

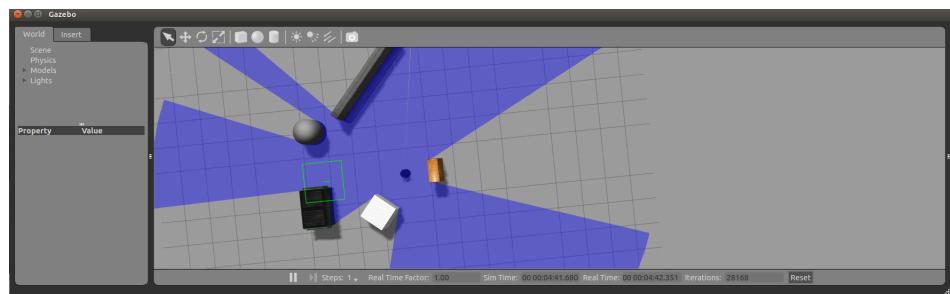


# UNIVERSITÉ AMAR TELIDJI DE LAGHOUAT

## DÉPARTEMENT D'ÉLECTRONIQUE

MÉMOIRE DE MASTER  
EN AUTOMATIQUE ET INFORMATIQUE INDUSTRIELLE

## LiDAR SLAM



Abir Zineb BLIDI  
Promo 2021

*Encadreur : Mme. Fatima CHOUIREB*  
*Co-Encadreur : M. Yacine AHMINE*

## JURY

**Mme. Flana Feltana** Directrice de Recherche, UATL  
**Mme. Flana Flan**, Professeur, UATL  
**M. Flan Feltan**, Maître de Recherche, UATL

Président du jury  
Examinateur  
Examinateur

1<sup>ier</sup> Juillet 2019 – 15 Septembre 2021

# Remerciements

Nous avons commencé à travailler sur ce mémoire depuis les vacances estivales, pendant les premiers mois, j'ai eu le plaisir et l'honneur d'être orientée par *M.Ouaddah* dans l'apprentissage de ROS qui s'est fait en ligne dans l'optique de m'inscrire pour un stage au sein du CDTA. Je suis très reconnaissante à *M.Ouaddah* et à son équipe de recherche NCRM au CDTA grâce à qui j'ai pu acquérir une certaine maîtrise de l'outil robotique.

C'est à lors que j'ai pu commencer à travailler avec *Mme.Chouireb*, mon encadreur qui m'a ouvert les yeux au monde du SLAM, et avec laquelle j'ai découvert les algorithmes ICP et NDT qui sont la base de ce travail de mémoire. Ce fut aussi grâce à mon co-encadreur *M.Ahmine* que j'ai pu comprendre les fondements de ces algorithmes. Je les remercie de tout cœur pour leur patience, leur temps, leur énergie et leur engagement à toute épreuve.

Le plan d'avancement a subit d'innombrables restructurations, nous avons avancé au gré des besoins, des résultats qu'on a obtenu, et des moyens à notre disposition. Pour résulter sur ce mémoire qui n'aurait pas vu le jour sans *Mme.Chouireb* qui est bien plus qu'une promitrice, elle a été le moteur d'avancement principal et la source des solutions aux difficultés qu'on a trouvé en chemin.

Ce fut grâce aux recommandations, et aux facilitations, de *M.Belkheiri* qui était chef d'option durant ma première année de travail sur ce mémoire.

Grâce aux conseils de *M.Bessalah*, doctorant à l'*Université de Frères Mensouri* pour l'inclusion de paramètres utiles dans le code MATLAB.

Et grâce à tout le corps enseignant du département d'électronique, d'électrotechnique de l'*Université de Amar Telidji* sans lesquels je n'aurais pas pu acquérir les connaissances en ma possession sur le domaine de l'automatique, de l'informatique industrielle et de la robotique.

Sans oublier mes enseignants en cycle préparatoire à l'*EPSTA* qui m'ont donné les bases de la programmation en C++ et de l'algorithme en général ce qui m'a été très utile dans ce projet.

Merci, à toutes les personnes qui ont apportées leur soutien, qui ont prêté leur attention, et qui se sont investi pour l'aboutissement de ce mémoire : Mes chers parents, ma petite sœur, mes enseignants, l'équipe de recherche au CDTA et celle du laboratoire de Télécommunications, signaux et systèmes.

Mais surtout à *Mme.Chouireb* pour la personne qu'elle est, pour ses efforts, sa compréhension, sa bonne humeur, et pour tout ce qu'elle a investi en ce mémoire. Et à ma petite famille sans laquelle je n'aurais jamais eu les moyens de mener à bien ce projet.

# **Abstract**

# Résumé

# Sommaire

<b>I Généralités sur les systèmes SLAM</b>	<b>15</b>
<b>1 Introduction</b>	<b>18</b>
<b>2 Définition et Origines</b>	<b>19</b>
<b>3 Formulation et Énonciation</b>	<b>20</b>
3.1 Localisation . . . . .	20
3.2 Cartographie . . . . .	21
3.2.1 Représentation de la carte . . . . .	22
3.2.2 Approche Directe . . . . .	22
3.2.3 Featured Based . . . . .	22
3.2.4 Grid Based . . . . .	23
3.2.5 L'approche topologique . . . . .	24
3.2.6 L'approche hybride . . . . .	25
3.2.7 Comparaison . . . . .	25
3.3 Localisation et Cartographie simultanées . . . . .	25
3.4 Perception et Capteurs . . . . .	27
3.4.1 Les systèmes de perception . . . . .	27
3.4.2 Les capteurs proprioceptifs . . . . .	27
3.4.3 Les capteurs extéroceptifs . . . . .	28
3.4.4 Le télémètre . . . . .	28
<b>4 Le SLAM</b>	<b>32</b>
4.1 Exemples de SLAM . . . . .	32
4.2 Workflow du SLAM . . . . .	32
4.2.1 Front End . . . . .	33
4.2.2 Back End . . . . .	39
4.3 Algorithmes Back End . . . . .	41
4.3.1 Filtre de Bayes . . . . .	41
4.3.2 Filtre de Kalman . . . . .	42
4.3.3 Filtre particulière . . . . .	43
4.3.4 Maximum de Vraisemblance (MLE) . . . . .	45
4.3.5 Comparaison . . . . .	47
4.4 Défis courants avec SLAM . . . . .	49
4.4.1 Exactitude des résultats . . . . .	49
4.4.2 Problème de positionnement . . . . .	49
4.4.3 Coût de calcul élevé . . . . .	50
4.5 Conclusion . . . . .	50

<b>II Scan Matching</b>	<b>51</b>
<b>1 Généralités sur le Scan Matching</b>	<b>53</b>
1.1 Point Cloud . . . . .	53
1.2 Scan Matching . . . . .	53
1.3 Le Scan Matching étape par étape . . . . .	54
<b>2 ICP</b>	<b>56</b>
2.1 Taxonomie de l'ICP . . . . .	57
2.1.1 Initialisation des points . . . . .	57
2.1.2 Matching des nuages de points . . . . .	58
2.1.3 Pondération des paires . . . . .	59
2.1.4 Réjection des paires . . . . .	59
2.1.5 Optimisation des erreurs . . . . .	62
2.2 Variantes . . . . .	64
2.3 Vanilla ICP . . . . .	64
2.3.1 Generate example data . . . . .	64
2.3.2 Correspondences computation . . . . .	64
2.3.3 ICP based on SVD . . . . .	64
2.4 IRLS-ICP . . . . .	66
2.5 ICP Point à Plan . . . . .	66
2.6 GICP . . . . .	66
2.7 Comparaison . . . . .	67
<b>3 NDT</b>	<b>68</b>
3.1 Représentation par des densités de probabilités . . . . .	68
3.1.1 Collecte des points 2D . . . . .	68
3.1.2 Calcul de la moyenne . . . . .	68
3.1.3 Calcul de la covariance . . . . .	69
3.2 L'alignement avec NDT . . . . .	69
3.3 Le processus d'optimisation avec la méthode de Newton . . . . .	70
<b>III Simulations en milieu Intérieur</b>	<b>74</b>
<b>1 SLAM 2D</b>	<b>77</b>
1.1 ICP Vanilla . . . . .	77
1.2 IRLS-ICP . . . . .	77
1.3 IRLS-ICP et Odométrie . . . . .	78
1.4 Pose Graph . . . . .	78
1.5 NDT . . . . .	79
1.6 EKF . . . . .	79
1.7 Comparaison . . . . .	79
1.7.1 Comparaison . . . . .	79
1.7.2 Occupancy Grid Map . . . . .	80
1.7.3 Comparaison . . . . .	80
<b>2 Scan Matching en 3D</b>	<b>82</b>
2.1 ICP en 3D . . . . .	82
2.2 ICP Vs. NDT . . . . .	83
2.2.1 Scan Matching de 2 nuages de points . . . . .	83
2.2.2 Comparaison . . . . .	83
2.2.3 Scan Matching pour cartographier un environnement . . . . .	83
2.2.4 Comparaison . . . . .	84

<b>IV</b>	<b>Simulations en milieu Extérieur</b>	<b>87</b>
<b>V</b>	<b>Conclusion Générale</b>	<b>89</b>
<b>VI</b>	<b>Annexes</b>	<b>91</b>
<b>A</b>	<b>Annexe A : Le filtre de Kalman</b>	<b>92</b>
A.1	Le filtre de Kalman-Shcmidt . . . . .	92
A.1.1	Équations de prédiction et de mise à jour . . . . .	93
A.1.2	Équation de mise à jour des mesures . . . . .	93
A.2	Le Filtre de Kalman étendu : EKF . . . . .	93
A.2.1	Équation de prédiction et de mise à jour . . . . .	93
A.2.2	Équations de mise à jour des mesures . . . . .	94
<b>B</b>	<b>Annexe B : Filtre Particulaire</b>	<b>95</b>
<b>C</b>	<b>Annexe C : Estimateurs robustes</b>	<b>96</b>
C.1	Définitions . . . . .	96
C.1.1	Variance . . . . .	96
C.1.2	Point de rupture . . . . .	96
C.1.3	Efficacité . . . . .	96
C.1.4	Robustesse . . . . .	97
C.2	Exemples . . . . .	97
C.2.1	Calculs de variance, efficacité variable . . . . .	97
C.2.2	M-estimateurs . . . . .	98
<b>D</b>	<b>Annexe D : La Décomposition SVD</b>	<b>100</b>
<b>E</b>	<b>Annexe E : Minimisation Point à Plan</b>	<b>101</b>

# Liste de Figures

1	Cycle See-Think-Act . . . . .	12
3.1	L'idée de base du SLAM . . . . .	20
3.2	La localisation . . . . .	21
3.3	La cartographie . . . . .	21
3.4	Carte en nuage de points . . . . .	22
3.5	Carte featured-based . . . . .	23
3.6	Grille d'occupation binaire . . . . .	24
3.7	Carte topologique . . . . .	25
3.8	Représentation du problème SLAM . . . . .	26
3.9	Chaîne fonctionnelle . . . . .	27
3.10	LRF : (a) Télémètre LASER (b) HDL-32E LiDAR. . . . .	28
3.11	Chaîne fonctionnelle . . . . .	29
3.12	Une même perspective photographiée via Caméra et acquise via LRF . . . . .	29
3.13	Principe du LiDAR atmosphérique . . . . .	30
4.1	Exemple de Slam Overall . . . . .	33
4.2	Blocs de construction de base des algorithmes SLAM . . . . .	33
4.3	Vision SLAM : (a) Structure du mouvement; (b) Enregistrement de nuages de points pour RGB-D SLAM . . . . .	34
4.4	LiDAR SLAM : (a) SLAM avec LiDAR 2D ; (b) SLAM avec 3D LiDAR . . . . .	35
4.5	Exemple de Fusion . . . . .	36
4.6	Robots Aspirateurs : (a) le Dyson utilise la fusion LiDAR et Vision ; (b) l'Ecovas qui utilise la fusion de deux caméras . . . . .	37
4.7	Modèle d'ADAS . . . . .	37
4.8	Exemple de Fusion . . . . .	38
4.9	Application de la robotique quantique à la navigation des voitures autonomes . . . . .	38
4.10	<i>Back End</i> du SLAM basé sur les raphes (a) Pose Graph (b) Factor Graph . . . . .	39
4.11	Cycle du filtre de Kalman . . . . .	42
4.12	Cycle du filtre Particulaire . . . . .	44
4.13	Principe général de l'IML . . . . .	45
4.14	Exemple Graph SLAM . . . . .	47
4.15	Minimisation de l'erreur de mesure . . . . .	49
1.1	Description d'un Nuage de Point . . . . .	53
1.2	Description d'un Nuage de Point . . . . .	53
1.3	Schamatisation du Scan Matching (MATLAB) . . . . .	55
2.1	L'organigramme général d'ICP . . . . .	56
2.2	Méthodes de Matching . . . . .	58
2.3	Exemple de BSP Tree 2D . . . . .	59
2.4	tableau . . . . .	60
2.5	icp rejection . . . . .	61

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. . . . .	62
2.7	Minimisation point à point . . . . .	63
2.8	Minimisation point à plan . . . . .	63
2.9	bla . . . . .	64
2.10	Première itération : (a) Make data centered; (b) Compute correspondences . . . . .	65
2.11	Première itération . . . . .	65
2.12	Itérations multiples . . . . .	66
2.13	bla . . . . .	66
2.14	bla . . . . .	67
3.1	L'organigramme général de la NDT . . . . .	73
1.1	scanMatchingExample . . . . .	77
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 . . . . .	78
1.3	pose graph full slam . . . . .	79
1.4	2 Nuages de Point en 2D . . . . .	79
1.5	Scan Matching 2D ICP . . . . .	80
1.6	Scan Matching 2D NDT . . . . .	80
1.7	occupancyGridMap NDT . . . . .	81
1.8	grid map full slam . . . . .	81
2.1	L'organigramme général d'ICP . . . . .	82
2.2	Scan Matching : ICP Point à Point . . . . .	83
2.3	Scan Matching : ICP Point à Point . . . . .	83
2.4	Vérité Terrain . . . . .	84
2.5	État initial de Scan Matching . . . . .	84
2.6	Scan Matching à base de l'ICP . . . . .	85
2.7	État initial de Scan Matching . . . . .	85
2.8	Scan Matching à base de l'NDT . . . . .	85
2.9	Scan Matching à base de l'ICP . . . . .	86
A.1	Schéma conceptuel du Filtre de Kalman . . . . .	92
A.2	Principe de l'UT . . . . .	94

# Liste de Tableaux

4.1	Tableau comparatif : Méthodes de SLAM . . . . .	48
2.1	Tableau récapitulatif : Fonctions de coût des estimateurs robustes . . . . .	61
C.1	Efficacité relative des estimateurs $\hat{\sigma}$ et $\hat{\epsilon}$ en fonction de $\epsilon$ . . . . .	98

# Liste des Acronymes

<b>2D</b>	Deux Dimensions
<b>3D</b>	Trois Dimensions
<b>ADAS</b>	Advanced Driver Assistance Systems
<b>AHRS</b>	Attitude & Heading Reference System
<b>AIIV</b>	Artificial Intelligence and Visual Interpretation
<b>AMCL</b>	Adaptative Monte Carlo Localisation
<b>ARE</b>	Adaptative Monte Carlo Asymptotic Relative Efficiency
<b>BoF</b>	Bag of Features
<b>BoVW</b>	Bag of Visual Words
<b>CDTA</b>	Centre de Développement des Technologies Avancées
<b>CPU</b>	Central Processing Unit
<b>DSO</b>	Direct Sparse Odometry
<b>DTAM</b>	Dense Tracking and Mapping
<b>EKF</b>	Extended Kalman Filter
<b>EPSTA</b>	École Préparatoire Sciences et Techniques d'Alger
<b>GM</b>	Generalised Median
<b>GNSS</b>	Géolocalisation et Navigation par un Système de Satellites
<b>GPS</b>	Global Positioning System
<b>ICP</b>	Iterative Closest Point
<b>IDC</b>	Iterative Dual Correspondences
<b>ICRA</b>	International Conference on Robotics and Automation
<b>IEEE</b>	Institute of Electrical and Electronics Engineers
<b>IML</b>	Incremental Maximum Likelihood
<b>IMU</b>	Inertial measurement unit
<b>INS</b>	Information Network System
<b>IR</b>	Infra-Rouge
<b>IROS</b>	International Conference on Intelligent Robots and Systems
<b>Kd tree</b>	$k$ -dimensional tree
<b>KF</b>	Kalman Filter
<b>KLD</b>	KullBack-Leibler Divergence
<b>LASER</b>	Light Amplification by Stimulated Emission of Radiation
<b>LASIK</b>	Laser-Assisted In-Situ Keratomileusis
<b>LiDAR</b>	LIght Detection And Ranging
<b>LRF</b>	Laser Range Finder
<b>LSD-SLAM</b>	Large-Scale Direct Monocular SLAM
<b>MAD</b>	Median of Absolute Difference
<b>MATLAB</b>	MATrix LABoratory
<b>ML-AMCL</b>	MultiLayer Adaptative Monte Carlo Localisation
<b>NCRM</b>	MATrix Navigation et Contrôle des Robots Mobiles
<b>NDT</b>	Normal Distributions Transform
<b>ORB-SLAM</b>	Oriented fast and Rotated Brief SLAM

<b>PhD</b>	Philosophiæ Doctor
<b>PTAM</b>	Parallel Tracking and Mapping
<b>RaDAR</b>	RAdio Detection And Ranging
<b>RANSAC</b>	RANdom SAmple Consensus
<b>RGB-D</b>	Red Green Blue Depth
<b>ROS</b>	Robot Operating System
<b>SfM</b>	Structure for Motion
<b>SIMD</b>	Single Instruction Multiple Data
<b>SoNAR</b>	SOund Navigation And Ranging
<b>LF</b>	Likelihood-field matching
<b>LOVe</b>	Instruction Multiple Data Logiciels d'Observation des usagers Vulnérables
<b>SfM</b>	Structure From Motion
<b>SIR</b>	Sampling Importance Resampling
<b>SLAM</b>	Simultanious Localization and Mapping
<b>SNSS</b>	Global Navigation Satelite System
<b>SVO</b>	Semi-direct Visual Odometry
<b>SVD</b>	Singular Value Decomposition
<b>ToF</b>	Time of Flight
<b>UAV</b>	Unmanned Aerial Vehicle
<b>UKF</b>	Unscented Kalman Filter
<b>vSLAM</b>	Visual SLAM

# Introduction Générale

## État de l'art

Durant ces trois dernières décennies, les applications de la Localisation et Cartographie Simultanés (SLAM) ont pris du terrain à l'instar de réalité augmentée[1] et conduite autonome[2]. Via SLAM, on peut non seulement dresser une estimation de la trajectoire d'un objet en mouvement, mais en plus, ça nous permet de reconstruire la scène environnante en 3D et en temps réel[3]. À ce jour une variété importante de SLAMs utilisant des capteurs 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, si bien que certains robots étaient aptes à faire certaines tâches sans interventions humaines, sous la contrainte de suivre des procédures prédéfinies comme c'est le cas, par exemple, des premiers robots aspirateurs. Avec le développement de la technologie les robots sont devenus de plus en plus intelligents, avec des aptitudes d'accomplissement en toute autonomie de tâches complexes telles 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] qui est aussi décrit dans [6] par le cycle *see-think-act* (voir la figure 1).

