IEEE, pp. 127–132.

[74] Thrun, S., Burgard, W., and Fox, D. A probabilistic approach to concurrent mapping and localization for mobile robots. *Autonomous Robots 5*, 3-4 (1998), 253–271.

[75] Vu, T.-D. *Vehicle Perception : Localization, Mapping with Detection, Classification and Tracking of Moving Objects*. PhD thesis, Institut National Polytechnique de Grenoble - INPG, Sept. 2009.

[76] Walter, E. *Méthodes numériques et optimisation, un guide du consommateur*. 2015.

[77] Weiss, G., and Puttkamer, E. v. A map based on laserscans without geometric interpretation. In *Intelligent Autonomous Systems* (1995), vol. 4, IOS Press, pp. 403–407.

[78] Wensing, P. M., Wang, A., Seok, S., Otten, D., Lang, J., and Kim, S. Proprioceptive actuator design in the mit cheetah : Impact mitigation and high-bandwidth physical interaction for dynamic legged robots. *IEEE Transactions on Robotics* (2017).

[79] Yue, Y., Wang, D., Senarathne, P., and Moratuwage, D. A hybrid probabilistic and point set registration approach for fusion of 3d occupancy grid maps. In *Systems, Man, and Cybernetics (SMC), 2016 IEEE International Conference on* (2016), IEEE, pp. 001975–001980.

[80] Zhang, L., Wang, H., and Yu, L. Enhancing model accuracy using odometry for relative location. In *Control Conference (CCC), 2016 35th Chinese* (2016), IEEE, pp. 6025–6031.

[81] Zhang, X.-b., Wang, C.-y., Fang, Y.-c., and Xing, K.-x. An extended kalman filter-based robot pose estimation approach with vision and odometry. In *Wearable Sensors and Robots*. Springer, 2017, pp. 539–552.

[82] Zhao, L., Huo, G., Wang, K., and Li, R. A multi-feature localization algorithm for mobile robots indoor environment mapping. In *Mechatronics and Automation (ICMA), 2016 IEEE International Conference on* (2016), IEEE, pp. 1732–1737.

[83] Zhao, Y., Liu, F., and Wang, R. Location technology of indoor robot based on laser sensor. In *Software Engineering and Service Science (ICSESS), 2016 7th IEEE International Conference on* (2016), IEEE, pp. 683–686.

[84] Zlotnik, D. E., and Forbes, J. R. Exteroceptive measurement filtering embedded within an so (3)-based attitude estimator. In *Decision and Control (CDC), 2016 IEEE 55th Conference on* (2016), IEEE, pp. 296–301.

Contribution à
La Localisation et la Cartographie Simultanées (SLAM) dans un Environnement Urbain Inconnu

ملخص — هذا العمل يعالج إشكاليات تحديد الموقع ورسم الخرائط بالنسبة لروبوت من نوع سيارة في بيئة خارجية غير معروفة مسبقًا. الروبوت مجهز بلاقط ليزر من نوع Sick LMS511 Pro ، بيانات الليزر هذه يتم استخدامها فيما بعد لإنجاز مهام تحديد الموقع ورسم الخرائط. في مثل هذه الحالة، لا يمكن فصل مشكلتي تحديد الموقع ورسم الخرائط عن بعض، ولهذا اخترنا مقاربة **تحديد الموقع ورسم الخرائط المتزامن** SLAM . الحل المقترح في عملنا هذا أطلقنا عليه اسم NDT-PSO ، وهو يرتكز على الـ Normal Distribution Transform (NDT) والـ Particle Swarm Optimization (PSO) ، الـ NDT هي طريقة مستعملة لتحديد الموقع ورسم الخرائط المتزامن وهي طريقة مستعملة لتحديد التحويلات الهندسية (التحويلات النقطية من انسحاب ودوران) بين مجموعتين من بيانات الليزر وذلك من خلال تقنية المحاذات Alignment الخاصة بها، تمثيلها للبيئة يرتكز على نمذجة كل النقاط ثنائية الأبعاد بمجموعة من دوال الكثافة الاحتمالية Probability Density Functions . في الطريقة المقترحة في عملنا هذا، قمنا باستخدام خوارزم PSO في مرحلة الاستمثال (التحسين) Optimization لإيجاد التحويلات النقطية الصحيحة والتي من خلالها يتم استنتاج موقع الروبوت ودرجة استدارته. قمنا ببرمجة الحل المقترح تحت نظام التشغيل ROS وباستخدام لغة البرمجة Python وقمنا بتجربته على الروبوت RobuCar في إطار مشروع للنقل الحضري ذاتي القيادة.

كلمات مفتاحية — روبوت، روبوت من نوع سيارة، تحديد الموقع، استمثال، تحسين، رسم الخرائط، تحديد الموقع ورسم الخرائط المتزامن.

Abstract — This work deals with the localization and mapping problem of a car-like mobile robot in an unknown urban environment. The robot is equipped with a *Sick LMS511 Pro* laser sensor, the laser data are exploited to accomplish the task of localization and mapping. In such a situation, the two problems can not be dissociated, therefore, we propose to adopt a Simultaneous Localization And Mapping (SLAM) approach. The developed solution is called NDT-PSO. It is based on the Normal Distribution Transformation (NDT) method and the Particulate Swarm Optimization (PSO) method. NDT is a SLAM method whose principle is to determine the geometric transformation between two successive laser scans using alignment techniques. Its representation of the environment is based on the modeling of the all 2D points reconstructed from a laser scan by a collection of local normal distributions. The PSO method is used in the transformation parameters optimization step , this parameters are used to determine the poses (positions and orientations) of the robot. The proposed algorithms are implemented on the Robot Operating System ROS using Python language and tested on the RobuCar mobile robot as part of an intelligent urban transport project.
Key words — Mobile robot, Localization, Mapping, SLAM, PSO, NDT, ROS, RobuCar, Car-like robot.

Résumé — Ce travail traite les problèmes de localisation et de cartographie pour un robot mobile de type voiture évoluant dans un environnement urbain inconnu. Le robot est équipé d'un capteur laser de type *Sick LMS511 Pro* dont les données sont exploitées pour l'accomplissement de ces tâches. Dans une telle situation, les deux problèmes ne peuvent pas être dissociés, c'est pourquoi, nous proposons d'adopter une approche de localisation et cartographie simultanées (SLAM). La solution développée dans ce manuscrit est nommée NDT-PSO. Elle est basée sur la méthode de la transformation de distribution normale (NDT) et la méthode d'optimisation par essaim particulaire (PSO). La NDT est une méthode SLAM dont le principe est de déterminer la transformation géométrique entre deux scans laser successives grâce à des techniques d'alignement. Sa représentation de l'environnement est basée sur la modélisation de tous les points 2D reconstruits à partir d'un scan laser par une collection de distributions normales locales. La méthode PSO est utilisée dans la phase d'optimisation des paramètres de transformation afin de déterminer les poses (positions et orientations) du robot. Les algorithmes proposés sont implémentés en langage Python sous le système ROS et testés sur le robot mobile RobuCar dans le cadre d'un projet de transport urbain intelligent.
Mots clés— Robot mobile, OES, PSO, Localisation, Cartographie, NDT, SLAM, ROS, RobuCar.

Abdelhak Bougouffa (abougouffa@fedoraproject.org)
Anis Hocine (hocine.anis.1992@gmail.com)
UMBB - FSI (exINGM)
CDTA - Division Robotique et Productique - Équipe NCRM