

Mémoire

2020

Remerciements

Ce sera là qu'on va parler des étapes d'évolution avec les parties prenantes dans la rédaction du mémoire

Abstract

Résumé

Sommaire

1	Introduction	12
1.1	État de l'art	12
1.2	Contexte	13
1.3	Les principales contributions du mémoire	13
1.4	Organisation du manuscrit	13
I	Généralités sur les systèmes SLAM	14
1	Introduction	15
2	Définition et Origines	16
3	Formulation	17
3.1	Localisation	17
3.2	Cartographie	18
3.2.1	Représentation de la carte	19
3.2.2	Approche Directe	19
3.2.3	Featured Based	20
3.2.4	Grid Based	20
3.2.5	L'approche topologique	21
3.2.6	L'approche hybride	22
3.2.7	Comparaison	22
3.3	Perception, Capteurs et Calib	22
3.3.1	Les systèmes de perception	22
3.3.2	Les capteurs proprioceptifs	23
3.3.3	Les capteurs extéroceptifs	23
3.4	Localisation et Cartographie simultanées	23
4	Le SLAM	25
4.1	Exemples de SLAM	25
4.2	Workflow du SLAM	25
4.2.1	Font End	26
4.2.2	Back End	29
4.3	Algorithmes	30
4.3.1	Filtre de Bayes	30
4.3.2	Filtre de Kalman	31
4.3.3	Filtre particulière	32
4.3.4	Maximum de Vraisemblance (MLE)	33
4.3.5	Comparaison	35
4.4	Défis courants avec SLAM	37
4.4.1	Exactitude des résultats	37
4.4.2	Problème de positionnement	37
4.4.3	Coût de calcul élevé	37

4.5 Conclusion	38
II Scan Matching	41
1 Généralités sur le Scan Matching	42
1.1 Point Cloud	42
1.2 Scan Matching	42
1.3 Le Scan Matching étape par étape	43
2 ICP	45
2.1 Taxonomie de l'ICP	46
2.1.1 Initialisation des points	46
2.1.2 Matching des nuages de points	47
2.1.3 Pondération des paires	48
2.1.4 Réjection des paires	49
2.1.5 Optimisation des erreurs	51
2.2 Variantes	53
2.3 Vanilla ICP	53
2.3.1 Generate example data	53
2.3.2 Correspondences computation	53
2.3.3 ICP based on SVD	53
2.4 IRLS-ICP	55
2.5 ICP Point à Plan	55
2.6 GICP	55
2.7 Comparaison	56
3 NDT	57
3.1 Représentation par des densités de probabilités	57
3.1.1 Collecte des points 2D	57
3.1.2 Calcul de la moyenne	57
3.1.3 Calcul de la covariance	58
3.2 L'alignement avec NDT	58
3.3 Le processus d'optimisation avec la méthode de Newton	59
III Simulations	63
1 Scan Matching en 2D	65
1.1 ICP Vanilla	65
1.2 IRLS-ICP	65
1.3 IRLS-ICP et Odométrie	66
1.4 IRLS-ICP et Pose Graph	66
1.5 NDT	67
1.6 Comparaison	67
1.6.1 Comparaison	67
1.6.2 Occupancy Grid Map	67
1.6.3 Comparaison	68
2 Scan Matching en 3D	70
2.1 ICP en 3D	70
2.2 ICP Vs. NDT	71
2.2.1 Scan Matching de 2 nuages de points	71
2.2.2 Comparaison	71
2.2.3 Scan Matching pour cartographier un environnement	71
2.2.4 Comparaison	72

IV Annexes	75
A Annexe A : Le filtre de Kalman	76
A.1 Le filtre de Kalman-Shcmidt	76
A.1.1 Équations de prédiction et de mise à jour	76
A.1.2 Équation de mise à jour des mesures	76
A.2 Le Filtre de Kalman étendu : EKF	77
A.2.1 Équation de prédiction et de mise à jour	77
A.2.2 Équations de mise à jour des mesures	77
B Annexe B : La Décomposition SVD	78
C Annexe C : Minimisation Point à Plan	79

Liste de Figures

1.1	See-Think-Act	12
3.1	L'idée de base du SLAM	17
3.2	La localisation	18
3.3	La cartographie	18
3.4	Carte en nuage de points	19
3.5	Carte featured-based	20
3.6	Grille d'occupation binaire	21
3.7	Carte topologique	22
3.8	Chaîne fonctionnelle	23
3.9	Représentation du problème SLAM	24
4.1	Exemple de Slam Overall	26
4.2	Vision SLAM : (a) Structure du mouvement; (b) Enregistrement de nuages de points pour RGB-D SLAM	27
4.3	LiDAR SLAM : (a) SLAM avec LiDAR 2D ; (b) SLAM avec 3D LiDAR	28
4.4	Exemple de Fusion	28
4.5	Robots Aspirateurs : (a) le Dyson utilise la fusion LiDAR et Vision ; (b) l'Ecovas qui utilise la fusion de deux caméras	29
4.6	Cycle du filtre de Kalman	31
4.7	Principe général de l'IML	33
4.8	Exemple Graph SLAM	35
4.9	Minimisation de l'erreur de mesure	39
4.10	Minimisation de l'erreur de mesure	40
4.11	Minimisation de l'erreur de mesure	40
1.1	Description d'un Nuage de Point	42
1.2	Description d'un Nuage de Point	42
1.3	Schamtisation du Scan Matching (Matlab)	44
2.1	L'organigramme général d'ICP	45
2.2	Méthodes de Matching	47
2.3	Exemple de BSP Tree 2D	48
2.4	tableau	49
2.5	icp rejection	50
2.6	(a) When two meshes to be aligned do not overlap completely (as is the case for most real-world data), allowing correspondences involving points on mesh boundaries can introduce a systematic bias into the alignment. (b) Disallowing such pairs eliminates many of these incorrect correspondences.	51
2.7	Minimisation point à point	52
2.8	Minimisation point à plan	52
2.9	bla	53
2.10	Première itération : (a) Make data centered; (b) Compute correspondences	54
2.11	Première itération	54

LISTE DE FIGURES

2.12	Itérations multiples	55
2.13	bla	55
2.14	bla	56
3.1	L'organigramme général de la NDT	62
1.1	scanMatchingExample	65
1.2	Pose Graph (a) estimation des nœuds pose et des arêtes les reliant; (b) Création de boucle en rajoutant une arête; (c) Rajouter un point nœud repère	66
1.3	pose graph full slam	67
1.4	2 Nuages de Point en 2D	67
1.5	Scan Matching 2D ICP	68
1.6	Scan Matching 2D NDT	68
1.7	occupancyGridMap NDT	69
1.8	grid map full slam	69
2.1	L'organigramme général d'ICP	70
2.2	Scan Matching : ICP Point à Point	71
2.3	Scan Matching : ICP Point à Point	71
2.4	Vérité Tarrain	72
2.5	État initial de Scan Matching	72
2.6	Scan Matching à base de l'ICP	73
2.7	État initial de Scan Matching	73
2.8	Scan Matching à base de l'NDT	73
2.9	Scan Matching à base de l'ICP	74

Liste de Tableaux

Liste des Acronymes

2D	Deux Dimensions
3D	Trois Dimensions
ADAS	Advanced Driver Assistance Systems
AIVI	Artificial Intelligence and Visual Interpretation
AMCL	Adaptative Monte Carlo Localisation
AHRS	Attitude & Heading Reference System
BoF	Bag of Features
BoVW	Bag of Visual Words
CPU	Central Processing Unit
DSO	Direct Sparse Odometry
DTAM	Dense Tracking and Mapping
EKF	Extended Kalman Filter
GM	Generalised Median
GNSS	Géolocalisation et Navigation par un Système de Satellites
GPS	Global Positioning System
ICP	Iterative Closest Point
IDC	Iterative Dual Correspondences
ICRA	International Conference on Robotics and Automation
IEEE	Institute of Electrical and Electronics Engineers
IML	Incremental Maximum Likelihood
IMU	Inertial measurement unit
INS	Information Network System
IR	Infra-Rouge
IROS	International Conference on Intelligent Robots and Systems
Kd tree	k -dimensional tree
KF	Kalman Filter
KLD	KullBack-Leibler Divergence
ML-AMCL	MultiLayer Adaptative Monte Carlo Localisation
LiDAR	Light Detection And Ranging
LRF	Laser Range Finder
LSD-SLAM	Large-Scale Direct Monocular SLAM
MATLAB	MATrix LABoratory
NDT	Normal Distributions Transform
ORB-SLAM	Oriented fast and Rotated Brief SLAM

PhD	Philosophiæ Doctor
PTAM	Parallel Tracking and Mapping
RANSAC	RANdom SAmple Consensus
RGB-D	Red Green Blue Depth
ROS	Robot Operating System
SfM	Structure for Motion
SIMD	Single Instruction Multiple Data
LF	Likelihood-field matching Instruction Multiple Data
SfM	Structure From Motion
SLAM	Simultaneous Localization and Mapping
SNSS	Global Navigation Satellite System
SVO	Semi-direct Visual Odometry
SVD	Singular Value Decomposition
ToF	Time of Flight
UAV	Unmanned Aerial Vehicle
vSLAM	Visual SLAM

1 Introduction

1.1 État de l'art

Durant les 3 dernières décennies, Les applications de la Localisation et Cartographie Simultanés (SLAM) ont pris du terrain dans les applications telles qu'en réalité augmentée[1] et conduite autonome[2]. Via SLAM, on peut non seulement estimer la trajectoire d'un objet en mouvement, mais en plus ça permet de reconstruire la scène environnante en 3D et en temps réel[3]. À ce jour une variété importante de SLAMs utilisant des capteurs -tel que lasers, IMUs, et caméras- ont été proposés[4].

Avant, les robots mobiles étaient principalement commandés par des opérateurs humains au lieu de disposer d'une navigation automatisée. Certains robots sont aptes à faire certaines tâches sans interventions humaines, mais en suivant des procédures prédéfinies, tels que le robot aspirateur. Avec le développement de la technologie et en suivant les bons algorithmes, les robots sont devenus de plus en plus intelligents, de sorte à ce qu'on attende d'eux de devenir capables de prendre en main des tâches complexes de manière autonome telle que l'assistance de personnes, le transport de marchandises, l'exploration de zones dangereuses, etc.

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[5] ou selon [6] le cycle see-think-act (voir la figure 1.1).

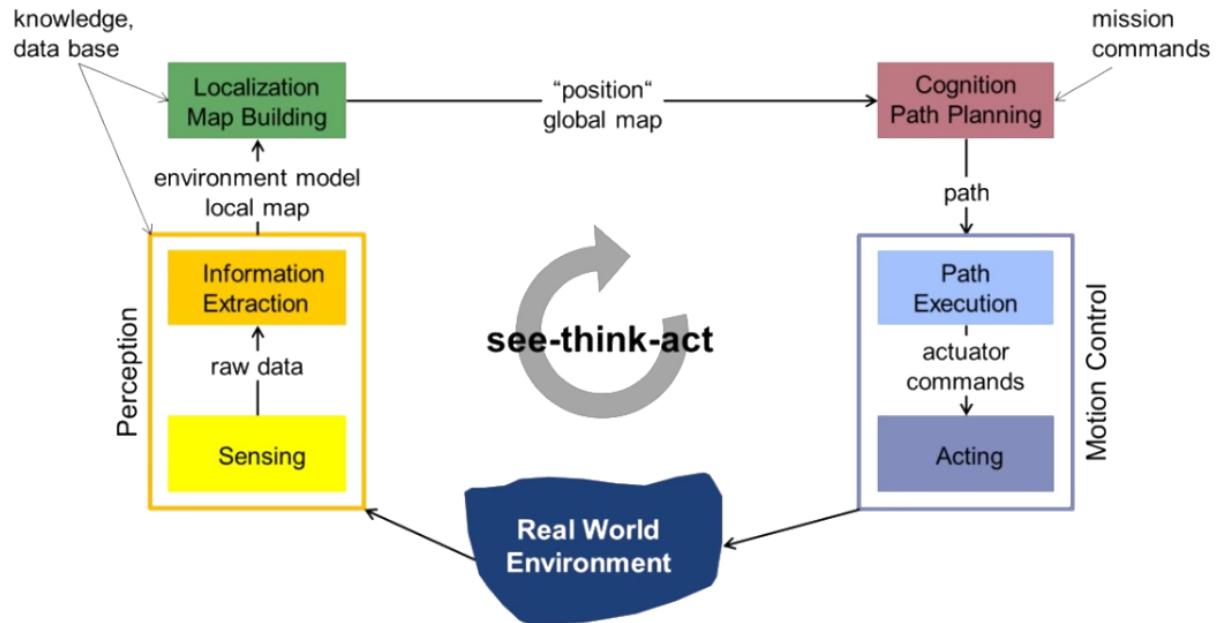


Figure 1.1: Le cycle perception, décision et action ou bien see-think-act[6]

Et donc, dans ce genre de scénarios, la capacité de Localisation et et de Cartographie Simultanées (SLAM) est une exigence de base pour les robots[7].

1.2 Contexte

Dans la vision du développement technologique et de la démocratisation de la robotique et de l'automatique, le milieu étudiant à Laghouat, bien qu'intéressé, il reste sceptique, et a des appréhensions face à l'engagement dans l'aventure de manière effective et efficace. L'objectif serait donc de fournir une brique à la base de l'édifice en fournissant un modèle réutilisable d'interface homme robot.

Mon objectif est de rédiger un mémoire claire, détaillée, et dans laquelle on peu aisément trouver les ressources nécessaires à retrouver les mêmes résultats.

Le thème étant de parvenir à retravailler des algorithmes déjà existants dans la localisation et le mapping simultanés pour qu'un dispositif électronique mobile parvienne à se situer dans l'espace.

1.3 Les principales contributions du mémoire

1.4 Organisation du manuscrit

Avec un parc de robots personnels de plus de 5 millions d'unités, et un cumul des ventes de robots professionnels s'élevant à \$13,2 milliards de dollars en 2009, la robotique de service va dépasser les \$22 milliards en 2013 et subir sa plus rapide accélération dans les 3 prochaines années.

La robotique de service désigne les robots qui rendent service à l'humain en se substituant à lui. La variété des tâches accomplies par ces robots est immense : on retrouve des robots de service dans presque tous les milieux hostiles : spatial, sous-marins, nucléaire, déminage, défense, sécurité, etc. En les retrouve également en remplacement des hommes dans les situations pénibles, dans le bâtiment ou l'agriculture. Ces robots sont dits de type professionnels : ils sont en général très cher unitairement et produits en nombre d'unités restreint par modèle. On retrouve ensuite des robots de service jusque dans nos maisons sous la forme d'aspirateur ou tondeuse automatique, ou encore sous forme de jouets, ou kit pour l'éducation. Leur prix est très faible pour entrer en masse chez les particuliers.

Chapitre I

Généralités sur les systèmes SLAM

1 Inroduction

Ce premier chapitre sera une introduction au SLAM, et ses principaux algorithmes(f)

2 Définition et Origines

Afin d'effectuer une navigation autonome, un robot qui se meut dans un environnement inconnu doit reconstruire incrémentalement une carte cohérente de son environnement tout en estimant sa position de sorte à éviter tout risque de collision ou de bug. Dans le cas de la localisation, la méthode probabiliste est largement appliquée dans le calcul du déplacement du robot qui est équipé de capteurs proprioceptifs tel qu'un encodeur à roues et un capteur d'inertie[7].

L'émergence du SLAM probabiliste a certainement été durant la conférence « IEEE Robotics and Automation Conference » en 1986 à San Francisco, Californie. Plusieurs chercheurs tentaient d'appliquer des méthodes théoriques d'estimation au problème de localisation et cartographie.

Les travaux de Smith et Cheeseman[8] et de Durant-Whyte[9] constituent une base des méthodes statistiques de description des relations entre les positions d'amers¹ dans un environnement et l'estimation de l'incertitude géométrique de la carte. L'un des éléments clés de ces travaux traite du degré de corrélation entre les estimations des positions des amers dans une carte[10].

L'inconvénient avec la méthode probabiliste, c'est qu'elle cumule les erreurs qui ne sont pas bornées au fur et à mesure que le robot se meut. Dans un environnement extérieur, on peut facilement palier à cette erreur en introduisant un système GPS afin d'estimer les erreurs accumulées. Cela dit, dans un environnement intérieur, le signal GPS est bloqué et est inutilisable en tant que référence globale. C'est pour cela que les capteurs externes (LRF, capteurs ultra-soniques, cameras, et capteurs RGBD) sont essentiels non seulement pour cartographier l'environnement, mais aussi afin de corriger la localisation. Inévitablement, les données qui sont obtenues via l'usage des capteurs externes afin d'explorer les alentours est aussi corrompu par les bruits. Dans le but de réaliser simultanément et correctement la cartographie et la localisation, les données acquises de différents capteurs doivent être fusionnées afin d'obtenir une estimation optimale. Ce problème est connu sous la dénomination de « Problème SLAM », qui a attiré l'attention de plusieurs chercheurs durant les dernières décennies[7], notamment grâce aux conférences internationales (ICRA, IROS...) qui attirent de plus en plus la communauté scientifique[10].

On a besoin d'une carte fiable pour se localiser, tandis qu'on a besoin de données de géolocalisation pour cartographier l'environnement. Ce problème aux dimensions de la question philosophique de la poule et de l'oeuf, fait du SLAM un problème assez compliqué à aborder. Une brève histoire de la recherche sur le SLAM et des solutions typiques de problèmes SLAM sont présentés dans[4]. Les problèmes majeurs dans la recherche SLAM, tel que la théorie de la complexité, l'association de données, et la représentation de l'environnement sont discutés dans[11]. Un aperçu général et une analyse détaillée du SLAM sont abordés dans [12] et [13].

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

3 Formulation

Le SLAM est composé d'un ensemble de méthodes permettant à un robot de construire une carte d'un environnement et en même temps de se localiser en utilisant cette carte. La trajectoire du véhicule et la position des amers dans la carte sont estimées au fur et à mesure, sans avoir besoin de connaissances a priori

Considérons un robot se déplaçant dans un environnement inconnu, en observant un certain nombre d'amers grâce à un capteur embarqué sur le robot. La figure 3.1 montre une illustration du problème.

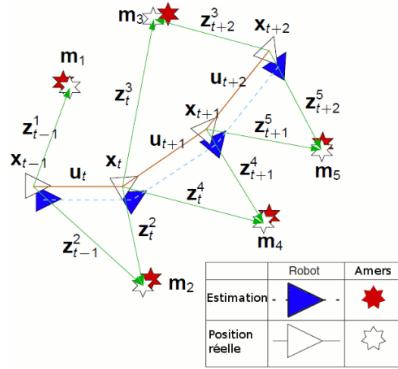


Figure 3.1: L'idée de base du SLAM

À l'instant k on définit les quantités suivantes :

- x_k : le vecteur d'état. Il contient la position du robot
- u_k : le vecteur de contrôle. L'application de u_k à l'instant $k - 1$ mène le robot de l'état x_{k-1} à l'état x_k
- m_i : vecteur contenant la position de l'amer i
- z_k : l'observation à l'instant k

On définit aussi les ensembles suivants :

- $X_{0:k} = \{x_0, x_1, \dots, x_k\} = \{X_{0:k-1}, x_k\}$: l'ensemble des vecteurs d'état jusqu'à l'instant k
- $U_{0:k} = \{u_0, u_1, \dots, u_k\} = \{U_{0:k-1}, u_k\}$: l'ensemble des vecteurs de commande jusqu'à l'instant k
- $Z_{0:k} = \{z_0, z_1, \dots, z_k\} = \{Z_{0:k-1}, z_k\}$: l'ensemble des observations jusqu'à l'instant k
- $m = \{m_1, m_2, \dots, m_n\}$: la carte de l'environnement contenant une liste d'objets statiques

3.1 Localisation

Le problème de localisation du robot consiste à estimer sa position dans un environnement donné, en utilisant l'historique de ses observations, l'historique des commandes et la connaissance de l'environnement. La figure 2.2 schématisé ce principe.

On peut analytiquement représenter cette opération par l'estimation de la probabilité de distribution :

$$P(x_k | Z_{0:k}, U_{0:k}, m)$$

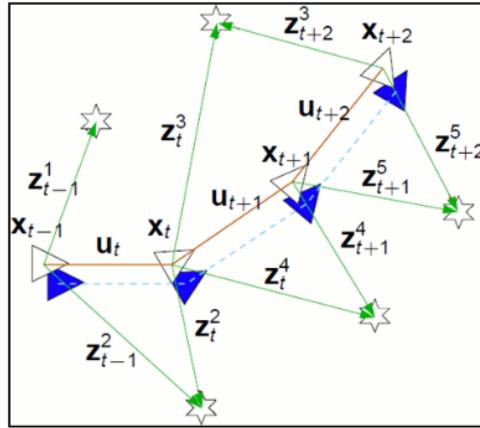


Figure 3.2: La localisation : le système cherche à estimer sa position en utilisant les informations sur l'environnement dont il dispose

L'estimation d'une telle quantité définit la localisation globale, dans la mesure où on utilise toutes les données de l'historique des observations et des commandes pour estimer la position. On obtient ainsi une estimation robuste de la position a posteriori, mais on augmente largement la complexité des calculs.

Afin de simplifier l'algorithme, on peut définir une localisation locale, où on utilise uniquement les données de l'instant ($k - 1$) pour estimer la position à l'instant k . On représente analytiquement cette opération par l'estimation de la distribution de probabilité :

$$P(x_k | z_{k-1}, u_{k-1}, x_{k-1}, m)$$

En utilisant cette méthode, on simplifie largement la complexité de l'algorithme, mais on risque de dévier de la position correcte du robot, sans pouvoir corriger cela.

3.2 Cartographie

Le problème de cartographie consiste à déterminer la carte d'un environnement, en utilisant les données des capteurs et l'historique des positions réelles du robot. Sur le schéma de la figure 2.3, le système connaît sa position exacte et estime la carte de l'environnement en utilisant les données de ses capteurs.

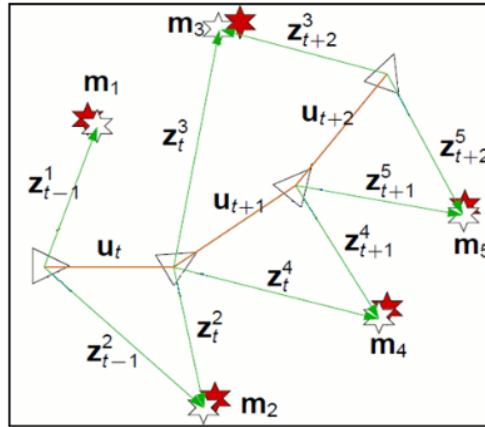


Figure 3.3: La cartographie : le système crée la carte de l'environnement en se basant sur sa position connue et les informations de ses capteurs

On peut exprimer cela analytiquement ainsi :

$$P(m_k | Z_{0:k}, X_{0:k})$$

Les positions réelles du robot peuvent être obtenues en utilisant des balises dans un environnement interne ou un récepteur GPS en externe. Ces positions doivent être précises et correctes afin d'obtenir une bonne cartographie.