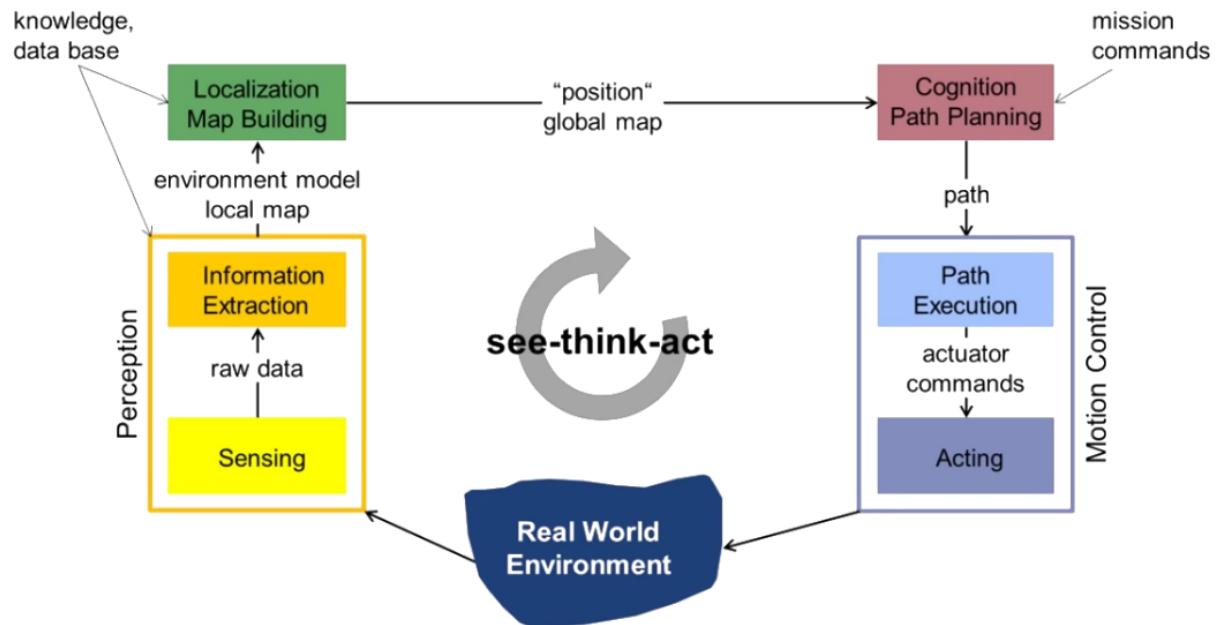


Figure 1: Le cycle Perception, Décision et Action (See-Think-Act)

Et donc, dans ce genre de scénarios, la capacité de Localisation et de Cartographie Simultanées (SLAM) est une exigence de base pour les robots[7], c'est l'étape primaire de traduction des données acquises via les capteurs en informations sémantiques sur l'environnement du Robot, c'est ce qu'on appelle *l'étape de perception*.

## Contexte

Dans l'optique de démocratiser la robotique et l'automatique au sein du milieu étudiant à Laghouat, ce mémoire a pour objectif de fournir une brique à la base de l'édifice en fournissant un modèle de SLAM réutilisable.

Et ce en réalisant un mémoire clair, détaillé, et dans lequel on peu aisément trouver les ressources nécessaires afin de retrouver les résultats atteints, et pour aller plus loin.

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. en exploitant des outils connus et reconnus du milieu de la recherche en robotique :

- **MATLAB** cette méthode a notamment été utilisée afe.
- **ROS** Melodic Linux distribution Ubuntu.
- **LiDAR** Nuages de Points.
- **Kitti** cette méthode a notamment été utilisée afe.

## Les principales contributions du mémoire

Ce mémoire présente différentes contributions dans les domaines de la navigation autonome des robots (plus particulièrement le SLAM basé sur une technologie de perception par LiDAR) dans un milieu qui est radicalement inconnu au dispositif électronique.

On présente quelques techniques fondées sur le principe d'optimisation de l'estimation probabiliste de la description de l'environnement et de la situation du robot (position et orientation) dites *Back End*. On décrit ensuite comment on peut interpréter les données et des mesures acquises via des capteurs extéroceptifs et proprioceptifs, des informations qui pourront être traiter par les algorithmes d'optimisation pour corrélérer les résultats des mesures cumulées par les capteurs embarqués sur le robots à chaque itération alors qu'il se meut dans un environnement nouveau, et on explicite un cas particulier qui mène à la reconstitution de l'environnement externe et de l'itinéraire entrepris via *Scan Matching*. Pour appliquer pratiquement des résultats impliquant des mesures de scans LiDAR, on propose et on caractérise deux algorithmes de superposition de nuages de points et qui sont l'ICP et la NDT, et on décrit quelques applications possibles en milieu intérieur. On décrit également, à partir des mêmes bases de données, comment on peut une utiliser une forme alternative, le filtre de Kalman et le filtre Particulaire, pour des problèmes de localisation pour remplacer la partie d'optimisation via des méthodes de minimisation des moindres carrées telles que Levenberg-Marquardt qui est souvent employé avec les méthodes de *Scan Matching*.

On s'intéresse ensuite de manière un spécifique au SLAM en milieu extérieur, qui est généralement dédié à la navigation des véhicules autonomes. Ceci permet notamment de démontrer que les algorithmes d'ICP et de NDT implantés sur Matlab sont performants dans différents cas de figure (environnement *Indoor* et *Outdoor*), et on compare leurs propriétés et caractéristiques, en mettant plus l'accent sur la méthode de l'ICP qui est l'axe de révolution autour duquel ce mémoire s'articule. Il sera mis à l'épreuve dans différents milieux avec plusieurs algorithmes d'optimisation, et nous utiliserons pour cela des variantes de l'ICP, de la version la plus classique à l'une des plus utilisées et plus modernisée.

On s'intéresse également au métasystème d'exploitation ROS qui est une plateforme de développement logiciel pour robot, grâce à laquelle nous pouvons obtenir des résultats de simulation crédibles et réalistes

sur des robots réels. Plus spécifiquement, nous avons étudié la navigation du robot TurtleBot3 dans le simulateur Gazebo en bénéficiant de la manipulation conjointe de la plateforme ROS et de l'environnement MATLAB, et en utilisant des extensions dites packages pré-codés en C++ et en python, pour la création d'un environnement de simulation, qui exploite les bases de données de référence pour la navigation.

En ce qui concerne l'aspect acquisition de données, on s'intéresse aux nuages de points acquis via télémètres lasers et au LiDAR qui est une version plus contemporaine, c'est un capteur ayant une architecture d'émetteur-récepteur numériques, et qui a besoin d'algorithmes de correction, de calibration et de contrôle spécifique d'où la consécration d'une branche dédiée en navigation, le LiDAR SLAM. Les contraintes posées pour la cartographie et la localisation via cette technologie nécessitent de palier aux lacunes via d'autres sources d'acquisition de mesures, ce qui nous a amené à sélectionner. On propose ici notamment une inclusion des données GNSS (Global Navigation Satellite System) dans les programmes LiDAR SLAM en milieu extérieur, permettant de fournir une précision à moindre coût permettant d'alléger la charge de calcul en limitant le nombre d'itération nécessaires dans la partie *Back End*.

Cette étude comparative permet d'établir les différences des diverses méthodes de SLAM sur plusieurs plans, via des indicateurs de performance dont la complexité et le temps d'exécution pour déterminer s'il est possible de les utiliser en temps réel ; la flexibilité en milieu intérieur et en milieu extérieur ; le besoin en optimisation via des outils software (optimisation) et hardware (capteurs supplémentaires). On prend aussi en considération et on illustre les conséquences néfastes d'une surexploitation des outils de SLAM, comme une densité trop importante des nuages de points, ou l'utilisation de plusieurs algorithmes d'optimisation ou de trop de capteur pour le même cas de figure. On propose et on caractérise plusieurs algorithmes, reposant sur des techniques de traitement des nuages de points, permettant d'obtenir des résultats pertinents.

## Organisation de l'ouvrage

Ce mémoire s'étale sur 4 chapitres qui permettent de décrire l'intégralité des points phares du travail réalisé en expliquant et décrivant les résultats significatifs obtenus.

Le chapitre 1<sup>er</sup> bref résumé de l'intro et de la conclusion.

Le 2<sup>nd</sup> chapitre : bref résumé de l'intro et de la conclusion.

Le 3<sup>ème</sup> chapitre : bref résumé de l'intro et de la conclusion.

Le 4<sup>ème</sup> chapitre : bref résumé de l'intro et de la conclusion.

Des annexes permettant de mieux appréhender certains aspects techniques phares sont détaillé à la fin de l'ouvrage.

# Chapitre I

## Généralités sur les systèmes SLAM

# Table des Matières du Chapitre 1<sup>er</sup>

---

<b>1</b>	<b>Introduction</b>	<b>16</b>
<b>2</b>	<b>Définition et Origines</b>	<b>17</b>
<b>3</b>	<b>Formulation et Énonciation</b>	<b>18</b>
3.1	Localisation . . . . .	18
3.2	Cartographie . . . . .	19
3.2.1	Représentation de la carte . . . . .	20
3.2.2	Approche Directe . . . . .	20
3.2.3	Featured Based . . . . .	20
3.2.4	Grid Based . . . . .	21
3.2.5	L'approche topologique . . . . .	22
3.2.6	L'approche hybride . . . . .	23
3.2.7	Comparaison . . . . .	23
3.3	Localisation et Cartographie simultanées . . . . .	23
3.4	Perception et Capteurs . . . . .	25
3.4.1	Les systèmes de perception . . . . .	25
3.4.2	Les capteurs proprioceptifs . . . . .	25
3.4.3	Les capteurs extéroceptifs . . . . .	26
3.4.4	Le télémètre . . . . .	26
<b>4</b>	<b>Le SLAM</b>	<b>27</b>
4.1	Exemples de SLAM . . . . .	27
4.2	Workflow du SLAM . . . . .	27
4.2.1	Font End . . . . .	28
4.2.2	Back End . . . . .	31
4.3	Algorithmes Back End . . . . .	33
4.3.1	Filtre de Bayes . . . . .	33
4.3.2	Filtre de Kalman . . . . .	34
4.3.3	Filtre particulière . . . . .	35
4.3.4	Maximum de Vraisemblance (MLE) . . . . .	36
4.3.5	Comparaison . . . . .	38

---

## TABLE DES MATIÈRES DU CHAPITRE 1<sup>ER</sup>

---

4.4 Défis courants avec SLAM . . . . .	40
4.4.1 Exactitude des résultats . . . . .	40
4.4.2 Problème de positionnement . . . . .	40
4.4.3 Coût de calcul élevé . . . . .	41
4.5 Conclusion . . . . .	41

---

# 1 Introduction

Ce premier chapitre sera une introduction au SLAM, les notions générales et ses principaux algorithmes seront présentés. Tout le contenu du mémoire est basé sur ce chapitre théorique explicatif.

Il nous permettra dans un premier temps de se familiariser à la question du SLAM en commençant par un brève aperçu historique et en acquérant une vue d'ensemble sur son évolution dans le temps avant de théoriser l'énonciation de la cartographie et de la localisation simultanés qui aurait pu être très simple si et seulement si la simultanéité de ces deux opérations n'était pas de mise. Effectivement, nous définirons chaque partie indépendamment ce qui nous permettra de mieux classifier les types de SLAM selon les cartes utilisées par exemple.

Puis nous nous attarderons sur la formulation probabiliste du SLAM dans son ensemble et dans toute sa complexité, ce qui nous permettra de remonter dans le cycle Perception Décision et Action pré définit dans l'[Introduction](#) et qui est la perception, pour ce faire nous avons listé les capteurs les plus utilisés en mettant l'accent sur le LiDAR qui est l'outil de mesure phare de cet ouvrage.

Dans la quatrième section, nous définirons le SLAM à base des notions déjà expliqués dans ce qui a précédé avec des exemples d'application pour souligner le large éventail de possibilités auquel on a accès avec le SLAM. Et que nous subdiviserons en deux parties principales dans le Workflow et qui sont le *Font End* et le *Back End*.

Le *Font End* dont nous expliquerons superficiellement les deux principales branches qui se sont démarquées dans le temps, le vSLAM et le LiDAR SLAM, c'est à cette dernière que nous consacrerons le deuxième chapitre sur le [Scan Matching](#).

Tandis que le *Back End* qui est un domaine où les avancées technologiques sont fulgurantes depuis que nous avons commencé à travailler sur ce sujet, on va détaillé les approches qu'il regroupe dits Bayésiennes (KF, PF et leurs variantes) ou les approches de minimisation des moindres carrés dont celles axées sur des approches graphiques et dont les méthodes IML qui et LM que nous allons comparer.

Avant de conclure, nous soulignerons les défis actuels du SLAM et que nous rencontrerons nous aussi dans les simulations du chapitre [Simulations en milieu Intérieur](#).

## 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 (exemples : encodeur à roues et capteur d'inertie dit IMU)[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'amers<sup>1</sup> 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 pour son manque de précision, car la plus grande différence entre la cartographie en milieu intérieur et en milieu extérieur est le besoin en précision, en effet une précision en unité de mètres fourni par un GPS est loin d'être suffisante. C'est pour cela que les capteurs externes (LRF, LiDAR, 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'œuf*, 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].

---

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

### 3 Formulation et Énonciation

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, et qui perçoit dans son champs de vision un certain nombre d'amers grâce à un capteur embarqué sur le robot, tel que la figure 3.1 l'illustre.

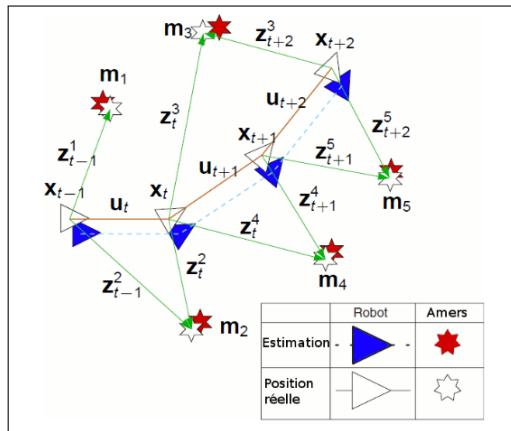


Figure 3.1: L'idée de base du SLAM

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  $k - 1$  commande le robot afin de passer de l'état précédent soit  $x_{k-1}$  à l'état présent dit  $x_k$
- $m_i$  : 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 schématisé dans la figure 3.2 consiste en l'estimation de la position relative du robot dans un environnement donné qui lui est inconnu, en utilisant l'historique de ses observations, l'historique des commandes et la connaissance de l'environnement acquise sur le tas.

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

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

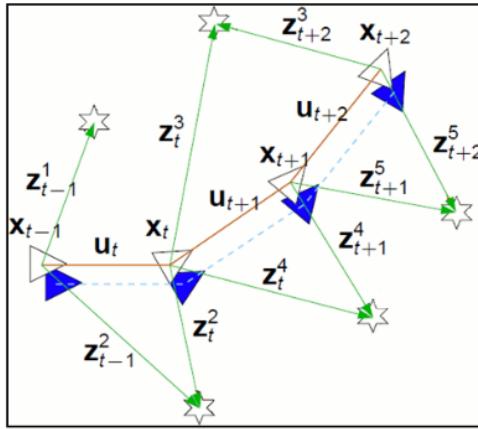


Figure 3.2: La localisation : le système cherche à estimer sa position en utilisant les informations sur l'environnement dont il dispose

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.

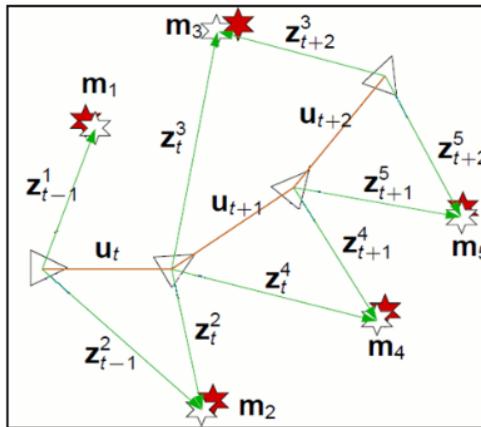


Figure 3.3: La cartographie : le système crée la carte de l'environnement en se basant sur sa position connue et les informations de ses capteurs

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'objets 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 3.4 montre un exemple d'une carte en nuage de points binaire.

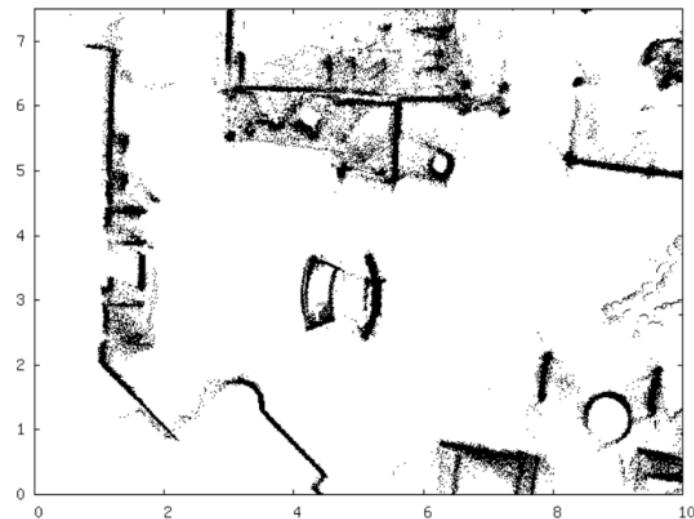


Figure 3.4: Carte en nuage de points

#### 3.2.3 Featured Based

Le principe de cette méthode se résume en l'exploitation d'une liste ou un vecteur cumulant l'intégralité des objets de l'environnement, cette liste contient des informations sur les objets telles 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 3.5.

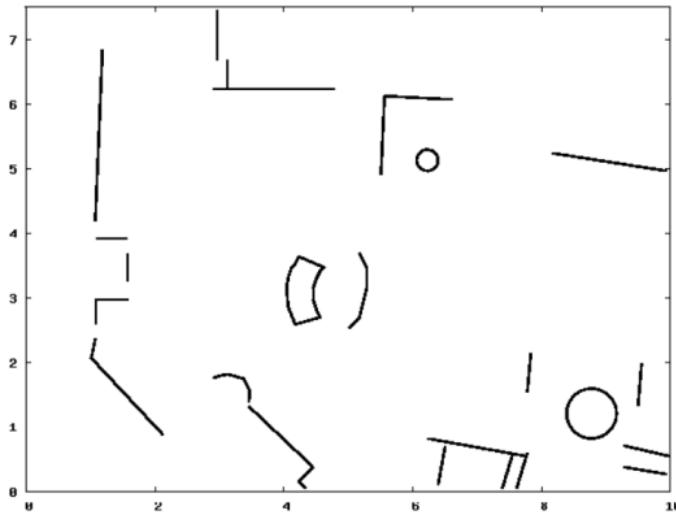


Figure 3.5: Carte basée sur l'extraction de caractéristiques géométriques de l'environnement

En se déplaçant, le robot prend des mesures de son environnement via son système de perception, afin d'acquérir des informations lui permettant de calculer 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 telles que des points, des lignes, des polygones, etc.

Une fois que la carte géométrique est construite, elle peut être utilisée par le robot pour sa localisation dans l'environnement, devenant ainsi  $m$  qu'on avait représenté dans le début du chapitre [Formulation et Énonciation](#). 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 3.6).

