Contribution à la localisation robuste embarquée pour la navigation autonome

Pierre Merriaux

Normandie Université

THÈSE

Pour obtenir le diplôme de doctorat

Spécialité AUTOMATIQUE, SIGNAL, PRODUCTIQUE, ROBOTIQUE

Préparée au sein de « ESIGELEC/IRSEEM »

Contribution à la localisation robuste embarquée pour la navigation autonome

Présentée et soutenue par
Pierre MERRIAUX

Thèse soutenue publiquement le 6 octobre 2016 devant le jury composé de		
M. Patrick Rives	Directeur de recherche INRIA	Rapporteur
M. Philippe Bonnifait	Professeur des universités UTC	Rapporteur
M. Philippe Martinet	Professeur des universités Ecole centrale de Nantes	Examinateur
M. Bruno Steux	Docteur Directeur R&D Nexter-Robotics	Examinateur
M. Pascal Vasseur	Professeur des universités Litis	Directeur de thèse
M. Xavier Savatier	Docteur HDR Enseignant Chercheur	Codirecteur de thèse
M. Yohan Dupuis	Docteur Chef de projet	Examinateur
M. Rémi Boutteau	Docteur Enseignant Chercheur	Examinateur

Thèse dirigée par Pascal VASSEUR, laboratoire Litis

ESIGELEC

SPMM ECOLE DOCTORALE

irseem

École doctorale nº 351 SPMII : Sciences Physiques, Mathématiques et de l'Information pour l'Ingénieur

THÈSE

pour obtenir le grade de docteur délivré par

l'Université de Rouen

Spécialité doctorale "Génie Informatique, Automatique et Traitement du Signal"

présentée et soutenue publiquement par

Pierre MERRIAUX

le 6 octobre 2016

Contribution à la localisation robuste embarquée pour la navigation autonome

Directeur de thèse : **PASCAL VASSEUR**
Co-encadrant de thèse : **XAVIER SAVATIER**

Jury

Patrick Rives,	Professeur	Rapporteur
Philippe Bonnifait,	Professeur	Rapporteur
Philippe Martinet,	Professeur	Examinateur
Bruno Steux,	Docteur	Examinateur
Pascal Vasseur,	Professeur	Directeur de thèse
Xavier Savatier,	Docteur HDR	Co-Directeur de thèse
Yohan Dupuis,	Docteur	Examinateur
Rémi Boutteau,	Docteur	Examinateur

ESIGELEC – IRSEEM – EA 4353
Pôle, informatique, instrumentation et système
Avenue Galilée, F-76801 St-Etienne du Rouvray, France

THESE

Remerciements

Je suis particulièrement reconnaissant envers Xavier Savatier pour m'avoir poussé dans cette aventure passionnante et permis d'organiser au mieux les travaux du laboratoire autour de mon activité de thèse. Je tiens également à remercier Pascal Vasseur pour avoir accepté de diriger cette thèse malgré mon parcours peu classique. Son expérience et son calme m'ont toujours apporté le bon conseil au bon moment.

Patrick Rives et Philippe Bonnifait qui malgré leurs emplois du temps chargés ont accepté le travail de rapporteur sur mes travaux. Leurs remarques éclairées ont participé à améliorer la précision de ce manuscrit. Philippe Martinet et Bruno Steux qui ont examiné mon travail, et participé à mon jury.

Rémi Boutteau pour son encadrement, et nos prises de tête sur les matrices de rotation. Yohan Dupuis, pour son encadrement bien sûr, mais surtout pour la collaboration fructueuse sur de nombreux sujets que nous avons développée. Pour nos innombrables repas sur le pouce devant un bout de code ou une figure Matlab inexplicable.

Les membres de l'équipe Argos : Romain Rossi, Pailin Chanuc, Vincent Vauchey, Benoît Decoux, Florent Rigaud, Florent Roger, Rémi et Xavier pour leur investissement sans limite, et leur esprit d'équipe. Nicolas Ragot pour nos discussions en courant le midi.

Nos enfants qui ont accepté avec une étonnante facilité nos dépaysements cyclopédiques. Et avant tout ma femme, qui en plus d'avoir supporté avec patience ces trois années un peu déjantées, a lu et relu ce manuscrit.

Table des matières

Liste des figures

Liste des tableaux

Introduction

Les robots. Longtemps enfermés dans des cages à l'usine ou cantonnés à une utilisation en laboratoire, ils s'apprêtent à envahir notre quotidien et occupent déjà une grande part de notre actualité. A la base, un robot n'est rien d'autre qu'une machine constituée d'électronique, de mécanique et d'un programme informatique qui va doter l'ensemble d'aptitudes pour lui permettre d'interagir avec notre environnement ; le plus souvent pour reproduire des tâches de manière répétitive (robotique manufacturière) ou encore pour aller là où l'homme ne peut aller (robotique d'exploration et d'intervention en milieu hostile) mais encore le plus souvent sous le contrôle de l'homme (robotique télé-opérée). Plusieurs facteurs concourent à la diffusion de ces nouvelles technologies dans notre société mais l'un des plus importants est probablement la capacité du robot à fonctionner de manière de plus en plus autonome. Les progrès dans les systèmes électroniques embarqués y sont pour beaucoup ; sur un robot, il est désormais possible d'embarquer de nombreux capteurs, fournissant un état exhaustif du système et de son environnement, et une puissance de calcul suffisante pour traiter toutes ces données en temps réel. Cette association capteurs et traitements (logiciels) embarqués vise à doter le robot d'une forme d'autonomie décisionnelle, au sens où il pourra percevoir et analyser de nombreuses informations en temps réel, pour décider de telle ou telle action.

Dans de nombreuses applications, en particulier celles de la robotique des services ou encore de la voiture à conduite automatisée, l'autonomie décisionnelle du robot passe par sa capacité à naviguer dans un environnement naturel ; donc par définition complexe. La navigation autonome est un thème de recherche largement étudié, dont le vocabulaire est d'ailleurs celui des premiers navigateurs : cartes, référentiels, navigation à l'estime, amers,... Naviguer, c'est aller d'un point A à un point B ; ce qui suppose que le robot sait localiser A (« où suis-je ? ») et B (« où vais-je ? ») dans un même référentiel. La localisation est une brique indispensable à la navigation autonome du robot. Sur cette question, les systèmes à positionnement par satellite (GPS) ont évidemment changé la donne. Cependant et malgré de nombreux progrès dans ce domaine, le positionnement par satellite s'avère inadapté dès lors que la navigation du robot exige une localisation précise (submétrique, impossible actuellement sauf avec des GPS différentiels coûteux et complexes d'usage) et une forte disponibilité de la mesure, point faible du GPS dans les environnements où la couverture satellitaire ne passe pas. Si le robot connaît son point de départ, une autre façon de le localiser pendant son déplacement est d'estimer au mieux ce déplacement (*egomotion*) en utilisant les informations internes (capteurs proprioceptifs) caractérisant ce mouvement et, à l'image des premiers navigateurs utilisant les étoiles pour se repérer, en exploitant des mesures de capteurs dits extéroceptifs qui vont fournir au robot des points de repère (amers) ou toute autre information lui permettant de se (re)localiser. Là encore, de nombreux travaux ont été menés depuis plusieurs décennies, qui vise à améliorer les informations de capteurs proprioceptifs telles que les centrales inertielles ou encore à tirer le meilleur parti d'informations issues de capteurs extéro-

ceptifs tels que des caméras, LiDARs [1], SONAR (pour la robotique sous-marine). Il reste cependant encore de nombreux défis, aussi bien autour des méthodes et algorithmes associés que des capteurs employés. Aucune solution n'est encore assez générique pour être robuste aux changements de conditions d'environnement, aux différents milieux rencontrés par le robot, tout en fournissant des indicateurs de qualité de service et des modes dégradés.

Mes travaux de thèse s'inscrivent dans cette problématique de localisation pour la robotique mobile autonome avec une extension au domaine spécifique du véhicule à conduite automatisée. Les recherches que j'ai effectuées sont nées de réflexions et travaux conduits pour plusieurs projets industriels dans le cadre de mes missions en tant que Responsable ingénierie au sein du laboratoire IRSEEM [2] de l'ESIGELEC. De prime abord, ces projets ont porté sur des applications disjointes : système de stabilisation de mouvements d'une passerelle de navire pour les opérations de maintenance en mer pour le projet NAVALIS [3], localisation du robot Viking dans le cadre du challenge international de robotique ARGOS [4] ou encore localisation de véhicule en environnement routier pour la logistique urbaine dans cadre du projet SAVEMORE [5]. Pourtant tous ces travaux ont eu un dénominateur commun : l'utilisation de la technologie LiDAR - dont l'évolution permet aujourd'hui des mesures temps réel 3D et sur 360° - dans des environnements de type industriels - que nous qualifions par la suite de complexes - avec la mise en correspondance de ces mesures avec des connaissances à priori sur l'environnement. Ces travaux ont donné lieu à trois principales contributions : 1) la mise en place d'une méthodologie et d'outils de mesure pour qualifier la performance de systèmes embarqués dédiés à la localisation en environnement complexe, 2) la conception et la mise en œuvre d'une solution de localisation précise pour le challenge de robotique Argos et 3) l'extension au domaine du véhicule autonome en couplant la brique de localisation Argos avec une méthode de localisation topologique.

Ces travaux présentés dans la suite de ce mémoire sont organisés en quatre chapitres :

Le **chapitre 1** dresse un état de l'art des méthodes de localisation. Après avoir expliqué les différentes manières de représenter l'environnement, nous aborderons les méthodes en elles-mêmes en fonction des capteurs envisagés. La contrainte opérationnelle du Challenge Argos nous a poussé à considérer plus particulièrement les systèmes LiDARs dans la mesure où ils sont peu sensibles aux variations d'environnement. Enfin, nous expliquerons le concept de localisation en deux couches composé tout d'abord d'une localisation de haut niveau peu précise mais disposant d'une capacité à converger à partir d'une connaissance très approximative de la position initiale ; puis d'une couche 2 suffisamment précise pour garantir le contrôle du véhicule.

Le **chapitre 2** présente une méthodologie de développement que nous avons mise en œuvre dans les chapitres 3 et 4. Les algorithmes de robotique sont relativement complexes à mettre au point, et nécessitent de nombreux essais et ajustements. Notre méthode permet de minimiser le temps de développement en fusionnant les tests en simulation, à petite échelle et pleine échelle. Afin d'objectiver la précision des algorithmes de nos tra-

1. LiDAR : Light Detection And Ranging, en robotique principalement utilisé comme télémètre LiDAR : laser range finder
2. IRSEEM : Institut de Recherche en Systèmes Électroniques Embarqués (http://www.esigelec.fr/fr/presentation-irseem)
3. NAVALIS : Projet ADEME AMI Navire du Futur
4. ARGOS : Autonomous Robot for Gas and Oil Sites, challenge de robotique organisé par la compagnie pétrolière et gazière *Total* avec le soutien de l'*ANR*
5. SAVEMORE : Projet INTERREG IV A

vaux, il est nécessaire de disposer d'une vérité terrain avec une meilleure précision que le système sous test. Les moyens d'essais de référence de l'IRSEEM seront conjointement introduits avec notre méthodologie, afin d'expliquer leur implication dans les différentes étapes.

Le **chapitre 3** concerne nos travaux sur la méthode de localisation basée LiDAR embarquée dans le robot Viking du Challenge Argos. Les domaines d'application visés sont les usines pétrochimiques. Ces milieux sont complexes à plusieurs titres : les scènes ne correspondent ni aux environnements intérieurs ni extérieurs de la littérature, les installations sont imbriquées avec les structures, et les coursives où se déplace le robot sont composées de multiples niveaux. Il est donc nécessaire de disposer d'une méthode de localisation robuste à six degrés de liberté. Nous obtenons une localisation embarquée actualisée en temps réel à 20Hz. L'équipe Viking a remporté les deux premières étapes du challenge Argos.

Dans le **chapitre 4**, nous proposons une méthode de localisation topologique de véhicule routier sur un graphe constitué par des données Open Street Map. Seuls les capteurs déjà présents dans les véhicules actuels, ABS et ESP, sont utilisés. Notre méthode permet une convergence malgré une très faible connaissance de la position initiale. La précision de positionnement atteinte est de la même classe que celle d'un GPS standard.

Dans la dernière partie de cette thèse, nous résumons les différents points clés abordés, et nous proposons des perspectives aux problèmes soulevés.

Chapitre 1

Localisation

« Si la variance σ tend vers 0, nous savons tout sur rien, et si σ tend vers l'infini, nous ne savons rien sur tout. »

Robert Vallée, 2015

Sommaire

En localisation, un mobile est représenté par un trièdre dans un repère dit global. Nous parlons de position pour définir le lieu où se trouve ce trièdre dans le repère global, et d'orientation pour définir la direction des axes du trièdre. Le terme pose regroupe quant à lui la position et l'orientation. Bien que venant du domaine aéronautique, il est courant également de rencontrer le terme attitude qui signifie la même chose que l'orientation.

Les problèmes de localisation peuvent être classés en deux catégories :

— Localisation locale (*tracking*), quand la position est connue avec une certaine précision à l'initialisation.

— Localisation globale (en robotique, problème connu sous le terme de *kidnapping robot*) si le système ne dispose d'aucune connaissance sur la position avant initialisation. Ce problème est beaucoup plus ardu, et fait encore l'objet de nombreuses recherches.

Différents moyens permettent de localiser un mobile autonome, le plus évident étant le *GPS* (Global Positioning System). En effet, des récepteurs bon marché et très largement diffusés permettent d'obtenir une position dans un référentiel terrestre. Cependant, il souffre de fortes limitations pour la localisation d'un mobile autonome. Sa précision ne permet pas de contrôler le véhicule à partir de la position fournie. L'orientation reste quant à elle inconnue et c'est un paramètre tout aussi important que la position pour le contrôle. Le système ne fonctionne pas en intérieur, et de manière générale, la précision peut se dégrader très fortement dès que l'environnement n'est plus dégagé. C'est pourquoi la communauté a imaginé d'autres solutions de localisation.

Depuis une vingtaine d'années, deux grands axes se sont dégagés : la localisation basée vision, c'est-à-dire utilisant une caméra, et la localisation LiDAR. A la différence du GPS, une caméra ou un LiDAR ne sont que des capteurs qui perçoivent une partie de leur environnement. Nous n'obtenons pas la position directement, il faut traiter l'information et souvent disposer de données initiales pour déterminer une pose. Si l'environnement est inconnu, le problème de localisation est intimement lié à celui de l'estimation de la géométrie de la scène. Dans ce cas, il faut déterminer à la fois la position et la carte de l'environnement ; nous parlons alors de SLAM (Simultaneous Localization And Mapping) qui est un domaine de recherche à part entière et particulièrement actif ces dernières années.