3.2.1 Représentation de la carte

Le choix de la représentation de la carte de l'environnement est une étape importante dans le SLAM. On peut distinguer trois approches fondamentales de représentation de l'environnement :

- L'approche directe ;
- L'approche basée sur les caractéristiques géométriques (feature-based) ;
- L'approche basée sur une grille d'occupation (grid-based) ;
- L'approche topologique basée 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.) ;
- L'approche hybride.

La carte de l'environnement peut aussi être représentée par une approche topologique. Mais cette méthode n'est pas analysée, dans la mesure où elle est basée sur un partitionnement des cartes de types feature-based ou grid-based en régions cohérentes.

3.2.2 Approche Directe

La méthode de représentation directe de la carte de l'environnement est généralement adaptée à l'utilisation des capteurs Laser . Cette méthode utilise les données brutes des mesures du capteur pour représenter l'environnement sans aucune extraction d'ameurs ou de caractéristiques prédéfinies (lignes, coins, etc).

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. On peut ainsi construire une carte simplement en superposant les différents points de mesure. On obtient ainsi une représentation en nuage de points. La figure ?? montre un exemple d'une carte en nuage de points.

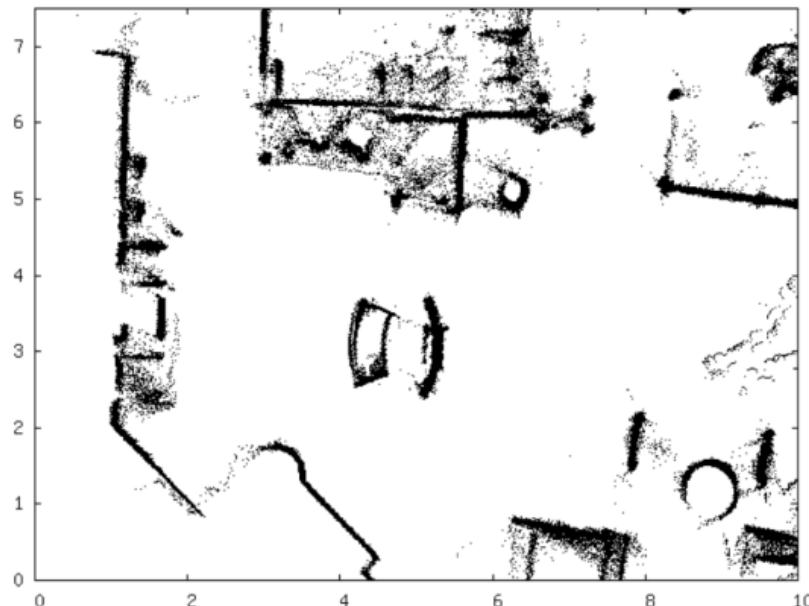


Figure 3.4: Carte en nuage de points

3.2.3 Featured Based

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 (un exemple d'une carte géométrique est illustré dans la figure 1.4).

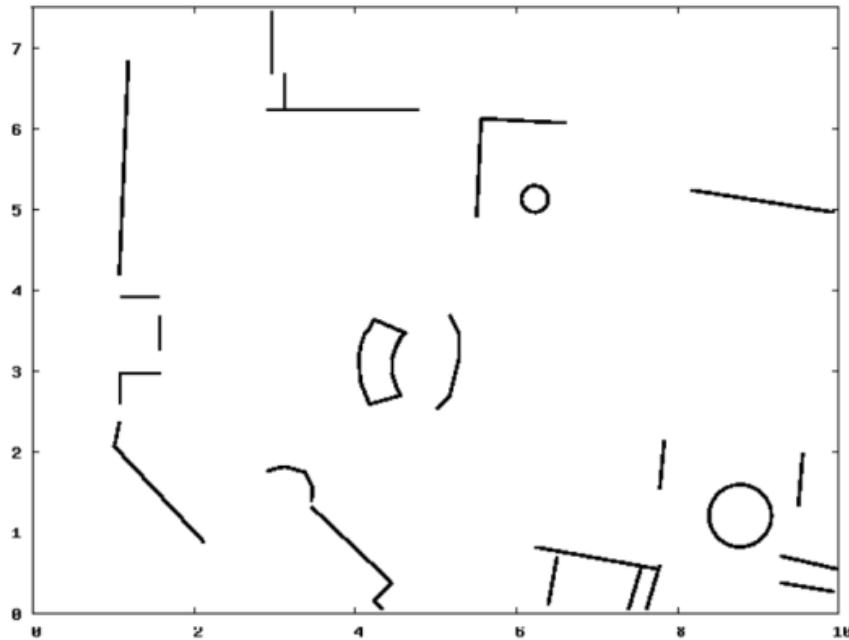


Figure 3.5: Carte basée sur l'extraction de caractéristiques géométriques de l'environnement

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 être 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.

Pour détecter les caractéristiques géométriques, plusieurs méthodes existent. Les plus connues sont :

- La méthode split-and-merge pour la détection des segments de lignes
- La transformation de Hough pour la détection des lignes ou des cercles
- RANSAC pour la détection des lignes ou des cercles.

Ce type de cartes est limité aux objets et formes modélisés et prédéfinis. Il est donc incompatible avec les environnements trop complexes et non structurés .

3.2.4 Grid Based

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. 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

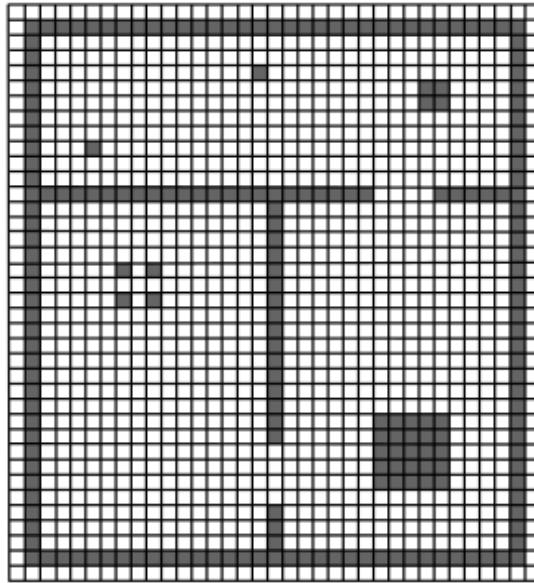


Figure 3.6: Exemple de grille d'occupation binaire. Les cellules blanches correspondent à des zones de l'environnement ne contenant aucun obstacle, les cellules grises correspondent à des zones occupées par des obstacle

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.

La mise à jour de l'état de chaque cellule se fait à la réception de nouvelles données.

On trouve dans la littérature plusieurs méthodes pour réaliser cette opération :

- Le filtrage bayésien : cette méthode a notamment été utilisée afin de modéliser la connaissance sur l'état de la cellule. Dans cette approche, on attribue à chaque cellule une probabilité entre 0 et 1. Une probabilité de 0 signifie qu'on a la certitude que la cellule est libre. La probabilité de 1 signifie qu'on a la certitude qu'elle est occupée.

- La théorie de Dempster-Shafer : dans cette approche on associe à chaque cellule deux poids probabiliste, P_f et P_e . P_f est une mesure de l'importance des informations fournies par les capteurs extéroceptifs qui vont dans le sens de l'hypothèse « la cellule est occupée ». P_e mesure l'importance des informations contraires. P_f et P_e varient entre 0 et 1, et leur somme est toujours inférieur ou égale à 1. Ainsi, un couple $(P_f, P_e) = (0, 0)$ indique l'absence d'information sur la cellule (c'est la valeur d'initialisation de la carte), alors que le couple $(P_f, P_e) = (1, 0)$ par exemple indique que la cellule est occupée avec certitude.

- La logique floue : l'état d'occupation de la cellule est modélisé par un ensemble flou. Chaque cellule peut exprimer à la fois deux états partiel

(E = vide et O = occupée) et le degré d'appartenance entre eux se détermine en utilisant la théorie des possibilités. Dans cette approche, chaque cellule peut avoir des données conflictuelles ($E \cap O$) fournies par le capteur, elle sera considérée comme une cellule ni libre ni occupée. Afin d'éliminer l'ambiguité sur l'état de ce type de cellule, on a besoin de plus de données en provenance des capteurs.

3.2.5 L'approche topologique

Les cartes topologiques 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[14].

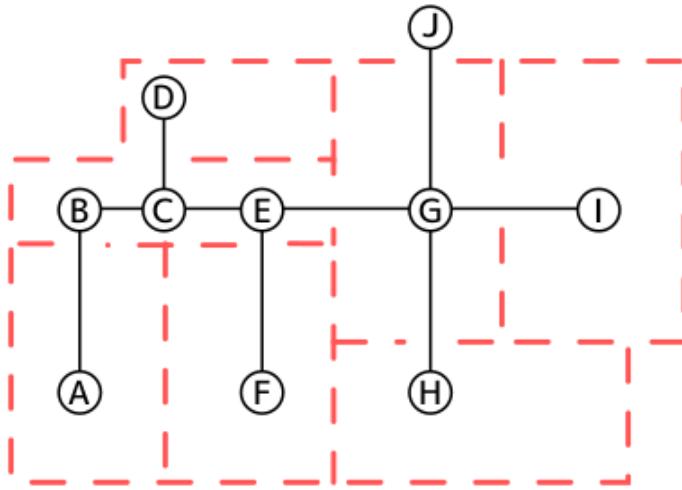


Figure 3.7: Exemple de carte topologique (en noir)

3.2.6 L'approche hybride

Les modélisations métriques (telle que l'approche géométrique et l'approche grid-based) 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 trouver des cartes hybride basées sur les grilles d'occupation à laquelle on ajoute des données topologiques afin de faciliter la planification de trajectoires[14].

3.2.7 Comparaison

Malgré sa simplicité, l'approche directe peut représenter tous les types des environnements. Mais elle présente l'inconvénient d'une grande consommation de mémoire et d'un manque de précision concernant la représentation de l'incertitude dans les mesures des capteurs.

Les cartes feature-based constituent une représentation compacte de l'environnement. Elles sont néanmoins basées sur l'extraction de caractéristiques connues et prédéfinies, ce qui limite leur utilisation aux environnements structurés et internes.

Les grilles d'occupation utilisent aussi une grande quantité de mémoire, mais elles offrent la possibilité de représenter tous les types d'environnements avec une prise en charge des caractéristiques des capteurs. Ce type d'approches est le mieux adaptés aux capteurs de profondeur comme les lasers ou les sonars.

3.3 Perception, Capteurs et Calib

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 suivante).

3.3.1 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.

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

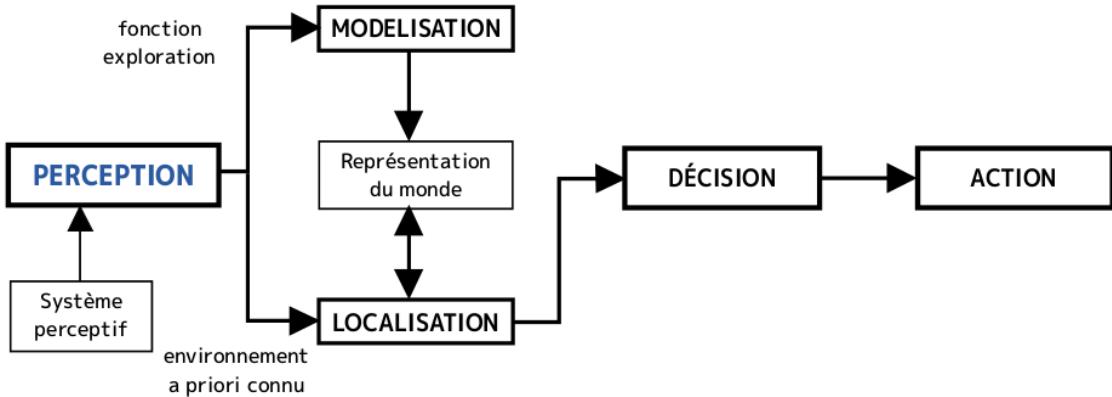


Figure 3.8: Chaîne fonctionnelle d'un système de navigation

- **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.

3.3.2 Les capteurs proprioceptifs

Ces capteurs fournissent, par intégration, des informations élémentaires sur les paramètres cinématiques ou dynamique du robot. Les informations sensorielles générées dans ce cadre sont généralement des vitesses, des accélérations, des angles de giration, des angles d'attitude. les capteurs proprioceptifs peuvent être regroupés en deux familles :

- **Les capteurs de déplacement :** qui comprennent (*les odomètres, les accéléromètres, les radars Doppler, les mesureurs optiques, etc.*). Cette catégorie permet de mesurer des déplacements élémentaires, des variations de vitesse ou d'accélération sur des trajectoires rectilignes ou curvilignes.
- **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.

3.3.3 Les capteurs extéroceptifs

Les capteurs extéroceptifs 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 utilisés 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 et la modélisation d'environnements*. Les principaux capteurs utilisés en robotique mobile sont[14] :

- **Les capteurs Passifs :** tel que les capteurs de contact (bumpers, capteurs d'effort), les magnétomètres, et les capteurs de vision.
- **Les capteurs actifs :** tel que les systèmes basés balises (localisation dans un repère fixe), les capteurs temps-de-vol (les sonars ou ultrasons et les télémètres lasers), les capteurs IR, les radars (ondes radio).

3.4 Localisation et Cartographie simultanées

La formulation probabiliste du problème de SLAM nécessite le calcul, à chaque instant k , de la quantité de probabilité :

$$P(x_k, m | Z_{0:k}, U_{0:k}, x_0)$$

Ce calcul est généralement effectué récursivement. On commence par la probabilité $P(x_{k-1}, m|Z_{0:k-1}, U_{0:k-1})$, puis on utilise le théorème de Bayes pour déduire la quantité $P(x_k, m|Z_{0:k}, U_{0:k})$ à partir de z_k et u_k . Afin d'effectuer cette déduction, nous avons besoin de connaître $P(x_k|x_{k-1}, u_k)$ et $P(z_k|x_k, m)$. Le terme $P(z_k|x_k, m)$ désigne le modèle d'observation. Il définit la probabilité d'avoir une mesure z_k connaissant l'état du véhicule x_k et une carte de l'environnement m . Le terme $P(x_k|x_{k-1}, u_k)$ définit le modèle de transition (modèle de mouvement du véhicule robotisé). Il permet de prévoir l'état x_k du système, qui ne dépend que de l'état précédent x_{k-1} et de la commande de contrôle appliquée. Dans ce cas, le processus de transition entre les états du système x_k est dit Markovien.

On définit donc le problème du SLAM en deux parties par les équations suivantes :

- Une partie de **mise-à-jour de la position** :

$$P(x_k, m|Z_{0:k}, U_{0:k}, x_0) = \int P(x_k|x_{k-1}, U_k) \times P(x_{k-1}, m|Z_{0:k-1}, U_{0:k-1}, x_0) dx_{k-1}$$

- Une partie de **mise-à-jour de l'observation** :

$$P(x_k, m|Z_{0:k}, U_{0:k}, x_0) = \frac{P(z_k|x_k, m) \times P(x_k, m|Z_{0:k-1}, U_{0:k}, x_0)}{P(z_k|Z_{0:k-1}, U_{0:k})}$$

En utilisant ainsi l'estimation a posteriori à l'instant $(k-1)$ donnée par le terme $P(x_{k-1}, m|Z_{0:k-1}, U_{0:k-1}, x_0)$, on peut calculer la prédiction et déduire ensuite l'estimation a posteriori à l'instant k . On a donc :

$$\begin{aligned} P(x_k, m|Z_{0:k}, U_{0:k}, x_0) &= \eta \times P(z_k|x_k, m) \times \int P(x_k|x_{k-1}, U_k) \\ &\quad \times P(x_{k-1}, m|Z_{0:k-1}, U_{0:k-1}, x_0) dx_{k-1} \end{aligned}$$

Sachant que η est une constante de normalisation dépendant du modèle d'observation et du modèle de transition.

$$\eta = \frac{1}{P(z_k|Z_{0:k-1}, U_{0:k})}$$

Cette structure du SLAM est représentée sur le schéma de la figure suivante. Sur ce schéma, les cercles gris représentent les données connues, tandis que les cercles blancs désignent les quantités à estimer.

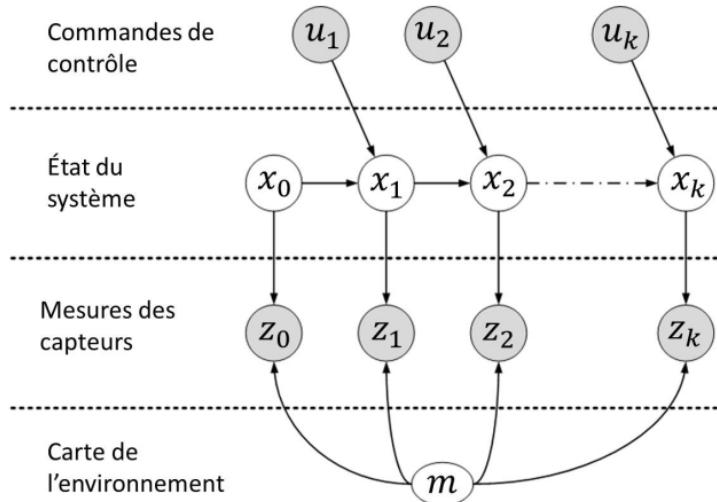


Figure 3.9: Représentation graphique du problème de SLAM

4 Le SLAM

SLAM (localisation et cartographie simultanées) est une méthode utilisée pour les véhicules autonomes qui vous permet de créer une carte et de localiser votre véhicule sur cette carte en même temps. Les algorithmes SLAM permettent au véhicule de cartographier des environnements inconnus. Les ingénieurs utilisent les informations de la carte pour effectuer des tâches telles que la planification de trajectoire et l'évitement d'obstacles.

SLAM fait l'objet de recherches techniques depuis de nombreuses années. Mais avec de vastes améliorations de la vitesse de traitement informatique et la disponibilité de capteurs à faible coût tels que des caméras et des télémètres laser, SLAM est maintenant utilisé pour des applications pratiques dans un nombre croissant de domaines.

Pour comprendre pourquoi SLAM est important, examinons certains de ses avantages et des exemples d'application.

4.1 Exemples de SLAM

Considérons un aspirateur robot domestique. Sans SLAM, il se déplacerait aléatoirement dans une pièce et ne pourra peut-être pas nettoyer toute la surface du sol, d'autant plus que cette approche consomme une énergie excessive, de sorte que la batterie s'épuise plus rapidement. D'autre part, les robots avec SLAM peuvent utiliser des informations telles que le nombre de tours de roue et des données provenant de caméras et d'autres capteurs d'imagerie pour déterminer la quantité de mouvement nécessaire. C'est ce qu'on appelle la localisation. Le robot peut également utiliser simultanément la caméra et d'autres capteurs pour créer une carte des obstacles dans son environnement et éviter de nettoyer deux fois la même zone. C'est ce qu'on appelle la cartographie.

SLAM est utile dans de nombreuses autres applications telles que la navigation dans une flotte de robots mobiles pour organiser des étagères dans un entrepôt, le stationnement d'une voiture autonome dans un endroit vide ou la livraison d'un colis en naviguant sur un drone dans un environnement inconnu. Permettant la localisation et le mappage simultanés avec d'autres tâches telles que la fusion de capteurs, le suivi d'objet, la planification de chemin et le suivi de chemin.

4.2 Workflow du SLAM

La solution SLAM est divisée en un *front end* et un *back end*. La partie frontale est chargée de la lecture des données de mesure acquise par les différents capteurs, et de les faire coïncider pour en construire la carte de l'environnement. La partie *back end* est responsable de la mise à jour de la carte à chaque nouvelle acquisition de données, et ce via la détection de fermetures de boucles et leur optimisation, le plus souvent, cette aspect du SLAM est fait via Pose Graph[15].

La figure 4.1 décrit un exemple de SLAM Overall ou complet qui utilise l'algorithme de l'ICP pour faire le scan matching dans le *front end*, le résultat est assigné à une carte où les poses sont des noeuds de Pose Graph. Dans la partie *back end* un algorithme VBRL pour la détection d'erreurs de fermeture, dès que la validité de la boucle est vérifiée une fonction d'optimisation du Pose Graph est lancée.

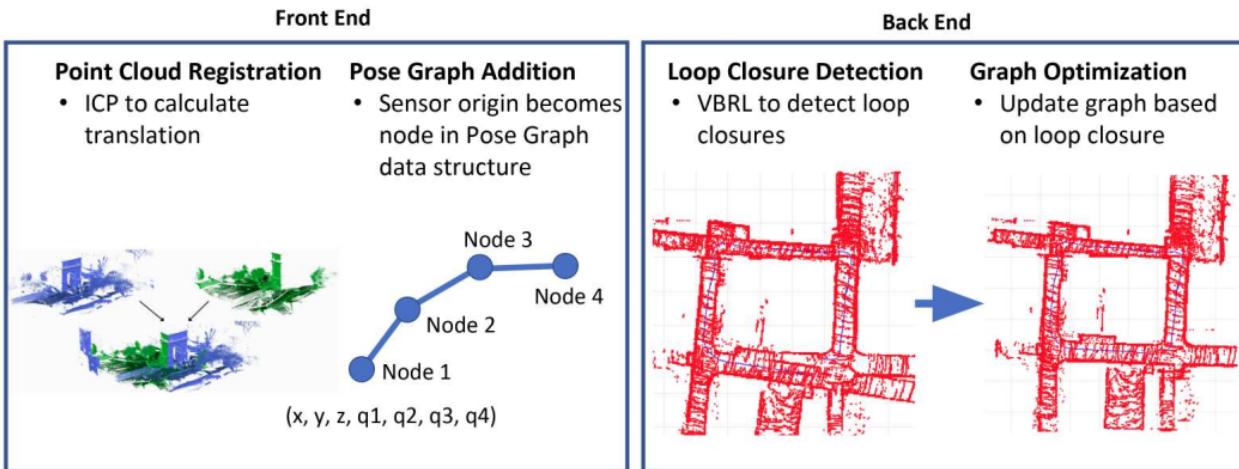


Figure 4.1: Exemple de Slam Overall

D'une manière générale, il existe deux types de composants technologiques utilisés pour réaliser le SLAM. Le premier type est le traitement du signal des capteurs, y compris le traitement frontal, qui dépend largement des capteurs utilisés. Le deuxième type est l'optimisation du graphe de pose, y compris le traitement *back end*, qui est indépendant du capteur. Dans ce mémoire on va se focaliser dans un premier temps sur la partie *front end*, sur le traitement de signal des capteurs et en considérant la méthode de Pose Graph comme un moyen d'optimisation.