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 obstacles. Il doit construire une carte du lieu seulement à partir des informations données par des capteurs

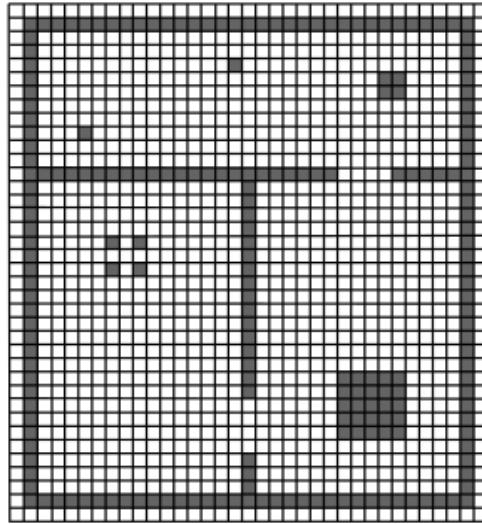


Figure 3.6: Exemple de grille d'occupation binaire. Les cellules blanches correspondent à des zones de l'environnement ne contenant aucun obstacle, les cellules grises correspondent à des zones occupées par des obstacle

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 partiels ( $E = \text{vide}$  et  $O = \text{occupée}$ ) 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'ambiguïté 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 (cf. figure 3.7). 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].

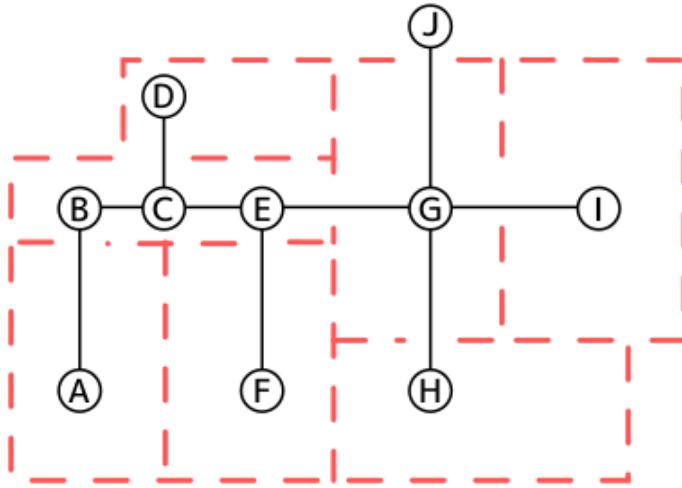


Figure 3.7: Exemple de carte topologique (en noir)

### 3.2.6 L'approche hybride

Les modélisations métriques (telles 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 hybrides basées sur les grilles d'occupation auxquelles 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.

Dans ce mémoire, nous retrouverons dans la partie pratique les résultats des SLAMs appliqués à différents environnement, sous forme de représentations directes de la carte et de modélisation par grille d'occupation.

## 3.3 Localisation et Cartographie simultanées

La structure du SLAM est représentée sur le schéma de la figure 3.8 avec des cercles gris représentant les données connues, tandis que les cercles blancs désignent les quantités à estimer.

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

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

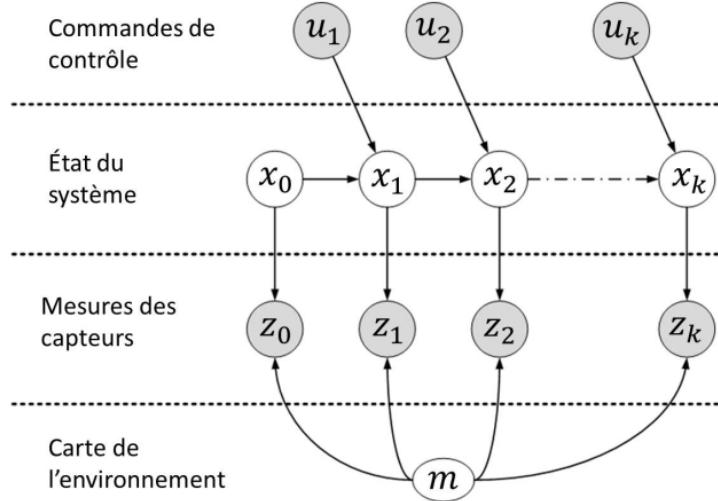


Figure 3.8: Représentation graphique du problème de SLAM

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 (voir la section [Filtre 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  $P(z_k|x_k, m)$  en gardant à l'esprit que 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  $\eta$  est une constante de normalisation dépendant du modèle d'observation et du modèle de transition.

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

## 3.4 Perception et Capteurs

La perception consiste globalement en le fait de recueillir des informations sensorielles dans le but de les interpréter et d'en acquérir une connaissance et une compréhension du milieu d'évolution du robot. Elle est préalable et indispensable aux tâches suivantes qui sont généralement pour un robot mobile autonome : les tâches de localisation et de cartographie, cette dernière est aussi appelée représentation du monde dans la figure 3.9.

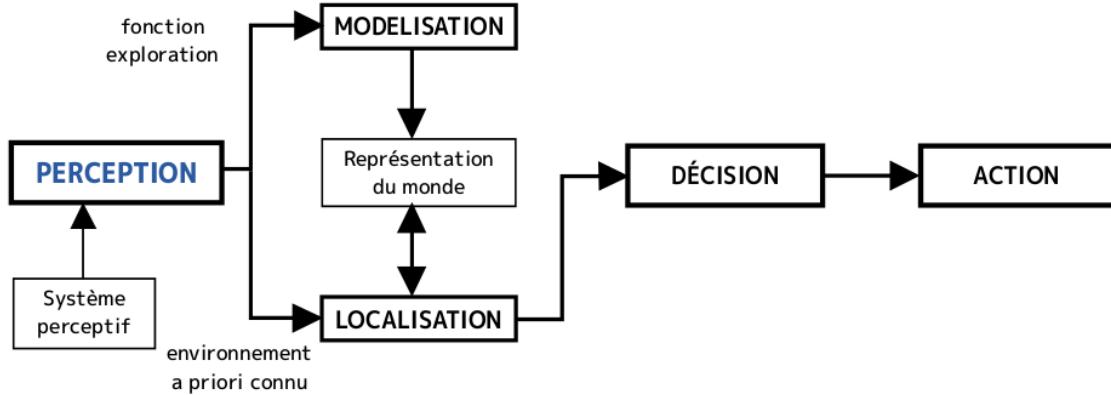


Figure 3.9: Chaîne fonctionnelle d'un système de navigation

### 3.4.1 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 :

- **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.

### 3.4.2 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.

### 3.4.3 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 présents 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** : tels que les capteurs de contact (bumpers, capteurs d'effort), les magnétomètres, et les capteurs de vision.
- **Les capteurs actifs** : tels que les systèmes de cartographie basés sur des balises (localisation dans un repère fixe), les capteurs temps-de-vol (les SoNARs ou ultrasons et les télémètres LASER ou LiDAR), les capteurs IRn et les radars (ondes radio).

Dans ce mémoire, nous serons souvent amenés à utiliser plus d'une source de perception, et donc plus d'un type de capteur. Cela dit le télémètre 2D ou 3D demeure la principale source de données, les autres capteurs seront complémentaires. C'est pour cela que nous le détaillerons davantage dans cette partie.

### 3.4.4 Le télémètre

Le télémètre LASER ou le RADAR LASER est un outil de mesure des distances tout comme le LiDAR, mais ce sont deux technologies différentes. Le télémètre était le plus utilisé il y a quelques années, mais depuis peu le LiDAR est devenu de plus en plus accessible et de plus en plus performant.

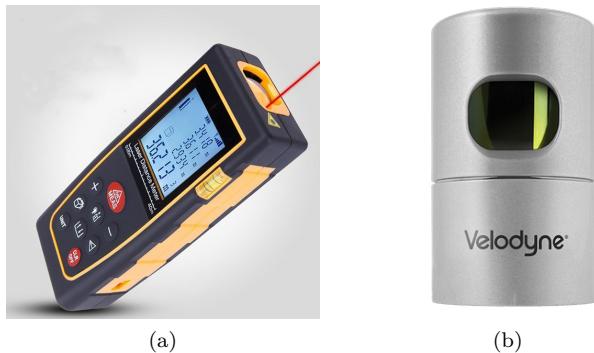


Figure 3.10: LRF : (a) Télémètre LASER (b) HDL-32E LiDAR.

#### Télémètre LASER

Le télémètre laser est un capteur actif, en cela qu'il émet une onde qui « sonde » l'espace, et qui permet de déterminer la distance au premier point d'impact (souvent par mesure du temps de vol, mais d'autres techniques, notamment fondées sur un déphasage de l'onde réfléchie ou sur l'effet Doppler sont possibles). Il fournit, de part la localisation de ce point et d'autre part l'information paradoxale de l'espace libre entre le point d'émission et celui-ci.

Une approche fréquente consiste à le définir en termes de probabilité d'occupation, celle-ci est en général négligeable entre le point d'émission et le point d'impact relevé, la probabilité est alors maximale au niveau du point d'impact, puis prolongée à son niveau maximal ou décroissante selon les besoins. La localisation du point d'impact peut être modélisée par une probabilité de présence continue, par exemple gaussienne, rendant compte de la stochastique du capteur, ou par une série de valeurs discrètes arbitraires, rendant parfois compte de différents besoins des algorithmes en aval dans le processus de traitement.

On présente un exemple via la figure 3.11 où la courbe en traits rouges uniformément discontinus révèle un obstacle ou un mur qui désigne la limite du champs possible de propagation du LASER, ça sera retracé en

blanc dans la grille d'occupation qu'on voit dans l'image (b) de la figure 3.12. Tandis que la courbe rouge en ligne *dash-dot* représente l'espace accessible, on le retrouve en gris sur la grille d'occupation. Et le trait vert évoque l'espace qui se trouve hors de porté du champs de motion, il est rendu inaccessible par les obstacles en blanc, et est symbolisé dans la carte en noir de même que le reste de l'environnement qui est inconnu et dont le télémètre n'a pas acquis de renseignement le concernant.

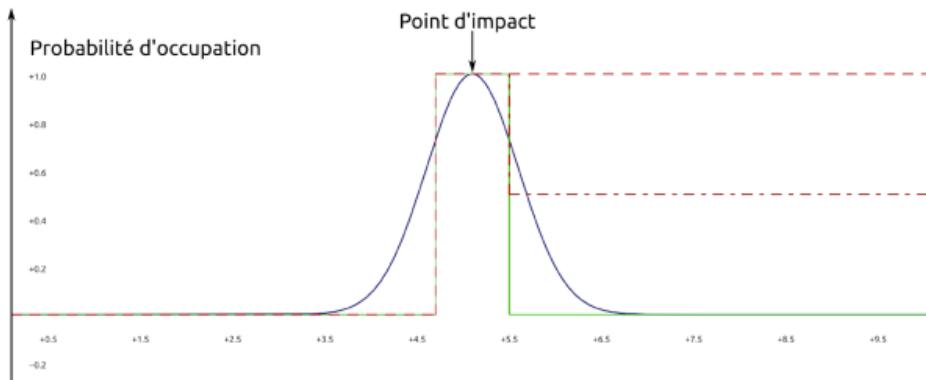


Figure 3.11: Exemple d'interprétation d'un signal acquis via LRF, définissant la probabilité d'occupation en fonction de la distance au point d'émission et au point d'impact. La courbe bleue exprime une probabilité d'occupation continue Gaussienne. Les courbes rouges et verte incarnent un modèle discret couramment utilisé dans les schématisations en grille d'occupation, et considérant différemment la probabilité d'occupation de la partie occulté par le point d'impact.

La généralisation des modèles discrets dans le cas de télémètres LASERs à balayage conduit souvent à une représentation dans le plan de rotation, dans laquelle la probabilité d'occupation de l'espace est échantillonnée selon une grille modélisée dans la figure 3.12, dont la scène est tirée des acquisitions du projet LOVE de protection des vulnérables sur la route[15].

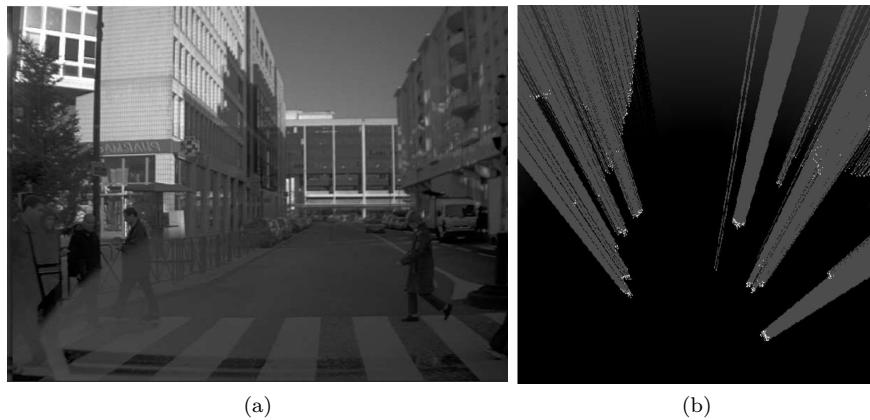


Figure 3.12: Même perspective : (a) Vue de la Caméra (b) Carte en grille d'occupation générée à partir des mesures d'un télémètre laser.

On constate aisément que ce modèle autorise une détection immédiate des possibles obstacles statiques ou dynamiques, par un simple critère de proximité. Les informations extraites du capteur sont cependant minimales dans ce cas (pas de segmentation, de classification, de dynamique par exemple).

Il n'est par ailleurs pas indispensable d'utiliser une vision probabiliste et basée sur l'occupation pour exploiter les données d'un télémètre laser. L'apparition de capteurs très denses et en trois dimensions (de type LiDAR) a rendu possible une exploitation plus directe des mesures de points d'impact, pas nécessairement basée sur le modèle précédent[16].

### LiDAR

Le principe de fonctionnement du LiDAR est voisin de celui du RADAR, il s'applique à la fois à l'instrument et à la méthode de télédétection correspondante. Il s'agit d'une technique optique active de mesure à distance par opposition aux techniques passives de télédétection d'une source de rayonnement naturel. Grâce à l'exploitation des propriétés de cohérence spatiale et temporelle spécifiques des sources LASERS, le LiDAR se différencie aussi des autres instruments de mesure à distance utilisant une source de lumière conventionnelle.

Un LiDAR comporte toujours un émetteur et un récepteur (cf. figure 3.13). La source est un LASER généralement impulsif émettant dans un domaine de transparence de l'atmosphère soit typiquement entre  $0.3\mu m$  et  $10\mu m$ .

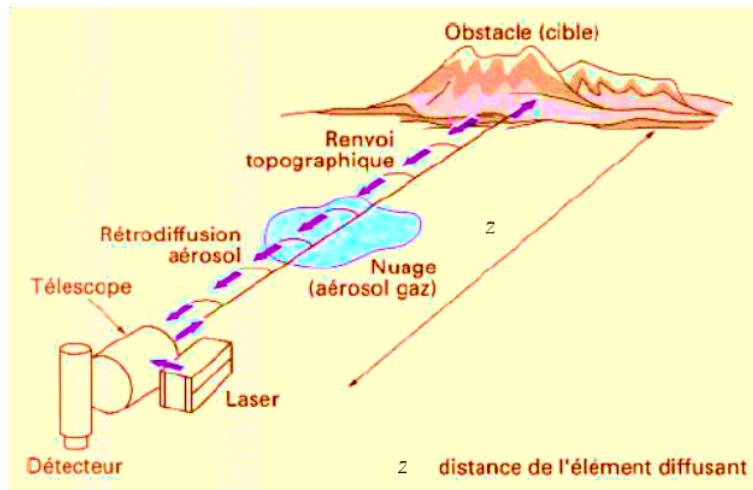


Figure 3.13: Schéma représentatif d'un exemple d'application du LiDAR atmosphérique

La propagation de l'impulsion LASER dans l'atmosphère est suivie de la réception d'une fraction du rayonnement réémis via diverses techniques classifiées selon le type de cible[17][18] :

- **LiDAR topographique** : dit aussi « à cible dure », la réflexion se produit une fois que le rayonnement rencontre un obstacle dense, exemple : le LiDAR télémètre LASER à balayage, c'est aussi le type de LiDAR le plus simple et le plus répandu, il balaye son environnement avec un faisceau LASER et mesure pour chaque point balayé sa distance au LiDAR, permettant la reconstruction d'un modèle tridimensionnel de la scène ; et le LiDAR à détection cohérente pour la mesure de vitesse.
- **LiDAR atmosphérique** : dit aussi « à cible diffuse », dans ce cas la rétrodiffusion se produit sur les constituants de l'atmosphère et milieux autres (océan, forêt) : molécules, aérosols, poussières. Ils sont utilisés par exemple pour détecter la vitesse du vent ou l'empreinte digitale des échanges énergétiques (diffusion Raman) qui se produisent à proximité.

La technologie LiDAR est de plus en plus efficiente. En effet, il existe à présent des capteurs très avancés comme le HDL-32E développé par Velodyne, une industrie américaine, qui concurrence d'autres leaders du marché dont Riegl (Autriche), Ibeo (Allemagne), Hokuyo (Japan) et Hi-Cloud (Chine)[19], le HDL-32E est le LiDAR utilisé pour générer la base de données Kitti exploitée dans le chapitre [Simulations en milieu Extérieur](#) et qui est représenté dans l'image (b) de la figure 3.10b. C'est un capteur LiDAR 3D en temps réel de haute

résolution ayant un large champs de vision dédié aux défis actuels des applications industrielles, le contrôle des véhicules autonomes inclus, ainsi que la cartographie mobile terrestre et aérienne. Tandis que d'autres constructeurs misent sur des capteurs à embarquer sur de petits robots et dispositifs électroniques tels que Apple, Samsung.

#### Comparaison

Il est très courant de confondre ces deux technologies d'autant plus que souvent les dénominations sont confondues, et l'acronyme LRF peut être utilisé pour LiDAR Range Finder autant que pour LASER Range Finder. Et on parle la plupart du temps de LASER scan ou de scans LASERs pour parler de balayage LiDAR. Voici d'autres différences qui permettent de les distinguer :

1. le LASER est un appareil qui recadre un faisceau de lumière intelligible monochromatique tandis qu'un LiDAR utilise des impulsions pénétrantes de lumière laser pour faciliter le processus de mesure de l'arrangement et de la structure de l'atmosphère. C'est une énorme différence entre les deux.
2. Ce sont tous les deux des acronymes, dont les significations sont très similaires. Ils traitent tous deux de l'amplification de la lumière par émission stimulée de rayonnement et détection et télémétrie de la lumière. Dans ce cas, le LIDAR utilise souvent des LASERs pour la simple raison qu'ils peuvent parcourir une longue distance et être réfléchis.
3. Le LiDAR est très similaire à la technique SONAR : il fait rebondir un faisceau de SONDE sur la cible. Dans ce cas, le temps de propagation de l'écho est en fait la distance qui le sépare de la cible. C'est ce qui facilite le processus de dessin d'une carte de la distance entre le capteur et la cible.
4. Le LiDAR est un faisceau dirigé qui est utilisé pour mesurer et déterminer la vitesse. Cela signifie qu'un faisceau LiDAR est un faisceau focalisé tandis que le scanner LASER est utilisé pour éclairer une grande zone.
5. Ils ont des utilisations différentes bien que ces deux technologies peuvent sembler similaires. En fait, un LiDAR est utilisé pour établir les constituants atmosphériques, les objets solides, la distance des objets et bien d'autres choses. L'utilisation de la technologie LiDAR augmente rapidement et nous prévoyons qu'à l'avenir, il y ait plus d'applications. D'autre part, la technologie LASER est utilisée de plusieurs manières. Les premiers et les principaux utilisateurs sont dans les interventions chirurgicales, y compris la chirurgie oculaire LASIK. Ils sont également utilisés pour la coupe, le perçage et la fabrication de plusieurs matériaux.
6. Les deux technologies ne sont pas nées en même temps. La technologie LiDAR a au moins 30 ans. Alors que la technologie LASER a été mise en place dans les années 60. Robert N. Hall a été la première personne à proposer une représentation fidèle de ce que nous appelons aujourd'hui la technologie LASER.