1.1 Représentation de l'environnement

Avant de parler de localisation, il est indispensable de définir par rapport à quel référentiel nous allons exprimer la position. Depuis que la robotique autonome existe, la question de la représentation de l'environnement se pose. Comme le présente Drouilly [2015], les différentes représentations mises au point dépendent du type de mobile, des capteurs employés, et bien sûr, du milieu envisagé.

1.1.1 Cartes métriques

Une carte métrique représente la géométrie de l'environnement dans un espace métrique continu. Il faut dans un premier temps être capable de transformer les mesures des capteurs en informations géométriques, c'est-à-dire disposer d'un modèle métrique du capteur. Par exemple, un capteur de luminosité sans hypothèse sur la luminosité ambiante ne nous donne pas d'information sur la géométrie de la scène.

L'information stockée dans la carte peut prendre différentes formes :

— Les cartes d'occupation, qui sont une discrétisation de l'environnement sous forme d'une grille (Elfes [1987]). Basées généralement sur le filtrage bayésien (section 1.3), chaque case représente une probabilité d'occupation. Elles sont très adaptées à l'utilisation de sonar et LiDAR mais comme la hauteur des obstacles n'est pas connue, elles sont peu transposables entre robots. Par exemple, si le capteur n'est pas installé à la même hauteur, les deux cartes obtenues seront différentes.

Pour pallier ce problème, des extensions ont été proposées en codant dans chaque case la hauteur de l'obstacle (Herbert et collab. [1989]). La carte obtenue peut être qualifiée de 2.5D, il faut pour cela faire l'hypothèse que l'environnement est représentable avec une surface continue unique. Pour contourner cette limitation qui ne permet pas de représenter l'environnement sous un pont par exemple, Triebel et collab. [2006] et Pfaff et collab. [2007] ont développé les *muli-level surfaces*.

La diversité des environnements à représenter a conduit à la généralisation des grilles d'occupation en 3D (Moravec [1996]) en remplaçant les cases par des voxels. Si l'environnement est de taille importante, l'espace de stockage peut rapidement devenir prohibitif. Pour cela, nous pouvons remplacer le maillage homogène de l'environnement par un Octree (Wurm et collab. [2010]), qui a entre autres l'avantage de ne coder finement que les zones utiles.

— Les cartes représentant des objets dans l'espace : à l'inverse des grilles d'occupation, l'environnement est représenté sous forme d'objets basés sur les mesures. Les objets peuvent être de bas niveau comme les nuages de points 3D issus d'un LiDAR (Cole et Newman [2006]), ou des points 3D et leurs descripteurs issus d'une reconstruction par vision (Royer [2006]). Mais il est également possible d'utiliser des amers de plus haut niveau extraits à partir du nuage de points, comme des lignes, des arcs de cercles ou des plans (Fallon et collab. [2012]).

Malgré les améliorations apportées aux cartes métriques, la place occupée en mémoire, la mise à jour peu réaliste dans un contexte dynamique réel, ainsi que l'inévitable dérive lors des multiples concaténations de données capteurs représentent des freins à leur déploiement sur des environnements à grande échelle.

1.1.2 Cartes topologiques

Une carte topologique contient les lieux qui peuvent être visités ainsi que la manière de s'y rendre. Contrairement à la carte métrique, il n'existe pas de notion de position métrique dans un repère global ou encore d'échelle. Sa représentation graphique est juste une vue humaine qui n'a pas forcément de corrélation avec la réalité de l'environnement. Elle peut très facilement être représentée informatiquement sous forme d'un graphe. Les données stockées pour chaque nœud dépendent de l'application et du capteur utilisé. Un exemple très courant de carte topologique est le plan de métro (figure **1.1**). Les stations y sont représentées ainsi que le moyen de s'y rendre, c'est-à-dire le numéro de ligne qu'il faut emprunter. La représentation graphique avec les stations à peu près correctement placées géographiquement est juste là pour aider la compréhension humaine.

Ces cartes sont adaptées à la représentation de grands espaces en gardant un niveau d'abstraction suffisant. Il n'y a plus de problème d'estimation métrique, et de vastes lieux peuvent être représentés par un unique nœud. Un autre avantage concerne la planification du chemin à emprunter par le robot, qui se détermine à haut niveau dans un graphe, plus simple d'un point de vue algorithmique que dans une carte continue.

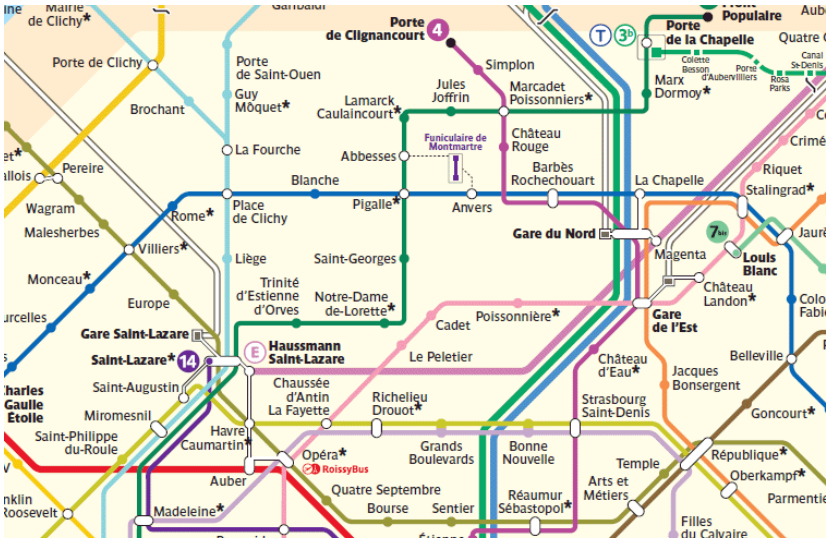


FIGURE 1.1 – Exemple courant de carte topologique, le plan de métro.

1.1.3 Cartes hybrides et hiérarchiques

Les représentations métriques et topologiques comportent des avantages et des inconvénients pour la robotique autonome : les cartes métriques proposent une représentation détaillée de l'environnement, mais coûteuse en ressources mémoire et temps de calcul. Les cartes topologiques quant à elles proposent une représentation plus compacte de l'environnement, mais sans représenter l'aspect local, ce qui pose problème lors de la navigation.

Rapidement, l'idée est venue d'exploiter les avantages des deux représentations. Ces cartes qui ne peuvent plus vraiment être classées dans l'une ou l'autre des deux catégories s'appellent hybrides ou hiérarchiques. Elles stockent les informations de différentes approches sur plusieurs niveaux hiérarchiques.

Différentes approches ont été explorées :

— Les cartes par superposition : les données métriques et topologiques représentant le même environnement sont stockées en parallèle. Les données topologiques sont construites par segmentation des informations métriques (Thrun [1997]). Même si le mixage des deux méthodes apporte des avantages, surtout au niveau de la navigation à long terme, elle cumule également les inconvénients de place de stockage et de complexité des deux méthodes.

— L'assemblage de cartes locales : l'idée consiste à conserver la précision des cartes locales métriques, et connecter les différentes cartes locales par un graphe topologique. Ceci permet de conserver l'aspect aisé de la navigation à long terme des représentations topologiques, et le non cumul des erreurs des dérives de reconstruction, les cartes locales étant par définition petites (par exemple Meilland et collab. [2011] ou Lisien et collab. [2005]).

Depuis les origines de la navigation autonome, la représentation d'environnement a toujours été un sujet de recherche intense. Une multitude de méthodes a été proposée, jusqu'aux cartes hybrides qui sont les plus évoluées. Malgré leurs avantages, la localisation dans de grands environnements reste problématique. Les recherches actuelles s'orientent entre autres vers les cartes sémantiques pour enrichir les données spatiales par d'autres concepts de plus hauts niveaux et donner du sens à la perception (cf travaux de Drouilly [2015]).

1.2 Capteurs

Ce paragraphe ne se veut pas une liste exhaustive de tous les capteurs existants permettant de déterminer la localisation d'un mobile. Selon le cas d'application, il existe une multitude de solutions possibles. Parmi les solutions « exotiques », citons juste par exemple Frassl et collab. [2013] qui proposent d'exploiter les champs magnétiques naturellement présents dans les bâtiments ou encore Koch et Zell [2014] qui utilisent des *tags RFID* (radio frequency identification).

Nous nous concentrerons donc ici sur les systèmes les plus courants dans le cadre de la localisation de mobiles autonomes, c'est-à-dire permettant une précision suffisante pour son contrôle :

— Le GPS, et pourquoi il ne peut pas répondre à tous les besoins ;

— Une brève présentation des capteurs proprioceptifs ;

— La vision qui est un thème très porteur ;

— Et le LiDAR, qui reste aujourd'hui une des solutions les plus robustes dans de nombreux cas d'applications.

Pour la vision et le LiDAR, nous aborderons également les méthodes associées permettant de calculer une localisation.

1.2.1 GPS

Le GPS (Global Positioning System) permet d'obtenir directement par triangulation une position sur le globe. Elle est obtenue par la mesure du temps de trajet d'ondes électromagnétiques entre une constellation de 24 satellites et un récepteur. Cependant, comme vu en introduction, la précision d'un récepteur bas coût étant de l'ordre de dix mètres, il est impossible d'envisager le contrôle d'un véhicule autonome avec un tel dispositif. De plus, un ensemble d'inconvénients vient réduire l'intérêt d'une solution en apparence idéale. Citons par exemple :

— Les problèmes de canyon urbain, quand il n'y a plus suffisamment de satellites visibles par le récepteur.

— Le multi-trajet générant des sauts violents de position : il se produit quand le signal GPS arrive au récepteur après plus d'un trajet à cause de réflexions près du récepteur ;

— Le fonctionnement impossible en intérieur ;

— Seule la position est disponible et non la pose (l'orientation reste inconnue) ;

— La faible fréquence d'actualisation de la position (de moins en moins vrai, certains récepteurs bas coût offrent une fréquence de rafraîchissement de 10Hz) ;

— Moindre précision sur l'axe Z (altitude) ;

— La dépendance avec les États-Unis

— La facilité à brouiller les signaux utilisés...

Bien sûr différentes solutions existent pour pallier une partie de ces inconvénients. D'autres constellations existent comme *Glonass* (Russe), *Galileo* (Européen) et *Beidou* (Chinois). De plus en plus, les récepteurs actuels utilisent l'ensemble de ces satellites, mais il est difficile de chiffrer l'apport en précision. Le GPS différentiel (DGPS) utilise deux récepteurs pour s'affranchir en partie des perturbations dues à la propagation dans les couches atmosphériques. Sommairement, un récepteur de référence est installé à une position

parfaitement connue, et la compare avec la position calculée. La différence obtenue est transmise au récepteur mobile pour corriger sa position. Le DGPS permet d'obtenir des précisions d'ordre centimétrique, mais pour un coût beaucoup plus important. Attention, il existe différentes « classes » de système différentiel :

— Le dispositif Egnos (European Geostationary Navigation Overlay Service) est un ensemble de 34 stations de références réparties sur le globe qui retransmettent les informations de correction par 3 autres satellites. Beaucoup de récepteurs « bas coût » supportent maintenant les signaux Egnos, la précision est ramenée autour de quelques mètres. Il est également souvent appelé DGPS.

— La solution différentielle avec deux « récepteurs privés » est quant à elle souvent dénommée par les fabricants RTK GPS (Real Time Kinematic). Il est communément admis que sa précision centimétrique se dégrade d'un centimètre par dizaine de kilomètres d'éloignement entre le mobile et le récepteur de référence.

Une autre solution couramment utilisée consiste à fusionner les données obtenues par le GPS avec une centrale inertielle (section 1.2.2.2). Du fait de la dérive des centrales, la précision absolue du positionnement ne sera pas améliorée, mais cette association permet une augmentation de la fréquence de rafraîchissement ainsi qu'un « lissage » des positions. La centrale inertielle présentera également l'avantage de déterminer l'orientation (au moins le roulis et le tangage ; le cas du cap étant plus compliqué). L'amélioration des performances est liée au prix investi dans la centrale inertielle, et dans ce domaine, il n'y a pas de limite !

1.2.2 Capteurs proprioceptifs

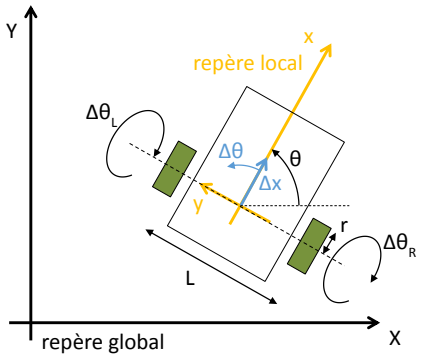
Les capteurs proprioceptifs vont effectuer des mesures par rapport à ce qu'ils perçoivent localement du déplacement du mobile, contrairement aux capteurs extéroceptifs qui vont les réaliser entre le mobile et l'environnement. Cette mesure étant directe, il est possible de calculer une position à partir d'une seule de ces informations. Ce n'est pas le cas des capteurs proprioceptifs qui nécessitent d'intégrer toutes les mesures de déplacements depuis un point initial pour calculer la position courante. Et toute mesure étant entachée d'erreur, cette intégration va poser problème dans le sens où le cumul des erreurs va rapidement amener à une dérive de la position. Il est donc nécessaire de recaler le calcul avec des mesures sur des références externes au mobile. Tout se passe comme si vous vous leviez la nuit dans le noir, vous faites trois pas tout droit (de longueur à peu près connue) puis vous tournez d'un quart de tour à droite (environ -90°, ça dépend du degré de réveil du mobile) et normalement vous êtes devant la première marche de l'escalier. Généralement, il est bon de tendre les bras en avant et de toucher la rampe ou un mur pour s'assurer que l'intégration des 4 mesures précédentes n'a pas trop dérivé.

1.2.2.1 Odométrie

L'odométrie consiste à « compter » le nombre de tours de roue ou mesurer la variation angulaire de patte pour déterminer le déplacement du mobile. Ce déplacement est à déterminer en linéaire Δx et en rotation $\Delta\theta$. Dans le cadre d'un robot différentiel à roues ou chenilles (figure 1.2a), nous pouvons déterminer un déplacement local à la plateforme avec les équations de la figure 1.2b. Les déplacements seront projetés dans le repère local (x, y) par la formule 1.1.

$$\begin{bmatrix} x \\ y \end{bmatrix}_i = \Delta x_i \begin{bmatrix} \cos(\Delta\theta_i/2) \\ \sin(\Delta\theta_i/2) \end{bmatrix}_i \tag{1.1}$$

Une fois les déplacements locaux obtenus, il faut les projeter dans le repère global afin de pouvoir les cumuler. Pour l'orientation α, c'est direct, il suffit d'intégrer les rotations (équation 1.4). Pour la position, il faut faire un changement de repère avant de pouvoir les intégrer (équation 1.5).