4.2.1 Font End

Afin de comprendre ce type de technologie appelée traitement frontal, nous prendrons l'exemple du Visual SLAM et LiDAR SLAM, deux différentes méthodes d'aborder le SLAM.

SLAM visuel

Comme son nom l'indique, le SLAM visuel (ou vSLAM) utilise des images acquises à partir d'appareils photo et d'autres capteurs d'image. Visual SLAM peut utiliser des caméras simples (caméras grand angle, fish-eye et sphériques), des caméras à œil composé (caméras stéréo et multi-caméras) et des caméras RGB-D (caméras de profondeur et ToF).

Visual SLAM peut être mis en œuvre à faible coût avec des caméras relativement bon marché. De plus, comme les caméras fournissent un grand volume d'informations, elles peuvent être utilisées pour détecter un point de repère (positions précédemment mesurées). La détection de points de repère peut également être combinée avec une optimisation basée sur des graphiques, ce qui permet une flexibilité dans la mise en œuvre SLAM.

Le SLAM monoculaire est lorsque vSLAM utilise une seule caméra comme seul capteur, ce qui rend difficile la définition de la profondeur. Cela peut être résolu en détectant des marqueurs AR, des damiers ou d'autres objets connus dans l'image pour la localisation ou en fusionnant les informations de la caméra avec un autre capteur tel que des unités de mesure inertielles (IMU), qui peuvent mesurer des quantités physiques telles que la vitesse et l'orientation. La technologie liée au vSLAM comprend la structure à partir du mouvement (SfM), l'odométrie visuelle et l'ajustement des faisceaux.

Les algorithmes Visual SLAM peuvent être globalement classés en deux catégories. Les méthodes creuses correspondent aux points caractéristiques des images et utilisent des algorithmes tels que PTAM et ORB-SLAM. Les méthodes denses utilisent la luminosité globale des images et utilisent des algorithmes tels que DTAM, LSD-SLAM, DSO et SVO.

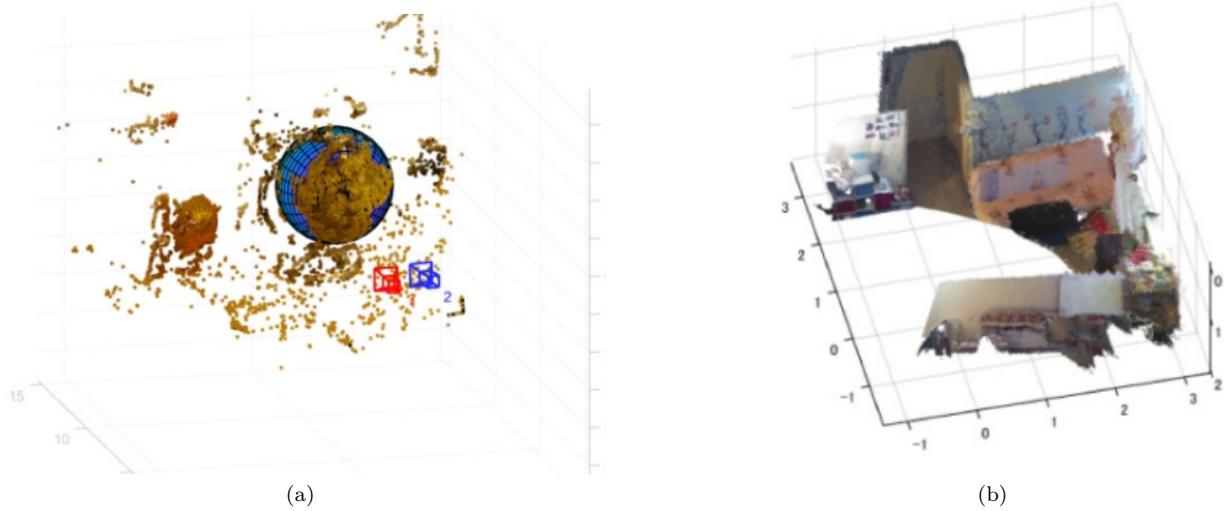


Figure 4.2: Vision SLAM : (a) Structure du mouvement; (b) Enregistrement de nuages de points pour RGB-D SLAM

LiDAR SLAM

La détection et la télémétrie de la lumière (LiDAR) est une méthode qui utilise principalement un capteur laser (ou capteur de distance).

Par rapport aux caméras, ToF et autres capteurs, les lasers sont nettement plus précis et sont utilisés pour les applications avec des véhicules en mouvement à grande vitesse tels que les voitures autonomes et les drones. Les valeurs de sortie des capteurs laser sont généralement des données de nuages de points 2D (x, y) ou 3D (x, y, z). Le nuage de points du capteur laser fournit des mesures de distance de haute précision et fonctionne très efficacement pour la construction de cartes avec SLAM. Généralement, le mouvement est estimé séquentiellement en faisant correspondre les nuages de points. Le mouvement calculé (distance parcourue) est utilisé pour localiser le véhicule. Pour l'appariement de nuages de points LiDAR, des algorithmes de transformation itérative de point le plus proche (ICP) et de distribution normale (NDT) sont utilisés. Les cartes de nuages de points 2D ou 3D peuvent être représentées sous forme de carte quadrillée ou de carte voxel.

En revanche, les nuages de points ne sont pas aussi finement détaillés que les images en termes de densité et ne fournissent pas toujours des fonctionnalités suffisantes pour l'appariement, encore moins en cas où on souhaiterait appliquer des fonctions d'AI qui permettraient de rajouter des informations à la carte telles que des données sémantiques. Par exemple, dans les endroits où il y a peu d'obstacles, il est difficile d'aligner les nuages de points et cela peut entraîner une perte de trace de l'emplacement du véhicule. De plus, la correspondance de nuages de points nécessite généralement une puissance de traitement élevée, il est donc nécessaire d'optimiser les processus pour améliorer la vitesse. En raison de ces défis, la localisation des véhicules autonomes peut impliquer la fusion d'autres résultats de mesure tels que l'odométrie des roues, le système mondial de navigation par satellite (GNSS) et les données IMU. Pour des applications telles que les robots d'entrepôt, le SLAM LiDAR 2D est couramment utilisé, tandis que le SLAM utilisant des nuages de points LiDAR 3D peut être utilisé pour les drones, le stationnement automatisé, ou les robots aspirateurs.

Cela dit, de nos jours les capteurs LiDAR sont aux prémices d'une vulgarisation de masse, en effet, plusieurs marques de smartphones (Galaxy S20 ou iPhone 12 Pro) se sont vu embarquer des capteurs laser dans leurs dispositifs, permettant de faire des scans en 3D et en couleurs.

Les capteurs Lidar intégrés aux nouveaux appareils téléphoniques peuvent être utilisés pour numériser des objets ou cartographier des pièces avec une rapidité surprenante. Ils ont une portée de balayage suffisant pour procéder à la télédétection d'objets ou de la plupart des pièces d'une maison. La télédétection étant le procédé qui consiste à mesurer un environnement à l'aide d'un scanner laser 3D ou d'un LiDAR capable de

4.2. WORKFLOW DU SLAM

mesurer des millions de points en peu de temps, puis résolvant une opération trigonométrie nécessaire pour calculer la taille de n'importe quel objet, ainsi que sa distance. Il fonctionne à l'intérieur comme à l'extérieur et a une précision de l'ordre du photon et une vitesse de l'ordre de la nanoseconde.

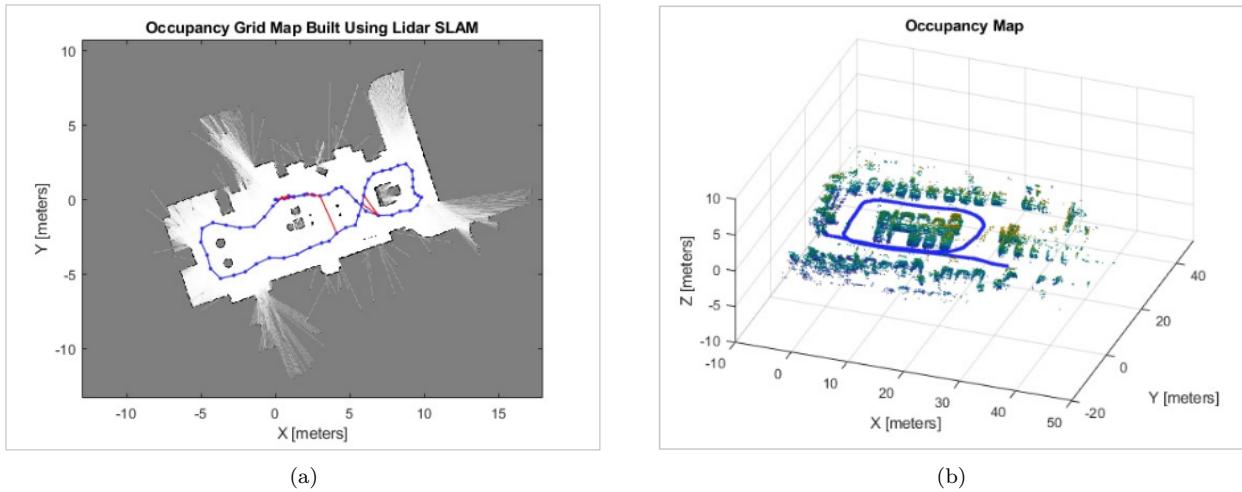


Figure 4.3: LiDAR SLAM : (a) SLAM avec LiDAR 2D ; (b) SLAM avec 3D LiDAR

Fusion

Depuis que le rendu des signaux GPS est surpassé dans les environnements en intérieur (*indoor*), ou même en milieu urbain, plusieurs chercheurs se focalisent sur le développement d'algorithme SLAM en utilisant des capteurs embarqués pour des robots ou d'autres dispositifs mobiles. De nos jours, les caméras sont des capteurs qui représentent une excellente alternative, appuyée par des facteurs de disponibilité et de moindre coût, ayant fait émergé ainsi le SLAM visuel.

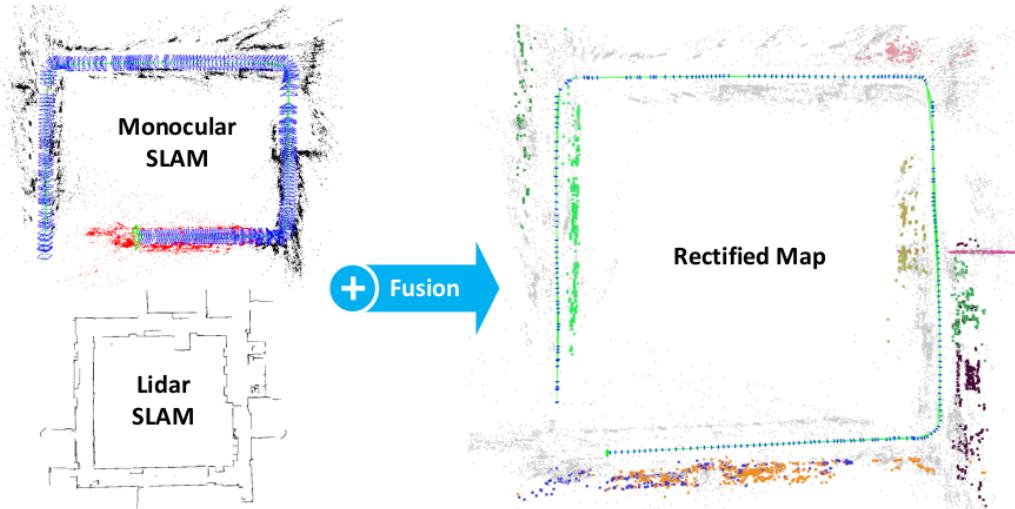


Figure 4.4: Les entrées de la carte de fusion consistent en une carte 3D de basse qualité générée par un SLAM Visuel Monoculaire, et carte prioritaire de haute qualité générée par des méthodes de LiDAR SLAM. La fusion de cartes va corriger la carte 3D en exploitant les plans verticaux qui sont généralement disponibles dans les deux cartes et génèrent en sortie une carte 3D davantage plus précise et plus proche de la réalité.

Les caméras RGB-D sont certes très disponibles, mais ne sont pas très fiables dans le cas d'exposition à la lumière du soleil surtout si celle ci est vive et éblouissante, ces capteurs sont alors aveuglés par la lumière.

Le télémètre est souvent considéré comme étant le capteur de cartographie le plus fiable, mais il est bien trop énergivore, et volumineux (encombrant) et coûteux pour plusieurs dispositifs mobiles.

Les dispositifs électroniques, ou certains petits robots, sont contraints d'utiliser une caméra monoculaire notamment du aux contraintes mécaniques architecturales, ou à cause de la contrainte énergétique.

Les algorithmes de SLAM Visuel monoculaire génèrent souvent des cartes de basse qualité à cause des difficultés de mise à l'échelle et de *drift* angulaire. Afin de répondre à cette problématique la fusion permet de générer des résultats à partir des deux méthodes qui se complémentent l'une l'autre, comme c'est le cas dans l'exemple de la figure 4.4. Tout comme on peut effectuer la fusion avec d'autres cartes du même milieu, dans Google MapsTM on trouve souvent des plans extérieurs de bâtisses[16].

Pour l'instant le LiDAR SLAM a prouvé sa supériorité face au vSLAM, que ce soit du point de vue précision, ou de vitesse, sans parler du fait qu'il n'ait pas besoin d'éclairage, mais les solutions basées uniquement sur la technologie LiDAR sont condamnés à la marginalisation dans le futur, lorsque l'électronique aura défi les défis de lourdeur des données à traiter dans le cas du vSLAM, et quand des fonctionnalités plus intéressantes seront accessibles uniquement grâce au format imagé. À lors, le LiDAR représentera un outil d'optimisation une fois fusionné à la vision.



Figure 4.5: Robots Aspirateurs : (a) le Dyson utilise la fusion LiDAR et Vision ; (b) l'Ecovas qui utilise la fusion de deux caméras

C'est ce qu'on peut déjà constater avec les aspirateurs intelligents, qui utilisent soit deux caméras, l'une pour cartographier le plafond et l'autre pour reconnaître les obstacles dans sa trajectoire (Dyson 360 EyeTM) ou qui utilisent un capteur LiDAR en tant qu'outil de navigation et de cartographie principal et une caméra permettant d'exploiter d'AIVI et de déterminer la nature des obstacles (Deebot OzmoTM960 de la marque Ecovacs) Que ce soit en LiDAR ou en vSLAM, l'avenir réside probablement dans la vision.

4.2.2 Back End

La partie *back end* de SLAM rentre en jeu à partir du moment où on dispose de la pose relative, et qu'on a une base de donnée solide de scans issus d'un télémètre laser. Les scans, s'ils sont acceptés vont être incorporés dans la structure de données du graphe de pose (voir section 4.3.4), et la trajectoire consiste en une série de noeuds représentants les positions estimées du robots, tel que chaque pose correspond à un scan qui décrit l'environnement.

Dans le *back end*, l'algorithme VBRL permet de faire les fermetures de boucle, qui représenteront par la suite des données sémantiques.

4.3 Algorithmes

Nous allons présenter dans cette partie trois méthodes largement utilisées pour la résolution du problème de SLAM. La majorité des autres méthodes et algorithmes en dérivent. Le premier exemple est le SLAM par Filtre de Kalman Etendu (EKF). C'est la plus ancienne méthode, encore largement utilisée. Le deuxième exemple utilise des techniques de filtrage statistique qu'on nomme généralement des filtres particulaires, ces deux premiers filtres sont très semblables étant donné leur structure commune qui découle du Filtre de Bayes.

La troisième méthode présentée se base sur la recherche du maximum de vraisemblance (Maximum Likelihood) dont la méthode de pose-graphe représente la méthode la plus exploitée pour faire du back-end SLAM, et représente donc une excellente méthode d'optimisation pour les méthodes de scan matching que nous allons voir dans le chapitre II.

4.3.1 Filtre de Bayes

Le filtre de Bayes est une technique d'estimation récursive d'état utilisée dans divers domaines de la robotique, comme par exemple dans la conduite de voitures autonomes, afin d'estimer l'état présent d'un système que ce soit des observations, des mesures ou des commandes de contrôle.

Ce qu'on appelle le filtre de Bayes est en effet une structure importante, cela dit, ce n'est pas une réalisation ou un filtre en soi utilisable dans la localisation ou l'estimation de la position du système dans un moment donné.

Le rôle effectif du filtre de Kalman est d'emmène une estimation d'état en temps réel (en ligne). Ceci implique qu'on va prédire une approximation de l'état en temps $t + 1$ à base des données qu'on a au temps t , ces données étant les plus récentes acquises par l'observateur et émises par le modèle de commande.

Cette méthode récursive d'estimation d'état consiste globalement en la prédition de l'état futur à partir d'une projection de l'état précédent. L'aspect récursive de cette approche apparaît dans la dérivation de l'équation de Bayes :

$$\begin{aligned} P(x_t, z_{1:t-1}) &= \sum_{x_{t-1}} p(x_t | x_{t-1}) p(x_{t-1} | z_{1:t-1}) \\ P(x_t, z_{1:t}) &= p(x_t | z_{1:t-1}) p(z_t | x_t) \end{aligned}$$

On commence par appliquer le processus de Markov sur plusieurs itérations de même que la loi des probabilités totales, et ce qu'après ça qu'on procède à la dérivation d'une équation.

Le filtre de Bayes utilise le modèle de commande avec soi le modèle d'observation ou le modèle de mesure ou les deux.

Le modèle d'observation décrit la probabilité d'obtenir l'observation z sachant l'état x .

$$p(z|x)$$

Le modèle de commande nous informe sur la probabilité d'évolution de l'état de x_t à x_{t+1} sachant commande u .

$$p(x_{t+1}|x_t, u)$$

Alors on peut voir que la commande u peut représenter par exemple la force de pression d'appui sur la pédale de l'accélérateur, ou la commande guidage d'un robot mobile.

Les données captées qui feraient guise d'entrées peuvent être issues d'un télémètre laser ou une caméra montée sur le véhicule, et le filtre récursif nous permet d'estimer les états de notre système récursivement.

Il existe diverses concrétisations du filtre de Bayes, les concrétisations les plus connues étant : le filtre de Kalman, le filtre de Kalman Étendu (EKF), le filtre de Kalman Incrémental (IKF), le filtre particulaire, le filtre de Monte Carlo, les filtres discrets tel que le filtre Histogramme.

Tous ces filtres suivent la même structure pour faire une estimation d'état en temps réel. Cependant, chacun fait des suppositions distinctes sur la problématique initiale. Exemples : Le filtre de Kalman présume que le monde est gaussien et que tous les systèmes sont linéaires ; Le filtre de Kalman Étendu fait la relaxation des hypothèses sur la linéarité, et linéarise les systèmes non linéaires avant de les traiter ; Le filtre particulaire lui fait la relaxation de la nature gaussienne permettant ainsi de représenter des systèmes qui répondent à des lois de probabilité arbitraires ce qui nécessite d'ailleurs une capacité et un coût de calcul supérieur. En effet, ces réalisations ont leurs avantages et leurs inconvénients, c'est pour cela que le choix de la méthode à appliquer en cas de besoin repose sur les caractéristiques de notre système[17][18].

4.3.2 Filtre de Kalman

Kalman propose une solution récursive au filtrage des données linéaires. Cette méthode, améliorée ensuite par Kalman lui-même et Bucy, 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 (voir figure 4.6) :

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

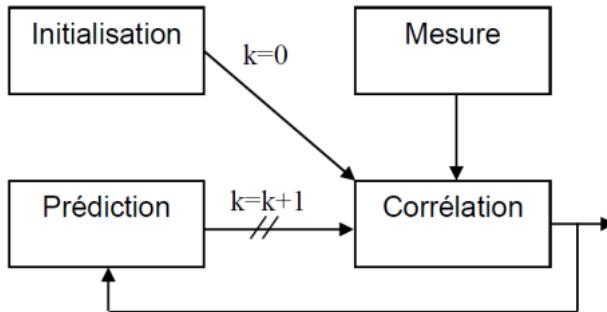


Figure 4.6: Le cycle du filtre de Kalman basé sur les deux étapes récursives : Prédiction et Correction

Le cycle du filtre de Kalman est constitué de deux étapes fondamentales :

Étape de prédiction : durant laquelle on estime l'état du système à l'instant t en utilisant l'estimation corrigée de l'instant $(t - 1)$.

Étape de correction : durant laquelle on corrige l'estimation de l'état du système à l'instant t en utilisant les informations sensorielles reçues à l'instant t .

Le filtre de Kalman Étendu 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 dans le cadre du programme spatial Apollo. D'ailleurs, l'ancien nom du filtre était *le filtre de Kalman-Schmidt*.

Dans le cadre du SLAM, cette méthode a été introduite pour la première fois pour estimer en un seul processus de la position du robot et de la localisation des amers dans l'environnement. Les erreurs dans l'estimation sont représentées par une matrice de covariance mise à jour régulièrement en utilisant un filtre de Kalman Étendu à chaque fois que le robot change de position. La taille de cette matrice grandit quadratiquement au fil des observations du robot.

Dans un algorithme de SLAM par EKF, on décrit le modèle de transition sous la forme :

$$P(x_k|x_{k-1}, u_k) \iff f(x_{k-1}, u_k) + w_k$$

Dans cette équation, f représente le modèle du véhicule robotisé et $w_k \sim N(0, Q_k)$ est un bruit gaussien de moyenne nulle et de variance w_k .

Le modèle d'observation se présente sous la forme :

$$P(z_k|x_k, m) \iff h(x_k, m) + v_k$$

où h décrit le modèle d'observation et $v_k \sim N(0, R_k)$ un bruit gaussien de moyenne nulle et de variance R_k . L'annexe B décrit plus en détails les formulations mathématiques du KF et de l'EKF.

La convergence d'un filtre de Kalman est prouvée analytiquement dans le cadre linéaire. Mais elle n'est pas toujours réalisable dans des cas réels où les données sont fortement non linéaires. Ce problème apparaît également avec un EKF ou un UKF¹ (Unscented Kalman Filter).

L'implémentation d'un filtre de Kalman évolue généralement quadratiquement en $O(n^2)$ (où n est le nombre d'amers de la carte) dans le temps et l'utilisation des ressources mémoire du système. Ainsi, avec l'évolution du système, l'algorithme atteindra un point où il ne pourra pas mettre à jour sa carte en temps réel. Ce problème vient du fait que chaque amer de l'environnement est corrélé à tous les autres. Cette corrélation se justifie par le fait que l'observation de chaque nouvel amer est faite par les capteurs du robot, l'erreur de localisation de l'amer est ainsi liée à l'erreur de localisation du robot lui-même et aux erreurs des autres amers dans la carte.