De nos jours le LiDAR s'impose en force, et c'est une technologie plus avancée qui a plus de fonctionnalités, plus d'avantages et dont le prix promet de devenir davantage plus accessible.

# 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 et c'est un sujet de recherche dans le domaine de la robotique depuis les années 1990. 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

Il existe une variété d'approches qui traitent la question du SLAM. Afin de comprendre toute la diversité de ces méthodes, il est important de distinguer entre *front end* et un *Back End*.

La partie frontale est chargée de la lecture des données de mesure brutes acquises par les différents capteurs, et de les convertir instantanément en une représentation sémantique intermédiaire, telle que : des contraintes dans un problème d'optimisation ou une distribution de probabilité sur l'emplacement d'un repère.

La partie *Back End* va prendre cette représentation intermédiaire pour remettre à jour 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[20].

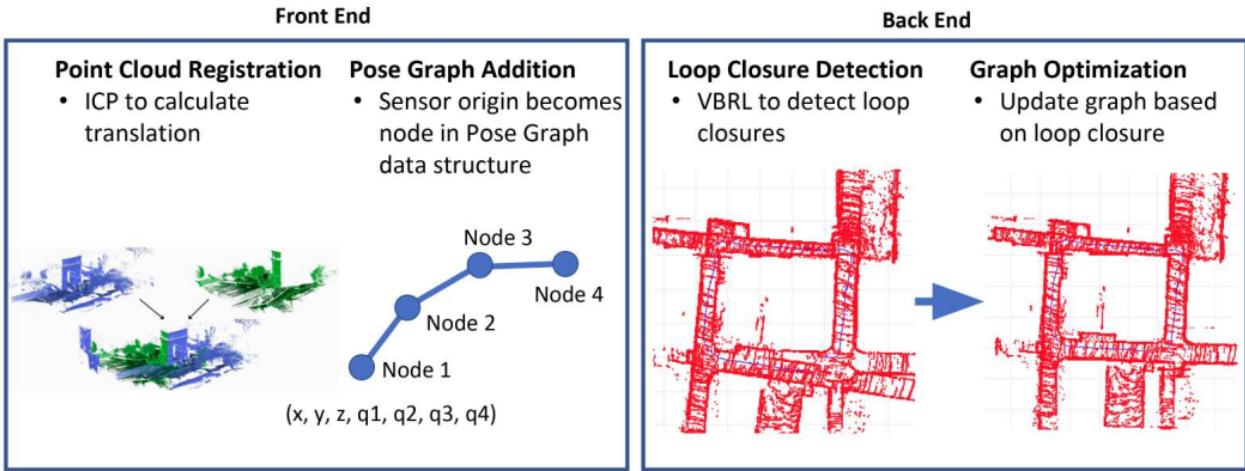


Figure 4.1: Exemple de Slam Overall

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 des boucles de fermetures, dès que la validité de la boucle est vérifiée une fonction d'optimisation du Pose Graph est lancée.

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.

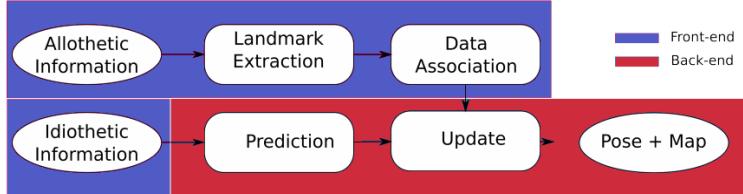


Figure 4.2: Schéma représentatif des principaux blocs de base de construction de l'approche SLAM basée sur les filtres montrant la distinction entre le front-end et le back-end.

### 4.2.1 Front End

Afin de comprendre ce type de technologie appelée traitement *Front End*, nous prendrons l'exemple du Visual SLAM et LiDAR SLAM, deux différentes méthodes d'aborder le SLAM. Mais pas les seules, la fusion de ces deux méthodes est aussi présentée, et il existe également des méthodes prometteuses de SLAM basées sur des technologies quantiques, c'est un cas qui ne sera pas exploité dans ce mémoire, mais qui représente la promesse d'une association du LiDAR et des technologies quantiques.

Ces méthodes d'acquisition de données utiliseront des méthodes de calcul différentes qui sont aussi viables pour le vSLAM, le LiDAR SLAM, auxquels nous avons consacré le chapitre [Scan Matching](#).

### 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.

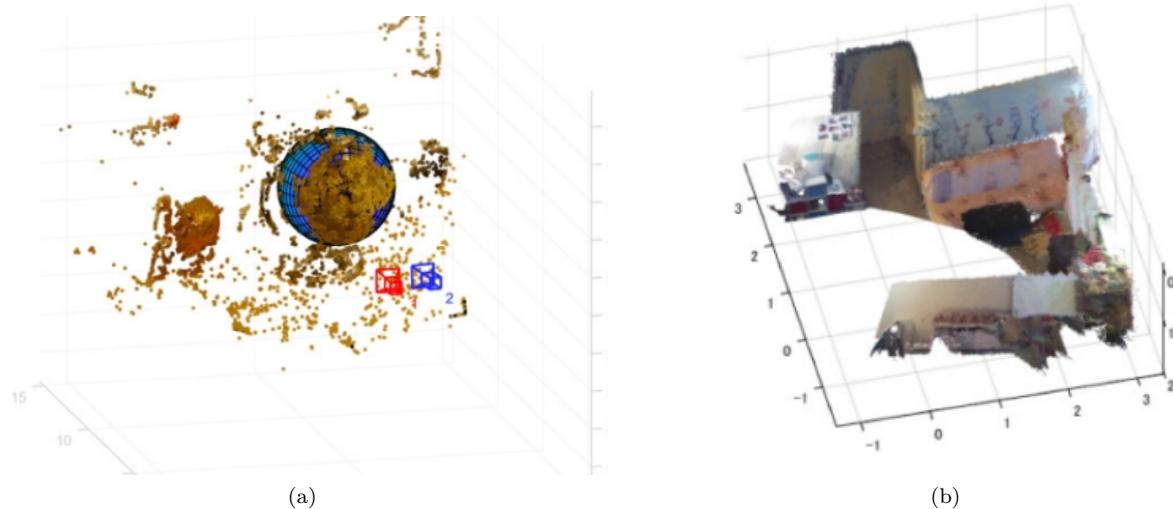


Figure 4.3: Vision SLAM : (a) Structure du mouvement; (b) Enregistrement de nuages de points pour RGB-D SLAM

### 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 en 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.

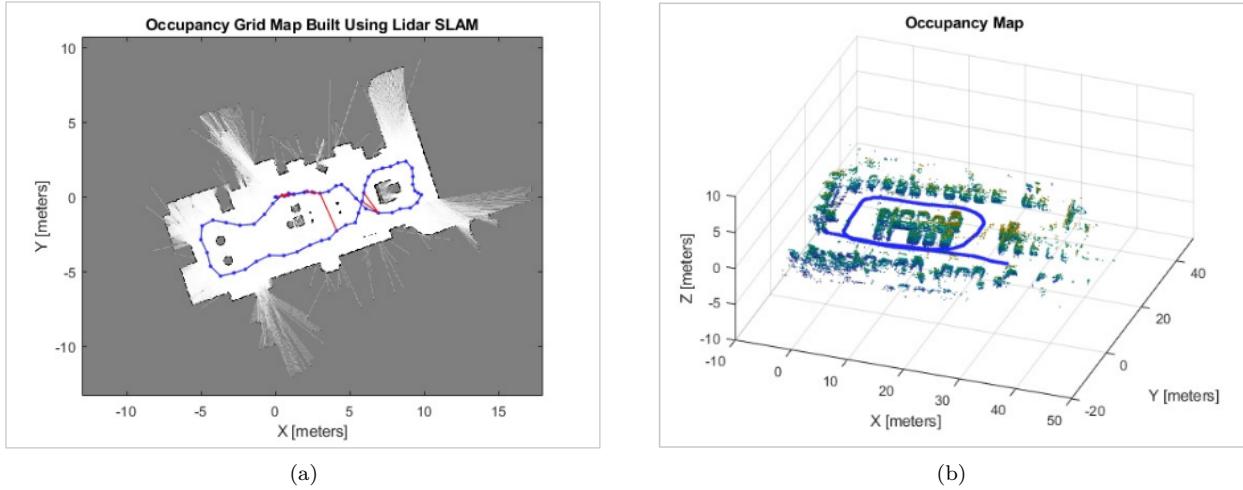


Figure 4.4: LiDAR SLAM : (a) SLAM avec LiDAR 2D ; (b) SLAM avec 3D LiDAR

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, c'est ce que nous pourrons voir plus en détail dans le chapitre [Scan Matching](#).

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 démocratisation en 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 suffisante 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 mesurer des millions de points en peu de temps, puis résolvant une opération trigonométrique 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.

### 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'algorithmes 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.

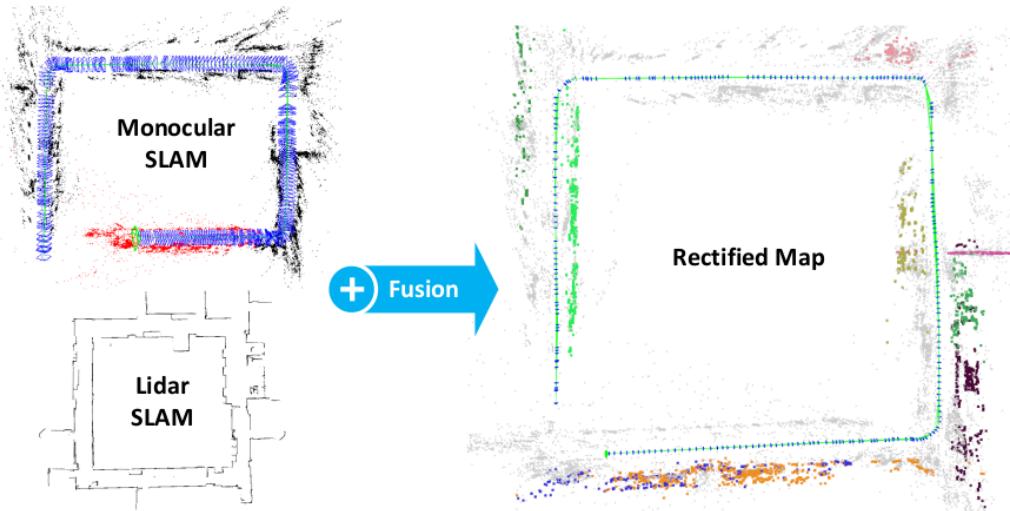


Figure 4.5: Les entrées de la carte de fusion consistent en une carte 3D de basse qualité générée par un SLAM Visuel Monoculaire, et carte prioritaire de haute qualité générée par des méthodes de LiDAR SLAM. La fusion de cartes va corriger la carte 3D en exploitant les plans verticaux qui sont généralement 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 4.5. Tout comme on peut effectuer la fusion avec d'autres cartes du même milieu, dans Google Maps<sup>TM</sup> on trouve souvent des plans extérieurs de bâtisses[21].

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'a 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.

C'est ce qu'on peut déjà constater avec les aspirateurs intelligents, qui utilisent soit deux caméras, l'une



Figure 4.6: Robots Aspirateurs : (a) le Dyson utilise la fusion LiDAR et Vision ; (b) l'Ecovacs qui utilise la fusion de deux caméras

pour cartographier le plafond et l'autre pour reconnaître les obstacles dans sa trajectoire (Dyson 360 Eye<sup>TM</sup>) 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 Ozmo<sup>TM</sup> 960 de la marque Ecovacs) Que ce soit en LiDAR ou en vSLAM, l'avenir réside probablement dans la vision.

### SLAM Quantique

L'industrie 4.0 a vu le jour grâce à la convergence de la production industrielle avec les technologies de l'information et de la communication. La quatrième révolution industrielle est fondée sur l'usine intelligente, qui se caractérise par une interconnexion des machines et des systèmes au sein des sites de production, mais aussi entre eux et à l'extérieur (clients, partenaires, autres sites de productions), ce concept s'apprête à vivre un rebond grâce à l'avènement des technologies quantiques.

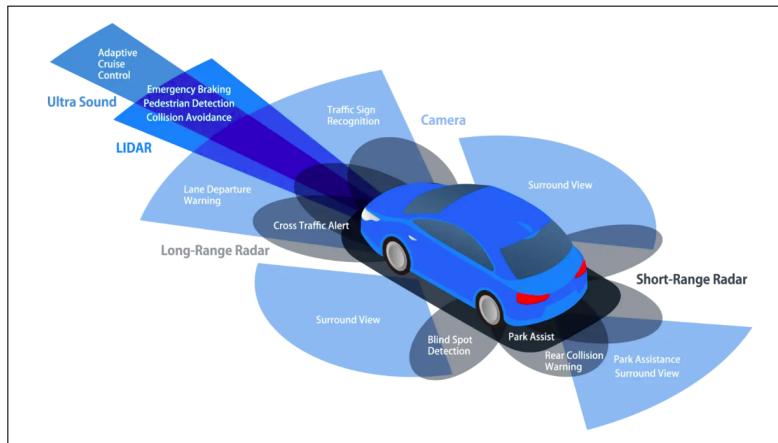


Figure 4.7: Modèle d'ADAS pour protéger des vies via la calibration précise de systèmes avancés d'assistance à la conduite, tels que des LiDARs, des Ultrasons, des Caméras, des télémètres LASERS,etc.

En effet, si le monde a connu la robotique spatiale, les essaims de drones hybrides, et le cycle de sécurité de l'ADAS représenté dans la figure 4.7, c'est bien grâce à fusion de deux sciences fondamentales, ce fut en combinant les systèmes mécaniques et les technologies informatiques classiques.

En fusionnant les technologies qui sont à notre disposition actuellement avec les technologies quantiques, F. Khoshnoud et son équipe de recherche développent ce qu'ils ont appelé la robotique quantique[22]. Qui apporte des solutions multiples pour diverses problématiques actuelles, et l'option qu'ils ont développé est celle d'un réseau de dispositifs mobiles[23] qui communiquent via cryptographie quantique.

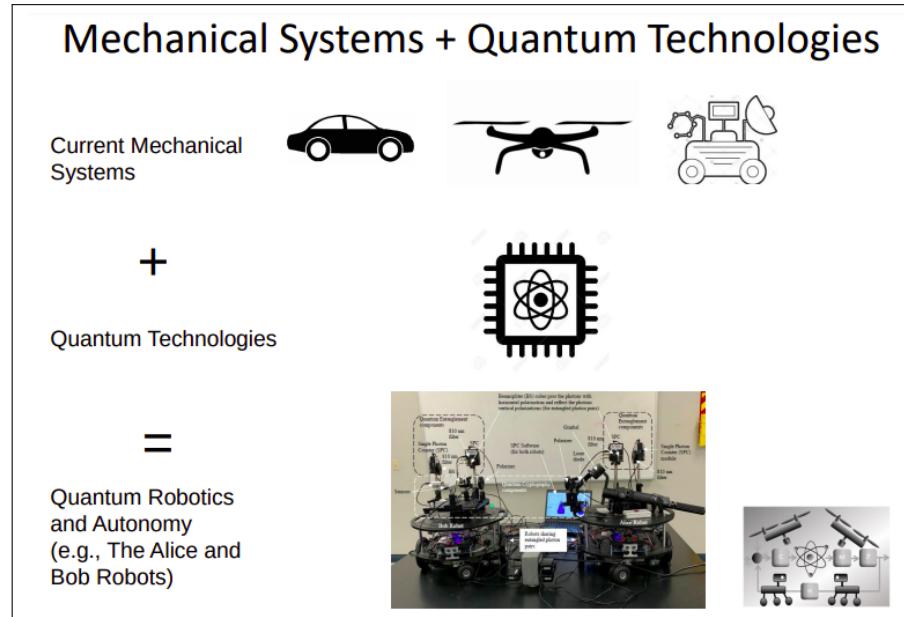


Figure 4.8: Les entrées de la carte de fusion consistent en une cart.

Dans la figure 4.8 nous pouvons observer le modèle Bob et Alice qui est leur première réalisation pratique qui est devenue une référence, de telle sorte à ce qu'ils peuvent configurer plusieurs dispositifs, pas seulement deux, et que ceux-ci peuvent être non seulement des robots mobiles terrestres, mais aussi des drones, ou tout type de véhicules autonomes.

Le projet qu'ils projettent de concrétiser est d'exploiter des systèmes ADAS pour des voitures autonomes afin d'optimiser leur modèle, ce serait un moyen de parfaire la technologie dans le sens sécuritaire. Et ce grâce à :

- La mise en œuvre expérimentale de coopération autonome quantique pour Robots (rayant des photons partagés) pour utiliser et activer la connexion quantique ;
- L'accès à une sécurité garantie pour la coopération autonome par cryptographie quantique ;
- La téléportation quantique pour les communications entre agents pseudo autonomes d'un SMA en téléportant des états quantiques, générant ainsi une intelligence artificielle distribuée.

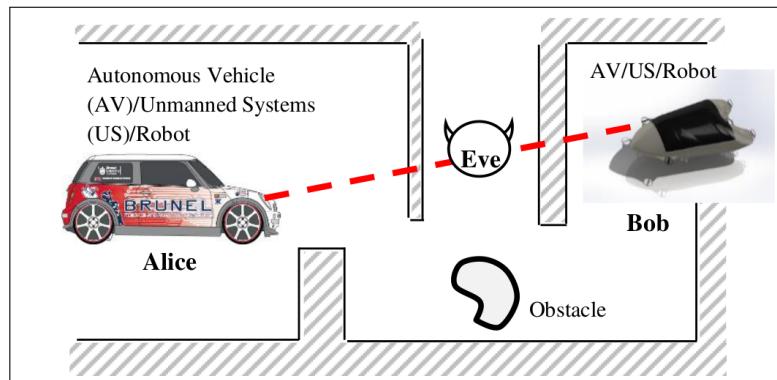


Figure 4.9: Application de la robotique quantique à la navigation des voitures autonomes en l'appliquant au modèle Alice, Bon & Eve ou les véhicules seraient des Alice

Ce concept étant bio-inspiré par le comportement en essaim, par exemple pour les chemins stochastiques typiques de cailloux poussés par des fourmis (rouges) en synergie quantique ainsi que par des fourmis autonomes (noires). Le même phénomène est observable pour des papillons, tel que la trajectoire de vol de deux papillons en somme de vols cours pour qu'ils se retrouvent est de moitié plus long dans le cas où les papillons sont en synergie.

La polarisation des photons en synergie sera convertie en une information numérique classique pour être exploitée par les applications de contrôle numérique (*Back End SLAM*) et d'autonomie. En effet, en utilisant de n'importe quel moyen de communication classique entre robots équipés de processeurs/calculateurs quantiques (lorsque les ordinateurs quantiques seront disponibles à l'avenir) peut donner une autonomie absolue au système ce qui le rendra davantage plus sécurisé.

### 4.2.2 Back End

La partie *beck end* permet de résoudre le problème d'estimation ou d'optimisation de l'état sous-jacent à la partie *front end*. En estimant l'emplacement des objets constituants l'environnement, ou en déterminant l'emplacement de ma plateforme dans le monde. Données grâce auxquelles la reconstruction de la carte de l'environnement est possible en les faisant coïncider.