(a) Repères local et global

$$\Delta x = r\frac{\Delta\theta_R + \Delta\theta_L}{2} \tag{1.2}$$

$$\Delta\theta = r\frac{\Delta\theta_R - \Delta\theta_L}{L} \tag{1.3}$$

r	:	rayon de la roue
L	:	voie du robot
$\Delta\theta_R$	:	angle (radian) de rotation roue droite
$\Delta\theta_L$	:	angle (radian) de rotation roue gauche
Δx	:	déplacement linéaire
$\Delta\theta$	:	rotation

(b) Equations pour le calcul de l'odométrie

FIGURE 1.2 – Calcul de l'odométrie sur un robot de type différentiel

$$\theta_i = \sum_{j=1}^{i} \Delta\theta_j \tag{1.4}$$

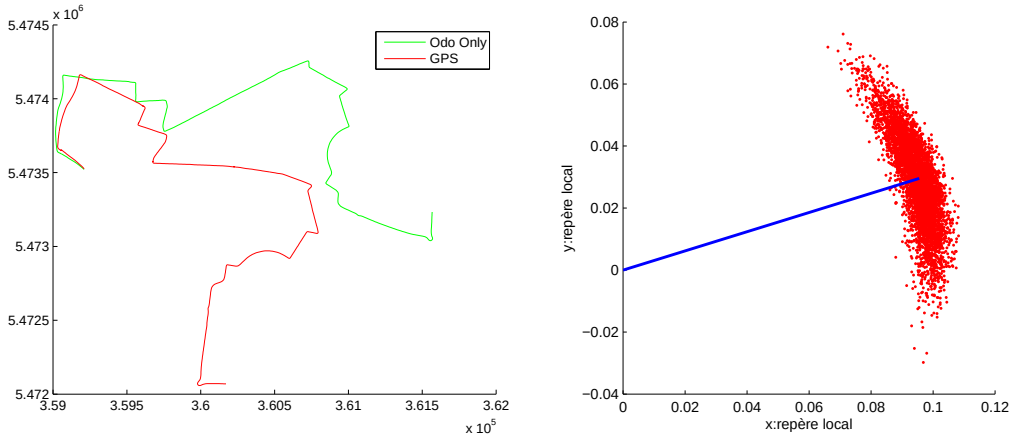
$$\begin{aligned}
\begin{bmatrix} X \\ Y \end{bmatrix}_i &= \sum_{k=1}^{i} \begin{bmatrix} \cos(\theta_k) & -\sin(\theta_k) \\ \sin(\theta_k) & \cos(\theta_k) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}_k \\
&= \sum_{k=1}^{i} \begin{bmatrix} \cos(\theta_k) & -\sin(\theta_k) \\ \sin(\theta_k) & \cos(\theta_k) \end{bmatrix} \begin{bmatrix} \Delta x_k \cos(\Delta\theta_k/2) \\ \Delta x_k \sin(\Delta\theta_k/2) \end{bmatrix}_k \\
&= \sum_{k=1}^{i} \Delta x_k \begin{bmatrix} \cos(\theta_k)\cos(\Delta\theta_k/2) - \sin(\theta_k)\sin(\Delta\theta_k/2) \\ \sin(\theta_k)\sin(\Delta\theta_k/2) - \cos(\theta_k)\sin(\Delta\theta_k/2) \end{bmatrix} \\
&= \sum_{k=1}^{i} \Delta x_k \begin{bmatrix} \cos(\theta_k + \Delta\theta_k/2) \\ \sin(\theta_k + \Delta\theta_k/2) \end{bmatrix}
\end{aligned} \tag{1.5}$$

En utilisant un vecteur d'état (notions abordées en section 1.3.1) la mise à jour étape par étape devient :

$$\begin{bmatrix} X \\ Y \\ \theta \end{bmatrix}_i = \begin{bmatrix} X \\ Y \\ \theta \end{bmatrix}_{i-1} + \begin{bmatrix} \Delta x_i \cos(\theta_{i-1} + \Delta\theta_i/2) \\ \Delta x_i \sin(\theta_{i-1} + \Delta\theta_i/2) \\ \Delta\theta_i \end{bmatrix}_i \tag{1.6}$$

Les différentes méthodes d'intégration de l'odométrie ainsi qu'une méthode de calibration de la voie L et du rayon r sont fournies en annexe C.

La qualité de l'estimation du déplacement dépend fortement de la qualité du contact roue/chenille sol qui n'est jamais ponctuel. De ce fait, un glissement va se produire et donc une erreur de mesure. Comme le montre la figure 1.3a, les déplacements linéaires sont nettement moins bruités que les rotations. Cela s'explique de la manière suivante : dans le cas d'un déplacement linéaire, si les accélérations sont correctement contrôlées, il y a aura peu de glissement pour ce type de mouvement. Seuls le diamètre des roues et éventuellement le taux de réduction moteur-roue sont mal connus et induisent des erreurs. Dans le cas d'une rotation, un robot différentiel ne peut tourner que par glissement. Nous constaterons donc une dérive importante en plus des sources d'inconnues précédentes. De plus, si les roues sont larges, le contact n'est pas ponctuel, et la détermination de la voie L n'est pas évidente. Nous nous apercevons par des expériences de calibration que cette voie varie en fonction du taux de rotation du mobile. L'influence de la géométrie des robots différentiels sur la qualité d'estimation de leurs rotations est abordée dans l'annexe C. Concrètement, dans les modèles nous n'appliquerons pas le même coefficient de dérive aux déplacements linéaires et aux rotations. Pour chaque déplacement effectué, nous obtenons une densité de probabilité en « forme de banane » (figure 1.3b).



(a) Dérive de l'odométrie sur Renault Mégane, trajet de 12 km

(b) Modèle probaliste du bruit odométrique, *bleu : trajectoire idéale - rouge : densité de probabilité après le déplacement*

FIGURE 1.3 – Dérive et modèle probabiliste de l'odométrie

A moins de disposer d'un modèle précis du terrain (modèle 3D, et contact sol), l'odométrie ne fournit qu'un déplacement 2D dans un plan.

Afin d'améliorer les performances de l'odométrie, il est possible de munir le robot de roues non propulsives qui serviront à la mesure de déplacement. De manière plus réaliste, si nous avons le choix, il est toujours préférable d'utiliser l'odométrie de roues non propulsives pour limiter les glissements. Par exemple sur une voiture classique, de type traction, nous utilisons l'odométrie des roues arrière. D'autre part, pour pallier la dérive du taux de rotation, il est courant de coupler l'odométrie à une centrale inertielle (section 1.2.2.2) ou du moins à un gyromètre (annexe C).

D'un point de vue pratique, sur des plateformes à propulsion électrique, si le jeu mécanique du réducteur est raisonnable, il est souvent possible d'utiliser les informations issues des codeurs d'asservissement des moteurs. Dans ce cas, on atteint facilement des résolutions inférieures au millimètre. Sur un véhicule routier thermique, les capteurs d'ABS/ESP sont disponibles sur le bus CAN et fournissent également une très bonne source d'odométrie. Certains fournisseurs proposent des systèmes sans contact avec le sol, donc

sans problème de glissement et de meilleure performance qu'avec un contact roue :

— radars doppler [1], pour la mesure de vitesse relative par rapport au sol

— système optique [2], analogue aux souris informatiques (existe en un ou deux axes)

Ils sont généralement chers et volumineux donc plutôt adaptés aux véhicules routiers expérimentaux.

1.2.2.2 Centrale inertielle

Une centrale inertielle (IMU : Inertial Measurement Unit) mesure les déplacements par intégration de capteurs inertiels, 3 accéléromètres et 3 gyromètres. L'idée est potentiellement très fiable car elle ne dépend pas de l'environnement et est difficilement perturbable. Malheureusement les capteurs inertiels sont bruités, ce qui amène à une dérive de la position au cours du temps. Notez que sur ce point, l'odométrie pour peu que la résolution soit suffisante, est très efficace quand la plateforme est immobile. Sa dérive dépend de la distance parcourue.

Les premières centrales inertielles sont mécaniques et utilisent une plateforme dite de Schüler. Un cardan constitué de 3 gyroscopes permet de stabiliser les 3 accéléromètres, qui se trouvent donc dans un référentiel d'orientation fixe. De manière très simplifiée, il « suffit » donc d'intégrer les accélérations pour retrouver la position (il faudrait au moins retrancher la gravité terrestre).

De nos jours, pour des raisons de simplicité de fabrication, d'évolution de capteur, de coût et de puissance de calcul disponible, la plupart des centrales inertielles sont de type *strap-down* ; c'est-à-dire que les 6 capteurs sont fixés sur le même référentiel. Il est donc nécessaire d'intégrer les gyromètres pour retrouver l'attitude, avant d'intégrer les accéléromètres.

La performance d'une centrale inertielle est intimement liée à la précision de ses capteurs ; et la précision des capteurs est directement liée à leurs coûts. Il existe de nombreuses technologies, aussi bien pour les accéléromètres que pour les gyromètres, mais toutes les centrales inertielles dites « bas coût » sont constituées de capteurs MEMS (Microelectromechanical systems) de par leur faible coût de production.

La position étant calculée en intégrant deux fois les accéléromètres, le moindre bruit ou biais entraîne une dérive très rapide. Dans les faits, la plupart des centrales « bas coût » ne fournissent que l'attitude et ne peuvent qu'interpoler la position entre les points fournis par un GPS. Il est relativement aisé de calculer le roulis et le tangage, en retrouvant la gravité avec les accéléromètres. La détermination du cap est plus compliquée. Soit le gyromètre a un biais très inférieur au taux de rotation de la terre (15°/h), et à condition de ne pas être sur un des deux pôles, il est possible de retrouver le Nord par projection du vecteur de rotation sur un plan horizontal (la précision obtenue dépend donc de la latitude sur le globe terrestre). Si ce n'est pas le cas, il faut une source extérieure pour recaler le gyromètre de cap : compas GPS différentiel, mesure de position (GPS) et hypothèses de déplacement, magnétomètres... Dans les centrales « bas coût », la solution magnétomètre est la plus employée. Elle comporte cependant plusieurs inconvénients :

— Premièrement, il faut calibrer le capteur avec les champs magnétiques générés par la plateforme porteuse. Cela consiste à faire tourner dans toutes les directions la plateforme avec l'IMU. Ce qui peut paraître aisé sur un robot de 5Kg l'est beaucoup moins sur une bouée instrumentée de plusieurs tonnes.

1. Delta DRS1000 : http://www.gmheng.com/speed_sensor.php
2. Correvit S-350 Aqua Sensors : http://datrontechnology.co.uk/products/speed-sensors/optical/

— Deuxièmement, malgré la calibration, si le mobile se déplace dans un environnement magnétique, typiquement juste sur une dalle de béton armée, l'estimation du cap sera complètement faussée.

En pratique, nous avons constaté des dérives de plus de 30° juste en roulant sur la dalle de notre laboratoire.

1.2.3 Capteurs extéroceptifs

Comme expliqué dans la section 1.2.2, les capteurs extéroceptifs mesurent une information par rapport à l'environnement. Il existe une multitude de capteurs pouvant s'inscrire dans cette catégorie. Les premiers utilisés dans le domaine de la localisation intérieure sont sûrement les télémètres ultrason (Elfes [1987]). Leur cône d'émission, autour de 30°, ne permet pas une grande directivité du capteur. Il existe également des télémètres infrarouge ; leur portée est faible, rarement plus d'un mètre. Mais, selon la méthode de localisation choisie, un capteur de luminosité ou magnétique peut être classé dans cette catégorie. Une autre voie assez prometteuse semble être l'utilisation des échos bruts d'un radar micro-onde (Jaud et collab. [2013]).

Dans les deux sections suivantes, nous nous intéresserons aux deux capteurs extéroceptifs les plus classiques ces dernières années en localisation : les caméras et les LiDARs.

1.2.4 Caméra

Une caméra mesure l'intensité lumineuse réfléchie par les différents objets constituant la scène. Ces rayons lumineux sont projetés sur un plan, c'est-à-dire que toute information de profondeur est perdue.

Si nous excluons les méthodes où des amers connus ont été installés dans l'environnement, comme Pinto et collab. [2012] qui se servent de mires actives infra-rouge, Wu et Tsai [2009] qui utilisent une caméra omnidirectionnelle et d'un ensemble de cercles noirs collés au plafond, ou encore Lim et Lee [2009] qui lisent des QR-codes disposés sur les murs, nous pouvons classer les méthodes de vision en deux grandes catégories : les méthodes directes et les méthodes indirectes.

Dans les deux cas, le principe général consiste à identifier la transformation qui s'est produite entre une image dite de référence et la vue courante issue de la caméra. Si l'image de référence est elle-même issue de la caméra, nous ne pouvons pas vraiment parler de localisation mais plutôt d'odométrie visuelle (Scaramuzza et Fraundorfer [2011], Fraundorfer et Scaramuzza [2012]). La trajectoire est déterminée en cumulant successivement les déplacements d'image en image (et par la même occasion les erreurs). Par contre si une base de référence est utilisée, par exemple des images géo-référencées, la pose calculée permet de déterminer la localisation dans le référentiel de la base de données. Ces bases de données peuvent prendre plusieurs formes : simples images géo-référencées, images satellites (Anirudh Viswanathan et Huber [2014]), une liste de points d'intérêts avec leurs coordonnées 3D (Royer [2006]), un modèle 3D de la scène (Caron et collab. [2014]),...