Afin de réduire ces exigences en puissance du matériel, le filtre de Kalman étendu compressé (CEKF) a été introduit. Il traite et maintient les informations liées à un espace local avec un coût de calcul proportionnel au carré du nombre d'amer de la carte locale. Ces informations sont ensuite transférées à la carte globale d'une manière similaire à l'algorithme ordinaire, mais en une seule itération. Le CEKF réduit l'utilisation de mémoire, mais nécessite la détection d'amers robustes et souffre du problème d'association des données. Ce problème est renforcé par l'inconsistance due aux différentes approximations de linéarisation introduites dans le filtrage de Kalman. Plusieurs recherches ont ainsi tenté de proposer des extensions de cette méthode, permettant d'améliorer l'association des données. On trouve notamment quelques travaux de Burgard, Fox et Thrun utilisant des techniques de statistiques avancées comme l'algorithme d'Espérance-Maximisation de Dempster. Mais cela engendre encore plus de calculs et augmente la complexité de l'algorithme.

4.3.3 Filtre particulaire

Un filtre particulaire est un filtre récursif qui permet d'estimer l'état a posteriori en utilisant un ensemble de particules. 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 Monte Carlo dans FastSLAM.

L'algorithme FastSLAM décompose le problème du SLAM en deux parties : un problème de localisation du robot et une collection de problèmes d'estimation d'amers liés à l'estimation de la position du robot. Dans cette configuration, chaque particule se charge de l'association des données locales qui lui sont liées. Par comparaison, un filtre EKF traite une seule hypothèse d'association de données pour tout le filtre. FastSLAM nécessite ainsi moins de mémoire et de temps de calcul que l'EKF.

¹L'UKF n'est applicable que dans le cas de bruits gaussiens, et nécessite plus de puissance de calcul que l'EKF

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.

4.3.4 Maximum de Vraisemblance (MLE)

Alors que qu'un filtre particulier ou un filtre de Kalman constituent des solutions probabilistes du problème de SLAM, la recherche de l'estimateur du maximum de vraisemblance (notée MLE pour Maximum Likelihood Estimator) est une approche d'optimisation, où on teste plusieurs hypothèses à la recherche de celle qui maximise la vraisemblance. La première approche qui nous vient à l'esprit en parlant de MLE, c'est bien l'IML.

Maximum de Vraisemblance Incrémentale (IML)

Contrairement aux différentes déclinaisons du filtre de Kalman ou des approches à maximisation globale de vraisemblance (Expectation Maximization [19]), qui essaient d'établir une estimation a posteriori sur les positions du robot et sur la carte, l'idée de l'IML est de construire une seule carte incrémentalement à chaque réception des données des capteurs sans garder un suivi de l'incertitude. Ce principe assure la simplicité de l'IML, qui reste son grand avantage comparé aux autres méthodes de SLAM.

En théorie l'Incremental Maximum Likelihood (IML) consiste à rechercher à chaque instant la meilleure correspondance entre l'observation courante (les données provenant du laser) et la carte courante (combinant la connaissance de l'environnement), et à remettre à jour la carte conformément à cette mise en correspondance. L'idée générale de cet algorithme est présentée dans le schéma de la figure 4.7.

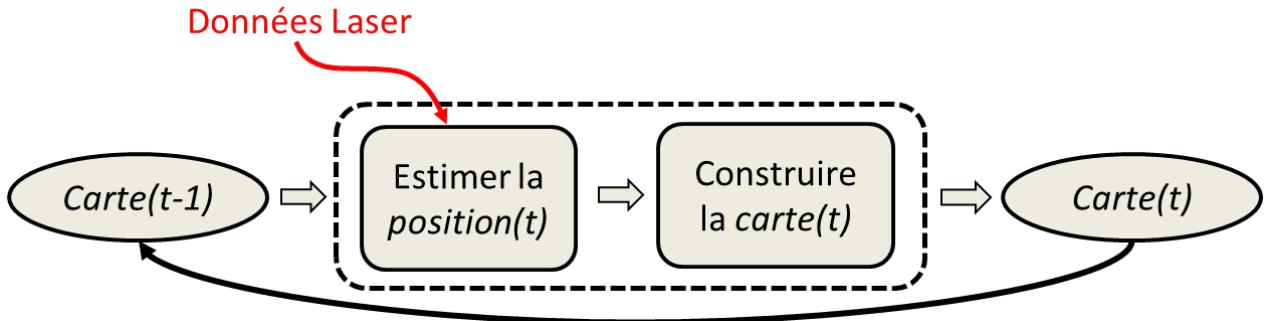


Figure 4.7: Principe général de l'IML

L'IML est un processus évidemment divergent, puisqu'on utilise la localisation du robot pour mettre à jour la carte, et la carte mise à jour pour trouver la localisation du robot.

Mathématiquement, l'idée de base de l'IML est de maintenir une série de cartes ($\hat{m}_1, \hat{m}_2, \dots$) et une série de positions du robot ($\hat{x}_1, \hat{x}_2, \dots$) maximisant la vraisemblance. La position du robot à l'instant k est calculée en utilisant l'estimation à l'instant $k - 1$ en maximisant la vraisemblance :

$$\{\hat{x}_k, \hat{m}_k\} = \underset{x_k, m_k}{\operatorname{argmax}} P(z_k | x_k, m_k) \times P(x_k, m_k | u_k, \hat{x}_{k-1}, \hat{m}_{k-1}) \quad (4.1)$$

La carte de l'environnement m_k est construite lorsque la position x_k est connue. Ainsi, en pratique, il suffit de chercher dans l'espace des positions du robot. Afin de déterminer la position \hat{x}_k maximisant la vraisemblance, l'IML nécessite simplement une recherche dans l'espace de toutes les positions x_k lorsque l'algorithme reçoit de nouvelles données des capteurs :

$$\hat{x}_k = \underset{x_k}{\operatorname{argmax}} P(z_k | x_k, m_{k-1}) \times P(x_k | u_k, \hat{x}_{k-1}) \quad (4.2)$$

¹Attention : la notation m_k utilisée ici est différente de la définition donnée dans 3. Dans la section 3, m_k désigne l'amer k de la carte, alors que dans cette section, elle désigne la carte construite à l'étape k .

Dans l'équation 4.2, le terme $P(z_k|x_k, \hat{m}_{k-1})$ décrit la probabilité d'observer les dernières mesures z_k des capteurs en utilisant la carte m_k construite à l'étape $k-1$ et la position du robot x_k . Le terme $P(z_k|u_k, \hat{x}_{k-1})$ représente la probabilité que le système soit à l'état x_k en supposant connu l'état \hat{x}_{k-1} et la commande u_k . Le résultat \hat{x}_k trouvé est utilisé afin de mettre à jour la carte en utilisant les données correspondantes z_k :

$$\hat{m}_k = \hat{m}_{k-1} \cup \{\hat{x}_k, z_k\} \quad (4.3)$$

La maximisation de l'équation 4.2 revient à trouver la position x_k du robot qui permet de satisfaire le modèle de mouvement du véhicule assurant une meilleure concordance entre les données des capteurs z_k et la carte m_{k-1} . Dans les différents travaux de recherche portants sur le SLAM par recherche du maximum de vraisemblance, on a souvent recourt aux méthodes de mise en correspondance des scans Laser (scan matching).

Ces méthodes diffèrent selon la représentation de la carte choisie (section 3.2.1). On peut ainsi utiliser un scan-matching direct [20], se baser sur les caractéristiques géométriques de l'environnement ou profiter de la grille probabiliste de la carte. L'une des techniques les plus utilisées dans le cadre du scan matching est l'ICP.

La simplicité et la rapidité du SLAM par recherche du maximum de vraisemblance permet de construire des cartes de l'environnement en temps réel, mais cette approche ne peut pas garder une notion d'incertitude dans les estimations. En plus, la nature incrémentale de l'algorithme limite les traitements sur une seule étape de calcul et néglige l'ensemble des données capturées dans les autres étapes (contrairement au filtre de Kalman par exemple).

Lorsqu'une position x_k du robot et une carte m_k ont été déterminées, elles sont fixées et ne peuvent plus être changées ou corrigées en utilisant les prochaines données (cas de fermeture de boucle par exemple). L'erreur d'estimation de la position x_k peut ainsi grandir sans limite. Afin de corriger ce problème, des propositions comme celle de Hähnel [21] ont été émises, il propose d'utiliser un arbre d'associations pour suivre plusieurs hypothèses de la carte de l'environnement. Si cette idée peut améliorer la qualité des résultats de l'algorithme, elle risque tout comme les autres propositions de nécessiter plus de charges de calcul, ce qui peut freiner l'utilisation temps-réel de l'IML dans le SLAM[10].

C'est justement ce qui fait de Pose Graph une approche de *full SLAM* (ou *off line SLAM*), or qu'on ne peut pas l'appliquer en temps réel.

Graph SLAM

Le GraphSLAM (pour *graph based SLAM*), ou Pose Graph, a été introduit pour la première fois par Lu et Milios en 1997[22], c'est un problème d'estimation de maximum de vraisemblance incrémentale, c'est sa variante la plus utilisée[23].

Cette approche constitue depuis une dizaine d'années un axe de recherche très actif au sein de la communauté de robotique. L'estimation de l'état du système est formulée par un problème d'optimisation.

Ce dernier fait appel à plusieurs techniques issues essentiellement de l'algèbre linéaire et de la théorie des graphes. Le GraphSLAM est une technique de SLAM basé optimisation de graphe, il modélise le problème de SLAM à l'aide d'un graphe. Comme l'illustre la figure 4.8, la trajectoire et la carte des amers sont représentées par des noeuds. Associées à un bruit Gaussien, les mesures des capteurs donnent des contraintes spatiales entre les noeuds. Ces contraintes spatiales sont modélisées par des arêtes. On distingue deux types d'arêtes : arêtes de mouvement et arêtes d'observation. Une arête de mouvement relie deux noeuds robot (pose) consécutifs. Une arête d'observation provient d'une observation d'un amer. Un amer est relié à une pose (noeud robot) s'il a été observé depuis celle-ci.

La figure 4.8 illustre également la procédure de marginalisation des amers que l'on peut appeler aussi réduction de graphe. Celle-ci est réalisée en utilisant le complément de Schur qui réduit la taille du système et ainsi son temps de résolution. Il transforme le système construit en un nouveau système plus petit, et ne contenant que les poses.

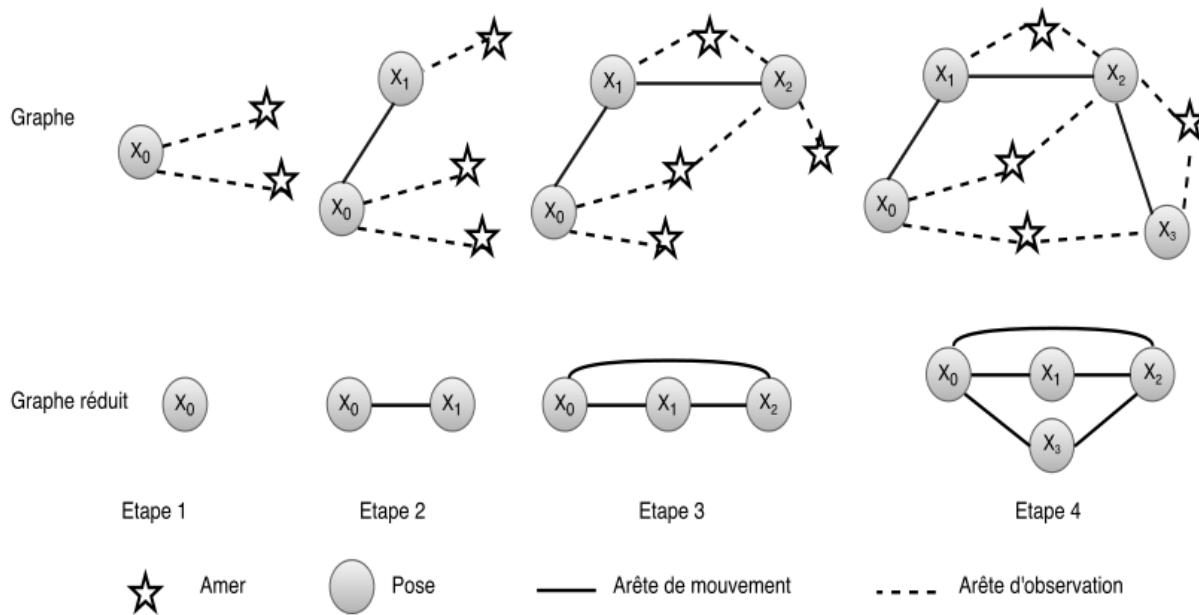


Figure 4.8: Exemple illustratif de la méthode Graph SLAM

La marginalisation consiste à éliminer des entrées de la matrice de variances-covariances (inverse de la matrice d'information) dans le but de diminuer la densité du système (le rendre plus épars).

C'est le complément de Schur dans le GraphSLAM qui réduit le graphe en marginalisant les amers. La marginalisation d'un amer implique la connexion deux à deux les poses depuis lesquelles cet amer a été observé. Le graphe réduit peut être également obtenu par une mise en correspondance directe des faisceaux Laser entre deux poses donnant, ainsi, la distance relative entre ces poses.

Pour efficacement résoudre de larges systèmes, cette technique consiste à utiliser un squelette du graphe (skeleton graph). Celui-ci est composé d'un sous-ensemble de frames suite à la marginalisation des amers.

Néanmoins, la marginalisation des amers génère de nouvelles contraintes entre les poses. Le nouveau système peut, donc, être moins épars, voire très dense par rapport au système initial. Le temps de résolution peut alors augmenter et devient très important. D'autre part, ce phénomène de remplissage peut être masqué par la forte réduction de la taille du système si le nombre d'amers est très grand par rapport au nombre de poses[10].

4.3.5 Comparaison

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 qu'on doit 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 grandissante et ses difficultés de paramétrage.

La MLE reste une solution intéressante grâce à sa simplicité et son efficacité en temps de calcul. En plus, il peut être appliquée à tous les types de cartes. Malheureusement, on ne peut estimer que les meilleures hypothèses en position à chaque étape de calcul, ce qui freine les possibilités de l'IML lors de la fermeture d'une boucle dans un environnement cyclique.

4.3. ALGORITHMES

	Avantages	Inconvénients
Filtre de Kalman		
Filtre Particulaire		
IML		
Scan Matching		
Scan Matching + Pose Graph		

4.4 Défis courants avec SLAM

Bien que SLAM soit utilisé pour certaines applications pratiques, plusieurs défis techniques empêchent une adoption plus générale. Chacun a une contre-mesure qui peut aider à surmonter l'obstacle.

4.4.1 Les erreurs de localisation s'accumulent, provoquant un écart important par rapport aux valeurs réelles

SLAM estime le mouvement séquentiel, qui comprend une certaine marge d'erreur. L'erreur s'accumule au fil du temps, provoquant un écart substantiel par rapport aux valeurs réelles. Cela peut également entraîner la réduction ou la distorsion des données cartographiques, rendant les recherches ultérieures difficiles. Prenons un exemple de conduite autour d'un passage de forme carrée. À mesure que l'erreur s'accumule, les points de départ et d'arrivée du robot ne correspondent plus. C'est ce qu'on appelle un problème de fermeture de boucle. Des erreurs d'estimation de pose comme celles-ci sont inévitables. Il est important de détecter la fermeture de boucle et de déterminer comment corriger ou annuler l'erreur accumulée.

Une **contre - mesure** consiste à se souvenir de certaines caractéristiques d'un lieu précédemment visité comme point de repère et à minimiser l'erreur de localisation. Les graphiques de pose sont construits pour aider à corriger les erreurs. En résolvant la minimisation des erreurs comme un problème d'optimisation, des données cartographiques plus précises peuvent être générées. Ce type d'optimisation est appelé ajustement groupé dans Visual SLAM.

4.4.2 La localisation échoue et la position sur la carte est perdue

La cartographie d'images et de nuages de points ne tient pas compte des caractéristiques du mouvement d'un robot. Dans certains cas, cette approche peut générer des estimations de position discontinues. Par exemple, un résultat de calcul montrant qu'un robot se déplaçant à 1 m/s a soudainement fait un bond de 10 mètres en avant. Ce type d'échec de localisation peut être évité soit en utilisant un algorithme de récupération, soit en fusionnant le modèle de mouvement avec plusieurs capteurs pour effectuer des calculs basés sur les données du capteur.

Il existe plusieurs méthodes pour utiliser un modèle de mouvement avec fusion de capteurs. Une méthode courante consiste à utiliser le filtrage de Kalman pour la localisation. Étant donné que la plupart des robots à entraînement différentiel et des véhicules à quatre roues utilisent généralement des modèles de mouvement non linéaires, des filtres de Kalman étendus et des filtres à particules (localisation Monte Carlo) sont souvent utilisés. Des filtres Bayes plus flexibles tels que des filtres Kalman non parfumés peuvent également être utilisés dans certains cas. Certains capteurs couramment utilisés sont des dispositifs de mesure inertuelle tels que l'IMU, le système de référence d'attitude et de cap ou AHRS , les systèmes de réseau d'information ou INS, les capteurs accélérométriques, les capteurs gyroscopiques et les capteurs magnétiques. Les encodeurs de roue attachés au véhicule sont souvent utilisés pour l'odométrie.

Lorsque la localisation échoue, une **contre - mesure** à récupérer consiste à se souvenir d'un point de repère comme une image clé d'un lieu précédemment visité. Lors de la recherche d'un point de repère, un processus d' extraction d' entités est appliqué de manière à pouvoir numériser à grande vitesse. Certaines méthodes basées sur des caractéristiques d'image incluent un sac de caractéristiques (BoF) et un sac de mots visuels (BoVW). Plus récemment, l'apprentissage en profondeur est utilisé pour comparer les distances des entités.

4.4.3 Coût de calcul élevé pour le traitement d'image, le traitement des nuages de points et l'optimisation

Le coût de calcul est un problème lors de la mise en œuvre de SLAM sur le matériel d'un véhicule. Le calcul est généralement effectué sur des microprocesseurs intégrés compacts et à faible consommation d'énergie qui ont une puissance de traitement limitée. Pour obtenir une localisation précise, il est essentiel d'exécuter le traitement d'image et la correspondance des nuages de points à haute fréquence. De plus, les

calculs d'optimisation tels que la fermeture de boucle sont des processus de calcul élevés. Le défi est de savoir comment exécuter un traitement aussi coûteux en calcul sur des micro-ordinateurs embarqués.

Une **contre - mesure** consiste à exécuter différents processus en parallèle. Des processus tels que l'extraction de caractéristiques, qui est le prétraitement du processus de correspondance, sont relativement adaptés à la parallélisation. L'utilisation de processeurs multicœurs pour le traitement, le calcul de données multiples à instruction unique (SIMD) et de GPU intégrés peut encore améliorer les vitesses dans certains cas. De plus, étant donné que l'optimisation du graphe de pose peut être effectuée sur un cycle relativement long, abaisser sa priorité et effectuer ce processus à intervalles réguliers peut également améliorer les performances.

4.5 Conclusion

Depuis l'identification de la problématique du SLAM de nombreuses approches pour y répondre ont apparues. Historiquement, les approches filtrées, de Kalman puis particulière, sont apparues les premières dans un contexte de SLAM général puis ont été adaptées au cas du SLAM visuel et monoculaire. Dans ce contexte sont ensuite apparues les approches *Structure From Motion*, issues de la communauté de vision; ces SfM permettent de reconstruire une scène ou un objet en 3D à partir d'images en 2D. Ce domaine reste en permanente évolution. Et pour

2.2.2.4 Comparaison entre les algorithmes

Le tableau 2.2 présente une liste d'avantages et d'inconvénients de certaines méthodes et algorithmes de SLAM ([3] et [53]).

	<i>Avantages</i>	<i>Inconvénients</i>
Filtre de Kalman	<ul style="list-style-type: none">- Prise en compte des incertitudes- Convergence prouvée- Fermeture de boucle	<ul style="list-style-type: none">- Hypothèses fortes- Complexité- Problème d'association des données- Utilisation possible sur un seul type de cartes
Filtre particulaire	<ul style="list-style-type: none">- Élimination des hypothèses du KF- Fermeture de boucle	<ul style="list-style-type: none">- Complexité- Paramétrage difficile
IML	<ul style="list-style-type: none">- Concept simple et rapide- Tous les types de cartes	<ul style="list-style-type: none">- Divergent par construction- Pas de fermeture de boucle

TABLE 2.2 – Tableau comparant les avantages et les inconvénients de quelques méthodes de 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 qu'on doit 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

Figure 4.9: Exemple de construction d'un graphe de pose et de minimisation des erreurs

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	<ul style="list-style-type: none"> - Prise en compte des incertitudes. - Convergence prouvée. - Fermeture de boucle. 	<ul style="list-style-type: none"> - Hypothèse forte. - Complexité. - Problème d'association des données. - Utilisation possible sur un seul type de carte.
Filtre particulaire	<ul style="list-style-type: none"> - Elimination des hypothèses du KF. - Fermeture de boucle. 	<ul style="list-style-type: none"> - Complexité. - Paramétrage difficile.
Scan matching	<ul style="list-style-type: none"> - Concept simple et rapide. - Tous les types de cartes. 	<ul style="list-style-type: none"> - 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 utilisent les données brutes du capteur qui sont des

Figure 4.10: Exemple de construction d'un graphe de pose et de minimisation des erreurs



Figure 4.11: Exemple de construction d'un graphe de pose et de minimisation des erreurs

Chapitre II

Scan Matching

1 Généralités sur le Scan Matching

1.1 Point Cloud

Un point cloud, ou un nuage de point est un ensemble de points dans un espace 3D. Ces nuages de points sont obtenus notamment à partir de scanners 3D, tels que des scans de télémètres laser. On retrouve ce concept dans la navigation et la perception des robots, l'estimation de la profondeur, la vision stéréo, l'enregistrement visuel et les systèmes avancés d'aide à la conduite (ADAS)[24]. <https://youtu.be/ktRqKxddjJk?t=1363> Densité, selection/segmentation, fréquence <https://youtu.be/ktRqKxddjJk?t=1009>

Descriptors Matrix		N Points			
Descriptor Labels					
		-0.6363	0.7386	...	0.7061
orientation	-0.4724	0.1594	...	0.2441	
	-0.7089	0.0997	...	-0.2981	
normal	-0.1964	-0.7534	...	-0.1655	
	-0.8481	-0.9007	...	-0.6322	
	-0.5202	-0.5201	...	0.8054	

Figure 1.1: Description de la matrice représentative d'un Nuage de Point

Descriptors Matrix		N Points			
Descriptor Labels					
		-0.6363	0.7386	...	0.7061
orientation	-0.4724	0.1594	...	0.2441	
	-0.7089	0.0997	...	-0.2981	
normal	-0.1964	-0.7534	...	-0.1655	
	-0.8481	-0.9007	...	-0.6322	
	-0.5202	-0.5201	...	0.8054	

Figure 1.2: Description de la matrice représentative d'un Nuage de Point

1.2 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. Il existe plusieurs algorithmes avec qui divergent dans leurs approches, qui ont pour but de faire du scan matching dont les plus connus : L'algorithme itératif du point le plus proche (ICP), la double correspondance itérative (IDC), la transformation de distribution normale (Normal Distribution Transform, NDT) et la méthode de champs de vraisemblance (LF)[14].