Nous trouvons généralement deux approches différentes de *Back End*, l'approche bayésienne et l'approche des moindres carrés :

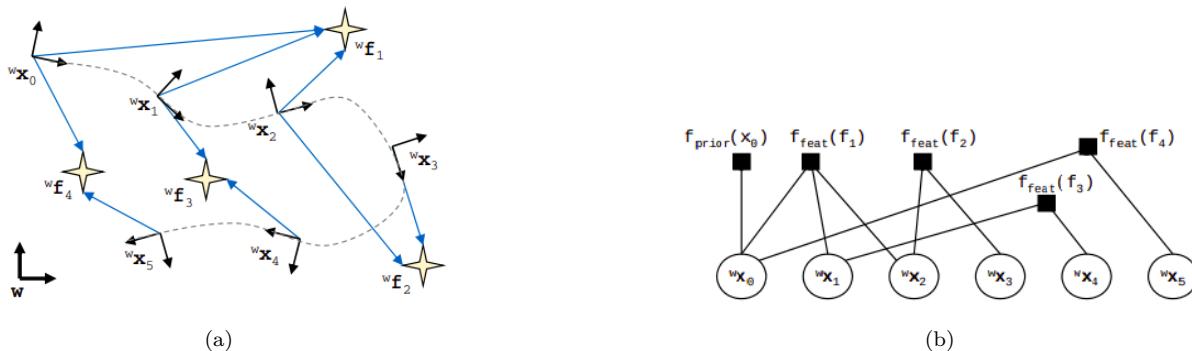


Figure 4.10: *Back End* du SLAM basé sur les raphes (a) Pose Graph (b) Factor Graph

- La première ayant vu le jour dans le domaine de la robotique étant celle des **filtres Bayésiens** voir la section [Filtre de Bayes](#), et à base de laquelle les chercheurs ont développé le Filtre Étendu de Kalman (EKF) voir la section [Filtre de Kalman](#), et le filtre particulier (FP) voir la section [Filtre particulier](#).
- Tandis que la deuxième approche est la plus communément employée de nos jours, c'est les **approches basées sur un graphe** où on représente la problématique sous forme de variables (nœuds) et de relation entre ces variables (arrêtes), et parmi les nombreux types de graphes qui existent de nos jours, les deux ayant gagné le plus de popularité sont le graphe de pose, et le graphe de facteurs :
  - **Pose Graph** : qui met l'accent notamment sur les poses (position et orientation) et néglige le reste de la carte dans le processus de calcul pour générer un résultat complet
  - **Factor Graph** : qui est un graphe à deux parties permettant d'exprimer des variables et les facteurs qui leurs sont associés. Tel que les variables représentent des variables inconnues et aléatoires as le problème d'estimation ou d'optimisation et que les facteurs représentent les contraintes probabilistes de ces variables, des contraintes dérivées de mesures ou d'informations pré-acquises. [24].

Ces méthodes sont aussi dites méthodes des moindres carrés, c'est ce qu'on utilise lorsqu'on base notre SLAM sur du Scan Matching uniquement.

Il y a plusieurs boîtes à outils[25] qui aident à traiter les problèmes d'optimisation dans cette partie *Back End* qui est généralement gourmande en temps de calcul, dont les plus populaires : GTSAM[26] qui est basée sur *Factor Graph* et sur les réseaux Bayésiens ;  $g^2o$ [27] ; et Ceres Solver[28].

Dans ce mémoire, 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 LiDAR. Dans le cas de Pose Graph, les scans, s'ils sont acceptés, sont incorporés dans la structure de données du graphe de pose (voir section [Graph SLAM](#)), et la trajectoire consiste en une série de nœuds 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 Back End

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 [Scan Matching](#).

### 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'emmêler 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 prédiction 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 :

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

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 montrée 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 présuppose 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 particulier 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[29][30].

### 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 4.11) :

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

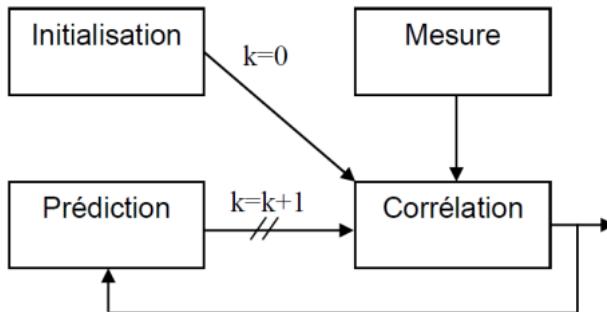


Figure 4.11: Le cycle du filtre de Kalman basé sur les deux étapes récursives : Prédiction et Correction

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 [Annexe A : Le filtre de Kalman](#) 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 UKF<sup>1</sup> (Unscented Kalman Filter).

L'implémentation d'un filtre de Kalman évolue généralement quadratiquement en  $O(n^2)$  (où  $n$  est le nombre d'amas 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'amas 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 figure [4.12](#) schématisé le filtre à particules qui lorsqu'un robot commence à un SLAM dans un environnement inconnu, il va tout d'abord définir la pose initiale du robot à l'origine de son système de coordonnées comme  $(x, y, \theta) = (0, 0, 0)$ , et construire une première carte en utilisant les informations du télémètre LASER en fonction de la pose initiale. Après avoir initialisé le système, il estimera à plusieurs reprises la pose correcte du robot à l'aide d'un filtre à particules et mettra à jour la carte de chaque particule par une technique de fusion de lignes à chaque fois que le robot se déplace au-delà d'une distance.

En général le SLAM basé sur un algorithme en filtre particulaire en guise de *Back End* peut être implémenté via un filtre *Sampling Importance Resampling* (SIR), qui est l'un des algorithmes de filtrage de

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

particules les plus couramment utilisés, en deça d'une technique de mise à jour cartographique. L'ensemble de la procédure se résume dans les étapes suivantes[31] :

- **Étape d'échantillonage (*Sampling*)** : De nouvelles particules sont générées à partir de la particule précédente à l'aide d'un modèle de mouvement.
- **Étape de Pondération d'importance (*Importance Weighting*)** : Chaque nouvelle particule se voit attribuer un poids d'importance pour déterminer la précision de la particule en fonction de la correspondance entre l'observation actuelle et la carte qu'elle a déjà construite.
- **Étape de rééchantillonnage (*Resampling*)**: durant laquelle les particules de faible poids de pondération sont susceptibles d'être remplacées par celles de poids plus important.
- **Étape de mise à jour de la carte (*Map Update*)** : durant laquelle la carte la plus récente observée par le télémètre laser est mise à jour pour chaque particule restante après l'étape de rééchantillonnage en fonction de sa pose individuelle, de sorte que chaque particule dispose de la carte la plus à jour de l'environnement.

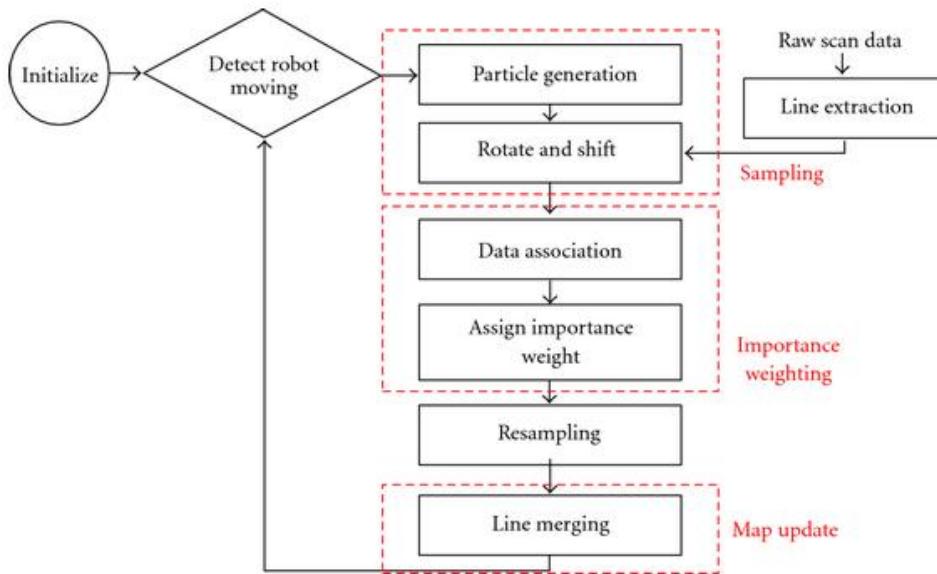


Figure 4.12: *Le cycle du filtre Particulaire basé sur les étapes récursives : Échantillonnage, Pondération, Rééchantillonnage, Mise à jour de la carte*

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, plus détails résident dans l'[Annexe B : Filtre Particulaire](#).

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.

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 [32]), 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 4.13.

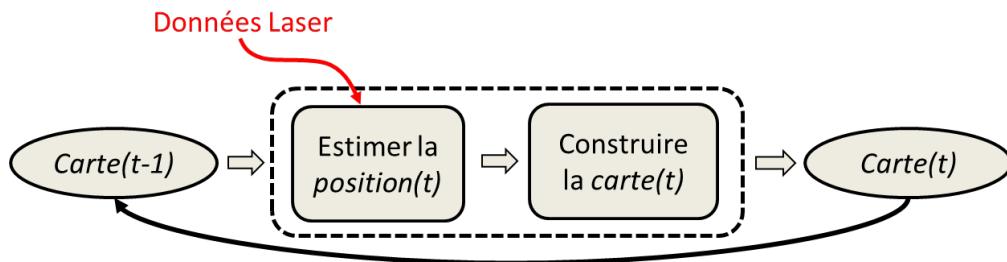


Figure 4.13: Principe général de l'IML

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 :

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

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  $P(x_k | u_k, \hat{x}_{k-1})$

<sup>1</sup> Attention : la notation  $m_k$  utilisée ici est différente de la définition donnée dans [Formulation et Énonciation](#). Dans la section [Formulation et Énonciation](#),  $m_k$  désigne l'amer  $k$  de la carte, alors que dans cette section, elle désigne la carte construite à l'étape  $k$ .

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 [Représentation de la carte](#)). On peut ainsi utiliser un scan-matching direct [33], 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 [34] 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 en 1997 par Lu et Milios[35], c'est un problème d'estimation de maximum de vraisemblance incrémentale, c'est sa variante la plus utilisée[36].

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 4.14, 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 4.14 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.

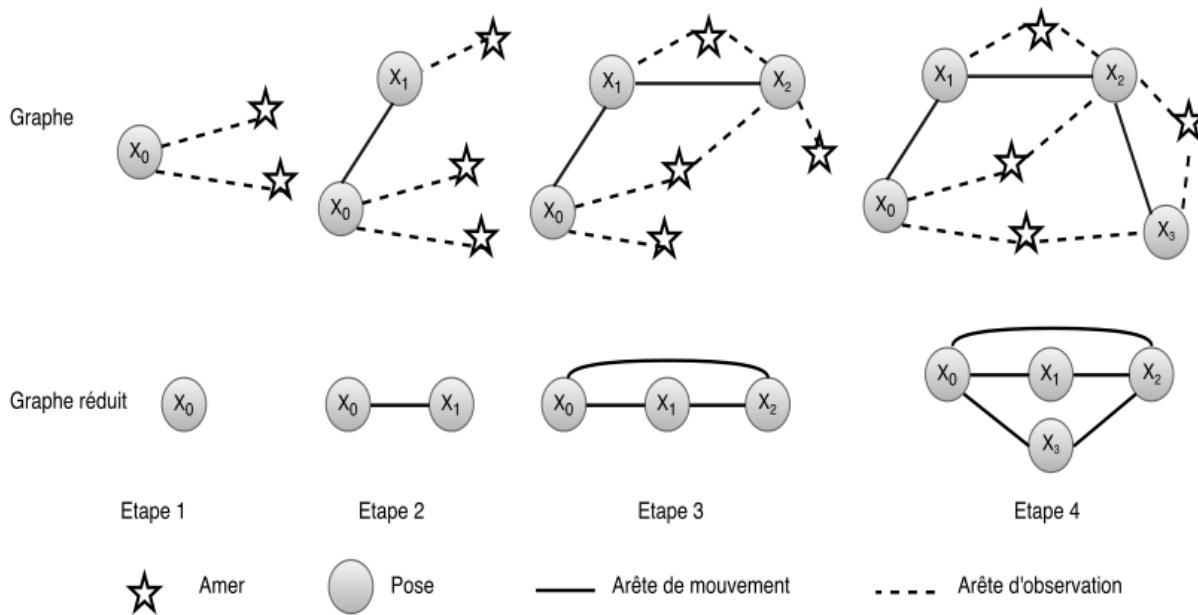


Figure 4.14: Exemple illustratif de la méthode Graph SLAM

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

Le tableau 4.1 présente une liste d'avantages et d'inconvénients de certaines méthodes et algorithmes 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 particulier Rao-Blackwellisé permet de maintenir une estimation a posteriori de l'état du système d'une manière plus rapide que le KF. Le filtrage particulier peut également être utilisé avec des cartes *grid-based* ou *feature-based*. Cette méthode souffre néanmoins de plusieurs problèmes à cause

### 4.3. ALGORITHMES BACK END

---

de sa complexité grandissantes 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.

	<i>Avantages</i>	<i>Inconvénients</i>
Filtre de Kalman	<ul style="list-style-type: none"> <li>- Prise en compte des incertitudes</li> <li>- Convergence prouvée</li> <li>- Fermeture de boucle</li> </ul>	<ul style="list-style-type: none"> <li>- Hypothèse fortes</li> <li>- Complexité</li> <li>- Problèmes d'association des données</li> <li>- Utilisation possible sur un seul type de cartes</li> </ul>
EKF		<ul style="list-style-type: none"> <li>- Complexité</li> <li>- Manque de flexibilité</li> </ul>
UKF		
Filtre Particulaire	<ul style="list-style-type: none"> <li>- Élimination des hypothèses du KF</li> <li>- Fermeture de boucle</li> </ul>	<ul style="list-style-type: none"> <li>- Complexité</li> <li>- Paramétrage difficile</li> </ul>
AMCL		
Fast SLAM		
IML	<ul style="list-style-type: none"> <li>- Concept simple et rapide</li> <li>- Tous les types de cartes</li> </ul>	<ul style="list-style-type: none"> <li>- Divergence par construction</li> <li>- Pas de fermeture de boucle</li> </ul>
Pose Graph	<ul style="list-style-type: none"> <li>- Fermeture de boucle</li> </ul>	

Tableau 4.1: Tableau comparatif entre les méthodes de SLAM les plus courantes

## 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.



Figure 4.15: Exemple de construction d'un graphe de pose et de minimisation des erreurs

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 inertie 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 multicoeurs 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 apparu. 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

# **Chapitre II**

## **Scan Matching**

# Table des Matières du Chapitre 2<sup>nd</sup>

---

<b>1 Généralités sur le Scan Matching</b>	<b>43</b>
1.1 Point Cloud . . . . .	43
1.2 Scan Matching . . . . .	43
1.3 Le Scan Matching étape par étape . . . . .	44
<b>2 ICP</b>	<b>46</b>
2.1 Taxonomie de l'ICP . . . . .	47
2.1.1 Initialisation des points . . . . .	47
2.1.2 Matching des nuages de points . . . . .	48
2.1.3 Pondération des paires . . . . .	49
2.1.4 Réjection des paires . . . . .	49
2.1.5 Optimisation des erreurs . . . . .	52
2.2 Variantes . . . . .	54
2.3 Vanilla ICP . . . . .	54
2.3.1 Generate example data . . . . .	54
2.3.2 Correspondences computation . . . . .	54
2.3.3 ICP based on SVD . . . . .	54
2.4 IRLS-ICP . . . . .	56
2.5 ICP Point à Plan . . . . .	56
2.6 GICP . . . . .	56
2.7 Comparaison . . . . .	57
<b>3 NDT</b>	<b>58</b>
3.1 Représentation par des densités de probabilités . . . . .	58
3.1.1 Collecte des points 2D . . . . .	58
3.1.2 Calcul de la moyenne . . . . .	58
3.1.3 Calcul de la covariance . . . . .	59
3.2 L'alignement avec NDT . . . . .	59
3.3 Le processus d'optimisation avec la méthode de Newton . . . . .	60

---

# 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)[37]. <https://youtu.be/ktRqKxddjJk?t=1363> Densité, selection/segmentation, fréquence <https://youtu.be/ktRqKxddjJk?t=1009>

Descriptors Matrix		N Points			
Descriptor Labels	orientation normal	-0.6363	0.7386	...	0.7061
		-0.4724	0.1594	...	0.2441
		-0.7089	0.0997	...	-0.2981
		-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	orientation normal	-0.6363	0.7386	...	0.7061
		-0.4724	0.1594	...	0.2441
		-0.7089	0.0997	...	-0.2981
		-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 21). 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[37].

## 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[37] :

- **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.

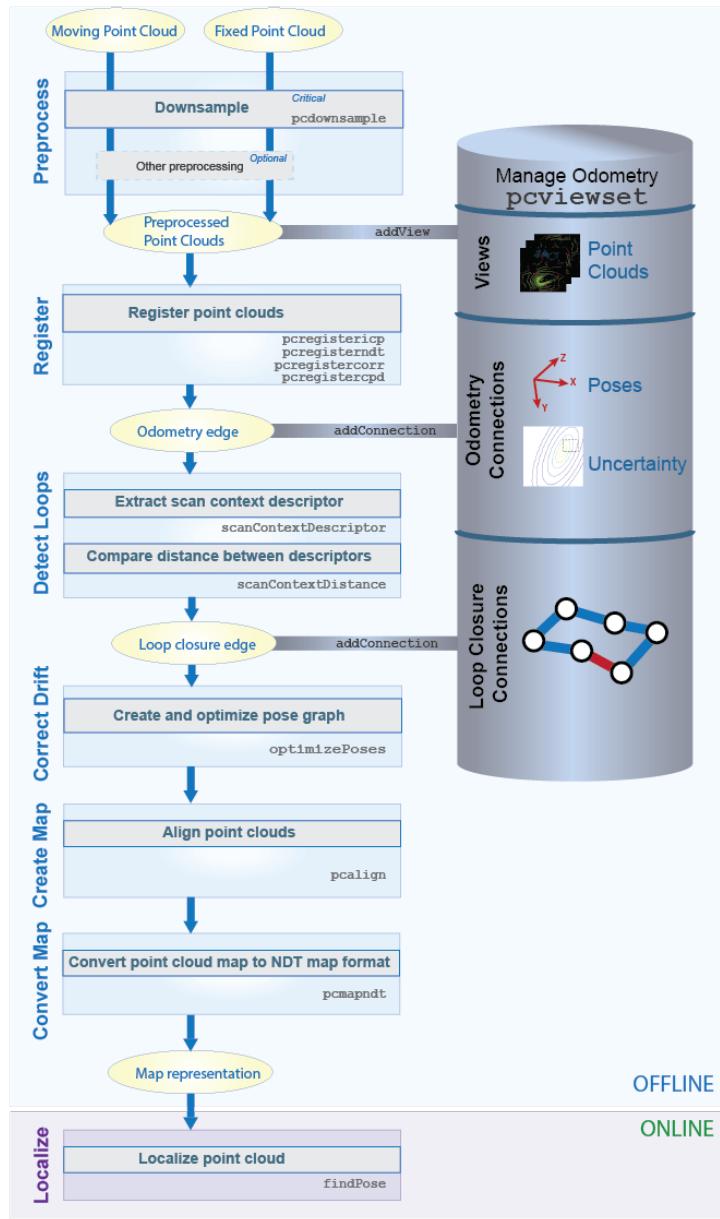


Figure 1.3: Schématisation du fonctionnement du Scan Matching selon le site web de MATLAB

## 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[38].

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).