1.2.4.1 Méthodes directes

Les méthodes directes utilisent une transformation affine (*Warping*) de l'image courante pour la déformer afin de « recoller » au mieux au point de vue d'une image de réfé-

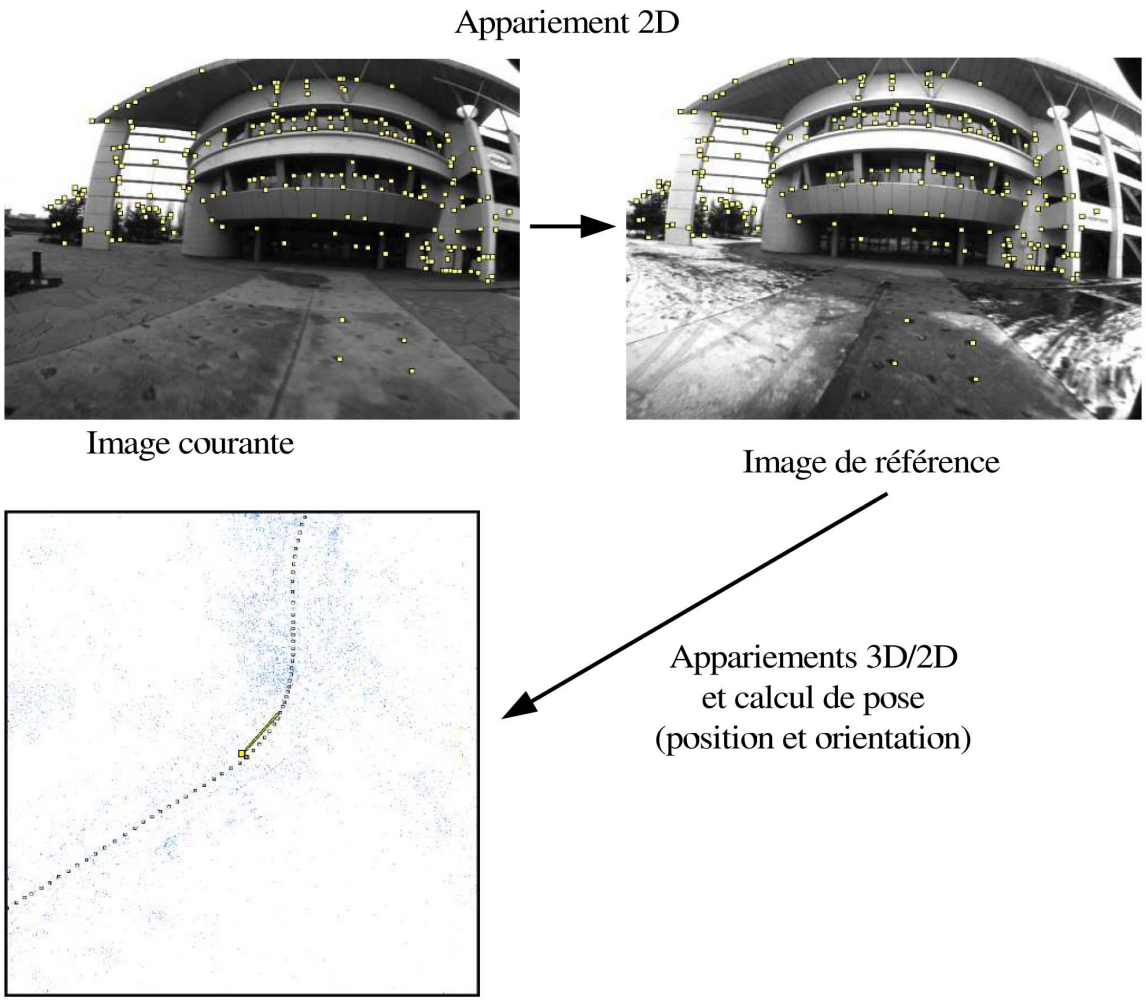


FIGURE 1.4 – Méthode indirecte de localisation basée vison : extraction de points de Harris, appariement par ZNCC, puis calcul de pose Royer [2006]

rence. La comparaison entre les deux images peut être réalisée par exemple en minimisant les erreurs de luminosité entre les images (Meilland [2012]). Un processus d'optimisation permet d'identifier la meilleure transformation, c'est-à-dire la pose entre l'image de référence et la vue courante.

1.2.4.2 Méthodes indirectes

Les méthodes indirectes utilisent les contraintes épipolaires afin de restituer l'information tridimensionnelle à partir de deux images. Elles consistent à extraire de l'image des amers (points caractéristiques, lignes, plans, cercles...) qui ont de bonnes chances d'être également identifiés dans une image précédente ou dans la base de données constituant la carte. Une fois ces amers obtenus, il faut les apparier entre eux afin de déterminer quel point dans l'image A correspond à tel autre dans l'image B. Ensuite des algorithmes basés sur la géométrie épipolaire, nécessitant un nombre minimum de points, permettent de recalculer le déplacement entre les deux images. L'appariement est une étape essentielle du processus. Il est donc souvent associé à un algorithme RANSAC (Random Sample Consensus : Fischler et Bolles [1981]) afin de supprimer les appariements qui ne respecteraient pas le modèle des contraintes épipolaires. A travers les travaux de Royer [2006], la figure 1.4 expose un exemple de localisation basée vision par une méthode indirecte.

1.2.4.3 Appariement

Il faut commencer par distinguer deux types d'appariements :

— Utilisant la totalité de l'image : qui ne peut être adapté qu'à des méthodes directes, puisque nous avons vu que les méthodes indirectes nécessitent l'utilisation de points d'intérêt.

— Utilisant un certain nombre d'amers dans l'image.

Plusieurs méthodes permettent de mesurer la similarité globale entre deux images (Bonin-Font et collab. [2008]). Nous pouvons citer entre autres l'information mutuelle, une corrélation sur la totalité de l'image (ZNCC par exemple), Gist, les signatures de Fourrier, les décompositions en ondelettes de Haar...

Tout comme pour les appariements sur la totalité de l'image, il existe de nombreuses solutions pour extraire des points d'intérêts, tels que les points de Harris (Harris et Stephens [1988]), ou des détecteurs plus évolués qui sont indépendants de l'échelle et des rotations comme SIFT et SURF (Lowe [2004], Bay et collab. [2006]). Pour l'appariement, il est possible d'évaluer la différence d'intensité lumineuse entre deux imagettes autour des deux points intérêts à tester, par une corrélation : SSD (Sum of Squared Differences), SAD (Sum of Absolute Differences) ou une ZNCC (Zero Normalised Cross Correlation). Les points d'intérêts de type SIFT ou SURF fournissent un descripteur invariant à l'échelle et en rotation, il suffit alors de minimiser la distance entre les descripteurs à apparier.

Malgré une très forte contribution de la communauté ces dernières décennies dans le domaine de la localisation basée vision, il reste de nombreux verrous à lever, parmi lesquels nous pouvons citer la robustesse aux variations d'environnement : position du soleil, saisons... (Tom Krajnik [2014], McManus et collab. [2014]).

1.2.4.4 Vision non conventionnelle

Les méthodes présentées ci-dessous sont indépendantes du type d'objectif qui détermine le champ de vue du capteur et donc de la perception de la scène. Si les caméras perspectives sont les plus courantes, de nombreux travaux portent sur les caméras omnidirectionnelles : catadioptriques (objectifs orientés vers un miroir, le plus souvent convexe) ou fisheye. Ces dispositifs permettent d'élargir le champ de vue de manière à percevoir une grande partie de l'environnement avec un capteur unique, ce qui est une propriété intéressante pour la robotique mobile.

D'autres capteurs, comme les caméras RGB-D (Red Green Blue - Depth) qui utilisent une projection de lumière structurée, permettent une mesure de profondeur de la scène. De par leur faible coût et leur grande résolution spatiale, elles ont donné lieu à de nombreux travaux dans la communauté. Cependant, elles souffrent de deux inconvénients majeurs, d'une part leur résolution et leur portée ne sont pas des plus mirobolantes et d'autre part, comme elles utilisent une émission de lumière infrarouge faible puissance sur un large champ de vue, elles sont très perturbées en extérieur.

Une autre famille de capteurs peut s'apparenter aux caméras RGB-D : les caméras temps de vol (Tof : Time of Flight). Elles émettent une impulsion lumineuse, puis chaque pixel mesure le temps de parcours de l'onde de manière à restituer l'aspect 3D d'une scène. La diffusion de ces capteurs reste encore marginale, et leur résolution assez faible. De par leur capacité à mesurer la profondeur de la scène, les méthodes associées aux caméras *RGB-D* et *TOF* s'apparentent plus à celles utilisées pour le *LiDAR*.

1.2.5 LiDAR

Il existe une multitude de types de LiDARs (*Light Detection And Ranging*) permettant d'effectuer différentes mesures : sonder l'atmosphère, mesurer des vitesses, l'intensité du vent... mais celui qui nous intéresse en robotique est le télémètre LiDAR (*laser range finder*). Il permet de mesurer des distances avec des obstacles, et donc d'obtenir l'emplacement des objets environnants.

Son principe de fonctionnement peut être résumé simplement (figure 1.5) : une impulsion laser infra-rouge est émise par une diode, qui va être réfléchie par le premier obstacle rencontré sur sa trajectoire. « L'écho » résultant parcourt le trajet inverse avant d'être détecté par le récepteur. La distance avec l'obstacle est directement proportionnelle au « temps de vol » de l'onde lumineuse selon la formule : $Distance = \frac{c}{2}(t_{reception} - t_{emission})$ (avec c : la célérité de la lumière).

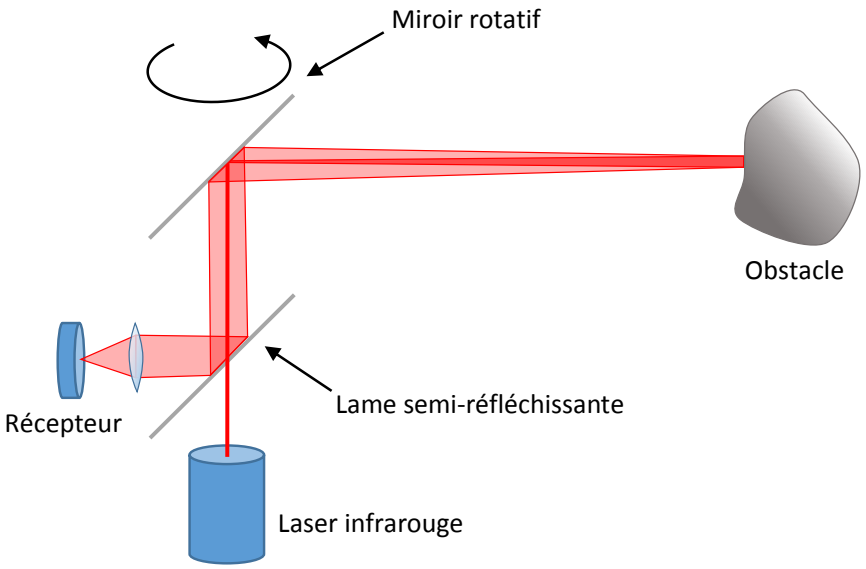


FIGURE 1.5 – Synoptique simplifié du fonctionnement d'un télémètre LiDAR (*LiDAR rangefinder*)

La plupart des LiDARs utilisés en robotique sont équipés d'un miroir tournant permettant au faisceau laser de balayer un plan, que l'on appelle *nappe*. Un codeur est utilisé pour obtenir l'angle correspondant à chaque tir laser. Avec l'angle et la distance mesurée, nous pouvons reconstruire directement des points 3D dans le référentiel du LiDAR par une transformation du repère sphérique vers un repère cartésien (équation 1.7).

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} cos(\theta) \\ sin(\theta) \\ Z/Distance \end{bmatrix} * Distance \tag{1.7}$$

La vitesse de rotation correspond à la fréquence de rafraîchissement des *scans* (un tour complet du miroir). L'angle du miroir entre deux impacts correspond à la résolution angulaire.

En plus de mesurer la distance, la plupart des LiDARs informent également sur l'intensité lumineuse de l'écho. Cela permet d'obtenir une idée de la réflectance[3] de la matière sur laquelle le faisceau a été réfléchi (figure 1.6). Des méthodes (cf section 1.2.5.1)

3. La réflectance, également nommée facteur de réflexion, est la proportion de lumière réfléchie par la surface d'un matériau.

utilisent directement cette réflectance pour calculer la vraisemblance de la mesure par rapport à une carte. C'est également une bonne solution pour extraire les marquages au sol du réseau routier (Yang et collab. [2012],Takahashi et collab. [2014], Kumar et collab. [2014]).

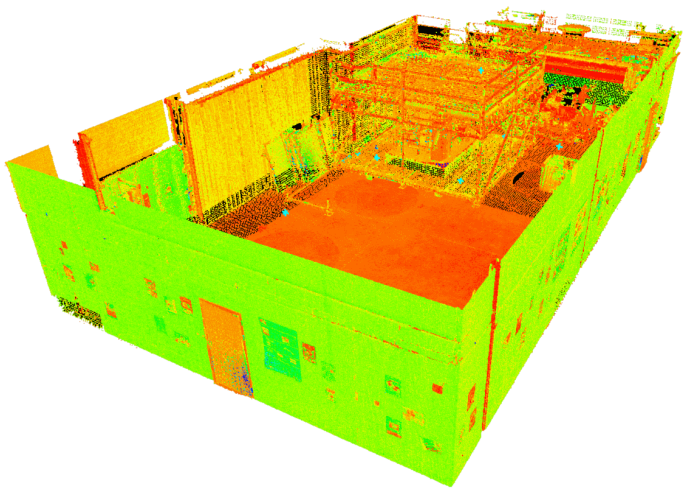


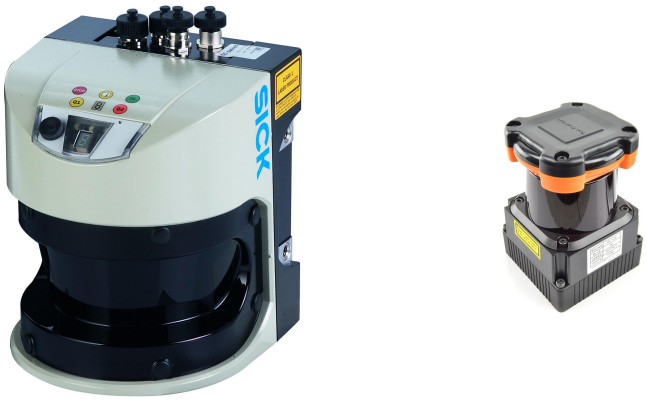
FIGURE 1.6 – Numérisation LiDAR du laboratoire de navigation autonome de l'IRSEEM, les couleurs du vert au rouge correspondent à l'échelle de l'intensité lumineuse reçue

Nous avons souvent tendance à imaginer les faisceaux laser vraiment ponctuels. Dans le cas des LiDARs, ils ne sont pas parfaitement collimatés. Sur un Sick LMS511 (figure 1.7a), il forme un angle de 4.7mrad, soit un spot mesurant 19cm de diamètre à 40m de distance. Supposons que ce spot impacte un tube de 10cm de diamètre, une partie du faisceau pourra continuer à se propager en aval du premier obstacle. S'il rencontre un nouvel objet, il y aura deux échos retournés vers le récepteur.