Afin de déterminer la localisation, on pourrait croire que l'odométrie suffirait à acquérir assez d'information sur le déplacement de notre robot, pour dessiner son itinéraire. Cependant, on pourra constater dans le chapitre suivant, que l'odométrie brute à elle seule est bien trop pauvre pour créer un environnement que le robot pourra exploiter pour naviguer dans un milieu cartographié via des données issues exclusivement de l'odométrie.

La superposition de nuages de points, est dite en *Image Registration*, ou en encore *Scan Matching* (dans ce mémoire, on va privilégier cette dernière dénomination). C'est le processus d'alignement de deux ou plusieurs nuages de points 3D de la même scène dans un système de coordonnées commun. La cartographie est le processus de construction d'une carte de l'environnement autour d'un robot ou d'un capteur (voir 3.2 à la page 18). Le scan matching et la localisation sont utilisés pour reconstruire une scène 3D ou créer la carte d'une route à des fins de localisation.

Bien que le scan matching est généralement utilisé pour la cartographie, il existe d'autres applications du scan matching, qui peuvent ne pas nécessiter de cartographie, telles que la poursuite de mouvement déformable.

Les algorithmes de *Computer Vision Toolbox* de Matlab fournissent des fonctions d'enregistrement et de cartographie de nuages de points en 3D. Le flux de travail comprend le prétraitement, l'enregistrement, la correction de la dérive et l'alignement des nuages de points. Nous utiliserons cette outil dans le chapitre suivant, de même que nous appliquerons des algorithmes de scan matching sans recourir à ces fonctions notamment dans le cas en 2D[24].

1.3 Le Scan Matching étape par étape

L'algorithme de cartographie et de localisation basé sur le scan matching adopté par la *Computer Vision Toolbox* de Matlab a une approche assez générale via laquelle on peut comprendre le concept aussi à l'aide d'un schéma explicatif dans la figure 1.3.

Il faut noter qu'il existe plus d'une méthode, et plus d'un outil pour faire du SLAM, mais les étapes qui vont suivre sont communes à plusieurs autres algorithmes qui ont pour objectif de superposer des nuages de points afin de dessiner une carte à partir d'une séquence de nuage de points, afin de localiser par la suite le véhicule sur la carte prédéfinie[24] :

- **Prétraiter les nuages de points :** Pour préparer les nuages de points à l'enregistrement, il faut procéder à leur échantillonnage et à la suppression des éléments indésirables et du bruit.
- **Scan Matching :** Superposer chaque nuage de points à celui qui le précède. Ces superpositions sont utilisés en odométrie, qui est le processus d'accumulation d'estimations successives de superpositions de trames de nuages de points. L'utilisation de l'odométrie seule peut conduire à une dérive entre les poses de vérité-mesurée et de vérité-terrain.
- **Déetecter les boucles :** Effectuer une détection de fermeture de boucle pour minimiser la dérive. La détection de fermeture de boucle est le processus d'identification du retour du capteur à un emplacement précédemment visité, qui forme une boucle dans la trajectoire du capteur.
- **Corriger la dérive :** Utiliser les boucles détectées pour minimiser la dérive grâce à l'optimisation du graphe de pose, qui consiste à créer progressivement un graphe de pose en ajoutant des noeuds et des arêtes, puis à optimiser le graphe de pose une fois qu'il dispose de suffisamment de boucles. L'optimisation du graphe de pose se traduit par un ensemble de poses absolues optimisées.

- **Assembler la carte :** Assembler une carte de nuages de points en alignant les nuages de points enregistrés à l'aide de leurs poses absolues optimisées. Sinon utiliser une carte de nuages de points prédéfinie pour la localisation du véhicule dans la carte.
- **Localiser :** Trouver la pose du véhicule sur la base de la carte assemblée.

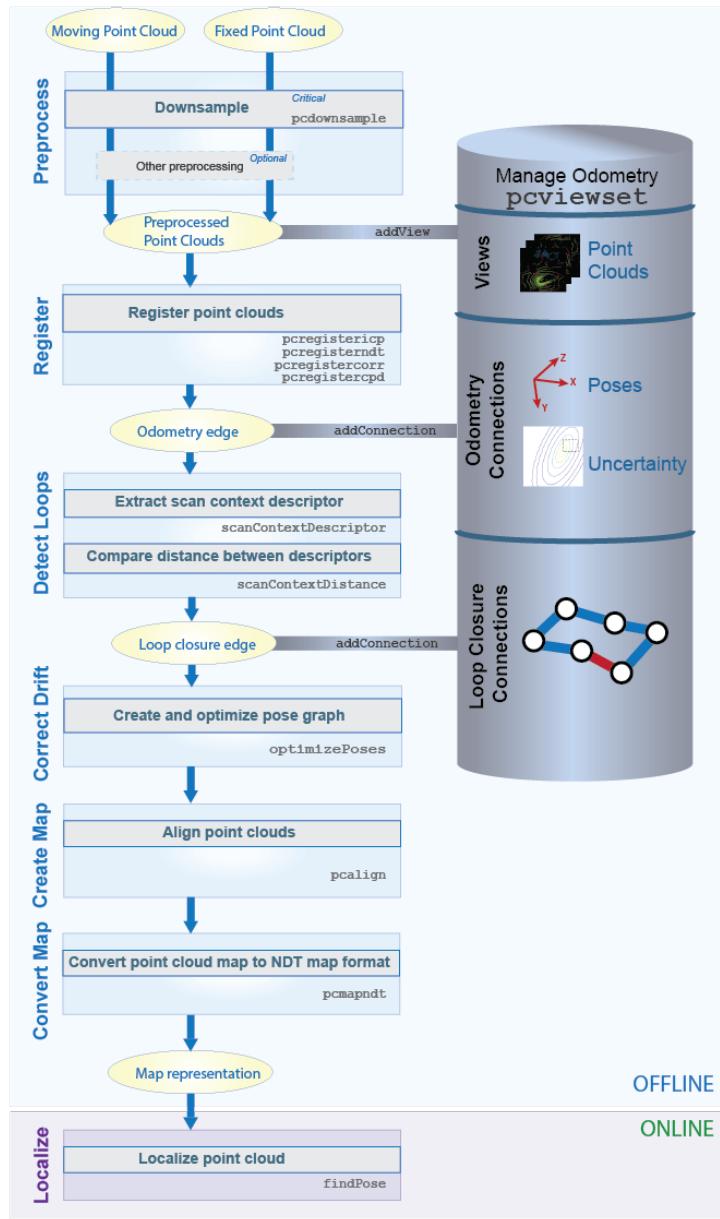


Figure 1.3: Schématisation du fonctionnement du Scan Matching selon le site web de Matlab

2 ICP

<https://www.youtube.com/watch?v=QWDM4cFdKrE&t=125s>

L'ICP ou *Iterative Closest Point* est une technique pour aligner deux nuages de points en cherchant de manière itérative les points les plus proches pour les faire correspondre et superposer des nuages de points. Elle a été introduit en 1992 par Besl et McKay[25].

Cette technique de SLAM est basée sur le scan matching est très utilisée, son 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).

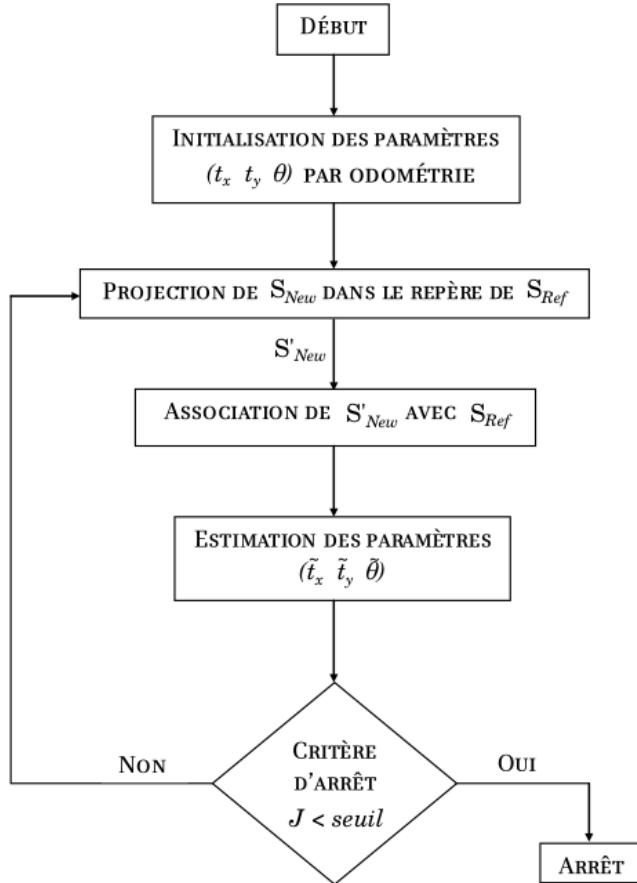


Figure 2.1: L'organigramme général d'ICP

Pour simplifier le concept on va commencer par un algorithme de simplification 2.1 sachant que l'initialisation ne se fait pas forcément en utilisant les données odométriques, bien que ce soit une option, on peut toujours initialiser la transformation à l'unité.

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[25].

Puis vient l'étape de la pondération où chaque paire de points est attachée 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 rejettés pour éviter une fausse estimation, L'organigramme de la figure 2.1 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 (on a impérativement besoin du vecteur de translation et la matrice de rotation qui décrit le déplacement qui a eu lieu pour passer d'une vue décrite par le premier nuage de point (premier scan) à la position où le robot perçoit un second nuage de point qui décrit une perspective légèrement différente comparée à la première.), et pour finir le seuil de sortie (on parle de seuil dans le but de simplifier le concept, voir la section 2.1.5) de la boucle est un seuil de nombre d'itérations permettant de déduire une position correcte, On peut aussi voir l'approche détaillée de l'algorithme adapté en fonction matlab[26][27].

Plusieurs chercheurs ont proposé des solutions pour adresser les problèmes générés par l'algorithme de l'ICP original (Vanilla ICP), ce qui a mené à différentes variantes de cet algorithme. Une taxonomie a des variantes de l'ICP a été suggérée par [28][29] et [7]. Elles décrivent cinq (05) étapes :

1. Sélection des points
2. Matching des nuages de points
3. Pondération des paires
4. Rejection des paires
5. Optimisation des erreurs

Cette brève vue d'ensemble permettra d'esquisser une meilleure image de ce en quoi consiste l'ICP en général, d'autant plus que les variantes peuvent être tellement différentes qu'il devient difficile de déterminer que ce sont des dérivées de l'ICP. Nous allons exploiter l'algorithme originale dans le chapitre dédié aux simulations pour comprendre les imperfections de l'algorithme Vanilla ICP qu'on va définir théoriquement dans la section 2.3. Puis nous verrons deux autres variantes qui sont toutes deux très utilisées, et qui donnent d'excellents résultats.

2.1 Taxonomie de l'ICP

2.1.1 Initialisation des points

La façon dont sont sélectionnés les points en entrée de ICP peut avoir un impact sur la convergence de celui ci. *Est il préférable de traiter toutes les données disponibles ou faut il les échantillonner ?* Il existe plusieurs approches pour sélectionner ces points. La documentation [30] utilise les méthodes suivantes d'échantillonnage, bien qu'il y en est plusieurs autres, celles ci donnent une idée sur ce qui existe :

- Sélection aléatoire d'un certain pourcentage de points (on le trouve dans [30] sous `RandomSampling` et on l'utilisera dans la partie 2 sous la dénomination `random`) ;
- Sélection du $n^{\text{ème}}$ élément de chaque échantillon de point, l'échantillon étant défini selon l'ordre des points du nuage de points de l'objet à échantillonner ; (`IntervalSampling`) ;
- Échantillonnage uniforme de points dans l'espace (`UniformSampling`) ;

- Sélection de points selon le poids de la normale de tangente des points appareillés (**MaxLeverageSampling**) ;
- Sélection de tous les points appartenant à un champs déterminé de forme quadrilatère ou polygonale (**Limits** ou **InPolygon**) ;
- Sélection de points appartenant à un VoxelHull spécifique, or des points appartenant à des parties communes à deux nuages de points, là où se produit un chevauchement (**InVoxelHull**) ;
- Sélection des k plus proches points voisins de chaque point dans un autre nuage de points (**KnnSearch**) ;
- Sélection des points constituant un contour vertical (**Profile**) ;
- Sélection de points selon un coefficient de robustesse pour la distance entre les points appareillés ou les ou pour l'angle entre la normale des points appareillés (**Attribute**). Selon [31] il est plus intéressant d'échantillonner ces points selon l'orientation de leurs normales, au lieu d'effectuer un tirage aléatoire sur l'ensemble des données.

Il est préférable de trier, d'abord, et regrouper dans des ensembles différents ces points en fonction de l'orientation de leurs normales. Ensuite, un échantillonnage uniforme est effectué sur chaque ensemble. Mais il arrive d'avoir besoin d'échantillonnage léger. Pour l'initialisation, on utilise des méthodes de filtrages et de dé-bruitage pour lisser l'image et/ou la suppression des points aberrants.

2.1.2 Matching des nuages de points

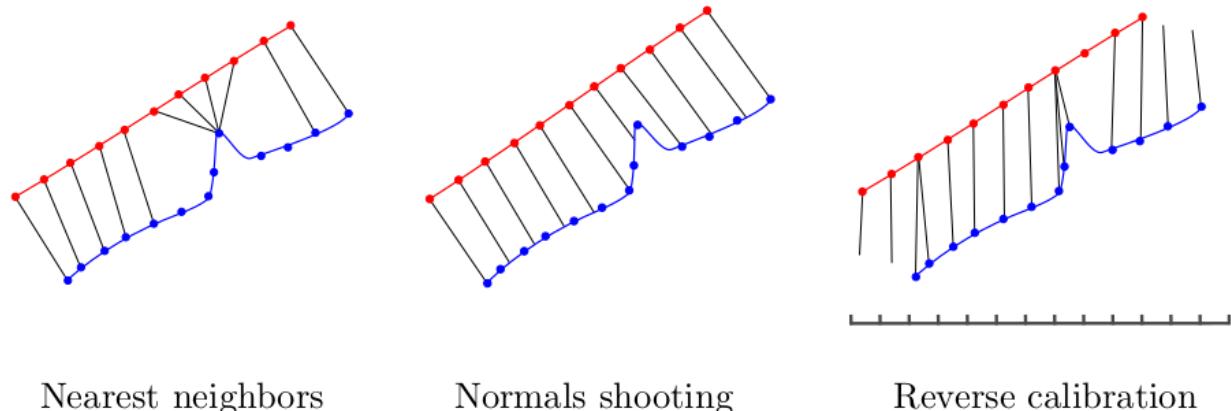


Figure 2.2: ICP matching strategies in two dimensions. Blue points represent the model point clouds to which red data points are matched. In the reverse calibration case, the points are projected onto the discretized, lower dimensional camera view space.

Des méthodes d'accélération de mise en correspondance existent tel que KD tree, cependant ça ne garantie pas de donner les mêmes résultats à chaque fois.

Le kd-Tree est selon [32] une structure permettant d'organiser des données présentes dans un espace à k -dimensions selon leur répartition spatiale. Cette structure est très utile pour de nombreuses applications, comme par exemple, pour accélérer la recherche de données dans un espace multi-dimensions, la recherche d'intervalles, ou encore la recherche de plus proches voisins.

Le kd-Tree est un cas particulier des « *Binary Space Partitionning (BSP) trees* ». Les BSP trees subdivisent l'espace à k -dimensions en coupant chaque volume englobant en deux sous-volumes par un plan de l'espace, et en ré-itérant récursivement sur ces deux nouveaux volumes ainsi obtenus. Les BSP trees peuvent donc être représentés par un arbre binaire où les deux sous-volumes sont les deux fils du nœud correspondant

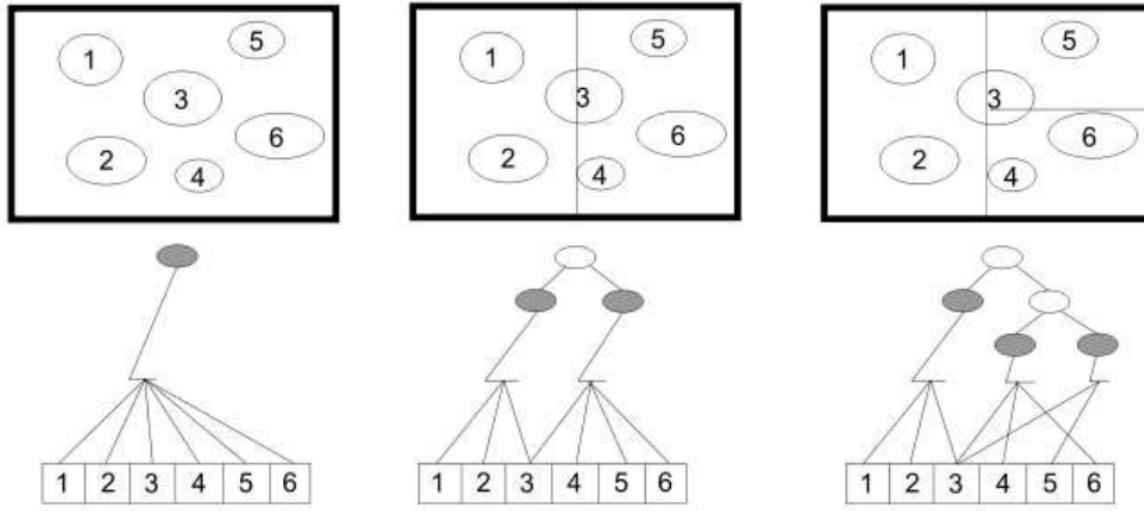


Figure 2.3: Exemple de BSP Tree dans un espace 2D

au volume englobant de niveau supérieur (cf. figure n o 1). Les plans de coupe peuvent être choisis en fonction de la répartition des données, afin qu'il y ait des volumes englobants de grande taille, là où il n'y a pas une grande concentration de données et inversement.

Dans le cas du kd-Tree, ces plans séparateurs sont toujours choisis de telle façon que leur normal soit un des axes du système de coordonnées de l'espace (plans toujours perpendiculaires aux axes comme dans la figure n o 1). Cela permet de simplifier la construction, mais aussi le parcours de l'arbre.

Le rôle du kd-Tree est double : il permet, d'une part, d'avoir une subdivision spatiale optimisée de l'espace permettant d'accélérer le traitement des données et donc de d'accélérer la mise en correspondance des nuages de points, d'autre part, de stocker les données sous la forme d'un arbre binaire.

2.1.3 Pondération des paires

La pondération des couples de points appariés a pour objectif de renforcer l'apport des appariements supposés être corrects et atténuer l'effet des faux appariements.

La convergence de ICP dépend beaucoup de la qualité des appariements utilisés. En effet, La présence de faux appariements, dans le meilleur des cas, ralentit la convergence de l'algorithme et peut, au pire des cas, causer sa divergence.

Une méthode de mise en correspondance robuste est, donc, indispensable. La distance géométrique euclidienne classique n'est, peut être, alors, plus suffisante pour établir des appariements relativement corrects et nécessaires à la convergence de ICP. Ainsi, d'autres critères qu'une simple distance euclidienne (ou en plus de celle ci) peuvent être utilisés.

2.1. TAXONOMIE DE L'ICP

Activités Visionneur de documents lun 20:47

O s3.eu-centr... | (1690) Nar... | Download | The KITTI | Build a Ma... | Localizatio... | Velodyne | Accès rapide | +

ComparaisonCauchyWelshHuberTukey.pdf 138,08% 3 sur 7

Vignettes

all pairs. Beyond L_2 , outlier filtering is all about the choice of this weight function $w(\cdot)$ and its configuration. Table I shows a list of outlier functions used in this paper, with a dedicated column highlighting their implementation of $w(\cdot)$.

TABLE I: Descriptive table of robust cost functions used in this analysis expressed with respect of their tuning parameter k and the scaled error e .

Functions	Conditions	Cost $\rho(e)$	Weight $w(e)$	M
L_2		$\frac{e^2}{2}$	1	✓
L_1		$\frac{1}{ e }$	1	✗
Huber	$ e \leq k$ otherwise	$\begin{cases} \frac{e^2}{2} \\ k(e - k/2) \end{cases}$	$\begin{cases} 1 \\ \frac{k}{ e } \end{cases}$	✓
Cauchy		$\frac{e^2}{2} \log(1 + (e/k)^2)$	$\frac{1}{1 + (e/k)^2}$	✓
GM		$\frac{e^2/2}{k+e^2}$	$\frac{k^2}{(k+e^2)^2}$	✓
SC	$e^2 \leq k$ otherwise	$\begin{cases} \frac{e^2}{2} \\ \frac{2ke^2}{k+e^2} - k/2 \end{cases}$	$\begin{cases} 1 \\ \frac{4k^2}{(k+e^2)^2} \end{cases}$	✓
Welsch		$\frac{k^2}{2} (1 - \exp(-(\frac{e}{k})^2))$	$\exp(-(\frac{e}{k})^2)$	✓
Tukey	$ e \leq k$ otherwise	$\begin{cases} \frac{k^2}{2} (1 - (1 - (\frac{e}{k})^2)^2) \\ \frac{k^2}{2} \end{cases}$	$\begin{cases} (1 - (e/k)^2)^2 \\ 0 \end{cases}$	✓
Student			$\frac{(k+3)(1+\frac{e^2}{k})^{-\frac{k+3}{2}}}{k+e^2}$	✗
Max. Dist.	$ e \leq k$ otherwise	$\begin{cases} \frac{e^2}{2} \\ \frac{k^2}{2} \end{cases}$	$\begin{cases} 1 \\ 0 \end{cases}$	✓
Trimmed	$e \leq P_f$ otherwise		$\begin{cases} 1 \\ 0 \end{cases}$	✗

Legend: M = Is the function a M-Estimator?

A. Hard rejection

Outlier filters categorized as hard rejection define the result of $w(\cdot)$ to be binary (i.e., either zero or one). The two most common solutions are *Max. distance* and *Trimmed*. The

The influence function $\psi(\cdot)$ is used to evaluate whether an M-estimator is robust or not. If $\psi(\cdot)$ is non-monotonic (i.e., redescending) and is null for an error that tends to infinity, the M-estimator is considered robust. As for the weight function $w(\cdot)$, it is given for convenience since it is the only part required to implement an M-estimator for an IRLS solution, as in Equation 3. In the soft rejection algorithms considered in this paper (shown in Table I), two are not M-estimators: L_1 has a singularity for an error that is equal to zero, and *Student* has an undefined $\rho(\cdot)$.