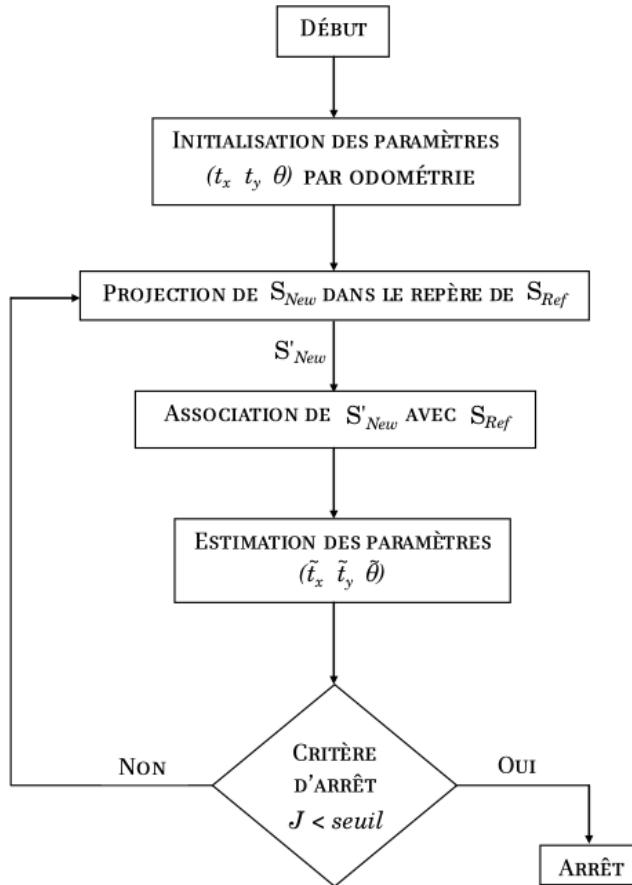


Figure 2.1: L'organigramme général d'ICP

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[38].

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 [Optimisation des erreurs](#)) 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[39][40].

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 [41][42] 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 [Vanilla ICP](#). 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 [43] 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 [43] sous `RandomSampling` et on l'utilisera dans la partie [Scan Matching en 3D](#) 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 [44] 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

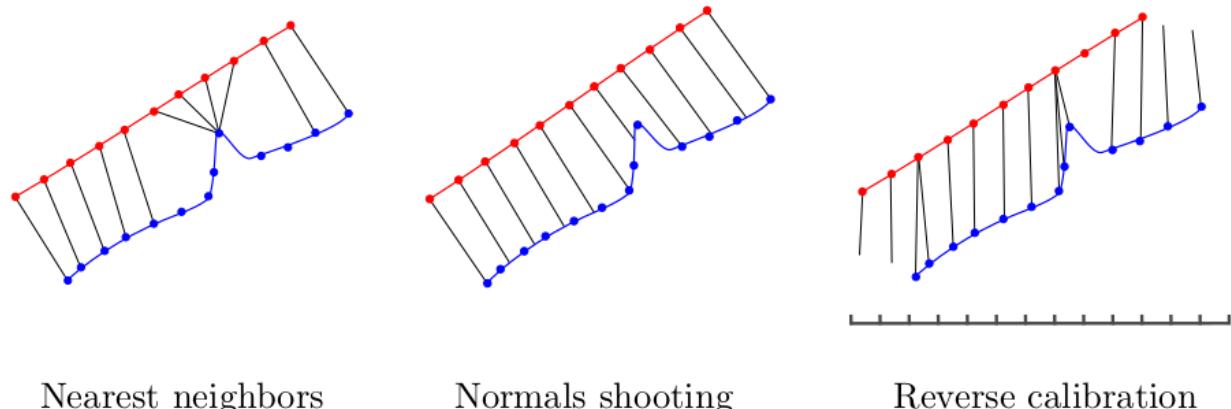


Figure 2.2: ICP matching strategies in two dimensions. Blue points represent the model point clouds to which red data points are matched. In the reverse calibration case, the points are projected onto the discretized, lower dimensional camera view space.

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 [45] 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

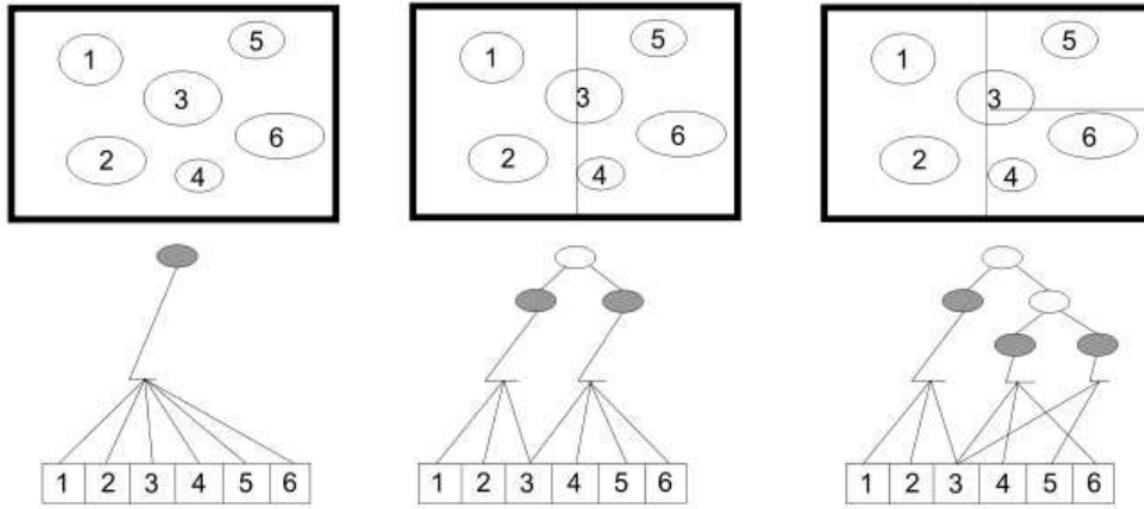


Figure 2.3: Exemple de BSP Tree dans un espace 2D

l'espace, et en re-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 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.

### 2.1.4 Réjection des paires

Tout comme l'étape précédente qui pondère les paires correspondantes, l'étape de rejection 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.

## 2.1. TAXONOMIE DE L'ICP

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

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

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

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>

Afin d'obtenir une estimation satisfaisante, la rejetion 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 minime 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 prenant 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 donné 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][42], 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

Fonctions	Condition	Coût $\rho(e)$	Poids $w(e)$	M
$L_2$				✓
$L_1$				✗
Huber	$\begin{cases} x \text{ si } x < -1 \\ 5x \text{ si } x \geq 3 \end{cases}$			✓
Cauchy				
GM				
Welsch				
Tukey				
Max. Dist				
Trimmed				

Tableau 2.1: Tableau récapitulatifs des principales fonctions de coût des estimateurs robustes par rapport au paramètres de configuration  $k$  et de l'altération de l'échelle dite erreur  $e$

exemple de cette méthode serait [46] 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 [47] qui rejette 10% des paires.

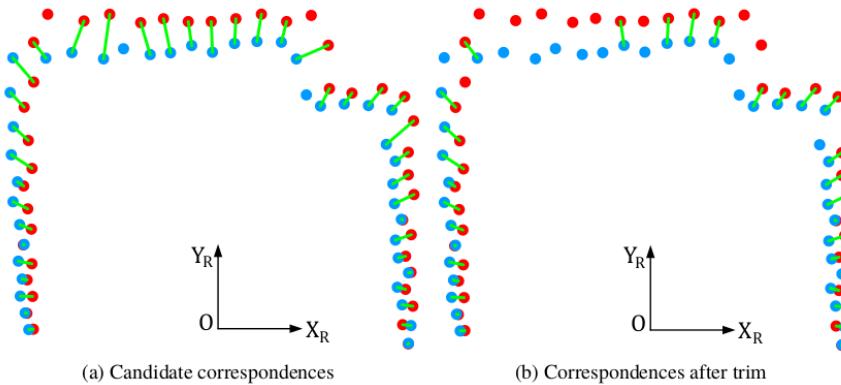


Figure 2.5: Méthode Trim de Rejection

- **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 [48] 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[49], 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.

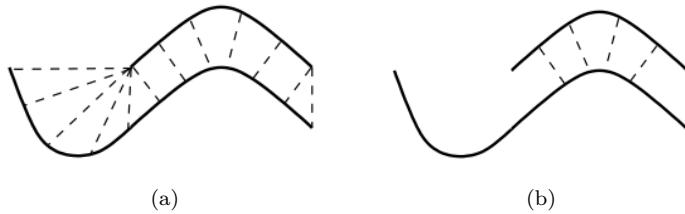


Figure 2.6: (a) When two meshes to be aligned do not overlap completely (as is the case for most real-world data), allowing correspondences involving points on mesh boundaries can introduce a systematic bias into the alignment. (b) Disallowing such pairs eliminates many of these incorrect correspondences.

Dans le document [42] 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 :

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

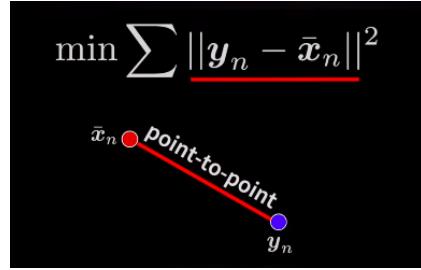


Figure 2.7: Minimisation point à point

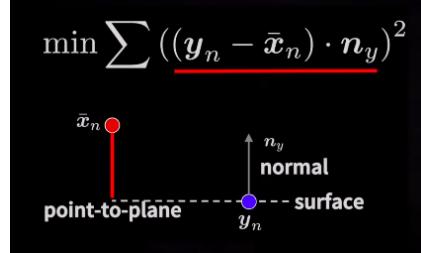


Figure 2.8: Minimisation point à plan

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 [Annexe D : La Décomposition SVD](#) et qu'on va appliquer pour simuler l'algorithme de l'ICP original (voir la théorie dans la section [Vanilla ICP](#) et la simulation dans [SLAM 2D](#)). 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[50], (quaternions unitaires [51], ou des quaternions duals[52])[44].

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 [Annexe E : Minimisation Point à Plan](#). 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 [53].

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[54], 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 :

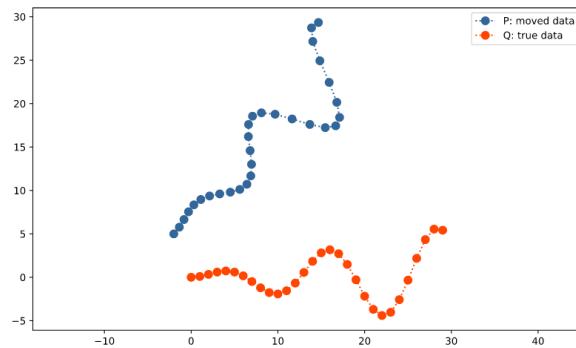


Figure 2.9: blablabla

### 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 Annexe D : La Décomposition SVD.

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.

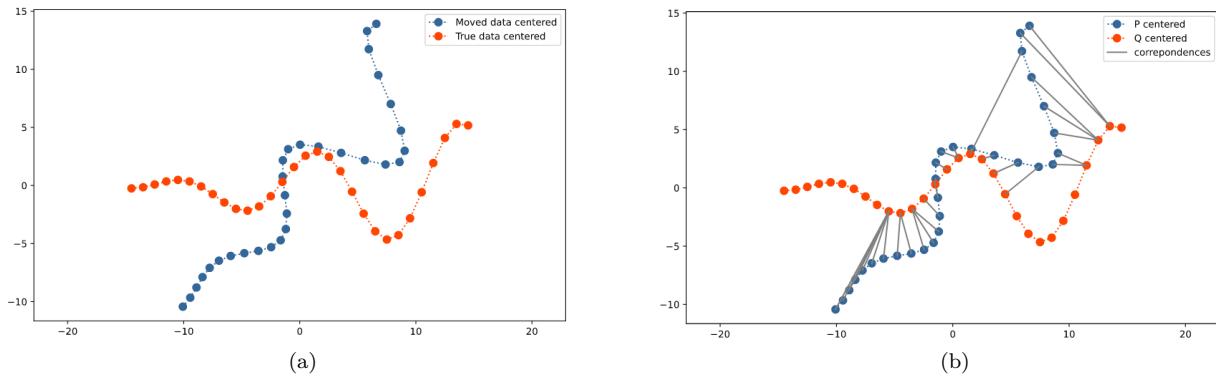


Figure 2.10: Première itération :  
 (a) Make data centered; (b) Compute correspondences

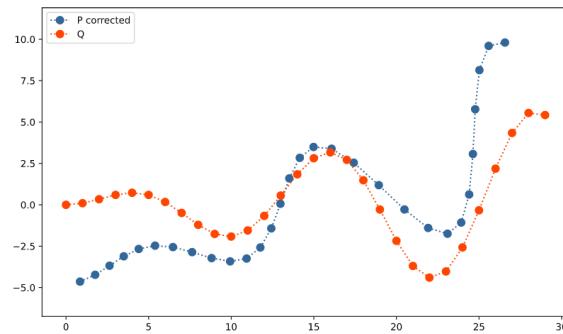


Figure 2.11:  $P$  corrigé après une itération d'ICP

#### 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.

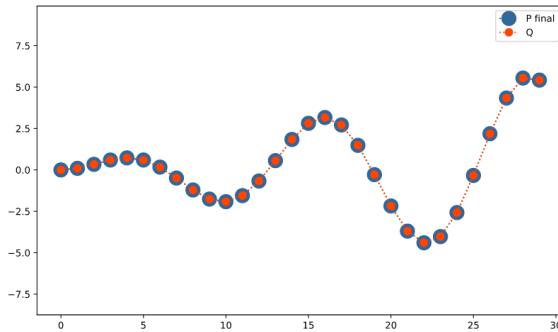


Figure 2.12:  $P$  corrigé après y avoir appliqué de multiples itérations de l'algorithme ICP

## 2.4 IRLS-ICP

### 13 FastAndRobust Iterative Registered Least Squares

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 appareilllements erronés en cas de

## 2.5 ICP Point à Plan

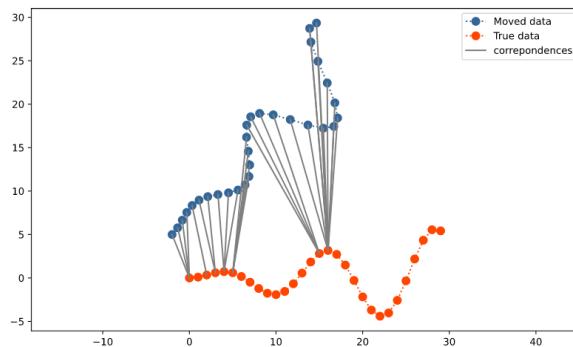


Figure 2.13: blablabla

## 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>

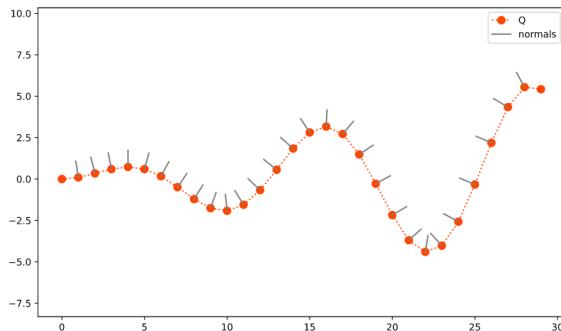


Figure 2.14: blablabla

## 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. La SVD (voir l'annexe D) peut aussi être utilisée comme méthode Font End seule avec une méthode Back End comme c'est le cas das ?? qui l'associe à l'UKF.

# 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 [55] comme une méthode pour l'alignement des scans 2D. Biber et Straßer ont ensuite développé la méthode dans un document commun avec Sven Fleck [56], 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[57].

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)[56]. 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  $2 \times 2$  :

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

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  $\Pi$  :

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

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  $\phi$ ) 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  $\phi$ ) (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  $\phi$ ) 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  $\phi$  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.
- $\Omega_i$  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  $\Omega_i$  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 [55].

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  $\Delta p$ , 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  $\lambda$  choisi de sorte que  $H'$  soit définie positive.

Cet algorithme est maintenant appliqué à la fonction  $\text{Score}$ "(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  $\text{Score}$ "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 :

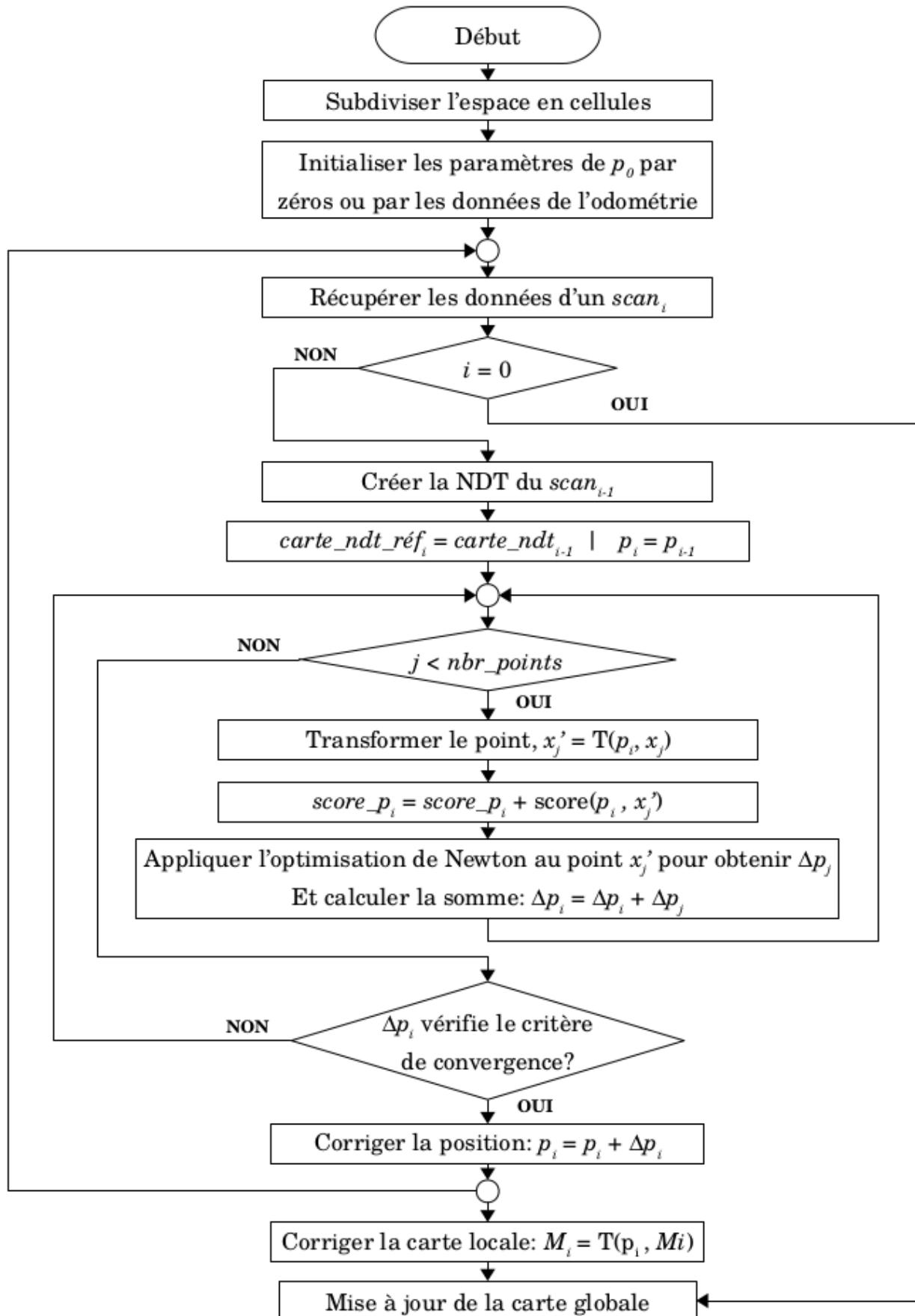
$$\frac{\partial^2 Q}{\partial p_k \partial p_j} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad (3.20)$$

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][58][59] 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 en milieu Intérieur**

# Table des Matières du 3<sup>ème</sup> Chapitre

---

<b>1 Scan Matching en 2D</b>	<b>66</b>
1.1 ICP Vanilla . . . . .	66
1.2 IRLS-ICP . . . . .	66
1.3 IRLS-ICP et Odométrie . . . . .	67
1.4 IRLS-ICP et Pose Graph . . . . .	67
1.5 NDT . . . . .	68
1.6 Comparaison . . . . .	68
1.6.1 Comparaison . . . . .	68
1.6.2 Occupancy Grid Map . . . . .	68
1.6.3 Comparaison . . . . .	69
<b>2 Scan Matching en 3D</b>	<b>71</b>
2.1 ICP en 3D . . . . .	71
2.2 ICP Vs. NDT . . . . .	72
2.2.1 Scan Matching de 2 nuages de points . . . . .	72
2.2.2 Comparaison . . . . .	72
2.2.3 Scan Matching pour cartographier un environnement . . . . .	72
2.2.4 Comparaison . . . . .	73

---

## TABLE DES MATIÈRES DU 3<sup>ÈME</sup> CHAPITRE

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 SLAM 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*.