Partant de ce principe, les fabricants ont développé des LiDARs dis « multi-échos ». La principale utilisation consiste à réduire les perturbations de types pluie ou embruns, par exemple en filtrant sur l'écho retournant la plus forte intensité. Mais sur certains équipements, il est également possible de récupérer tous les échos (LMS511, figure 1.7a), ce qui peut enrichir la perception de l'environnement en milieu complexe. Par exemple, Reymann et Lacroix [2015] classifient les matériaux de la scène grâce aux multi-échos et à la réflectance.

Afin de pouvoir percevoir l'environnement en 3D et non uniquement sur un plan, certains chercheurs ont mis au point des systèmes pour faire osciller des LiDARs mononappes. Dans le cas de mouvements relativement lents, ces solutions peuvent être très performantes comme le montrent les travaux de Zhang et Singh [2014]. Roberge [2013] présente les avantages et inconvénients des différents choix d'axe de balayage ainsi qu'une méthode de calibration de la géométrie du système. Il existe des solutions encore plus surprenantes, comme Ryde et Hu [2008] qui utilisent un miroir pivotant pour dévier la nappe du LiDAR. Les scans se retrouvent par conséquent fortement déformés. Nous pouvons également citer Bosse et collab. [2012] qui ont installé un LiDAR et une centrale inertielle à l'extrémité d'un ressort, les mouvements du mobile porteur assurant le balayage de l'environnement.

Des solutions multi-nappes intégrées existent également. Techniquement elles consistent en plusieurs émetteurs/récepteurs assemblés sur le même axe de rotation d'azimut, mais décalés en élévation. Les plus connus dans le domaine de la recherche sont ceux de la société *Velodyne* (figure 1.7b). A noter que le dernier modèle, le VLP-16, dispose



(a) LiDARs mono-nappe : à gauche Sick LMS511 80m 5 échos 190°, à droite Hokuyo UTM30-LX 30m 1 écho 270°

(b) LiDARs multi-nappes Velodyne 360˚de gauche à droite : VPL-16, HDL-32 et HDL-64, respectivement 16, 32 et 64 nappes.

FIGURE 1.7 – Exemple de LiDARs courants en robotique autonome

de 16 nappes 2 échos sur 360˚en azimut et -15˚et 15˚en élévation pour le même ordre de grandeur de coût qu'un Sick LMS511 mono-nappe.

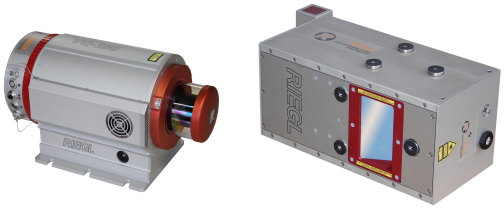
Malgré une forte réduction des coûts et de l'encombrement, ces LiDARs multi-nappes restent peu adaptés au domaine du véhicule autonome. En effet, les considérations de design sont assez peu compatibles avec un « champignon » sur le toit ! Les constructeurs s'orientent actuellement vers l'intégration de plusieurs LiDARs plus simples dans les boucliers avant et arrière pour couvrir l'ensemble du champ de vue autour du véhicule. Par exemple, la société *Ibeo Automotive*[4] propose des LiDARs quatre nappes qui devraient coûter autour de 200€ pièce courant 2016.

D'autres LiDARs télémètres, de par leur fréquence de rafraîchissement rapide et leur portée importante comme les *Riegl* (figure 1.8b, VQ-450 : 200scans/s, 800m de portée), sont plutôt destinés à la numérisation d'environnements pour en faire des cartes que de la localisation à proprement parler. Les LiDARs de type *Leica ScanStation C10* (figure 1.8b) sont composés d'un télémètre mono-point et d'une caméra motorisés sur deux axes. Ils sont également destinés à la numérisation d'environnement, mais pas de manière embarquée, plutôt dans le domaine des géomètres. La caméra permet d'obtenir des nuages de points colorisés.

Malgré l'utilisation de LiDAR multi-nappes 3D, l'information recueillie sur l'environnement est nettement moins riche qu'avec des caméras. Cependant, ils permettent d'obtenir directement des points 3D et non juste leurs projections sur un plan, les temps de calcul des algorithmes sont donc souvent plus faibles. De plus, la fiabilité des mesures obtenues et la fréquence élevée de rafraîchissement en font un capteur performant pour la localisation dans les domaines de la robotique.

4. http://www.ibeo-as.com/start.html

(a) Leica ScanStation C10, numérisation pour les géomètres



(b) LiDAR Riegl : à gauche VQ-250 destiné à être embarqué sur véhicule routier, à droite LMS-Q680i aéroporté

FIGURE 1.8 – LiDARs destinés à la numérisation d'environnement

1.2.5.1 Méthodes de localisation basées LiDAR

De nombreuses années de recherche sur la localisation LiDAR ont démontré la maturité de plusieurs méthodes. Cependant trois difficultés restent complexes à contourner :

— Les milieux intérieurs comportent souvent des symétries et une relative monotonie, les algorithmes rencontrent alors ce que nous appelons le *perceptual aliasing*. C'est-à-dire que du point du vue du capteur, plusieurs endroits de l'environnement semblent similaires. La perception n'est pas assez riche, et les mesures semblent équivalentes, ce qui fait qu'il est impossible de distinguer catégoriquement les ambiguïtés.

— Les environnements réels ne sont que rarement statiques. Deux causes principales peuvent amener à ce que la carte ne soit pas à jour : la géométrie du milieu est modifiée faiblement, porte ouverte ou fermée, ou de manière plus importante par exemple dans un entrepôt avec déplacement d'un ensemble de palettes ou encore le cas d'un parking. La deuxième cause est tout ce qui est dénommé sous le terme *obstacles dynamiques*. Par définition, ils ne sont pas présents dans la carte, et vont venir perturber la corrélation que nous essayons de réaliser entre la carte et la mesure de l'environnement réel.

— La gestion des environnements vastes sans connaissance *a priori* sur la position reste un problème. Comme nous le verrons avec plus de détails dans la section 1.4, cela pose deux problèmes principaux : d'une part, la représentation de l'environnement doit suffisamment compresser l'information pour des raisons de stockage tout en étant assez discriminatif. Et d'autre part, la capacité à tester beaucoup d'hypothèses pour explorer un espace important reste très coûteuse en temps de calcul.

Dans cette section, nous allons donc lister les méthodes utilisant le LiDAR pour se localiser. Comme nous le mettrons en évidence dans la section 1.3 et surtout 1.4, les notions d'appariement et de fonction de vraisemblance sont essentielles à ces algorithmes. C'est pourquoi, nous commencerons par une énumération des différentes méthodes existantes pour corréler un *scan* LiDAR avec une carte. Le capteur nous fournit une vision partielle de l'environnement et le problème de la localisation consiste à mettre en correspondance cette observation avec l'environnement complet que représente la carte.

Dans le cadre de la localisation métrique, ces méthodes peuvent travailler selon deux grands principes :

— en extrayant des amers (coins, lignes, plans, objets ...) dans les scans, avant toutes autres opérations ;

— directement sur le nuage de points brut mesuré par le LiDAR sans transformation préalable.

Les méthodes utilisant les *scans* bruts sont plus riches en terme d'information, mais généralement plus gourmandes en temps de calcul. Le fait d'extraire des amers permet une « certaine compression » de l'information, et donc une diminution des ressources nécessaires, mais elles imposent forcément de faire des hypothèses sur le contenu de l'environnement. Si nous utilisons des cercles pour amers, comme les poteaux de lampadaires, une hypothèse est prise sur le contenu de l'environnement. En milieu naturel, l'algorithme ne pourra pas fonctionner.

La méthode est fortement liée à la représentation choisie pour la carte. Plus exactement, il faut trouver une représentation de la carte en adéquation avec la méthode choisie, de manière à optimiser les performances. Du fait de la quantité d'information à traiter, une carte d'occupation sera efficace pour y projeter un nuage de points, alors qu'une liste d'objets pourra convenir pour des amers plus disparates.

1.2.5.2 La fonction de vraisemblance

La fonction de vraisemblance va permettre de répondre à la question « Est-ce que pour une position précise la pièce de puzzle s'emboîte correctement dans la carte ? ». Elle consiste à déterminer la similarité entre une mesure et la carte ou entre deux mesures. En d'autres mots, comment traiter les données LiDAR pour être capable de comparer deux scans ? C'est elle qui permet d'évaluer la probabilité 1.23 de l'étape *measurement update* des filtres de la section 1.3, c'est donc le cœur de la perception.

Il existe de nombreuses solutions pour réaliser cet appariement qui dépendent des méthodes choisies. Le paragraphe qui suit en décrit les principales, regroupées en deux catégories : méthodes basées sur l'extraction d'amers et méthodes exploitant les données brutes du LiDAR.

Méthodes basées sur l'extraction d'amers dans les scans.

Amers artificiels Comme en localisation basée vision, certains travaux utilisent des amers artificiels constitués par des balises aux formes géométriques connues et placées au préalable dans la scène. La plupart du temps, ces balises prennent la forme de cylindres réfléchissants, disposés dans l'environnement à hauteur du plan de la nappe LiDAR.

Cette méthode est très courante en milieu industriel pour la localisation des AGV (*Automatic Guided Vehicle*) et comporte au moins deux avantages. Le fait d'utiliser des amers artificiels garantit leur invariance dans le temps. Dans le cadre d'une localisation dans un entrepôt, du fait du déplacement des marchandises, l'environnement est modifié continuellement. L'extraction d'amers « naturels », tels que des coins ou des lignes peut alors poser problème. D'autre part, cette approche permet de développer des « LiDARs simplifiés », donc moins coûteux, ne mesurant pas la distance mais juste la réflexion et l'angle. Loevsky et Shimshoni [2010] montrent l'application d'une telle localisation.

Ronzoni et collab. [2011] proposent une localisation d'*AGV* toujours avec des amers artificiels, mais en mesurant cette fois l'angle et la distance. Le fait d'utiliser la distance en plus de l'angle permet de minimiser les erreurs d'appariement entre les amers et la carte et donc d'améliorer la précision. Dans le cadre

de surfaces importantes, le nombre de balises peut atteindre plusieurs centaines (cf figure 1.9).

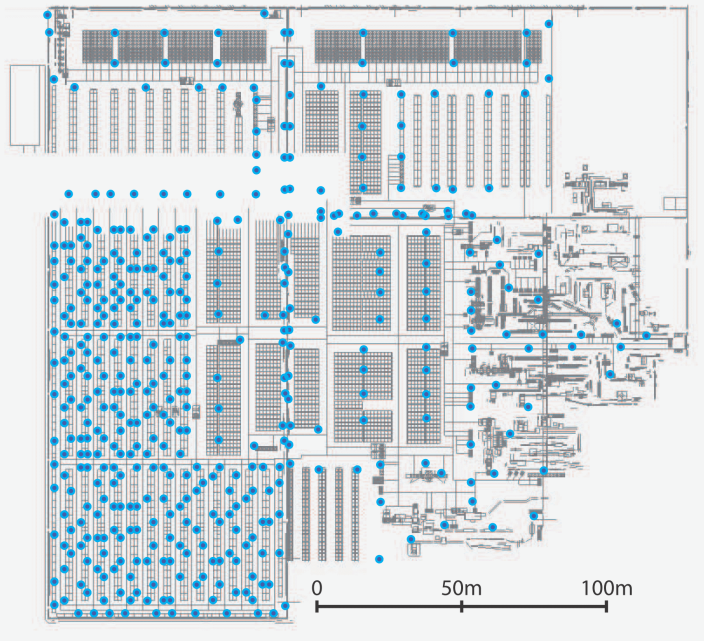


FIGURE 1.9 – Ronzoni et collab. [2011] : Carte d'un entrepôt industriel, couvert avec 450 balises artificielles

Il faut souligner que ces solutions basées amers artificiels en milieu industriel sont bien maîtrisées car proposées par des industriels comme *Sick* avec leur solution *NAV 350*[5].

Bien sûr, les méthodes basées amers artificiels ne sont qu'un petit sous-ensemble des recherches autour de la localisation basée LiDAR essentiellement dans le but de répondre à des contraintes fortes de fiabilité et robustesse des milieux industriels.

Amers naturels Pour s'affranchir des balises artificielles, de nombreuses méthodes ont été proposées pour extraire des amers naturels d'un environnement. En voici une liste non exhaustive :

— Transformée de *Hough* généralisée (Duda et Hart [1972]) permettant de détecter des lignes et des courbes.

— Utilisation des nuages de points issus de données LiDAR, surtout mononappe, ne sont pas très denses comparés à une image. Pour extraire des lignes ou des cercles, l'algorithme de *Ransac* (Ramdon Sample Consensus : Fischler et Bolles [1981]) est souvent préféré pour son efficacité en temps de calcul par rapport à la transformée de Hough.

— Un peu à la manière des points d'intérêts utilisés en traitement d'image, Tipaldi et Arras [2010] proposent un détecteur d'amer basé sur la variation de courbure du scan pour en extraire les discontinuités donc les coins de l'environnement (*FLIRT* : Fast Laser Interest Point Transform). Cette détection est associée à un descripteur utilisant une grille d'occupation dans

5. Sick NAV 350 https://www.sick.com/media/dox/3/43/143/Operating_instructions_NAV350_Laser_Positioning_Sensor_en_IM0040143.PDF

un référentiel polaire. Bosse et Zlot [2009] ont évalué 3 détecteurs descripteurs similaires adaptés aux scans 2D.

Il est intéressant de noter que les LiDARs automobiles embarquent de plus en plus des fonctions de labellisation d'objet. Zindler et collab. [2014] exploitent cette caractéristique et s'affranchissent donc de la phase d'extraction et d'appariement d'amers en utilisant directement les objets labellisés par un LiDAR multi-nappes de type *Ibeo Lux* pour localiser un véhicule.

Méthodes basées sur les données brutes sans traitement préalable.

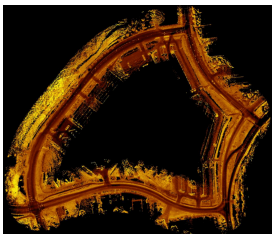
Appariement de scans (*scan registration*) L'ICP (*Iterative Closest Point* : Besl et McKay [1992]) est une des méthodes les plus connues pour retrouver la transformation géométrique (rotation et translation) entre deux nuages de points. L'algorithme peut succinctement se résumer en 3 étapes :

1. Choix d'une transformation initiale.
2. Détermination de la distance entre les deux nuages de points en calculant le carré des erreurs de la distance aux plus proches voisins.
3. Optimisation de la transformation avec ce critère de coût.