It is worth noting that some soft rejection functions are related. For instance, *Huber* uses a parameter k to combine L_1 and L_2 , in order to avoid the singularity at $e = 0$ of L_1 . Also, *Switchable-Constraint* (labeled *SC* hereafter), typically used in pose graph Simultaneous Localization and Mapping (SLAM), was expressed as a combination of L_2 and *Geman-McClure* (labeled *GM* hereafter) using a parameter k . It shares the same cost function as Dynamic Covariance Scaling [16]. *SC* is expected to have similar results to *GM* for extreme values of k .

C. Estimating the scale

As expressed in Equation 2, we used a scaled error in our ICP implementation. Contrary to the filter parameter k , which should be globally constant, the scale s is related to the point clouds and can be either fixed or estimated at every iteration. The scale s relates to the uncertainty for which paired points with a certain error should be considered as outliers. There are multiple estimators for the scale, two of the most interesting are: 1) Haralick *et al.* [21] used the Median of Absolute Deviation (MAD) as a scale estimator

Figure 2.4: provisoire : Descriptive table of robust cost functions used in this analysis expressed with respect of their tuning parameter k and the scaled error e . file: <:///home/blad/Documents/Études/M2/Mémoire/SLAM/notYet/ComparaisonCauchyWelshHuberTukey.pdf>

Fonctions	Condition	Coût $\rho(e)$	Poids $w(e)$	M
L_2				
L_1				
Huber				
Cauchy				
GM				
Welsch				
Tukey				
Max. Dist.				
Trimmed				

2.1.4 Réjection des paires

Tout comme l'étape précédente qui pondère les paires correspondantes, l'étape de rejet assigne un poids binaire, et rejette certaines paires jugées comme erronées, ou comme outliers, ce qui est très bénéfique lors de l'application des moindres carrés.

Afin d'obtenir une estimation satisfaisante, la rejet à partir d'un seuil D_{icp} fait de ce paramètre l'un des plus importants dans l'algorithme ICP, il permet en effet d'exclure les "outliers" des points candidats à l'appareillage. Si la valeur de ce seuil est minimale alors ça pourrait amener à des correspondances inadéquates et alors l'estimation est condamnée à converger vers un optimum local. Dans le cas contraire, une valeur trop importante pourrait introduire des correspondances incorrectes qui vont détériorer la précision de l'estimation et la vitesse de convergence en prendra un coup.

Il est très important de rejeter les appariements supposés faux. Ceux-ci induisent en erreur les estimateurs par moindres carrés. La difficulté réside dans la définition même d'un mauvais appariement". *Sur quel critère se baser pour décider qu'un appariement est faux ?* La façon la plus basique d'éliminer ces points, est d'utiliser un seuil de distance géométrique ou mixte (comportant un terme photométrique) fixe. Ce seuil serait difficile

à fixer dans le cas de ICP, étant donnée que d'une itération à l'autre la distance moyenne entre les points appariés est sensée décroître. Diverses méthodes déterminent et mettent à jour la rejetion par seuillage[7][29], dont :

- **Constant (Distance) :** C'est la méthode la plus directe pour définir un seuil, on détermine manuellement une valeur fixe. Dans chaque itération, toutes les paires candidates dont la distance est plus importante que celle de la valeur du seuil sera rejetée, et considérée comme une correspondance invalide. Le concept est simple, mais la difficulté réside dans le fait de déterminer la valeur adéquate du seuil d'autant plus que l'erreur de transformation entre les deux scans décroît au cours des itérations.

Constant (Angle) : De même que dans la méthode précédente un seuil est fixé, cela dit cette fois ce n'est pas une distance qui le caractérise mais plutôt un angle, ou plus précisément l'angle entre les normales des points appareillés ne doit pas dépasser un certain seuil.

Médian : Dans cette méthode le seuil équivaut à la somme des moyennes et de la déviation standard des distances entre les paires candidates, les paires rejetées sont celles dont la distance point à point est plus importante que certains multiples de la déviation standard des distances. Un exemple de cette méthode serait [33] qui rejette les paires dont la distance euclidienne représente plus de 2.5 fois la déviation standard. Cette méthode est elle aussi directe et ne requiert pas de configuration manuelle des paramètres, cependant elle suppose que la distribution des distances est gaussienne, ce qui est loin d'être vrai dans certains cas réels.

Median Plan : Maximum acceptable de la robustesse des plans extraits, et pour cette fois c'est la dérivée du plan qui est utilisée.

Median Généralisé : Une limite de distance pour les correspondances point à plan afin de filtrer les points aberrants (cette méthode est un estimateur médian (M-estimateur) robuste connu sous *Generalised Median* (GM) pour une base de donnée supposée être conforme à une distribution gaussienne).

Trim : Cette approche commence par en compte toutes les paires selon leurs distance, et ce n'est qu'après qu'elle inclus les paires candidates dont la distance correspond à un certain pourcentage par rapport à toutes les distances de toutes les paires candidates. Cette méthode dépend moins de la forme de la distribution, cependant, elle a besoin de prendre en compte l'intégralité des paires dans chaque itération ce qui augmente le coût de calcul. Dont la méthode proposée par [34] qui rejette 10% des paires.

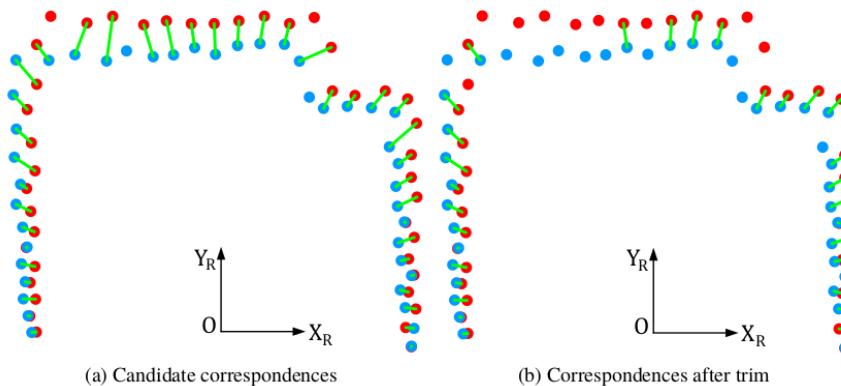


Figure 2.5: Méthode Trim de Rejection

Max.Dist : Ou la rejetion des paires qui ne consistent pas de paires avoisinantes, en assumant que la surface est rigide lorsque qu'elle se mouvoit. Ce schéma classe les deux appareillements (p_1, q_1) et (p_2, q_2) comme invalides si :

$$|Dist(p_1, p_2) - Dist(q_1, q_2)| > D_{icp} \quad (2.1)$$

Alors selon [35] on applique

$$0.1 \times \max \text{Dist}(p_1, p_2), \text{Dist}(q_1, q_2) \quad (2.2)$$

en tant que seuil.

Maillages de bordures : Ou maillage aux limites, cette stratégie exclue les points se situant sur les bordures du maillage[36], c'est spécialement utile pour éviter des appareillements erronés (causant ainsi une erreur systématique dans l'estimation de la transformée) dans le cas de superpositions de scans qui ne correspondent à deux perspectives légèrement décalées, comme c'est le cas dans la figure 2.6.

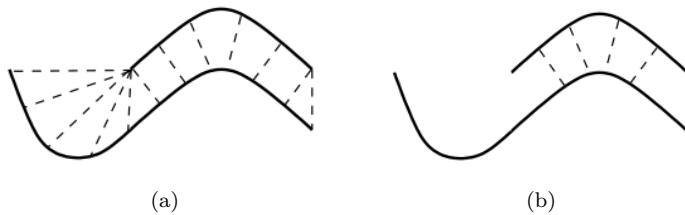


Figure 2.6: (a) When two meshes to be aligned do not overlap completely (as is the case for most real-world data), allowing correspondences involving points on mesh boundaries can introduce a systematic bias into the alignment. (b) Disallowing such pairs eliminates many of these incorrect correspondences.

Dans le document [29] la méthode par excellente serait celle du maillage de bordures, cela dit, dans nos simulations on a utilisé :

2.1.5 Optimisation des erreurs

On parle de minimisation d'Indice Clé de Performance aussi appel ICP, à ne pas confondre avec l'ICP(*Iterative Closest Point*). Il existe divers indicateurs d'erreur de performance tel que MSE (*Mean Squared Error*) ou l'erreur quadratique moyenne, RMSE (*Root Mean Square Error*) ou Racine de l'erreur quadratique moyenne, ou encore les MASE, MAPE, SMAPE, etc. Dans le cas de l'ICP, l'optimisation des erreurs peut être représentée en deux étapes :

La minimisation des critères de distance

Le calcul de la meilleure transformation revient à minimiser un critère de distance. L'erreur du critère de distance (que ce soit la distance entre deux point, ou deux normales du vecteur de distance, ou autre) est minimisée via l'une de ces 3 approches phares :

Minimisation Point à Point qui fait la somme des distances carrés de nuages de points d'un modèle, qu'on peut exprimer via :

$$E = \sum_{i=1}^N \|Rp_i + \vec{T} - q_i\|^2 \quad (2.3)$$

Ce critère est revient à minimiser une fonction de coût qui représente physiquement les distances entre la paire de points correspondants appartenant à deux nuages de points distincts.

Minimisation Point à Plan optimise la somme des distances des normales tangentes aux plans auxquels appartiennent les paires de points correspondants, mathématiquement formulé, cette fonction de coût correspond à :

$$E = \sum_{i=1}^N [(Rp_i + \vec{T} - q_i) \cdot \vec{n}_i]^2 \quad (2.4)$$

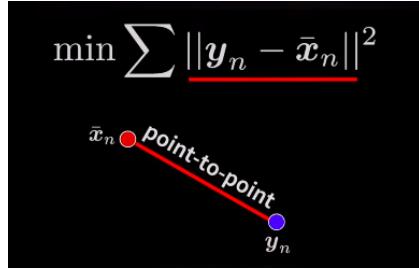


Figure 2.7: Minimisation point à point

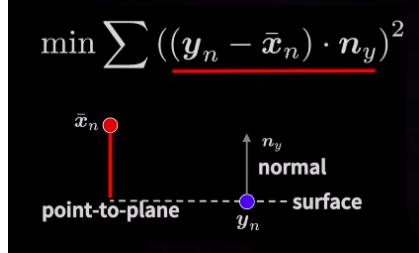


Figure 2.8: Minimisation point à plan

Où \vec{n}_i représente l'estimation de la normale tangente au $i^{\text{ème}}$ modèle de points. L'interprétation physique de cette fonction de coût est que les arrêtes entre les points de la même paire sont libre de se déplacer et de changer dans le modèle de point tangent aux plans, et fixes par rapport au nuage de point, c'est une caractéristique très avantageuse qui fait la force de l'algorithme de l'ICP point à plan.

Il existe d'autres fonctions de coût comme la norme L_1 qui peut être utilisée à la place de la norme euclidienne de l'équation 2.3.

Localisation

Une fois les points créés on a besoin d'identifier la pose (position et orientation) du robot, et pour ce divers Algorithmes existent qui vont calculer une solution à chaque itération :

Pour l'ICP point à point, il est très courant d'utiliser la SVD (Décomposition en valeur singulière) décrite dans l'annexe B et qu'on va appliquer pour simuler l'algorithme de l'ICP original (voir la théorie dans la section 2.3 et la simulation dans 1). Cependant il existe d'autres méthodes pour obtenir le vecteur de translation \vec{T} et la matrice de rotation R dont une formulation en *quaternions* équivalente à la SVD[37], (quaternions unitaires [38], ou des quaternions duaux[39])[31].

La minimisation point à plan trouve une expression explicite de la solution (*closed form solution*) après linéarisation de la matrice de rotation R , ce qui est détaillé dans l'annexe C. La linéarisation introduit un erreur, mais à force de réitérer l'algorithme, l'erreur est minimisée et la matrice de rotation linéarisée se rapproche fortement de la réalité. Cette solution a été proposé par [40].

Pour d'autres critères d'optimisation, l'existence d'une solution explicite devient invraisemblable. Celà dit on peut toujours utiliser des méthodes non linéaires, bien que ce soit des approches délicates où l'équilibre entre forte précision et moindre coût de calcul devient le défaut phare. Celà dit ça nous offre la liberté de modéliser la fonction convoitée à notre bon vouloir.

Reste enfin les algorithmes dédiés, comme l'AMCL et la ML-AMCL qui sont utilisés dans le cas du filtre particulier avec un filtrage KLD, c'est un système 2D de localisation probabiliste pour des robots mobiles. C'est ce que fait ce noeud ROS[41], qui à base d'une carte, de scans, et de données de télémètre laser, transforme les messages en des situations de pose (position et orientation). C'est une approche dédié

2.2 Variantes

On abordera la fonction originale de l'ICP, mais puisqu'elle n'est pas très utilisée, on va aborder d'autres variantes plus présentes dans la pratique.

<https://youtu.be/ktRqKxddjJk?t=887>

<https://www.youtube.com/c/CyrillStachniss/search?query=ICP>

2.3 Vanilla ICP

Vanilla ICP ou ICP originale est l'algorithme de base qu'on va appliquer dans

Having two scans $P = p_i$ and $Q = q_i$ we want to find a transformation (rotation R and translation t) to apply to P to match Q as good as possible. In the remainder of this notebook we will try to define what does as good as possible mean as well as ways to find such a transformation.

2.3.1 Generate example data

Throughout this notebook we will be working with generated data that looks like figure 2.9 :

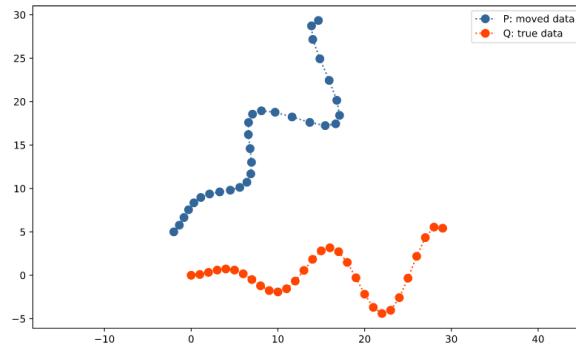


Figure 2.9: blablabla

2.3.2 Correspondences computation

We compute correspondences from P to Q , i.e. for every p_i we search the closest q_j to it.

2.3.3 ICP based on SVD

Il existe diverses alternatives à la SVD, mais c'est une méthode qui donne d'excellents résultats, qui est détaillée dans l'annexe B.

If the scans would match exactly, their cross-covariance would be identity. Therefore, we can iteratively optimize their cross-covariance to be as close as possible to an identity matrix by applying transformations to P . Let's dive into details.

Single iteration

In a single iteration we assume that the correspondences are known. We can compute the cross-covariance between the corresponding points. Let $C = \{i,j : p_i \approx q_j\}$ be a set of all correspondences, also $|C| = N$. Then, the cross-covariance K is computed as:

$K = \frac{1}{N} \sum_{(i,j) \in C} (p_i - \bar{p})(q_j - \bar{q})^T$ Each point has two dimensions, that is $p_i, q_j \in \mathbb{R}^2$, thus cross-covariance has the form of (we drop indices i and j for notation simplicity):

$K = [\text{cov}(px, qx)\text{cov}(py, qx)\text{cov}(px, qy)\text{cov}(py, qy)]$ Intuitively, cross-covariance tells us how a coordinate of point q changes with the change of p coordinate, i.e. $\text{cov}(px, qx)$ tells us how the x coordinate

of q will change with the change in x coordinate of p given that the points are corresponding. Ideal cross-covariance matrix is an identity matrix, i.e., we want the x coordinates to be ideally correlated between the scans P and Q , while there should be no correlation between the x coordinate of points from P to the y coordinate of points in Q . In our case, however, the position of P is derived from the position of Q through some rotation R and translation t . Therefore, whenever we would move the scan Q , scan P would move in a related way, but perturbed through the rotation and translation applied, making the cross-covariance matrix non-identity.

Knowing the cross-covariance we can compute its SVD decomposition:

$SVD(K) = USV^T$ (5) The SVD decomposition gives us how to rotate our data to align it with its prominent direction with UVT and how to scale it with its singular values S . Therefore:

$Rt == UVT^T Q R P (6)$ (7) Let's try this out: Make data centered : résultat dans la figure ??

Compute correspondences : résultat dans la figure ??

Compute cross covariance

Find R and t from SVD decomposition : Here we find SVD decomposition of the cross covariance matrix and apply the rotation to Q

Apply a single correction to P and visualize the result, voir figure 2.11 : This is the result after just one iteration. Because our correspondences are not optimal, it is not a complete match.

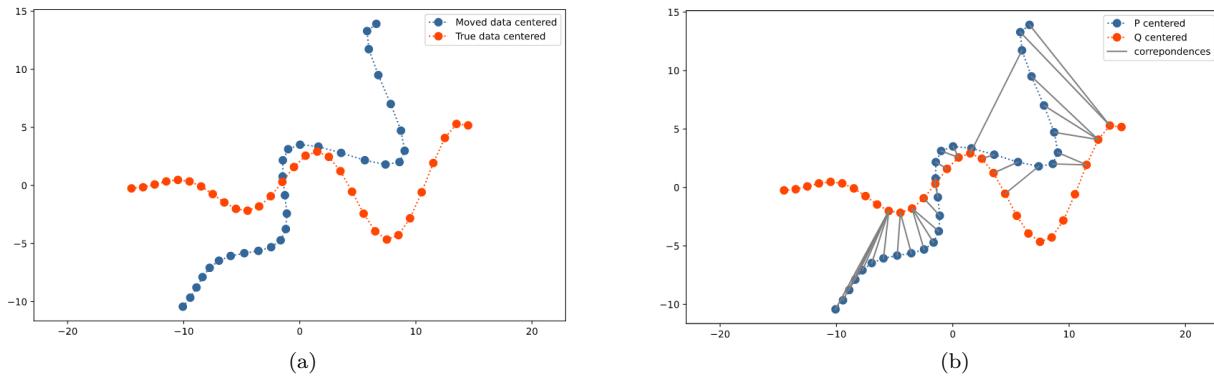


Figure 2.10: Première itération :
 (a) Make data centered; (b) Compute correspondences

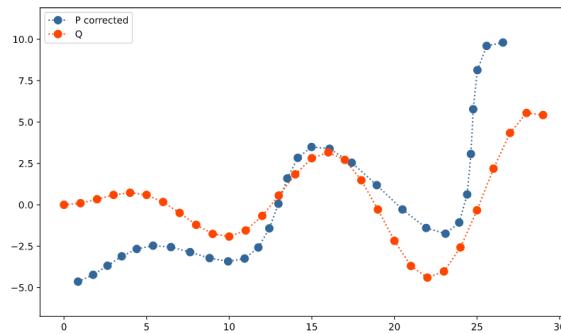


Figure 2.11: P corrigé après une itération d'ICP

Let's make it iterative

If we would know the correct correspondences from the start, we would be able to get the optimal solution in a single iteration. This is rarely the case and we need to iterate. That consists of the following steps:

Make data centered by subtracting the mean Find correspondences for each point in P Perform a single iteration by computing the cross-covariance matrix and performing the SVD Apply the found rotation to P

Repeat until correspondences don't change Apply the found rotation to the mean vector of P and uncenter P with it. Working example As we want to work with centered data and we will be iteratively centering the data, searching for rotation on centered data and uncentering the data at the end of each iteration. It is not the most elegant or efficient way, but it allows us to visualize the clouds nicer.

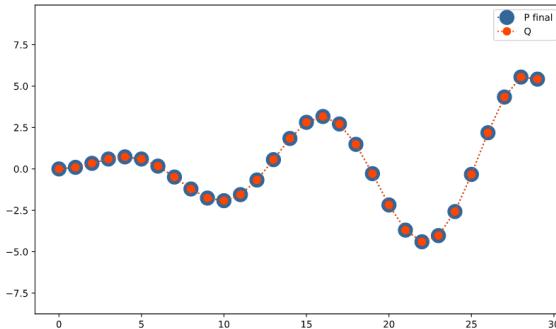


Figure 2.12: P corrigé après y avoir appliqué de multiples itérations de l'algorithme ICP

2.4 IRLS-ICP

13 FastAndRobust Iterative Registered Least Square

L'ICP classique mesure l'erreur d'alignement en utilisant la distance l_2 , qui pénalise les larges déviations de n'importe quel point du nuage de point source P au nuage de point Q . Cette méthode permet effectivement de parvenir à des résultats explicites (dits aussi en forme fermée) dans l'étape de l'alignement, mais peut tout de même amener à des appareillements erronés en cas de

2.5 ICP Point à Plan

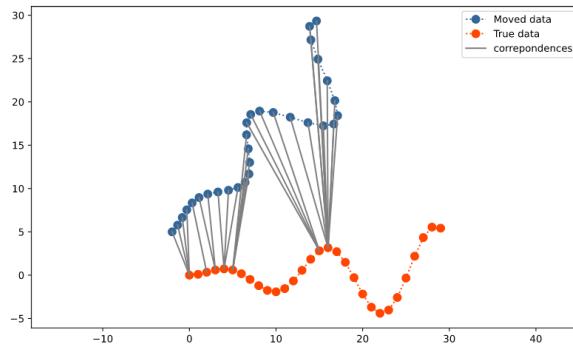


Figure 2.13: blablabla

2.6 GICP

L'ICP Généralisé ou Global ICP est un algorithme particulièrement résilient. <https://www.geo.tuwien.ac.at/downloads/pg/pctools/publish/globalICPHelp/globalICP.html>

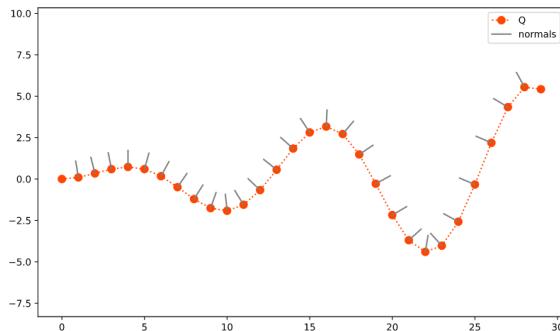


Figure 2.14: blablabla

2.7 Comparaison

L'ICP point à plan de même que le GICP estiment que les objets scannés sont des surfaces et non pas des points distincts.

Selon ces deux variantes de l'ICP, lorsqu'on dispose d'un télémètre laser qui fait la computation "des points sur les surfaces des objets scannés, ça n'implique pas pour autant qu'on scanne toujours les mêmes points. C'est pourtant bel et bien ce que l'ICP point à point fait, il essaie de minimiser la distance entre deux scans. Alors que les objets que nous balayons sont des surfaces, et donc que les points qui sont générés et traités résident quelque part sur cette surface, et c'est exactement ce que Le GICP et l'ICP point à plan prennent en considération.