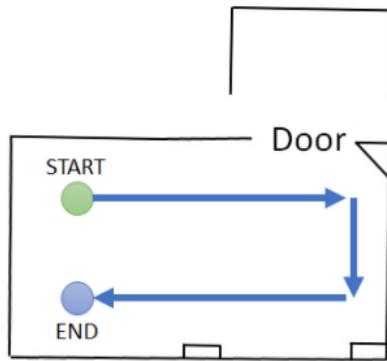


Figure 1.1: scanMatchingExample

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[60], 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 algorithme 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 [IRLS-ICP](#)).

## 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 Pose Graph

Afin d'appliquer l'optimisation via Pose Graph, nous ne pouvons pas utiliser les résultats obtenus dans la section [IRLS-ICP et Odométrie](#) 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 [IRLS-ICP](#).

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 noeud, 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 noeuds représentant les poses[61].

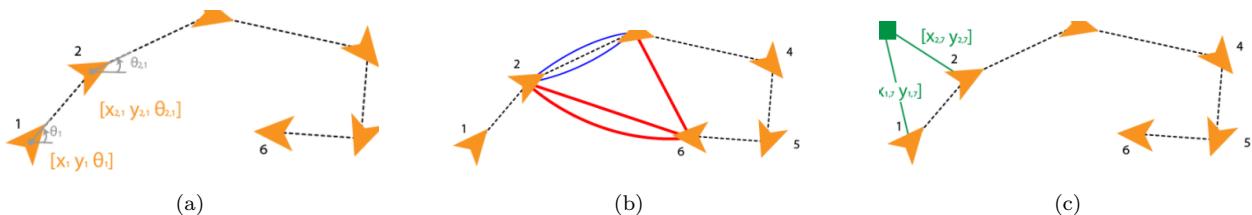


Figure 1.2: Pose Graph (a) estimation des noeuds pose et des arêtes les reliant; (b) Créeation de boucle en rajoutant une arête; (c) Rajouter un point noeud repère

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'environnement [3.2](#). Parmi elles une grille d'occupation binaire :

figure Occupancy Binary Grid

## 1.5 NDT

## 1.6 EKF

## 1.7 Comparaison

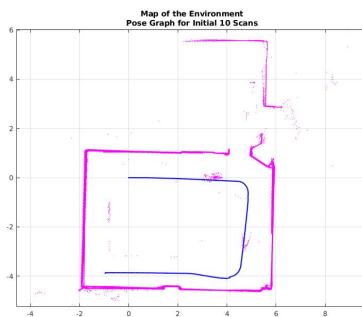


Figure 1.3: À corriger, ce ne sont pas les 10 premiers scans qui ont été pris en compte. C plutôt kamel les 690 scans

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

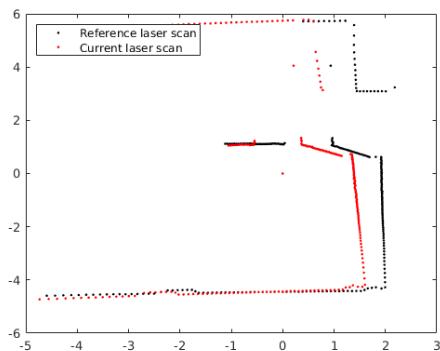


Figure 1.4: Les deux nuages de points en 2D à superposer

### 1.7.1 Comparaison

Nous pouvons constater une meilleure performance de l'algorithme de la NDT de 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.

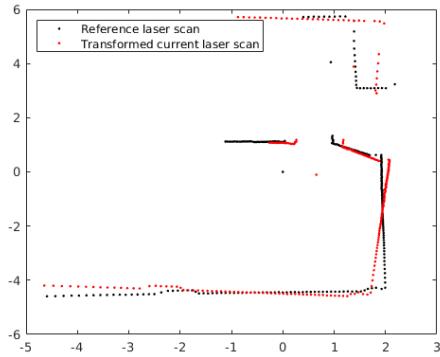


Figure 1.5: Scan Matching des deux nuages de points via l'algorithme de l'ICP

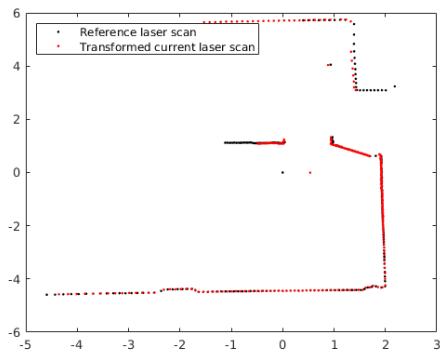


Figure 1.6: Scan Matching des deux nuages de points via l'algorithme NDT

### 1.7.2 Occupancy Grid Map

Comparaison entre le full slam w le slam en temps réel (icp et 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[62].

[https://fr.mathworks.com/videos/implement-simultaneous-localization-and-mapping-slam-with-MATLAB-152.html?s\\_eid=PSM\\_15028](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.

### 1.7.3 Comparaison

Ils sembles tous identiques.

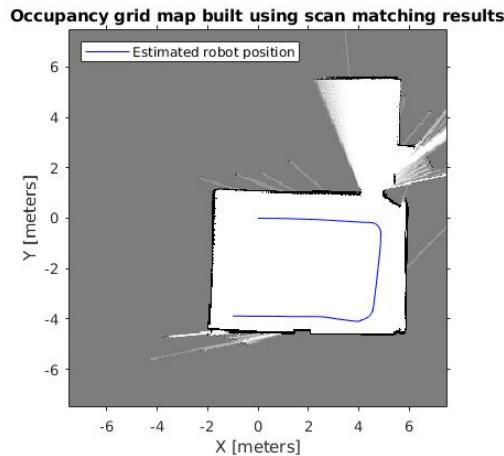


Figure 1.7: occupancyGridMap NDT occupancyGridMap NDT

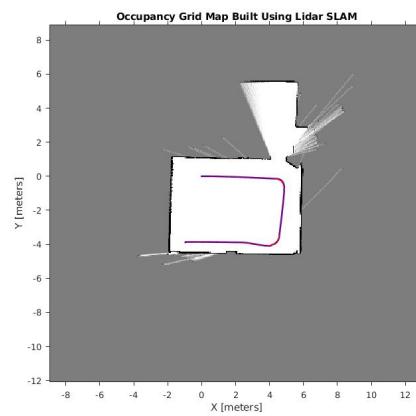


Figure 1.8: grid map full slam grid map full slam

## 2 Scan Matching en 3D

### 2.1 ICP en 3D

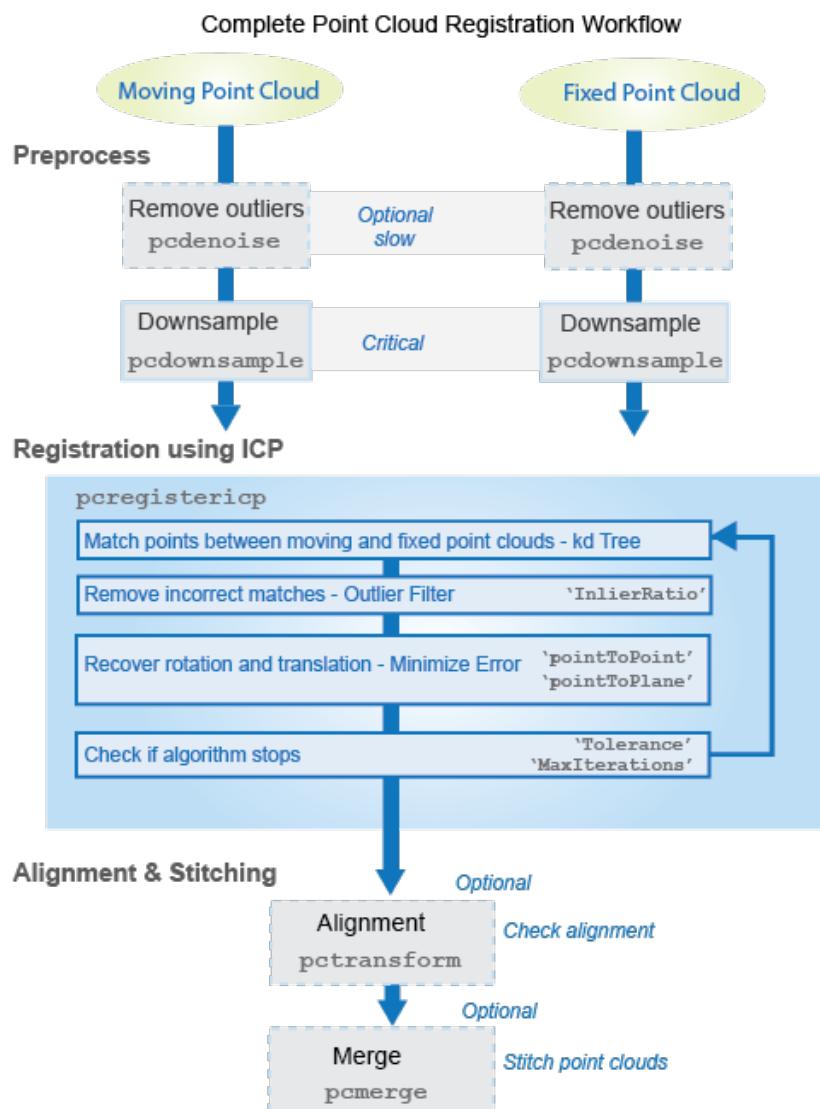


Figure 2.1: L'organigramme général d'ICP

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 :

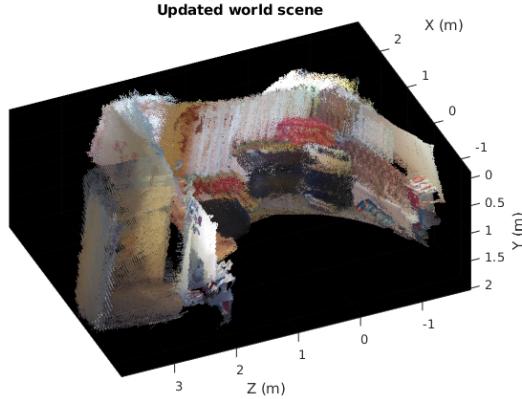


Figure 2.2: Résultat de la cartographie via scan matching basé sur la variante Point à Point de l’Algorithme ICP

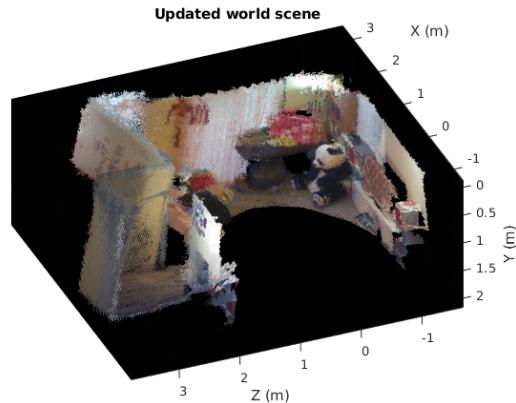


Figure 2.3: Résultat de la cartographie via scan matching basé sur la variante Point à Plan de l’Algorithme ICP

## 2.2 ICP Vs. NDT

ICP point à Point Vs. NDT Remarque : le fait de rallonger le pas diminue le temps et rend les résultat 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

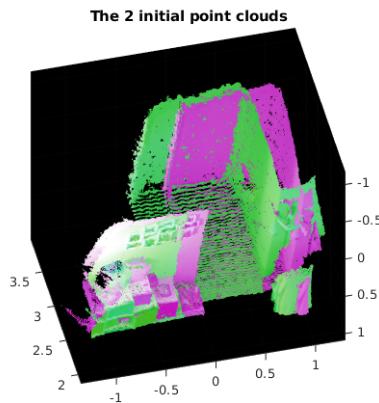


Figure 2.5: État initial de Scan Matching : 2 nuages de points consécutifs

#### 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 R2018a<sup>1</sup> 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[63].

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 lestelement plus flous en plus de ne pas aligner les scans aussi bien que `gridAverage`.

<sup>1</sup>Attention 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

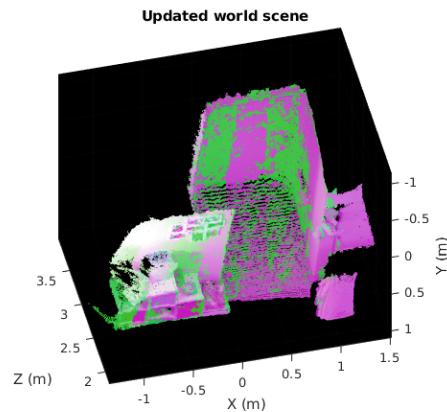


Figure 2.6: Scan Matching des 2 nuages de points via l'algorithme de l'ICP

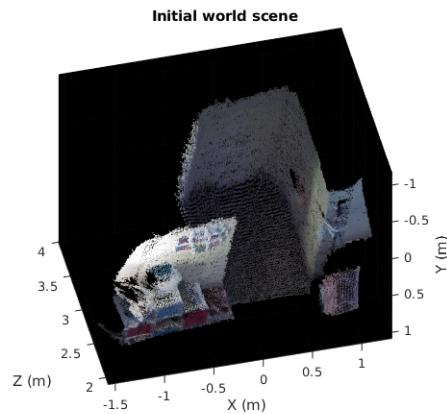


Figure 2.7: État initial de Scan Matching : 1er nuage de point

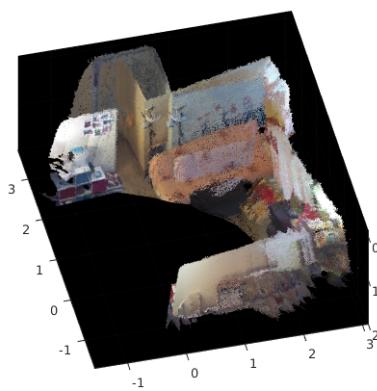


Figure 2.8: Scan Matching des 2 nuages de poits via l'algorithme de l'NDT

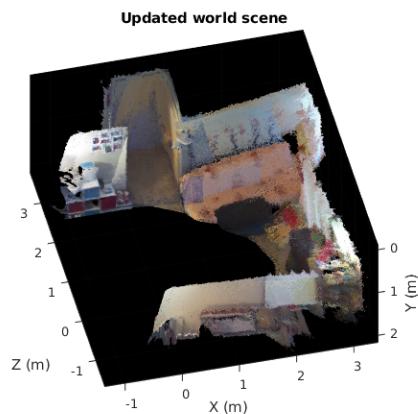


Figure 2.9: Scan Matching pour cartographier un environnement via l'algorithme de l'ICP

# **Chapitre IV**

## **Simulations en milieu Extérieur**

---

Dans un premier temps il a fallu changer de système d'exploitation, j'ai donc migré vers la distribution Ubuntu de Linux qui était en ce temps là encore indispensable pour exploiter ROS pleinement, en 2021, plusieurs systèmes d'exploitations peuvent être utilisés de manière efficiente dans l'usage de la plateforme ROS, mais Ubuntu reste une le système d'exploitation par excellence pour en profiter.

# **Chapitre V**

## **Conclusion Générale**

---

Le ciel est bleu.

## **Chapitre VI**

## **Annexes**

# A Annexe A : Le filtre de Kalman

Le filtre dit de Kalman a fait l'objet de plusieurs publications fondatrices entre 1958 et 1961, de plusieurs auteurs (notamment Bucy et R.E. Kalman [Kalman, 1960, Kalman and Bucy, 1961]) mais seul Kalman est finalement passé à la postérité. Il sert à l'estimation dans le temps de la valeur de variables aléatoires gaussiennes, en fonction de valeurs mesurées et d'un modèle d'évolution temporelle. Comme de nombreux filtres bayésiens, il fonctionne en deux étapes, l'état futur du filtre est prédit, puis comparé à la nouvelle mesure, ce qui permet finalement d'en estimer un état corrigé.

Une prédiction de l'état futur des variables que l'on cherche à estimer est tout d'abord réalisée, à partir de l'état précédent, d'une fonction de propagation et de paramètres de contrôle éventuels qui influent sur l'évolution prévisible de la variable. Cet état est ensuite rapproché de la nouvelle observation, grâce à une fonction de mesure qui permet d'obtenir à partir de l'état prédit la mesure correspondante (cette fonction est simplement l'identité si l'on est capable de mesurer tous les éléments du vecteur d'état). On peut alors calculer l'innovation, c'est à dire la différence entre la mesure prédite et celle mesurée, et corriger la prédiction pour obtenir une estimation de l'état courant (mis à jour). Ces étapes sont décrites schématiquement sur la figure A.1. Ce filtre est itératif, la prise en compte de l'historique des mesures étant contenue dans ces deux étapes (on suppose que l'état estimé est descriptible par une chaîne de Markov de rang 1).

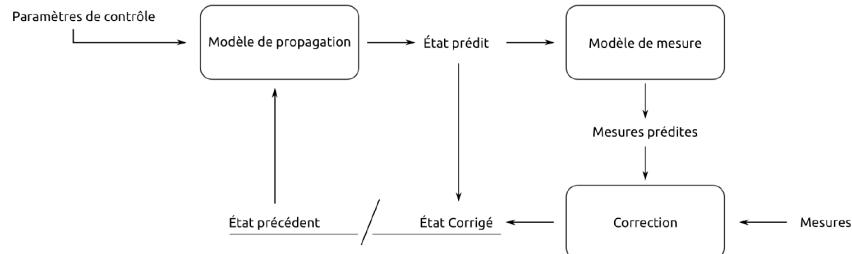


Figure A.1: Schéma conceptuel d'un Filtre de Kalman, tel que les rectangles représentent des fonctions, tandis que les vecteurs représentent des vecteurs de variables aléatoires

## 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 prédictive :

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

- 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.

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

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

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

$R$  est la covariance du bruit de mesure.

Le filtre de Kalman Étendu (EKF) est probablement l'algorithme d'estimation Non Linéaire le plus largement utilisé, cependant, les 4 dernières années de recherche au sein de la communauté de l'estimation ont démontré la complexité d'implémentation, de configuration et la fiabilité conditionnée par la quasi linéarité sur l'échelle de temps pour les *mises à jour*, des difficultés issues de la linéarisation[64].