Les deux dernières étapes sont répétées jusqu'à convergence (figure 1.13). Il existe de nombreuses versions de chaque étape énumérées par Pomerleau et collab. [2015]. Bien que très efficace, la méthode est très coûteuse en temps de calcul, particulièrement la mesure des plus proches voisins entre les deux nuages de points. La méthode *NDT* (Normal Distributions Transform : Biber et StraBer [2003]) propose de regrouper les points sous forme de gaussiennes de 2 ou 3 dimensions, ce qui réduit grandement le coût de comparaison. Magnusson et collab. [2009] ont réalisé un comparatif des performances ICP versus NDT.

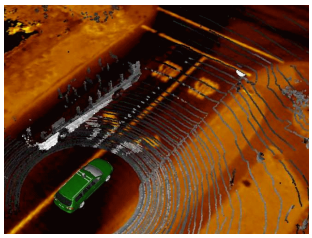
Lancer de Rayon Une autre idée consiste à « simuler » les « impacts du LiDAR » pour une pose donnée par du lancer de rayon (*raytracing*) dans la carte contenant un modèle 3D de l'environnement. Cela permet de déterminer la mesure que nous aurions dû obtenir pour ladite pose. L'erreur entre les distances d'impacts obtenues en *raytracing* et la mesure du vrai capteur sert de résultat à la fonction vraisemblance. Cette méthode bien qu'intuitive est très coûteuse en temps de calcul. Le *raytracing* impose que tous les points de passage du rayon laser entre le LiDAR et le premier obstacle dans l'environnement soient testés. Fallon et collab. [2012] emploient cette méthode en 3D en utilisant un capteur RGB-D. Sa carte contient les plans de l'environnement supérieurs à $1m^2$. Pour des raisons de temps de calcul, le nombre d'impacts est limité à 20×15.

Réflectance LiDAR Certaines méthodes associent les positions dans l'espace des impacts LiDAR et leur taux de réflectance. Par exemple, Levinson et Thrun [2010] constituent une carte 2D de la réflectance de la chaussée (figure 1.10a). Puis, ils localisent un véhicule équipé d'un LiDAR 3D de type Velodyne (cf figure 1.7b). Les impacts du LiDAR sont projetés sur le plan de la route contenu dans la carte (figure 1.10c), puis la réflectance est utilisée pour l'évaluation par rapport aux données mesurées. La carte est de type probabiliste, la moyenne et la variance y sont stockées. Sur certaines parties de la route, la réflectance ne peut pas être considérée comme invariante ; par exemple les traces où les pneumatiques marquent l'enrobé (figure 1.10b). Lors de l'évaluation de la mesure LiDAR, la variance contenue dans la carte permet d'affecter plus ou moins



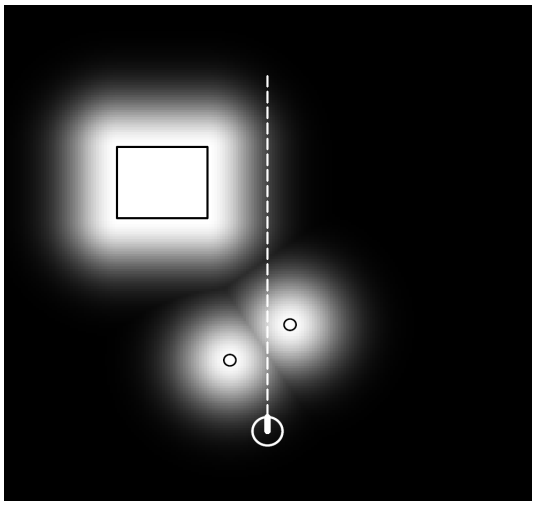
(a) Carte de réflectance servant à la localisation, cellules de $15 \times 15cm$

(b) La variance de la carte met en évidence les traces de véhicules comme des zones de faible confiance

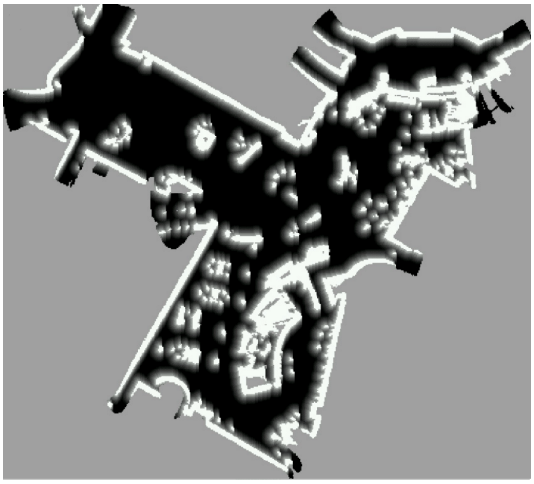


(c) Nappes LiDAR projetées sur la carte, pour la comparaison de la réflectance

FIGURE 1.10 – Localisation basée sur la réflectance LiDAR (issue des travaux de Levinson et Thrun [2010])



(a) Illustration du principe du champ de vraisemblance sur 3 obstacles



(b) Champ de vraisemblance du musée de San Jose Tech

FIGURE 1.11 – Exemple de champs de vraisemblance (figures issues du livre Thrun et collab. [2005])

de poids aux différents impacts. Il est à noter que cette méthode nécessite un calibrage de la réflectance en fonction des différents récepteurs du LIDAR et de l'incidence d'impact du rayon avec la surface.

Champ de vraisemblance (*Likelihood field*) Le champ de vraisemblance est une fonction permettant de connaître la probabilité d'un impact LiDAR pour un point donné (Thrun [2001]). Nous pouvons le voir comme une carte contenant les probabilités pré-calculées des différents impacts LiDARs. Comme l'illustre la figure 1.11a, sur un obstacle la probabilité d'avoir un impact est maximum, et plus nous nous en éloignons et plus cette probabilité devient faible, jusqu'à devenir nulle. Par rapport au lancer de rayon, cette méthode est beaucoup plus efficiente en temps de calcul. Au lieu de tester l'environnement sur l'ensemble de la trajectoire du rayon, il suffira de lire la valeur de la probabilité de la carte directement à la position de l'impact. Les impacts d'un même *scan* LiDAR sont combinés entre eux par le produit des différentes probabilités.

Cette méthode sera étudiée avec de plus amples détails dans le chapitre 3, où nous avons proposé une extension de ce concept en 3 dimensions.

1.2.5.3 Application de la localisation LiDAR

Comme nous l'avons entrevu à travers l'étude de la fonction de vraisemblance, les applications de localisation LiDAR sont innombrables. Nous allons donc lister quelques méthodes dans cette section. Attention, nous parlons bien ici de localisation, c'est-à-dire avec une connaissance *a priori* de l'environnement et non de *SLAM* (Simultaneous Localization And Mapping). Dans le cas du SLAM, la carte est construite à fur et à mesure de la découverte de l'environnement par le mobile, et comme son nom l'indique la localisation se déroule simultanément avec la connaissance actuelle de la carte. Ceci dit, les méthodes de *SLAM* en ligne effectuent bien la fonction de localisation et peuvent donc également servir d'inspiration.

De manière générale, les milieux intérieurs (*indoor*) sont plus structurés que les milieux extérieurs (*outdoor*). Ils seront donc généralement plus adaptés à une méthode basée sur l'extraction d'amers, particulièrement pour les formes géométriques (coins, lignes, cercles...). En même temps, une carte d'occupation permettant de représenter des structures plus complexes ne compresse pas énormément l'information. Dans le cadre d'environnements extérieurs vastes, ce volume de données peut rapidement poser problème. Alsayed et collab. [2015] proposent une solution à ce problème en chargeant dynamiquement des tuiles de la carte en fonction de la position du véhicule.

En localisation comme ailleurs, il n'y a pas de solution idéale, mais que des solutions adaptées à un problème donné. Aujourd'hui aucune méthode ne peut « se vanter » d'être suffisamment « universelle » pour couvrir toutes les situations.

Indoor Comme déjà abordé plus haut, il existe plusieurs solutions dans le cadre d'entrepôts industriels basées sur des amers artificiels. Paluri et collab. [2010]; Reinke et Beinschob [2013] proposent des solutions avec extraction d'amers « naturels » sous forme de lignes et de coins (figure 1.12). Ces méthodes ont largement été employées dans des environnements intérieurs plus classiques, comme des bureaux (Arsénio et Ribeiro [1998]). De nombreux travaux ont également porté sur les grilles d'occupation pour de la localisation en milieu intérieur (El Hamzaoui [2012]).

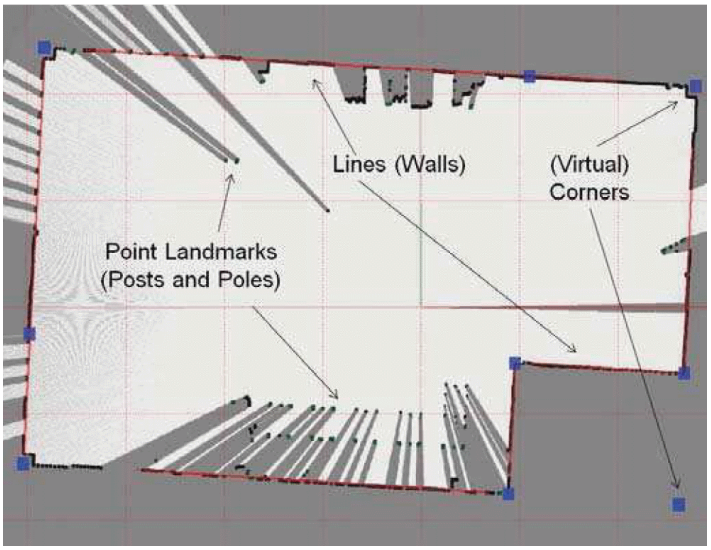


FIGURE 1.12 – Reinke et Beinschob [2013] : exemples d'amers naturels extraits en milieu industriel

La localisation et la cartographie d'anciennes mines posent des problèmes proches du milieu extérieur. Le sol n'étant pas plan, une localisation à 3 degrés de liberté ne

peut plus convenir. Nuchter et collab. [2004]) utilisent une méthode *ICP* pour apparier les scans successifs d'un LiDAR oscillant et reconstituer la trajectoire suivant les 6 degrés de liberté du mobile (figure 1.13).

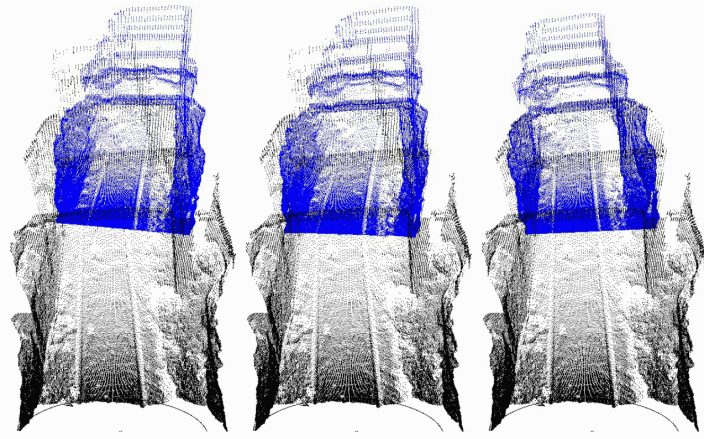


FIGURE 1.13 – Nuchter et collab. [2004] : *ICP* appliqué dans l'exploration de mines désaffectées. A gauche appariement avec l'odométrie, au centre *ICP* après 5 itérations, à droite appariement après alignement final.

Des travaux ont également été menés pour développer des localisations robustes aux cartes succinctement décrites, ce qui peut simplifier grandement l'étape de cartographie. Winterhalter et collab. [2015] proposent une localisation selon 6 degrés de liberté en utilisant les capteurs d'une tablette du projet *Google Tango*[6] et d'une carte construite sur la base d'un plan de masse du bâtiment. Les murs sont simplement extrudés verticalement à partir du plan 2D. Behzadian et collab. [2015] ont même développé une localisation 2D à partir d'une carte dessinée approximativement à la main.

Outdoor La classification *indoor* - *outdoor* n'est pas toujours évidente. Ainsi les mines ou encore les parkings souterrains (Kümmerle et collab. [2009]) présentent des caractéristiques communes aux deux milieux

Kuemmerle et collab. [2011] émettent une idée intéressante, une technique de traitement d'image permet d'extraire les contours d'une photo aérienne. Puis ces contours sont utilisés pour constituer un champ de vraisemblance. Un robot équipé d'un LiDAR mono-nappe est ensuite localisé en utilisant cette carte.

Nous trouvons également des applications dans le milieu agricole. Jagbrant et collab. [2013] utilisent un LiDAR mono-nappe installé verticalement pour obtenir le profil des arbres. Une segmentation de chaque arbre est réalisée, puis un descripteur est calculé sur le profil de l'arbre qui servira d'amer. Cette méthode est utilisée pour la localisation d'un robot dans un verger d'amandiers.

Comme abordé lors de l'étude des fonctions de vraisemblance, Levinson et collab. [2007] utilisent deux LiDARs mono-nappe à l'arrière du véhicule pour corréler une carte de réflectance infrarouge. Dans leur premier article, cette corrélation est assurée par un filtre particulaire (Les méthodes de filtrage pour la fusion de données seront abordées dans la section suivante 1.3, et plus particulièrement la section 1.3.4.3 pour le fitre particulaire). Dans un deuxième temps, Levinson et Thrun [2010] utilisent un LiDAR 3D de type Velodyne (cf figure 1.7b) et un filtre d'histogramme local (section 1.3.4.2) pour la corrélation.

6. https://www.google.com/atap/project-tango/

La route pouvant être considérée comme plane, ces deux dernières localisations se déroulent dans le plan donc selon 3 degrés de liberté. Quand les surfaces rencontrées ne sont plus aussi planes, comme un environnement naturel, les grilles d'occupations codant la hauteur du terrain (*level surface*) sont couramment utilisées. Nous parlons alors de localisation 2.5D (Herbert et collab. [1989]). En combinant plusieurs cartes de niveau les unes sur les autres (*multi surface level*), Pfaff et collab. [2007] ont pu gérer des environnements extérieurs et intérieurs. La carte du terrain est utilisée pour déplacer le robot dans les 3 dimensions, puis la corrélation de la mesure est faite en cherchant le point le plus proche sur la carte, un peu à la manière des champs de vraisemblance (figure 1.14).