Point à Plan : Le concept de la variante Poit à Plan de l'ICP est très proche de l'ICP point à point, cependant les différences qui les séparent mènent à des changements considérables dans la fonction de coût qu'on cherche à minimiser.

Dans l'ICP point à point nous prenons des nuages de points et recherchons dans l'autre nuage de points une correspondance pour tous chacun points. Par la suite, nous allons essayer de minimiser les distances carrées entre ces deux nuages de points.

Tandis que dans l'ICP point à plan, on prend additionnellement en compte les normales des surfaces des scans qu'on souhaite superposer ou aligner, puis on projette le vecteur d'erreur point à point (e.i. l'écart entre chacun des deux points les plus proches des deux nuages de points) sur la normale de la surface qui contient le vecteur d'erreur. Ce n'est fondamentalement rien de plus que de calculer le produit scalaire de deux vecteurs.

L'approche point à plan peut adopter la même solution de Vanilla ICP, la SVD qui permettrait de calculer la matrice de rotation et le vecteur de translation, ou on pourrait également l'exploiter avec d'autres méthodes tel que celle des quaternions afin de parvenir à exploiter la méthode des moindres carrés.

3 NDT

La transformation de distribution normale (NDT) peut être décrite comme une méthode pour représenter de façon compacte une surface. Cette méthode a été proposée par Biber et Straßer en 2003 [42] comme une méthode pour l'alignement des scans 2D. Biber et Straßer ont ensuite développé la méthode dans un document commun avec Sven Fleck [43], également dans le contexte de l'alignement et de la cartographie. Dans leurs travaux, 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[44].

La NDT est une méthode qui consiste à associer deux scans 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)[43]. Similaire 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 étant occupée, la NDT représente la probabilité de mesure d'un échantillon pour chaque position dans la cellule[14].

3.1 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 :

3.1.1 Collecte des points 2D

Cette phase consiste à collecter tous les points 2D $X_{i=1 \dots n}$ contenus dans cette cellule (n points). Chaque point X est représenté par ses coordonnées cartésiennes, soit :

$$X_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix}_{i=1 \dots n} \quad (3.1)$$

3.1.2 Calcul de la moyenne

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

$$Q = \begin{pmatrix} q_x \\ q_y \end{pmatrix} = \frac{1}{n} \sum_{i=1}^n X_i \quad (3.2)$$

3.1.3 Calcul de la covariance

Calculer la matrice de covariance

$$\Omega = \frac{1}{n} \sum_{i=1}^n (X_i - Q)(X_i - Q)^t \quad (3.3)$$

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

$$\begin{aligned} \Omega &= \frac{1}{n} \sum_{i=1}^n (X_i - Q)(X_i - Q)^t \\ &= \frac{1}{n} \sum_{i=1}^n \begin{pmatrix} x_i - q_x \\ y_i - q_y \end{pmatrix} \begin{pmatrix} x_i - q_x & y_i - q_y \end{pmatrix} \\ &= \frac{1}{n} \sum_{i=1}^n \begin{pmatrix} x_i - q_x^2 & (x_i - q_x)(y_i - q_y) \\ (x_i - q_x)(y_i - q_y) & y_i - q_y^2 \end{pmatrix} \end{aligned} \quad (3.4)$$

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 :

$$\Pi(X) = \exp\left(-\frac{(X - Q)^t \Omega^{-1} (X - Q)}{2}\right) \quad (3.5)$$

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.

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 !

3.2 L'alignement avec NDT

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

$$T : \begin{pmatrix} x' \\ y' \end{pmatrix} \mapsto \begin{pmatrix} \cos\phi & \sin\phi \\ \sin\phi & \cos\phi \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix} \quad (3.6)$$

Où $(t_x t_y)^t$ décrit la translation et 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 différentes (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 3.6.
 - Déterminer les distributions normales correspondantes pour chaque point transformé en utilisant l'équation 3.5.
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.
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 3.2 et 3.3. Les données d'odométrie pourraient être utilisées pour initialiser l'estimation. La transformation du deuxième scan est effectuée en utilisant T (équation 3.5) et la distribution normale correspondante est une recherche en utilisant l'équation 3.4 dans la grille NDT construite.

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

- $p = (p_k)^t_{k=1 \dots 3} = (t_x t_y \phi)^t$: Le vecteur des paramètres à estimer.
- X_i : Le point 2D de l'échantillon i du nouveau scan laser (correspondant à 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 laquelle 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 :

$$score(p) = \sum_{i=1}^n \Pi(X'_i) = \sum_{i=1}^n \exp\left(-\frac{(X'_i - Q_i)^t \Omega_i^{-1} (X'_i - Q_i)}{2}\right) \quad (3.7)$$

Ce "score" sera optimisé dans la section suivante.

3.3 Le processus d'optimisation avec la méthode de Newton

La NDT ne nécessite pas d'association point à point, et parvient à aligner les scans simplement en maximisant itérativement le "score" en utilisant l'algorithme de Newton[7].

Le fonctionnement de la NDT dépend de la maximisation du "score" (fonction 3.7). La méthode d'optimisation utilisée dans la NDT d'origine proposée dans [42].

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 fonction "score".

L'algorithme de Newton cherche itérativement les paramètres p qui minimisent une fonction f , tel que $p = (p_k)^t_k \in [1, 3]$. Chaque itération résout l'équation suivante :

$$H \Delta p = -g \quad (3.8)$$

Où g est le gradient transposé de f avec les entrées :

$$g_k = \frac{\partial f}{\partial p_k} \quad \text{avec } k \in [1, 3] \quad (3.9)$$

Et H est la matrice Hessienne de f avec des entrées

$$H_{kj} = \frac{\partial^2 f}{\partial p_k \partial p_j} \quad \text{avec } (k, j) \in [1, 3] \times [1, 3] \quad (3.10)$$

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

$$p \longleftarrow p + \Delta p \quad (3.11)$$

Cette étape est une étape dans une direction de descente, à condition que la matrice Hessienne H soit définie 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 choisi de sorte que H' soit définie positive.

Cet algorithme est maintenant appliqué à la fonction Score^* (équation 3.7). Le gradient et la matrice Hессienne sont construits en collectant les dérivées partielles de tous les sommets de l'équation.

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

$$Q = X'_i - Q_i \quad (3.12)$$

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 .

$$\begin{aligned} \frac{\partial Q}{\partial p_k} &= \frac{\partial X'_i - Q_i}{\partial p_k} \\ &= \frac{\partial X'_i}{\partial p_k} - \frac{\partial Q_i}{\partial p_k} \end{aligned} \quad (3.13)$$

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

$$\frac{\partial Q_i}{\partial p_k} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad (3.14)$$

Et alors l'équation 3.13 donne

$$\frac{\partial Q_i}{\partial p_k} = \frac{\partial X'_i}{\partial p_k} \quad (3.15)$$

Un élément de la somme de la fonction Score^* 3.7 est alors donné par :

$$\begin{aligned} s &= -\exp \frac{(X' - Q)^t \Omega^{-1} (X' - Q)}{2} \\ &= -\exp \frac{-Q^t \Omega^{-1}}{2} \end{aligned} \quad (3.16)$$

Les composants du gradient g sont alors :

$$\begin{aligned} g_k &= \frac{\partial s}{\partial p_k} \quad \text{avec } k \in [1, 3] \\ &= \frac{\partial s}{\partial Q} \frac{\partial Q}{\partial p_k} \\ &= Q^t \Omega^{-1} \frac{\partial Q}{\partial p_k} \exp \frac{-Q^t \Omega^{-1} Q}{2} \end{aligned} \quad (3.17)$$

Les dérivées partielles de Q par rapport à p sont données par la matrice Jacobienne J de T (voir équation 3.6) :

$$J = \begin{pmatrix} 1 & 0 & -x \sin \phi - y \cos \phi \\ 0 & 1 & x \cos \phi - y \sin \phi \end{pmatrix} \quad (3.18)$$

Les entrées de la matrice Hessienne H sont données par :

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

Les deuxièmes dérivées partielles de Q sont (voir l'équation 3.18) :

- si $k = j = 3$:

$$\frac{\partial^2 Q}{\partial p_k \partial p_j} = \begin{pmatrix} 1 & 0 & -x \sin \phi - y \cos \phi \\ 0 & 1 & x \cos \phi - y \sin \phi \end{pmatrix} \quad (3.19)$$

- sinon :

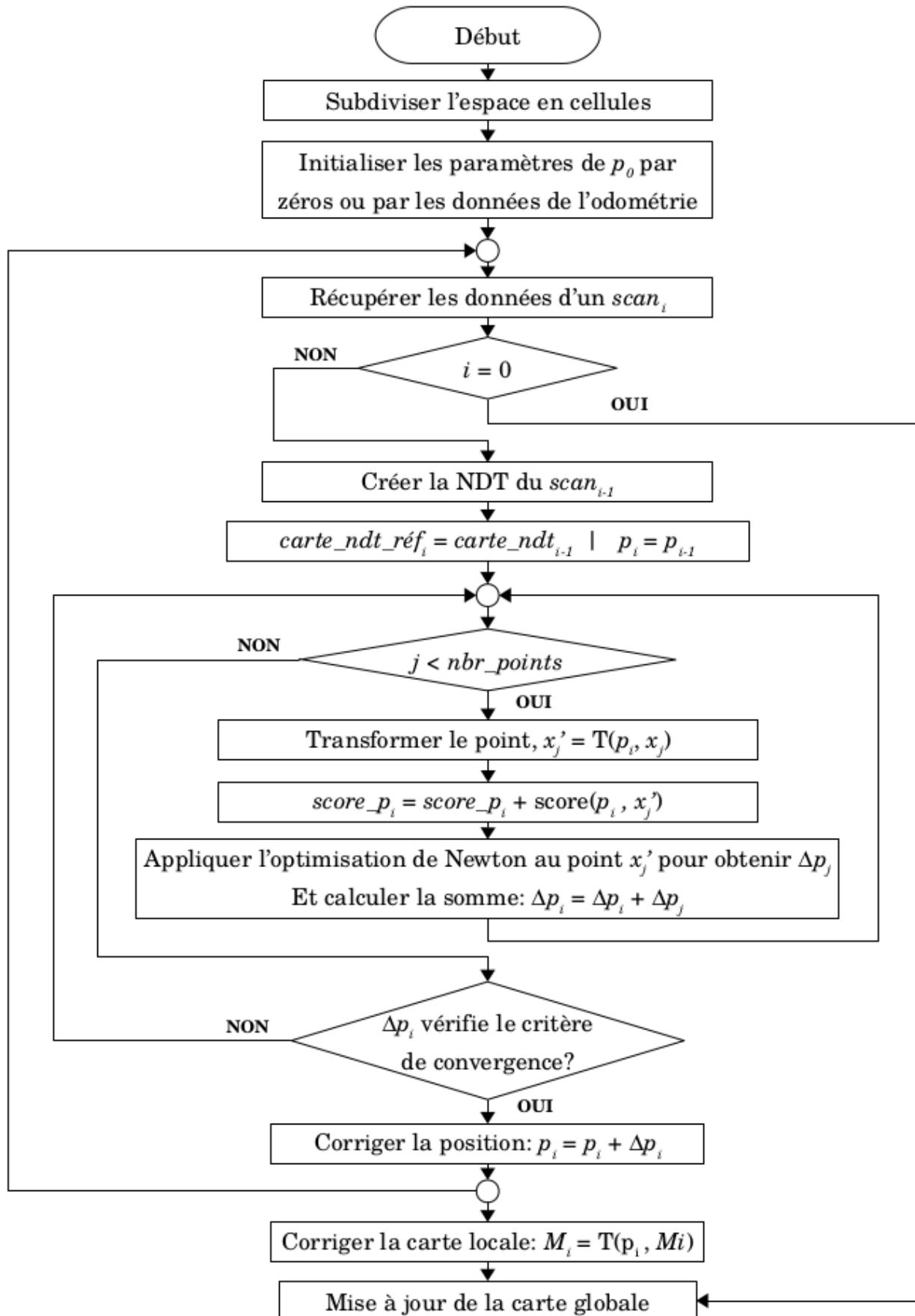
$$\frac{\partial^2 Q}{\partial p_k \partial p_j} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad (3.20)$$

L'intégralité des étapes citées sont résumées dans l'organigramme de la méthode de la NDT (voir figure 3.1)

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, les ouvrage [14][45][46] proposent de remplacer la méthode de Newton par l'optimisation par essaim particulaire, qui est une méthode évolutionnaire méta-heuristique très efficace.



Chapitre III

Simulations

Dans ce chapitre nous allons tester les algorithmes ICP et NDT dans l'optique de comparer leurs performances en 2D et en 3D. Afin de choisir le

1 Scan Matching en 2D

On va réaliser un SLAM sur un environnement de simulation Matlab, en commençant par implémenter un Scan Matching à base d'ICP Vanilla, sur une base de donnée disponible sur la toolbox Robotics de R2018. Le résultat devrait consister en la figure 1.1. Avec une carte de l'environnement, qu'on devrait reconstituer à base des données acquise d'un télémètre laser embarqué sur le robot alors qu'il réalise la trajectoire du point *Start* au point *End*.

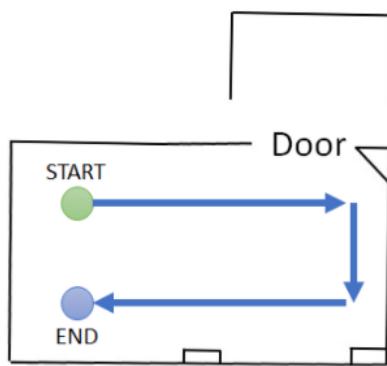


Figure 1.1: scanMatchingExample

En principe les données du télémètre laser devraient suffire pour cartographier et pour localiser le robot en calculant les poses relatives et tracer sa trajectoire. Mais on dispose également de données odométriques qui sont absolues et qui sont particulièrement précises dans ce cas, alors qu'en temps normal, les données acquises par l'odométrie ont tendance à manquer d'exactitude à force d'accumuler des erreurs au fil des mesures jusqu'à devenir inutilisable dans certains cas[47], d'où des approches de fusion données odométrique et d'autres approches, dont le meilleur exemple serait la VO.

Dans notre simulation nous allons commencer par faire un Scan Matching via l'ICP Vanilla, qui sera optimisé par la suite, et via la NDT, afin de comparer ces deux algorithmes.

1.1 ICP Vanilla

1.2 IRLS-ICP

L'algorithme a tendance à planter si on ne rend pas l'algorithme plus robuste, et pour ce on dispose de diverses méthodes de M-Estimateurs :

Figure ScanMatching-IRLS-ICP 2x2 avec les quatre méthodes pour des itérations proches de la phase de virage.

On constate que Welsch donne les meilleurs résultats, avec Cauchy qui côtoie de près son degré de précision, Tukey et Huber démontrent de moins bons résultats, qui restent tout de même bien meilleurs que la méthode l_2 .

Figure SLAM-IRLS-ICP-Welsch

Les résultats sont excellents tant que la rotation est minime, mais dès qu'elle devient importante, l'IRLS-ICP est très peu résilient dans les virages, et on peut voir de la figure refSLAM-IRLS-ICP-Welsch qui exprime la trajectoire et la carte résultant d'un SLAM via le M-estimateur de Welsh pour l'IRLS ICP.

Les résultats ne sont donc pas satisfaisants dans le cas de cette application de l'algorithme de l'ICP bien que ce soit une méthode robuste de la variante point à point, elle distingue mal les contours réels des limites du champs d'atteinte du télémètre laser. La robustesse de cet algorithmes réside dans l'exploitation d'une méthode de MLE qui est les M-Estimateurs, dont ici on a pu voir une comparaison entre l'estimateur d'Huber, de Welsch, de Cauchy, et un estimateur bi-weighted de Tukey(voir section 2.4).

1.3 IRLS-ICP et Odométrie

Étant donné que la dérive de la localisation accentue la dérive de la cartographie, on va calculer les poses à partir des données odométriques, et dessiner la carte de l'environnement à partir des données du télémètre laser et des poses déterminées via l'odométrie.

1.4 IRLS-ICP et Pose Graph

Afin d'appliquer l'optimisation via Pose Graph, nous ne pouvons pas utiliser les résultats obtenus dans la section 1.3 car le bouclage nécessite d'avoir une mesure absolue de l'environnement comme c'est le cas avec les données du télémètre laser ou d'une caméra ou d'un autre capteur extéroceptif. On l'appliquera donc au cas développé dans la section 2.4.

On ne peut pas faire un bouclage en utilisant un capteur relatif tel qu'un IMU ou un encodeur à roues car il n'y a pas de moyen de rapprocher les mesures d'une certaine itération à celles d'une pose précédente. On pourrait croire être proche d'un nœud, mais sans vérifier l'environnement on ne peut pas en être certain.

Bien que nous ne soyons pas dans un cas de bouclage de trajectoire nous pouvons créer des points repères, des points relatifs et les lier via des arêtes, et faire un bouclage par rapport à ces arêtes qui ne se résume pas qu'aux arêtes entre deux nœuds représentant les poses[48].

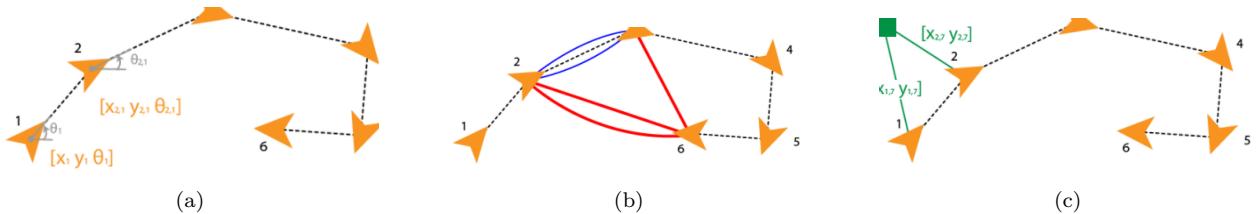


Figure 1.2: Pose Graph (a) estimation des nœuds pose et des arêtes les reliant; (b) Crédit de boucle en rajoutant une arête; (c) Rajouter un point nœud repère

Et on obtient le résultat : une figure ta3 les résultats du pose graph

Après avoir acquis le *Pose Graph* qu'on a optimisé, on peut à présent construire une carte de l'environnement à bas de toutes ces données de points. Il existe diverses méthodes pour représenter un modèle de l'environnement3.2. Parmi elles une grille d'occupation binaire :

figure Occupancy Binary Grid

1.5 NDT

1.6 Comparaison

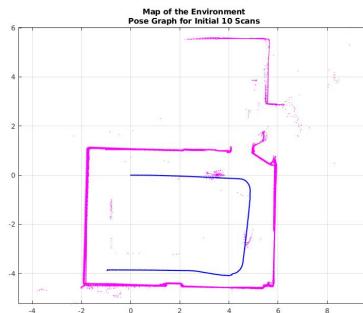


Figure 1.3: À corriger, ce ne sont pas les 10 premiers scans qui ont été pris en compte. C'est plutôt kamel les 690 scans

(exempleScanMatching)
 1 => NDT
 3 => ICP
 2 => Full Slam (LiDAR SLAM)
 4 => odometrie

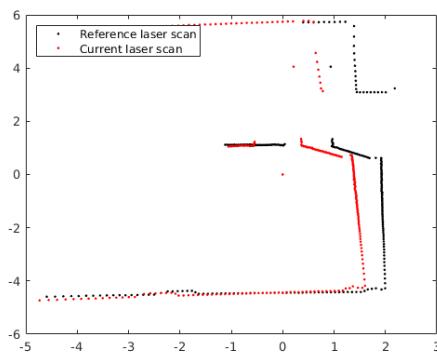


Figure 1.4: Les deux nuages de points en 2D à superposer

1.6.1 Comparaison

Nous pouvons constater une meilleure performance de l'algorithme de la NDT par la qualité de la superposition, et de l'exactitude de la mise en correspondance des nuages de points.

Même si on utilise les données odométriques, avec l'ICP pour reconstruire la carte de l'environnement à base des données recueillies par le télémètre laser, sachant que les données odométriques sont pertinentes et offrent une base solide pour dessiner avec une assez bonne exactitude afin de permettre au robot de naviguer en se basant sur l'odométrie pour le calcul de sa trajectoire. Les résultats obtenus via la NDT sont toujours meilleurs que ceux obtenus via l'ICP.

1.6.2 Occupancy Grid Map

Comparaison entre le full slam w le slam en temps réel (icp et ndt)

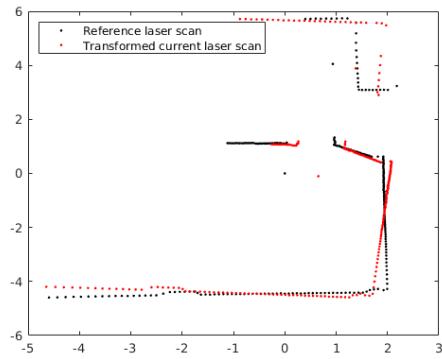


Figure 1.5: Scan Matching des deux nuages de points via l'algorithme de l'ICP

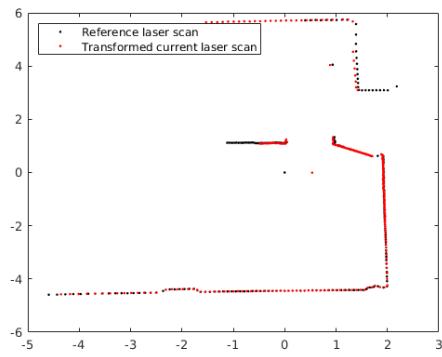


Figure 1.6: Scan Matching des deux nuages de points via l'algorithme NDT

Comprendre bien l'algo du full slam. On sait qu'il a générer les positions à partir des données du Télémètre laser. Le tout dans un algorithme de Pose Graphe[49].

https://fr.mathworks.com/videos/implement-simultaneous-localization-and-mapping-slam-with-matlab-152.html?s_eid=PSM_15028

<https://youtu.be/saVZtgPyyJQ>

À voir si l'environnement n'est vraiment pas complet. Ça serait intéressant de tester les mêmes algo en 2D sur un autre environnement ly on dispose de la réalité terrain ta3ou.

1.6.3 Comparaison

Ils sembles tous identiques.