### Filtre de Kalman Inodore : UKF

Le filtre UKF est un filtre de Kalman non linéaire qui semble prometteur en tant qu'amélioration par rapport à l'EKF. Dans l'UKF, la densité de probabilité est approximée par un échantillonnage déterministe de points qui représentent la distribution sous-jacente sous la forme d'une gaussienne. La transformation non linéaire de ces points est destinée à être une estimation de la distribution a posteriori, dont les moments peuvent ensuite être déduits des échantillons transformés. La transformation est connue sous le nom de transformation inodore. L'UKF a tendance à être plus robuste et plus précis que l'EKF dans son estimation de l'erreur dans toutes les directions[65].

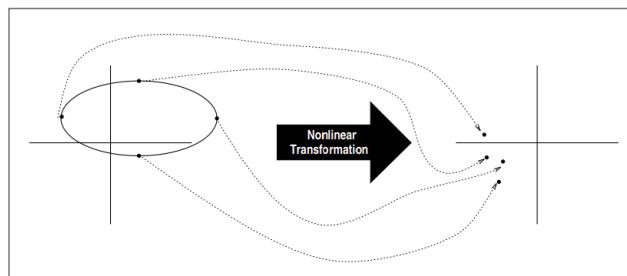


Figure A.2: Principe de base de la transformation indolore de l'UKF

### 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.

## B Annexe B : Filtre Particulaire

### Approche FastSlam

FastSLAM splits the SLAM problem into a robot localization problem, and a collection of landmark estimation problems that are conditioned on the robot pose estimate. It uses a modified particle filter for estimating the posterior over robot paths. According to a study by [23], FastSLAM allows autonomous mobile robots an ability to learn a consistent model of its environment, which is its prerequisite. FastSLAM is efficient to be applied to environments mapping because it can process far larger data than could be handled by the EKF [24]. This algorithm is capable of building maps with orders of magnitude with more landmarks than Kalman Filter. It can also handle a small number of particles which works well regardless of the number of landmarks under certain conditions. In addition, it is able to recover from false data association and can pursue multiple data associations simultaneously [25]. Long term process is an inconsistent stochastic filter but, as a heuristic estimator, it can be both tractable and highly accurate, while in short-term process it is able to produce consistent results given enough particles [26]. The advantages of the fastSLAM is that it does not consider the measurement acquired at time; instead, the measurement is through resampling [27]. Its current form cannot produce consistent estimates in the long-term and may produce quite accurate results but the estimate of its accuracy soon becomes optimistic. Many researchers underestimate fastSLAM due to its uncertainties because of the higher landmarks, or precise sensors or more frequent observations but will improve accuracy and also speed up particle depletion [26]. Nowadays, robot simulators have robust physics engines, high-quality graphics, and convenient interfaces, affording researchers to substitute physical systems with their simulation models in order to pre-estimate the performance of theoretical findings before applying them to real robots.

*/home/blad/SLAM/Memoire/bibli/24\_simulation\_of\_simultaneous\_localization\_and\_mapping.pdf*

### Approche Amers

# C Annexe C : Estimateurs robustes

On détaille ici le domaine des « statistiques robustes », dénommées ainsi par Peter J. Huber dans [66]. On cherche dans ce domaine à définir des estimations robustes à partir d'échantillons qui ne sont pas susceptibles d'être significativement altérées dans le cas où une ou plusieurs valeurs aberrantes par rapport à un ensemble de référence dans l'échantillon de mesure. Il ne s'agit pas d'un domaine intrinsèquement lié *Scan Matching* ou au SLAM, mais les techniques qui y sont développées peuvent être exploitées dans ces cas de figure.

On ne considère pas ici une densité de probabilité gaussienne pour les échantillons, celle-ci étant parfaitement mesurée par les opérateurs courants de moyenne géométrique et de variance. Comme nous avons déjà eu l'occasion de l'expliquer, différents phénomènes peuvent éloigner significativement les mesures de cette distribution dans certains problèmes.

## C.1 Définitions

On définit tout d'abord quelques métriques nous permettant ensuite d'illustrer les avantages d'une approche robuste des statistiques.

### C.1.1 Variance

Il s'agit d'une mesure de la fonction de distribution d'une variable aléatoire, qui correspond à son second moment statistique. Elle se définit en général pour une variable discrète par :

$$Var(X) = E((X - \mu)^2) \quad \text{avec } E \text{ l'espérance et } \mu = E(X) \quad (\text{C.1})$$

On ne prend pas en compte de l'erreur dans cette définition, c'est-à-dire que toutes les observations sont supposées correspondre à la variable  $X$ . Dans le cas d'observations bruitées, d'autres estimateurs convergeant vers la variance de la distribution visée sont possibles et parfois souhaitables (cf. section [Calculs de variance, efficacité variable](#)).

### C.1.2 Point de rupture

Cette quantité est définie comme étant le pourcentage de points marginaux (aberrations) acceptés par un estimateur avant de voir sa valeur altérée. En notant  $\Sigma$  un estimateur, on peut noter la définition du point de rupture (*Breakdown Point*) :

$$BP_{\Sigma} = \min\left(\frac{N_{Outliers}}{N_{Inliers}}\right) | \Sigma(Outliers) \neq \Sigma(Inliers + Outliers) \quad (\text{C.2})$$

### C.1.3 Efficacité

On peut par ailleurs définir l'efficacité relative d'un estimateur  $\tilde{\theta}$  par rapport à l'estimateur  $\hat{\theta}$  par le rapport inversé de leurs variances :

$$RE(\tilde{\theta}, \hat{\theta}) = \frac{var(\hat{\theta})}{var(\tilde{\theta})} \quad (\text{C.3})$$

Ce ratio n'a de sens que si les estimateurs en question ne sont pas biaisés, dans le sens où leur variance n'est pas inférieure à la variance réelle des échantillons. Il est possible dans ce cas d'estimer ce biais, ou encore de définir une efficacité invariante (par exemple en utilisant la variance du logarithme de l'estimation). On note ARE (Efficacité Relative Asymptotique) la limite de  $RE$  quand le nombre d'échantillons tend vers  $\infty$ .

Cette mesure est souvent utilisée pour comparer un estimateur optimal pour une distribution Gaussienne à un estimateur dont la convergence est plus lente, mais plus résistante au bruit.

### C.1.4 Robustesse

La définition proposée par Huber[66] pour quantifier la robustesse d'un estimateur est liée à une notion de continuité des valeurs estimées. Elle s'écrit formellement (en reprenant les notations de Huber), avec  $(T_n)$  la séquence d'estimations,  $F$  la fonction de distribution des échantillons,  $F_0$  une fonction de distribution de référence (par exemple gaussienne), et  $d$  une métrique sur l'espace des distributions considéré :

(C.4)

## C.2 Exemples

### C.2.1 Calculs de variance, efficacité variable

La comparaison de la médiane et de la moyenne, estimateurs couramment utilisés du centre d'une fonction de distribution, est intéressante lorsqu'appliquée à ces différentes métriques à partir d'une distribution gaussienne : la moyenne est beaucoup plus efficace que la médiane ( $ARE(\text{médiane}, \text{moyenne}) = 0.64$ ), mais l'ajout de points marginaux (par exemple issus d'une autre distribution gaussienne) l'affecte beaucoup plus rapidement.

On peut facilement s'en convaincre, la valeur de la moyenne étant modifiée dès l'altération d'un unique échantillon. La médiane est en ce sens plus robuste, la moitié des échantillons devant être modifiés avant de changer la valeur de cet estimateur (si l'on suppose une altération aléatoire et uniforme).

L'estimation de la largeur d'une distribution est l'objet d'un autre exemple de Tukey[67], que nous reprendons ici, illustrant quantitativement ce phénomène. Cet exemple est également repris dans les notes de cours de B. Ripley[68], dont [16] a repris les calculs

Considérons  $n$  observations  $Y_i$  tirées d'une fonction de distribution donnée. Supposons tout d'abord que cette fonction de distribution soit une loi normale, de moyenne  $\mu$  et de variance  $\sigma^2$  (notée  $N(?, \sigma)$ ). On considère les estimateurs par rapport à l'estimateur  $\hat{\sigma}$  et  $\tilde{\sigma}$  tels que :

(C.5)

avec  $\bar{Y}$  le centre de masse des points  $Y_i$  et

(C.6)

La normalisation de  $\tilde{\sigma}^2$  est choisie pour que ces opérateurs soient égaux pour une distribution normale. Leur efficacité relative (pour un nombre d'échantillons tendant vers inf en reprenant C.3) vaut :  $ARE(\hat{\sigma}^2, \tilde{\sigma}^2) = 0.88$ . On peut montrer que  $\hat{\sigma}$  est effectivement l'estimateur optimal pour la loi normale.

On propose, pour rendre compte d'une source de bruit indépendante, de générer une séquence de points tels que chaque point a une probabilité  $1 - e$  d'être selon la loi normale  $N(\mu, \sigma)$  et une probabilité  $e$  d'être selon la loi normale  $N(\mu, 9\sigma)$ . L'évolution de l'efficacité relative de nos deux estimateurs en fonction de  $e$  est visible sur le tableau C.1.

$\epsilon$ (%)	$ARE(\hat{\sigma}^2, \tilde{\sigma}^2)$
0	0.88
0.1	0.95
0.2	1.0
1	1.4
5	2.0

Tableau C.1: Efficacité relative des estimateurs  $\hat{\sigma}$  et  $\tilde{\sigma}$  en fonction de  $\epsilon$

La variance de ces observations reste proportionnelle à  $\sigma^2$ , et notre métrique est invariante d'échelle. On constate aisément que l'avantage en efficacité du calcul de la variance classique est très vite remis en cause en présence de bruit, 0.2% de points faux étant suffisants en pratique pour que le calcul basé sur la somme des écarts absolu soit plus efficace.

D'autres opérateurs sont possibles pour estimer la variance de la distribution, celui présenté dans cet exemple étant mis en avant pour illustrer les différences de comportement possibles en présence de bruit. Les deux opérateurs présents ont notamment un point de rupture de 0%, ce qui est en général préjudiciable, et peut être mitigé par d'autres opérateurs basés par exemple sur l'usage d'une médiane. On pourra notamment introduire l'opérateur MAD (Médiane des écarts à la médiane), qui est utilisé dans notre implémentation :

(C.7)

Cet opérateur n'est pas très efficace<sup>1</sup>, mais très robuste aux points aberrants. Le coefficient de 0.6745 fait office de normalisation, rendant l'opérateur MAD exactement égal à  $\hat{\sigma}$  dans le cas de la loi normale.

### C.2.2 M-estimateurs

Si l'on considère le problème de l'estimation de la valeur centrale d'une fonction de densité de probabilité, plusieurs estimateurs sont couramment utilisés. On pourra citer la moyenne et la médiane bien sûr, mais aussi la moyenne tronquée (on néglige les *ailes* de la fonction de distribution, par le biais d'un coefficient  $\alpha$  qui désigne la fraction des points extrêmes - les plus éloignés de la moyenne - à négliger).

Il est cependant courant d'utiliser les *M-estimateurs*, estimateurs basés sur le calcul du maximum (d'où le M) de vraisemblance. Cette vraisemblance est déterminée par rapport à la fonction de densité de probabilité modèle  $f$ , et on introduit souvent la fonction  $\rho = -\log(f)$  qui simplifie les calculs. L'estimateur maximisant la vraisemblance conduit à minimiser  $\rho$ , en supposant que les échantillons soient indépendants :

(C.8)

---

<sup>1</sup> Au sens de l'ARE, celle ci étant de 0.37 si on le compare à  $\hat{\sigma}$  pour une loi normale.

La fonction de densité de probabilité doit cependant être connue pour maximiser la vraisemblance et obtenir un estimateur, et celle-ci n'est pas nécessairement déterminée de manière exacte. Cette formule est par ailleurs exacte pour un grand nombre d'échantillons, mais n'assure pas nécessairement une convergence rapide de l'estimateur. Il est possible, comme le propose Huber de choisir une fonction  $\rho = \log(f)$  arbitraire dont on minimise la somme des contributions sur les échantillons.

(C.9)

## D Annexe D : La Décomposition SVD

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

La dérivation de la minimisation point à point donnée dans l'annexe E requiert la maîtrise d'une méthode de factorisation matricielle populaire appelée Décomposition en Valeurs Singulières (ou SVD, de l'anglais singular value decomposition). On retrouve souvent la SVD dans les processus de minimisation des problèmes linéaires.

Bien qu'étant très semblable à factorisation de la matrice en une forme canonique via la décomposition d'une matrice en éléments propres, mais la SVD se révèle être plus efficace puisque c'est un isomorphisme qu'on applique entre deux espaces vectoriels de dimensions

## **E Annexe E : 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] Pierre Merdrignac. Système coopératif de perception et de communication pour la protection des usagers vulnérables.
- [16] Benjamin Lefauveux. Detection, localisation and tracking of obstacles and moving objects, from a stereovision setup.
- [17] Francois Jehin. Le lidar.
- [18] Wikipedia. Lidar.
- [19] Xin Wang, Kai Guo HuaZhi Pan, and Sheng Luo Xinli Yang. The evolution of lidar and its application in high precision measurement. (502).
- [20] Zachary S. Nahman. Robot learning for loop closure detection and slam.

- [21] Yan Lu, Shu-Hao Yeh Joseph Lee, Baifan Chen Hsin-Min Cheng, and Dezhen Song. Sharing heterogeneous spatial knowledge: Map fusion between asynchronous monocular vision and lidar or other prior inputs.
- [22] Shayan J AVAHERIAN Behnam B AHR Farbod K HOSHNOUD, Ibrahim I. E SAT. Quantum entanglement and cryptography for automation and control of dynamic systems.
- [23] Shayan J AVAHERIAN Behnam B AHR Farbod K HOSHNOUD, Ibrahim I. E SAT. Quantum cooperative robotics and autonomy.
- [24] D. Koller and N. Friedman. Probabilistic graphical models: Principles and techniques.
- [25] Andela Juri, Ivan Markovic Filip KendeÅj, and Ivan Petrovic. A comparison of graph optimization approaches for pose estimation in slam.
- [26] F. Dellaert, A. Jain V. Agrawal, M. Sklar, and M. Xie. <https://github.com/borglab/gtsam>.
- [27] G. Grisetti R. KÄmmerle, K. Konolige H. Strasdat, and W. Burgard. G2o: A general framework for graph optimization. page pp. 3607â3613.
- [28] K. Mierle S. Agarwal and Others. Ceres solver.
- [29] Pieter Abbeel. Advanced robotics(<https://people.eecs.berkeley.edu/~pabbeel/cs287-fa09/lecture-notes/lecture21-6pp.pdf>).
- [30] Cyril Stachniss. Bayes filter(<https://www.youtube.com/watch?v=oUq0a8jHSQg>).
- [31] Yung-Chang Chen | Bor-Woei Kuo, Hsun-Hao Chang and Shi-Yu Huang. A light-and-fast slam algorithm for robots in indoor environments using line segment map.
- [32] N. Laird A. Dempster and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. 39:1–38.
- [33] F. Lu and E. Milios. *Globally consistent range scan alignment for environment mapping*, volume Vol. 4, pages 333–349.
- [34] B. Wegbreit D. Hähnel, S. Thrun and W. Burgard, editors. *Towards lazy data association in slam*.
- [35] E.Milios F. Lu. Globally consistent range scan alignment for environment mapping. pages 333–349.
- [36] Nikolaus Correll. *SLAM as a Maximum-Likelihood Estimation Problem*, chapter Graph-based SLAM, pages 185–194.
- [37] Matlab. Scan matching overview(<https://www.mathworks.com/help/vision/point-cloud-processing.html>).
- [38] P. J. Besl and N. D. McKay. *Method for registration of 3-d shapes*.
- [39] Y. Chen and G. Medioni. *Image Vision Computing*, volume Vol. 10. Issue 3 edition.
- [40] N. D. McKay Besl, Paul J. A method for registration of 3-d shapes. Vol. 14:pp. 239–256.
- [41] Hans Martin Kjer and Jakob Wilm. Evaluation of surface registration algorithms for pet motion correction.
- [42] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm.
- [43] Philipp Glira. Point cloud tools for matlab ([https://github.com/pglira/Point\\_cloud\\_tools\\_for\\_Matlab](https://github.com/pglira/Point_cloud_tools_for_Matlab)).
- [44] Marie-José Aldon et André Crosnier Lounis Douadi. Variantes de l'algorithme icp pour le recalage de données 3d couleur. page 9.
- [45] CÃ©dric Fleury. Le kd-tree : une mÃ©thode de subdivision spatiale.

- [46] K. Sakaue T. Masuda and N. Yokoya. Registration and integration of multiple range images for 3-d model construction.
- [47] K. Pulli. Multiview registration for large data sets.
- [48] J. Weng C. Dorai and A. Jain. Registration and integration of multiple object views for 3d model construction. Vol. 20(No. 1).
- [49] G. Turk and M Levoy. Zippered polygon meshes from range images.
- [50] B.K.P. Horn. Closed-form solution of absolute orientation using unit quaternions. 4:629–642, 1987.
- [51] O. Faugeras and M. Hebert. The representation, recognition and localisation of 3d objects. Vol. 5(No. 3).
- [52] L. Shao M. Walker and R. Volz. Estimating 3d location parameters using dual number quaternions. Vol.5(No. 3).
- [53] 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.
- [54] Dieter Fox. Wiki : Amcl (<http://wiki.ros.org/amcl>).
- [55] W. StraBer P. Biber, editor. *The normal distributions transform : A new approach to laser scan matching*, volume 3. IEEE/RSJ International Conference on (2003).
- [56] Peter Biber, Sven Fleck, and Wolfgang Strasser. A probabilistic framework for robust and accurate matching of point clouds. In Carl Edward Rasmussen, Heinrich H. Bülthoff, Bernhard Schölkopf, and Martin A. Giese, editors, *Pattern Recognition*, pages 480–487. Springer Berlin Heidelberg.
- [57] M. Magnusson. The three-dimensional normal-distributions transform : an efficient representation for registration, surface analysis, and loop detection.
- [58] A. El Dor. Improvement of particle swarm optimization algorithms : applications in image segmentation and electronics.
- [59] M. Clerc and P. Siarry. Une nouvelle métaheuristique pour l'optimisation difficile : la méthode des essaims particulaires.
- [60] Fabio Morbidi. Localisation et navigation de robots([https://home.mis.u-picardie.fr/~fabio/Eng/documents/Teaching/LNR20-21/LNR\\_p1.pdf](https://home.mis.u-picardie.fr/~fabio/Eng/documents/Teaching/LNR20-21/LNR_p1.pdf)).
- [61] G.Grisetti, C. Stachniss R. Kummerle, and W. Burgard. A tutorial on graph-based slam. Vol. 2(No. 4):pp. 31–43.
- [62] Mihir Acharya. Implement simultanious localization and mapping (<https://www.mathworks.com/matlabcentral/fileexchange/66284-implement-simultaneous-localization-and-mapping-slam-with>).
- [63] Matlab. pcdownsample documentation <https://fr.mathworks.com/help/vision/ref/pcdownsample.html#bupqqn1-1-gridAverage>.
- [64] Simon J. Julier and Jeffrey K. Uhlmann. Unscented filtering and nonlinear estimation. Vol. 92(No. 3).
- [65] Wikipedia. Extended $kalman$  filter.
- [66] P. J. Huber. *Robust Statistics*. Wiley Series in Probability and Statistics.
- [67] J. W. Tukey. *A Survey of Sampling from Contaminated Distributions*, volume 2.
- [68] B. D. Ripley. *Robust statistics*.