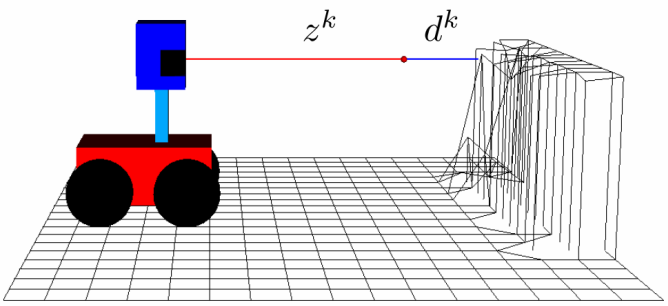


FIGURE 1.14 – Pfaff et collab. [2007] : Modèle *Endpoint* utilisé avec les *multi surface level* pour la localisation Indoor et Outdoor

Les milieux industriels présentés ici sont plutôt structurés, usines, entrepôts,... La localisation dans ces environnements est généralement traitée par l'extraction d'amers, coins ou lignes. Dans le cadre d'entreprises pétro-chimiques, le milieu est constitué d'une multitude de tubes, cuves, tuyaux, chemins et passerelles sur différents niveaux,... C'est dans ce cadre que nous avons développé une extension du champ de vraisemblance 3D. Les difficultés que présente un tel environnement, ainsi que les méthodes employées seront détaillées dans le chapitre 3.

1.3 Filtrage probabiliste pour la localisation

Nous avons vu dans la section précédente que chaque capteur a ses avantages et inconvénients mais à part le système GPS, aucun d'eux ne fournit directement l'information de localisation. Ils mesurent soit une information sur le robot lui-même (capteurs proprioceptifs) soit sur l'environnement qui les entoure (capteurs extéroceptifs). Il faut donc utiliser des méthodes (sections 1.2.4.1, 1.2.4.2 et 1.2.5.1) pour déterminer une pose. Ces algorithmes s'appuient sur des méthodes ensemblistes (Guyonneau [2013]) ou plus souvent sur le filtrage probabiliste que nous allons détailler dans cette section. Elle est écrite à partir de l'ouvrage Thrun et collab. [2005].

1.3.1 Le robot et son interaction avec l'environnement

L'objectif de la localisation est de déterminer la pose du robot. Cette pose est définie à travers un vecteur d'état pour chaque pas de temps. Par exemple dans le cas d'un robot se déplaçant sur un plan, comme celui de la figure 1.2a, le vecteur d'état comprend 3 dimensions (équation 1.8)

$$\mathbf{X_t} = \begin{bmatrix} x & y & \theta \end{bmatrix}^{\mathrm{T}} \tag{1.8}$$

Où :

x, y : Position en mètre (m)
θ : Orientation en degré (°)

Le but d'un algorithme de localisation consiste à estimer ce vecteur d'état. Pour cela, il a accès aux commandes de déplacement du robot $\mathbf{u_t}$, aux mesures des capteurs $\mathbf{Z_t}$ et à une carte $\mathcal{M}$.

Dans l'exemple simpliste de la figure 1.15, le vecteur d'état ne comporte qu'une seule dimension : $\mathbf{X_t} = \begin{bmatrix} x \end{bmatrix}$. Les commandes (qui peuvent aussi être issues de l'odométrie) ne peuvent prendre que trois valeurs : $\mathbf{u_t} = 0$, le robot reste sur place ; $\mathbf{u_t} = 1$, le robot avance d'une case et inversement $\mathbf{u_t} = -1$, le robot recule d'une case. Le vecteur de mesures ne peut quant à lui prendre que deux valeurs : $\mathbf{Z_t} = 0$, la case est lue de couleur noire et $\mathbf{Z_t} = 1$, la case est lue de couleur blanche.

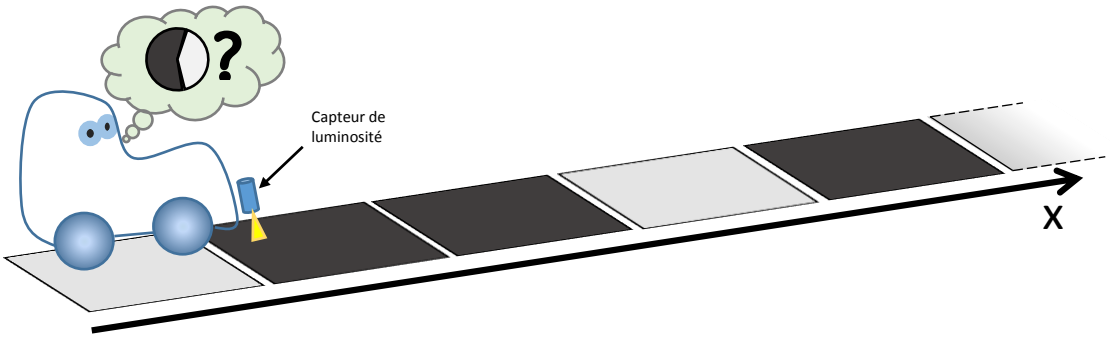


FIGURE 1.15 – Robot 1D, équipé d'un capteur de luminosité et se déplaçant sur un damier noir et blanc

1.3.2 Notions de probabilité

Avant d'aborder à proprement parler la localisation probabiliste, plantons le décor avec quelques notions simples de probabilités. Chaque dimension du vecteur d'état est représentée par une variable aléatoire, qui peut prendre des valeurs de manière à couvrir l'ensemble de l'environnement sur la dimension concernée. Par exemple dans le vecteur 1.8, l'orientation θ peut varier de $]-\pi, \pi]$ de façon à prendre toutes les orientations possibles.

Soit Θ une variable aléatoire représentant l'orientation du robot et θ une hypothèse d'orientation possible ; alors $P(\Theta = \theta)$ désigne la probabilité que Θ est pour valeur θ, c'est-à-dire la probabilité que l'orientation du robot soit de θ. Comme le robot n'a qu'une orientation à la fois, mais forcément une :

$$\sum_{\forall \theta \in]-\pi,\pi]} P(\Theta = \theta) = 1 \tag{1.9}$$

Cela revient également à dire que l'orientation du robot a forcément 100% de chances de valoir entre $-\pi$ et π. Nous pouvons représenter les variables aléatoires par leur fonction de densité de probabilité (PDF : Probability Density Function) $P(\theta)$, qui fournit une probabilité pour toute valeur de Θ.

De nombreux phénomènes physiques peuvent être approximés par une densité de probabilité gaussienne, appelée loi normale : $X \sim \mathcal{N}(\mu, \sigma^2)$ (figure 1.16 et équation 1.10).

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{1.10}$$

μ correspond à la moyenne et σ^2 à la variance, qui signifie son étalement. 68% des valeurs se trouvent dans l'intervalle $[\mu - \sigma, \mu + \sigma]$, et 99.7% dans l'intervalle $[\mu - 3\sigma, \mu + 3\sigma]$.

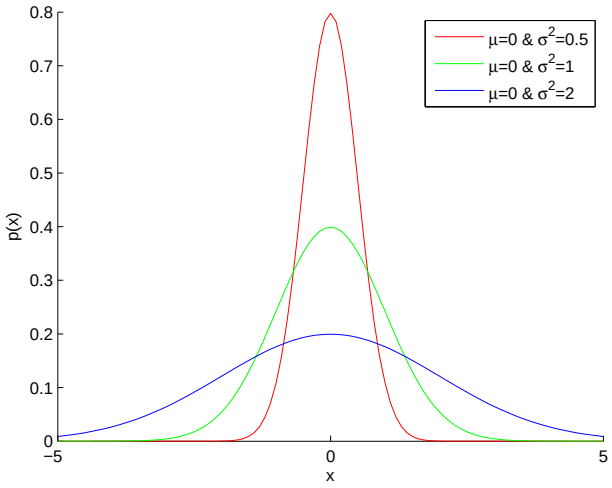


FIGURE 1.16 – Loi normale de moyenne nulle ($\mu = 0$), pour différentes variances ($\sigma^2 \in [0.5, 1, 2]$)

La probabilité jointe entre deux variables permet d'exprimer que deux événements se produisent, et se note :

$$P(x, y) = P(X = x \; et \; Y = y) \tag{1.11}$$

Si les deux variables ne s'influencent pas mutuellement, nous disons qu'elles sont indépendantes, et la probabilité jointe peut s'exprimer :

$$P(x, y) = P(x)P(y) \tag{1.12}$$

Si au contraire, des variables aléatoires conditionnent l'état d'autres variables, nous parlons de probabilité conditionnelle :

$$P(x \mid y) = \frac{P(x, y)}{P(y)} \tag{1.13}$$

Qui s'exprime :

$$P(x \mid y) = P(X = x \mid Y = y) \tag{1.14}$$

Cette relation 1.13 peut s'expliquer comme $P(x)$ sachant que l'événement $P(y)$ s'est produit, donc $P(x \mid y)$ multiplié par $P(y)$ est égal à $P(x, y)$ (soit : $P(x \mid y)P(y) = P(x, y)$).

Et si X et Y étaient indépendantes, la connaissance de l'événement $P(Y = y)$ n'apporterait aucune information sur la probabilité de l'événement $P(X = x)$:

$$P(x \mid y) = \frac{P(x)P(y)}{P(y)} = P(x) \tag{1.15}$$

1.3.3 Principe de fonctionnement du filtrage

Pour rappel, le but de la localisation est d'identifier le vecteur d'état $\mathbf{X_t}$ pour déterminer la pose du mobile considéré. Il existe différents filtres probabilistes qui permettent d'obtenir cette information $\mathbf{X_t}$. Ils sont tous composés de deux étapes principales qui se répètent :

— Prédiction ou plus communément appelée en robotique *motion update*
A l'aide d'un modèle du robot, des commandes ou mesures proprioceptives $\mathbf{u_t}$ et de la pose actuelle $\mathbf{X_{t1}}$, nous faisons une prédiction de la position du robot afin de déterminer un nouveau vecteur d'état $\overline{\mathbf{X}}_{\mathbf{t2}}$.
Par exemple dans le cas simpliste du robot 1D figure 1.15, le robot ne pouvant qu'avancer ou reculer sur une ligne d'un déplacement d'une case, le modèle résultant est plutôt facile à mettre en place. Comme son nom l'indique, cette étape va permettre de déplacer l'hypothèse $\mathbf{X_t}$ en fonction du déplacement du mobile.

— Correction, *measurement update* en robotique
Dans cette étape, la cohérence de la mesure obtenue $\mathbf{Z_t}$ pour la position $\mathbf{X_t}$ est vérifiée dans la carte $\mathcal{M}$. C'est le rôle de la fonction de vraisemblance (section 1.2.5.2).
Toujours dans l'exemple de la figure 1.15, la carte $\mathcal{M}$ correspond à la connaissance du damier (case 1 blanche, case 2 noire, case 3 noire, ...), et la mesure $\mathbf{Z_t}$ au retour du capteur de luminosité (noir ou blanc). L'étape *measurement update* vérifie que la position prédite dans l'étape *motion update* est conforme à la valeur mesurée par le capteur.

Ces deux étapes sont répétées tant qu'il y a des déplacements et des mesures. Le cumul de plusieurs mesures différentes pour une succession de poses permet d'identifier la séquence la plus probable contenue dans la carte.

Exemple :
Le robot est sur la case 1 $\mathbf{X_{t1}} = 1$, il se déplace en avant d'une case $\mathbf{u_{t1}} = 1$, nous prédisons alors qu'à l'instant suivant le robot devrait se trouver sur la case 2 $\overline{\mathbf{X}}_{\mathbf{t2}} = 2$ qui est noire. Nous vérifions alors que le capteur retourne bien la valeur noire $\mathbf{Z_{t2}} = 0$. Si c'est le cas, l'hypothèse $\mathbf{X_{t2}} = 2$ est renforcée, c'est-à-dire que sa probabilité $P(\mathbf{x_{t2}})$ augmente. Si la mesure est blanche, $P(\mathbf{x_{t2}})$ diminue. Il peut y avoir plusieurs causes possibles :

— la mesure $\mathbf{Z_{t2}} = 0$ est erronée
— la carte $\mathcal{M}_{\text{case2}}$ s'est avérée fausse
— le déplacement $\mathbf{u_{t1}} = 1$ s'est mal exécuté
— la position initiale $\mathbf{X_{t1}} = 1$ était déjà erronée

Le rôle des méthodes de filtrages abordées plus bas consiste à prendre en compte ces différentes erreurs possibles, pour calculer la meilleure estimation de $\hat{X}_t$. Évidemment, si vous avez pris un « capteur à moitié aveugle », une carte fausse, installé des roues lisses sur le robot pour qu'il patine tout le temps et en plus vous ne savez d'où vous partez... les méthodes ne vont pas faire des miracles !

Ceci dit, il est toujours surprenant de constater comment, dans des cas concrets, ces filtres donnent des résultats satisfaisants avec des données qui pourtant paraissaient peu intelligibles pour un être humain.

1.3.3.1 Lien avec les probabilités

En terme de probabilité, pour l'étape de *motion update* le théorème des probabilités totales est utilisé :

$$P(x) = \sum_{\forall y} P(x|y)P(y) \quad \text{(cas discret)} \tag{1.16}$$

$$P(x) = \int_{\forall y} P(x|y)P(y)dy \quad \text{(cas continu)} \tag{1.17}$$

Avec des termes plus adaptés à la localisation, nous pouvons écrire :

$$P(x_{t2}) = \sum_{\forall x_{t1}} P(x_{t2}|x_{t1})P(x_{t1}) \quad \text{(cas discret)} \tag{1.18}$$

$$P(x_{t2}) = \int_{\forall x_{t1}} P(x_{t2}|x_{t1})P(x_{t1})dx_{t1} \quad \text{(cas continu)} \tag{1.19}$$

$P(x_{t1})$ représente la probabilité d'être bien en x à l'instant $t1$.
$P(x_{t2}|x_{t1})$ représente la probabilité de transition de l'état X_{t1} vers X_{t2} que l'on aurait pu écrire $p(u_t)$, soit la probabilité que le déplacement se soit bien passé.

En faisant apparaître la probabilité de transition, la formule des probabilités totales peut encore être écrite sous la forme :

$$P(x_{t2}) = \sum_{\forall (x_{t1}, u_{t1}) \rightsquigarrow x_{t2}} P(u_t)P(x_{t1}) \tag{1.20}$$

La probabilité $P(x_{t2})$ de la position prédite $\hat{X}_{t2}$ est donc la somme de la probabilité de transition que multiplie la probabilité que la position précédente soit correcte ; et ce pour tous les couples de positions précédentes et déplacements qui peuvent mener en position X_{t2}.