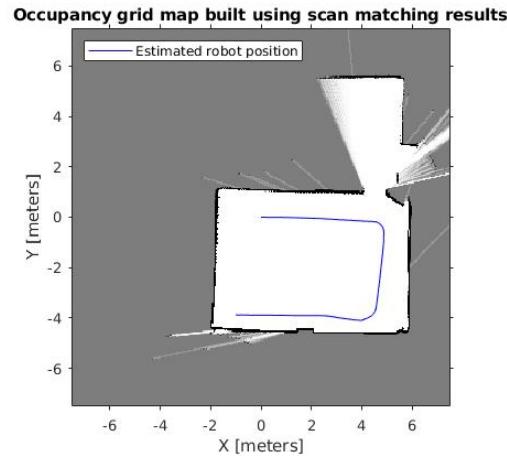


Figure 1.7: occupancyGridMap NDT occupancyGridMap NDT

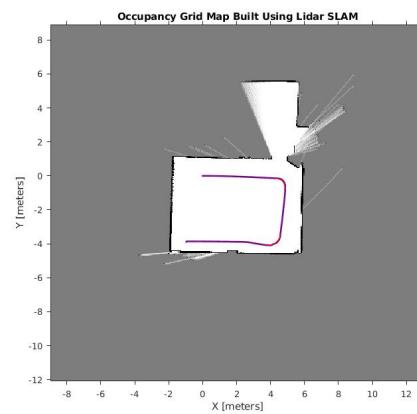


Figure 1.8: grid map full slam grid map full slam

2 Scan Matching en 3D

2.1 ICP en 3D

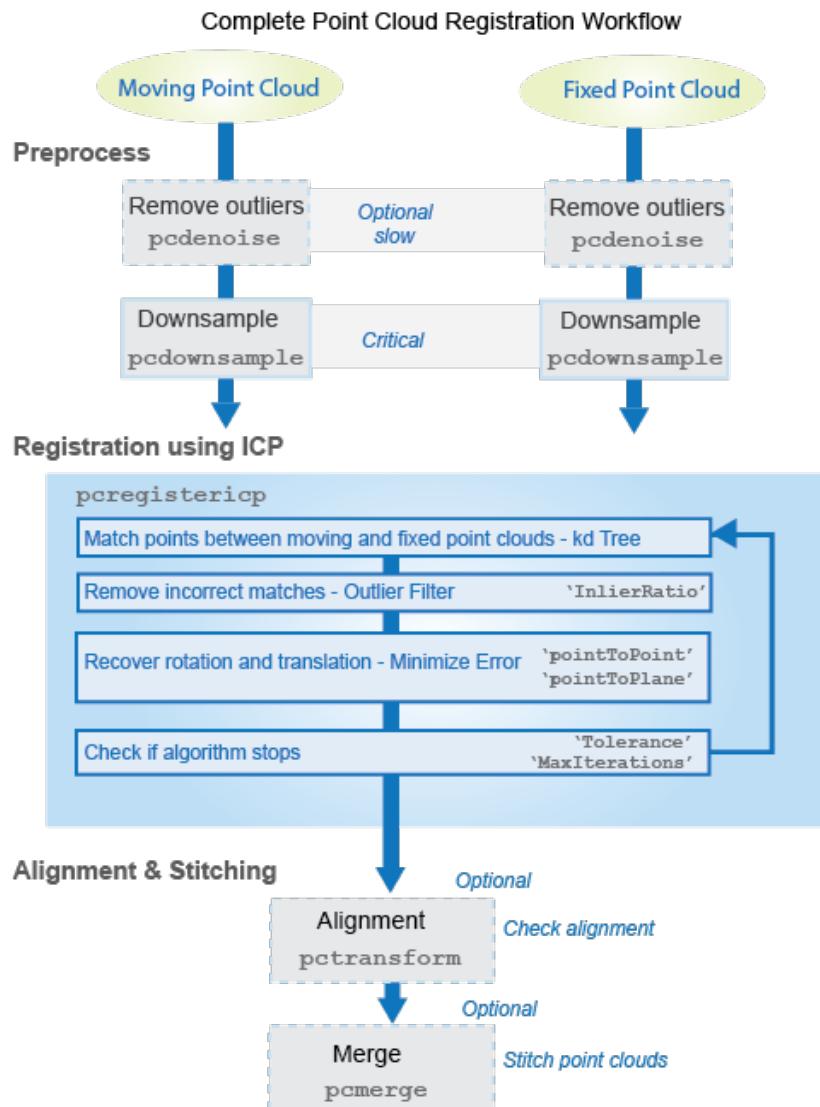


Figure 2.1: L'organigramme général d'ICP

Point to Point Vs. Point to Line Le temps de réponse est quasiment identique, et on ne remarque pas de différence entre le scan matching de deux nuages points, bessa7 quand on prend une frame"de nuages de

points alors, on constate la différence :

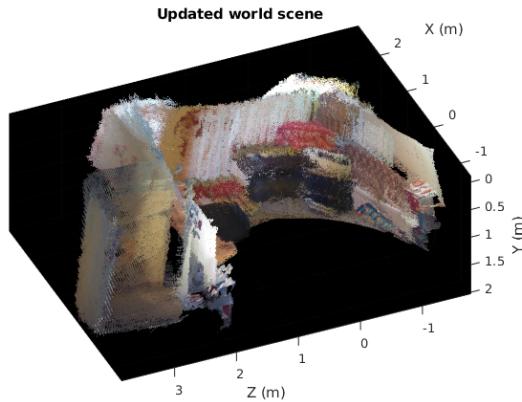


Figure 2.2: Résultat de la cartographie via scan matching basé sur la variante Point à Point de l’Algorithme ICP

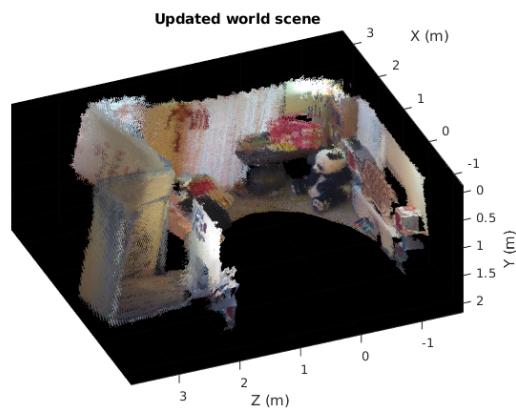


Figure 2.3: Résultat de la cartographie via scan matching basé sur la variante Point à Plan de l’Algorithme ICP

2.2 ICP Vs. NDT

ICP point à Point Vs. NDT Remarque : le fait de rallonger le pas diminue le temps et rend les résultats complètement sans sens. Il est impératif de trouver le juste milieu entre optimisation de la qualité de la carte et optimisation du temps de réponse.

2.2.1 Scan Matching de 2 nuages de points

2.2.2 Comparaison

ICP 3 secondes et des miettes NDT ma bin 6 et 7 sec

2.2.3 Scan Matching pour cartographier un environnement

Une séquence de ... de scans qui ont été superposés l’un après l’autre pour reconstruire une carte de l’environnement qui était jusque là inconnu.

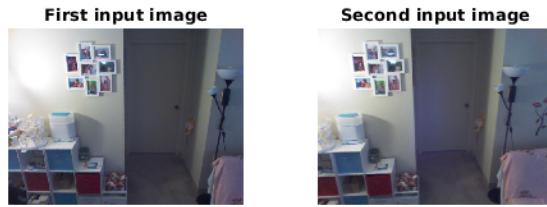


Figure 2.4: 2 images représentant la vérité terrain correspondant à 2 nuages de points consécutifs

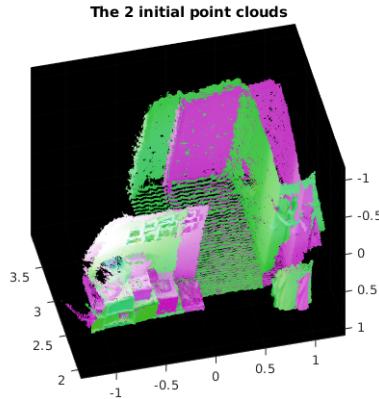


Figure 2.5: État initial de Scan Matching : 2 nuages de points consécutifs

2.2.4 Comparaison

ICP 16 secondes

NDT 60 sec min jusqu'à 90

Pour le filtrage, les fonctions `pcregistericp` et `pcregisterndt` sont des fonctions matlab qu'on utilise avec `pcdownsample` qui fait l'échantillonage, et qu'on retrouve dans la toolbox Vision de Matlab R2018a¹ et qui ont pour options de faire un échantillonnage `random` (aléatoire) ou `gridAverage` (qui calcul la moyenne des points faisant partie d'une même unité d'espace dans le cas où une unité d'espace contiendrait plus d'un point) cette seconde méthode devrait théoriquement préserver la forme du nuage de point mieux que l'échantillonnage aléatoire[50].

En pratique en remarque que la sélection de points `random` prend bien moins de points que la sélection `gridAverage` et est légèrement plus lente en plus de renvoyer des résultats lestement plus flous en plus de ne pas aligner les scans aussi bien que `gridAverage`.

¹Attention à utiliser une version qui inclut les ressources nécessaires pour lancer les fonctions `pcdownsample` et `pcmerge`, ce qui n'est pas le cas de la version 1 de Matlab R2021a

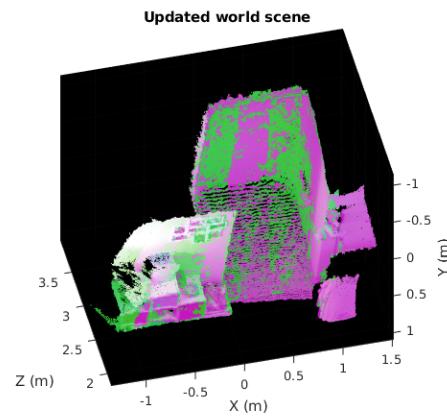


Figure 2.6: Scan Matching des 2 nuages de points via l'algorithme de l'ICP

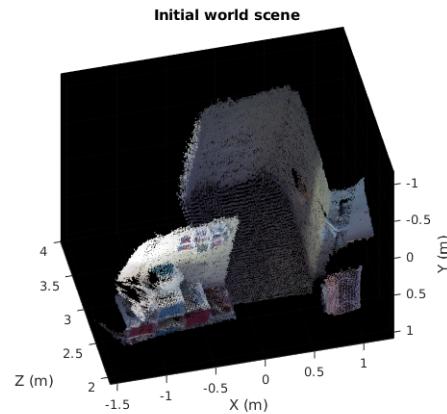


Figure 2.7: État initial de Scan Matching : 1er nuage de point

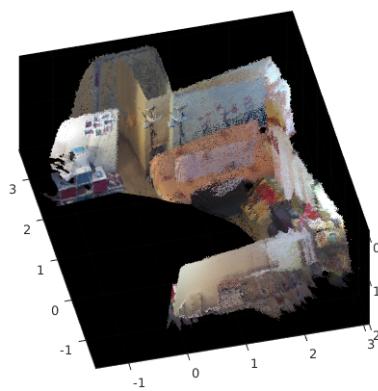


Figure 2.8: Scan Matching des 2 nuages de poits via l'algorithme de l'NDT

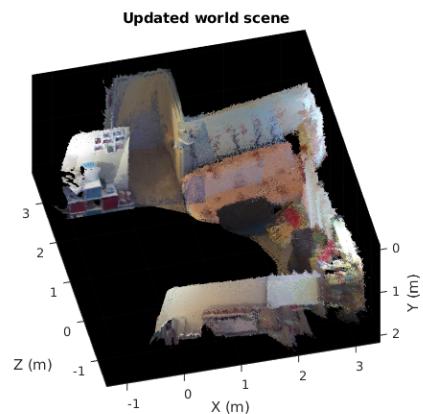


Figure 2.9: Scan Matching pour cartographier un environnement via l'algorithme de l'ICP

Chapitre IV

Annexes

A Annexe A : Le filtre de Kalman

A.1 Le filtre de Kalman-Shcmidt

Le filtre de Kalman traite le problème d'estimation de l'état x d'un processus discret déterminé par l'équation suivante :

$$x_t = Ax_{t-1} + Bu_t + w_{t-1} \quad (\text{A.1})$$

avec des mesures z ayant un bruit gaussien v qu'on peut exprimer ainsi :

$$z_t = Hx_t + v_t \quad (\text{A.2})$$

où :

- v_t représente le bruit des mesures $v \sim N(0, R)$
- w_t représente le bruit du processus $w \sim N(0, Q)$

A.1.1 Équations de prédiction et de mise à jour

L'état du système au moment de la réception de la prochaine mesure peut être prédit :

$$x_{t+1/t} = Ax_{t/t} + Bu_t \quad (\text{A.3})$$

La covariance de la prédiction de l'état :

$$P_{t+1/t} = AP_{t/t}A^T + Q_t \quad (\text{A.4})$$

A.1.2 Équation de mise à jour des mesures

L'innovation pondérée par le gain du filtre, plus l'état prédit, à partir de l'estimation de l'état mise à jour :

$$x_{t+1/t+1} = x_{t+1/t} + W_{t+1}v_{t+1} \quad (\text{A.5})$$

La covariance de l'état mise à jour :

$$P_{t+1/t+1} = P_{t+1/t} - W_{t+1}S_{t+1}S_{t+1}^T \quad (\text{A.6})$$

où : - L'innovation v est la différence entre la mesure réelle et la mesure prédite :

$$v_{t+1} = z_{t+1} - Hx_{t+1/t} \quad (\text{A.7})$$

- Le gain de Kalman W est :

$$W_{t+1} = P_{t+1/t}H_{t+1/t}^TS_{t+1}^{-1} \quad (\text{A.8})$$

- La covariance de l'innovation S est :

$$S_{t+1} = H_{t+1/t}P_{t+1/t}H_{t+1/t}^T + R_{t+1} \quad (\text{A.9})$$

où R est la covariance du bruit de mesure.

A.2 Le Filtre de Kalman étendu : EKF

Les équations de transition et de mesure ne sont pas toujours linéaires. Le filtre EKF est une extension du filtre de Kalman permettant de traiter ces problèmes de non linéarité. Les simplifications mathématiques introduites ont toutefois un inconvénient : les distributions de probabilités ne sont plus modélisées correctement et la linéarisation cause des inconsistances. Cependant, en pratique, les résultats sont souvent satisfaisants.

A.2.1 Équation de prédiction et de mise à jour

$$x_{t+1/t} = f(x_{t/t}, u_t) \quad (\text{A.10})$$

$$P_{t+1/t} = (\nabla_x f)_{t/t} P_{t/t} (\nabla_x f)_{t/t}^T + Q_t \quad (\text{A.11})$$

où : - f est l'équation de mise à jour de l'état - $x_{t/t}$ est l'estimation de l'état à l'instant t en se basant sur l'information à l'instant t - $x_{t+1/t}$ est l'estimation de l'état à l'instant $t+1$ en se basant sur le modèle de transition (sans intégrer l'information de mesure) - P correspond à la matrice de covariance - Q représente la matrice de covariance du bruit du processus

A.2.2 Équations de mise à jour des mesures

$$x_{t+1/t+1} = x_{t+1/t} + W_{t+1} v_{t+1} \quad (\text{A.12})$$

$$P_{t+1/t+1} = P_{t+1/t} + W_{t+1} S_{t+1} W_{t+1}^T \quad (\text{A.13})$$

Les équations de mise à jour des mesures ajoutent des informations à partir des nouvelles mesures afin de corriger les estimations faites à partir du modèle de transition. v est appelée l'innovation et correspond à l'ensemble de l'information non prédictive ayant été obtenue à partir des mesures. W est le gain de Kalman, et il exprime le degré de confiance qu'on a dans les mesures.

$$v_{t+1} = z_{t+1} - h x_{t+1/t} \quad (\text{A.14})$$

$$W_{t+1} = P_{t+1/t} (\nabla_x h)_{t/t}^T + S_{t+1}^{-1} \quad (\text{A.15})$$

$$S_{t+1} = (\nabla_x h)_{t+1/t} P_{t+1/t} (\nabla_x h)_{t+1/t}^T + R_{t+1} \quad (\text{A.16})$$

R est la covariance du bruit de mesure.

Filtre d'information étendu

Remarque le filtre d'information traite les systèmes qui acquièrent des données linéaires. Cependant le Filtre d'information étendu linéarise les systèmes non linéaires une fois, donc il donne de bons résultats localement donc si la linéarisation est correctement faite.

Approche FastSlam

Approche Amers

B Annexe B : La Décomposition SVD

Annexe A file://home/blad/Documents/Études/M2/Mémoire/ICP/bscthesis.pdf

C Annexe C : Minimisation Point à Plan

Annexe D file:///home/blad/Documents/Études/M2/Mémoire/ICP/bscthesis.pdf

Bibliographie

- [1] V. Gay-Bellile, D. Larnaout S. Bourgeois, and M. Tamaazousti. *Fundamentals of Wearable Computers and Augmented Reality*. 2nd ed. edition.
- [2] A. Geiger H. Lategahn and B. Kitt. Robotics and automation.
- [3] Y. Li, C. P. Chen B. Yu, W. Zhang N. Maitlo, and L. Mi. Deep neural network-based loop detection for visual simultaneous localization and mapping featuring both points and lines.
- [4] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part i. Vol.13(issue. 02):99 – 110.
- [5] C. Drocourt. Localisation et modelisation de l'environnement d'un robot mobile par coopération de deux capteurs omnidirectionnels.
- [6] R.I. Nourbakhsh R. Siegwart and D. Scaramuzza. *Introduction to Autonomous Mobile Robots*. 2nd ed edition.
- [7] Jixin LV. Scan matching and slam for mobile robot in indoor environment.
- [8] P. Cheeseman R. C.Smith. On the representation and estimation of spatial uncertainty. vol. 5:pp. 56,68.
- [9] H. Durrant-Whyte. Uncertain geometry in robotics. vol. 4.
- [10] Oussama El Hamzaoui. Localisation et cartographie simultanées pour un robot mobile équipé d'un laser à balayage : Coreslam.
- [11] H. Durrant-Whyte T. Bailey. Simultaneous localization and mapping (slam): Part ii. vol. 13(no. 2):pp.108–117.
- [12] W. Burgard S. Thrun and D. Fox. Probabilistic robotics (intelligent robotics and autonomous agent).
- [13] U. Frese. *A discussion of simultaneous localization and mapping*, volume vol. 20.
- [14] A. Hocine A. Bougouffa. Contribution à la localisation et la cartographie simultanées (slam) dans un environnement urbain inconnu.
- [15] Zachary S. Nahman. Robot learning for loop closure detection and slam.
- [16] Shu-Hao Yeh Hsin-Min Cheng Baifan Chen Yan Lu, Joseph Lee and Dezhen Song. Sharing heterogeneous spatial knowledge: Map fusion between asynchronous monocular vision and lidar or other prior inputs.
- [17] Pieter Abbeel. Advanced robotics(<https://people.eecs.berkeley.edu/~pabbeel/cs287-fa09/lecture-notes/lecture21-6pp.pdf>).
- [18] Cyrill Stachniss. Bayes filter(<https://www.youtube.com/watch?v=oUq0a8jHSQg>).
- [19] N. Laird A. Dempster and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. 39:1–38.
- [20] F. Lu and E. Milios. *Globally consistent range scan alignment for environment mapping*, volume Vol. 4, pages 333–349.

- [21] B. Wegbreit D. Hähnel, S. Thrun and W. Burgard, editors. *Towards lazy data association in slam*.
- [22] E.Milios F. Lu. Globally consistent range scan alignment for environment mapping. pages 333–349.
- [23] Nikolaus Correll. *SLAM as a Maximum-Likelihood Estimation Problem*, chapter Graph-based SLAM, pages 185–194.
- [24] Matlab. Scan matching overview(<https://www.mathworks.com/help/vision/point-cloud-processing.html>).
- [25] P. J. Besl and N. D. McKay. *Method for registration of 3-d shapes*.
- [26] Y. Chen and G. Medioni. *Image Vision Computing*, volume Vol. 10. Issue 3 edition.
- [27] N. D. McKay Besl, Paul J. A method for registration of 3-d shapes. Vol. 14:pp. 239–256.
- [28] Hans Martin Kjer and Jakob Wilm. Evaluation of surface registration algorithms for pet motion correction.
- [29] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm.
- [30] Philipp Glira. Point cloud tools for matlab (https://github.com/pglira/Point_cloud_tools_for_Matlab).
- [31] Marie-José Aldon et André Crosnier Lounis Douadi. Variantes de l'algorithme icp pour le recalage de données 3d couleur. page 9.
- [32] Cédric Fleury. Le kd-tree : une méthode de subdivision spatiale.
- [33] K. Sakae T. Masuda and N. Yokoya. Registration and integration of multiple range images for 3-d model construction.
- [34] K. Pulli. Multiview registration for large data sets.
- [35] J. Weng C. Dorai and A. Jain. Registration and integration of multiple object views for 3d model construction. Vol. 20(No. 1).
- [36] G. Turk and M Levoy. Zippered polygon meshes from range images.
- [37] B.K.P. Horn. Closed-form solution of absolute orientation using unit quaternions. 4:629–642, 1987.
- [38] O. Faugeras and M. Hebert. The representation, recognition and localisation of 3d objects. Vol. 5(No. 3).
- [39] L. Shao M. Walker and R. Volz. Estimating 3d location parameters using dual number quaternions. Vol.5(No. 3).
- [40] Y. Chen and G. Medioni. Object modeling by registration of multiple range images. In IEEE, editor, *International Conference on Robotics and Automation*, volume Vol.3, pages 2724–2729.
- [41] Dieter Fox. Wiki : Amcl (<http://wiki.ros.org/amcl>).
- [42] W. Straßer P. Biber, editor. *The normal distributions transform : A new approach to laser scan matching*, volume 3. IEEE/RSJ International Conference on (2003).
- [43] Peter Biber, Sven Fleck, and Wolfgang Strasser. A probabilistic framework for robust and accurate matching of point clouds. In Carl Edward Rasmussen, Heinrich H. Bühlhoff, Bernhard Schölkopf, and Martin A. Giese, editors, *Pattern Recognition*, pages 480–487. Springer Berlin Heidelberg.
- [44] M. Magnusson. The three-dimensional normal-distributions transform : an efficient representation for registration, surface analysis, and loop detection.
- [45] A. El Dor. Improvement of particle swarm optimization algorithms : applications in image segmentation and electronics.

BIBLIOGRAPHIE

- [46] M. Clerc and P. Siarry. Une nouvelle métaheuristique pour l'optimisation difficile : la méthode des essaims particulaires.
- [47] Fabio Morbidi. Localisation et navigation de robots(https://home.mis.u-picardie.fr/~fabio/Eng/documenti/Teaching/LNR20-21/LNR_p1.pdf).
- [48] G.Grisetti, C. Stachniss R. Kummerle, and W. Burgard. A tutorial on graph-based slam. Vol. 2(No. 4):pp. 31–43.
- [49] Mihir Acharya. Implement simultanious localization and mapping (<https://www.mathworks.com/matlabcentral/fileexchange/66284-implement-simultaneous-localization-and-mapping-slam-with>).
- [50] Matlab. pcdownsample documentation <https://fr.mathworks.com/help/vision/ref/pcdownsample.html#bupqqn1-1-gridAverage>.