Toujours avec notre exemple simpliste (figure 1.15) : après un déplacement d'une case en avant, si nous voulons estimer la probabilité de se retrouver sur la case 2 ; il faut prendre la probabilité de se trouver actuellement sur la case 1 que l'on multiplie par la probabilité que le déplacement en avant se déroule correctement. Il faut ajouter la probabilité de se trouver déjà sur la case 2 que multiplie la probabilité que le déplacement ne se fasse pas. Et à cela il ne faut pas oublier d'ajouter la probabilité de se trouver sur la case 3 que multiplie la probabilité que le déplacement se passe vraiment mal et qu'en fait le robot ait reculé. Dans l'exemple, du fait que la distance de déplacement soit limitée à une case maximum, il n'existe pas d'autre couple (x_{t1}, u_{t1}) pouvant mener à la case 2.

Pour l'étape de *measurement update*, c'est le théorème de Bayes qui sert de base :

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)} = \frac{P(y|x)P(x)}{\int_{\forall x'} P(y|x')P(x')} \quad \text{(cas discret)} \tag{1.21}$$

Nous pourrons noter qu'il découle de la relation 1.13 et que du fait que les probabilités jointes soient commutatives, $(P(x,y) = P(y,x))$.

Avec des termes plus adaptés à la localisation, nous pouvons écrire :

$$P(x_t|z_t) = \frac{P(z_t|x_t)P(x_t)}{P(z_t)} \tag{1.22}$$

Dans l'étape de correction *measurement update*, nous cherchons à déterminer si l'hypothèse émise $P(x_t)$ lors du *motion update* correspond bien à la mesure Z_t ; c'est-à-dire calculer la probabilité de x_t connaissant la mesure obtenue Z_t. Pour déterminer la probabilité après correction (*posterior*) $P(x_t|z_t)$, nous utilisons la probabilité d'être bien en x_t avant correction (*prior*) que l'on multiplie à la probabilité d'obtenir la mesure Z_t pour la

position x_t ($P(z_t|x_t)$). La manière d'estimer $P(z_t|x_t)$ correspond à la fonction de vraisemblance vue à la section 1.2.5.2.

$P(y)$ de l'équation 1.21 peut être vue comme un coefficient de normalisation permettant de respecter que la somme de toutes les probabilités valle 1. Dans la pratique, il est souvent plus facile de déterminer cette somme que de calculer $P(z_t)$. Nous notons donc le coefficient normalisateur η, et la relation de Bayes devient :

$$P(x_t|z_t) = \eta P(z_t|x_t)P(x_t) \tag{1.23}$$

A travers notre exemple simpliste (figure 1.15), l'étape *measurement update* donne : la probabilité estimée de la prédiction $\overline{X}_{t2}$ est multipliée par la probabilité que la mesure corresponde à la carte pour la position X_{t2} ; ce qui nous permet d'obtenir la probabilité corrigée par la mesure de X_{t2}.

Vous noterez que les exemples présentés ci-dessus ne nécessitent pas d'autre information que celle portée par l'état précédent. Nous parlons dans ce cas d'état complet, et de système correspondant à une chaîne de Markov de premier ordre (figure 1.17).

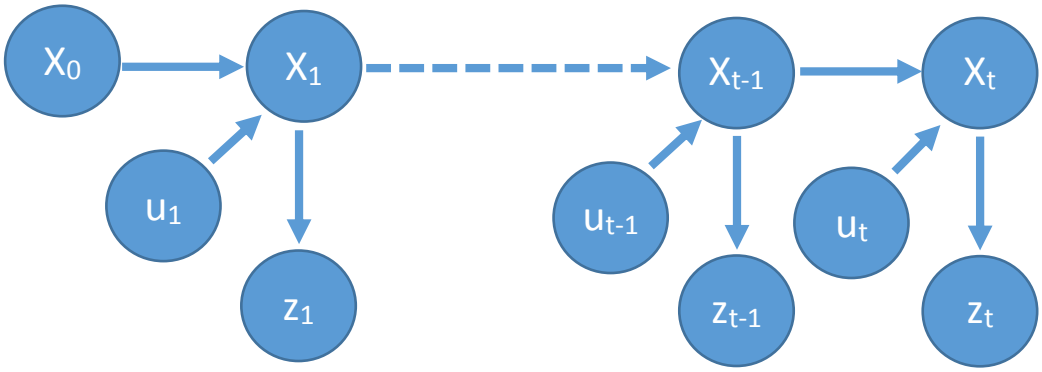


FIGURE 1.17 – Chaîne de Markov de 1^{er} ordre : l'état X_t ne dépend que de X_{t-1} et de la commande u_{t-1}. La mesure z_t ne dépend quant à elle que de l'état x_t

Le filtre aura une connaissance de l'environnement à travers la densité de probabilité de x_t que l'on appelle croyance (*believe* en anglais). La croyance d'un état x_t, $bel(x_t)$ est notée :

$$bel(x_t) = P(x_t|z_{1:t}, u_{1:t}) \tag{1.24}$$

Au moment de la prédiction, avant l'étape de correction et prise en compte de la dernière mesure :

$$\overline{bel}(x_t) = P(x_t|z_{1:t-1}, u_{1:t}) \tag{1.25}$$

1.3.4 Les différents types de filtres

Il existe plusieurs formes d'implémentation des filtres bayésiens qui peuvent être classés en deux catégories :

— les filtres paramétriques, qui représentent la probabilité de l'espace d'état sous forme de gaussiennes, comme le filtre de Kalman.

— Les filtres non paramétriques, qui représentent la probabilité de l'espace d'état sous forme discrète, comme le filtre d'histogramme ou le filtre particulaire.

1.3.4.1 Filtre de Kalman

Le filtre de Kalman (*KF* : Kalman Filter, Kalman [1960]) est l'une des plus vieilles méthodes pour mettre en œuvre les estimations bayésiennes. Il est applicable sous certaines conditions :

— Tout d'abord, le système doit être linéaire. C'est-à-dire que l'étape de prédiction $P(x_t|x_{t-1}, u_t)$ doit pouvoir s'écrire sous la forme suivante :

$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t \tag{1.26}$$

$x_t = \begin{bmatrix} x_{1,t} & x_{2,t} & \dots & x_{n,t} \end{bmatrix}^T$ représente le vecteur d'état de dimension n, et
$u_t = \begin{bmatrix} u_{1,t} & u_{2,t} & \dots & u_{m,t} \end{bmatrix}^T$ le vecteur de commande de dimension m.
A_t est une matrice de dimension $n \times n$ qui décrit la transition de l'état x_{t-1} à l'état suivant x_t. B_t (de dimension $n \times m$) quant à elle décrit l'influence de la commande sur l'état x_t. ε_t est une variable aléatoire gaussienne de dimension n, qui représente le bruit de modèle. En gros, toutes les imperfections de modèle sont modélisées par ce bruit. Il doit être de moyenne nulle.

— L'étape de correction doit aussi pouvoir être représentée linéairement $P(z_t|x_t)$:

$$z_t = C_t x_t + \delta_t \tag{1.27}$$

$z_t = \begin{bmatrix} z_{1,t} & z_{2,t} & \dots & z_{k,t} \end{bmatrix}^T$ est le vecteur de mesure. C_t une matrice de dimension $n \times k$ représentant le modèle de mesure permettant de relier les mesures au vecteur d'état. δ_t est une variable aléatoire gaussienne de dimension k modélisant le bruit de mesure. Ce bruit doit avoir une moyenne nulle.

— Et dernier point pour que le filtre de Kalman soit applicable, la croyance initiale $bel(x_0)$ doit correspondre à une distribution gaussienne. Nous notons μ_0 et Σ_0 la moyenne et la variance initiale :

$$p(x_0) = bel(x_0) = \frac{1}{\sqrt{det(2\pi\Sigma_0)}} e^{-\frac{1}{2}(x_0-\mu_0)^T(\Sigma_0^{-1})(x_0-\mu_0)} \tag{1.28}$$

L'équation ci-dessus est la version multidimensionnelle de la fonction gaussienne (équation 1.10).

Si ces conditions sont respectées, il est démontré que le filtre Kalman est l'estimateur optimal pour ce type de problème.

Les systèmes réels ayant généralement une certaine invariance, les matrices A_t, B_t et C_t sont en fait indépendantes du temps t.

Algorithme 1.1 filtre de Kalman $(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$

1: $\overline{\mu}_t = A_t \mu_{t-1} + B_t u_t$
2: $\overline{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$
3: $K_t = \overline{\Sigma}_t C_t^T (C_t \overline{\Sigma}_t C_t^T + Q_t)^{-1}$
4: $\mu_t = \overline{\mu}_t + K_t(z_t - C_t \overline{\mu}_t)$
5: $\Sigma_t = (I - K_t C_t)\overline{\Sigma}_t$
6: **return** μ_t, Σ_t

Dans l'algorithme 1.1 du filtre de Kalman, les lignes 1 et 2 correspondent à la prédiction, où est déterminée la croyance définie par $\overline{\mu}_t$ et $\overline{\Sigma}_t$. Les mesures z_t sont utilisées pour

corriger $\overline{bel}(x)$ en $bel(x)$ de la ligne 3 à 5. Le gain de Kalman K_t exprime la part de mesure et de modèle qui intervient dans le résultat final. R_t et Q_t sont respectivement les matrices de covariance des bruits de modèles et de mesure (équations 1.26 et 1.27).

Le filtre de Kalman est assez efficace d'un point de vue calculatoire. Sa complexité est $O(4k^2 + n^2)$

— $O(4k^2)$ correspond aux meilleurs algorithmes pour l'inversion de matrice de la ligne 3 de dimension du vecteur de mesure k.

— $O(n^2)$ à la multiplication de la ligne 5, n étant la dimension du vecteur d'état x_t.

Dans beaucoup d'applications de localisation ou de cartographie, le vecteur de mesure z_t est de dimension très inférieure au vecteur d'état x_t. Dans ce cas, la multiplication, bien que moins coûteuse à l'inversion d'une matrice de dimension égale peut rapidement devenir prédominante.

Le filtre de Kalman présente quelques sérieuses limitations d'implémentation. Tout d'abord, les bruits doivent être modélisables sous forme d'un bruit gaussien de moyenne nulle ; ce qui n'est pas toujours le cas des capteurs, comme par exemple le gyromètre qui présente un offset d'allumage. Il faudra donc intégrer ce biais dans le vecteur d'état pour pouvoir le prendre en compte. La croyance est également de forme gaussienne, ce qui rend impossible le suivi de plusieurs hypothèses, par exemple avant convergence dans un environnement présentant des symétries pour la perception. Nous disons que le filtre est uni-modal (figure 1.18). Une des limitations les plus sérieuses du filtre reste que le système doit être linéaire. Dans le cadre de la localisation, si pour un suivi 2D les équations du modèle peuvent être linéaires, dès qu'une rotation entre en jeu, cette condition n'est plus respectée.

Pour pallier les limitations concernant les systèmes non linéaires, le filtre de Kalman étendu a été mis au point (*EKF* : Extended Kalman Filter). Les matrices A_t et B_t de l'équation de prédiction 1.26 sont remplacées par une fonction non linéaire $g(x_{t-1}, u_t)$ et de même dans l'équation de correction 1.27, C_t est remplacée par $h(x_t)$.

Les applications linéaires de fonctions gaussiennes restent des gaussiennes, mais dans le cas de fonctions non-linéaires, il est nécessaire de linéariser afin de pouvoir continuer à appliquer les principes mathématiques sous-jacents au filtre de Kalman. Pour l'*EKF*, un développement de Taylor permet de calculer des jacobiennes et de linéariser localement les fonctions g et h autour de x_{t-1} pour calculer les covariances. La croyance sera donc approximée par une gaussienne. Cela fonctionne correctement tant que la confiance dans x_t est suffisante et que les non-linéarités ne sont pas trop importantes localement.

Il existe d'autres solutions pour linéariser, comme le *UKF* (Unscented Kalman Filter : Julier et Uhlmann [1997]) qui contrairement à l'*EKF* utilise plusieurs points pour linéariser les fonctions g et h.

1.3.4.2 Filtre d'histogramme

Le filtre d'histogramme, connu aussi sous le nom de filtre bayésien discret, tient son nom de sa manière de représenter l'espace d'état sous forme discrète. L'espace d'état est discrétisé, et la croyance va être calculée sur l'intégralité de la plage des dimensions. Il n'y a plus de limitation à des représentations gaussiennes, et la densité de probabilité peut prendre une forme quelconque, et devenir multi-modale (figure 1.18).

L'algorithme d'actualisation 1.2 est extrêmement simple, il découle directement des équations 1.20 et 1.23. Pour chaque case k représentant la croyance, l'étape de prédiction puis l'étape de correction vont être calculées.

Algorithme 1.2 Filtre d'histogramme ($P(x_{k,t-1}, u_t, z_t)$)

1: **for all** k **do**
2: $\overline{P}(x_{k,t}) = \sum_{\forall (x_{t-1}, u_t) \rightsquigarrow x_{k,t}} P(u_t) P(x_{t-1})$
3: $P(x_{k,t}) = \eta P(z_t | x_k, t) \overline{P}(x_{k,t})$
4: **end for**
5: **return** $P(x_{k,t})$

— La ligne 2 correspond à l'étape de prédiction (*motion update)* : pour tous les couples de déplacement (u_t, x_{t-1}) menant à la case k, nous allons sommer la probabilité que le déplacement se soit bien déroulé $P(u_t)$ par la probabilité que nous soyons bien dans la bonne case avant le déplacement $P(x_{t-1})$.

— La ligne 3 correspond à l'étape de correction (*measurement update)* : pour chaque case k, la probabilité que la mesure corresponde à la position de la case k ($P(z_t|x_k, t)$) est multipliée par la probabilité *a priori* d'être dans la case k ($\overline{P}(x_{k,t})$).

D'un point de vue pratique, généralement l'étape de *motion update* est calculée séparément du *measurement update*. La constante de normalisation η est calculée après coup. Il suffit de sommer les $P(x_{k,t})$ puis de les normaliser $P_{norm}(x_{k,t}) = P(x_{k,t}) / \sum_i P(x_{i,t})$.

Du fait de discrétiser l'espace d'état et de le calculer dans sa totalité, le filtre d'histogramme est coûteux en temps de calcul. Par exemple la fonction de vraisemblance entre la mesure et la carte sera évaluée pour chaque case k à la ligne 3 de l'algorithme 1.2.

Pour un filtre au vecteur d'état x de dimension n, si $D_i, i \in [[1; n]]$ représente le nombre de cases discrètes pour chaque dimension. La complexité est $O(\prod_{i=1}^{n} D_i)$.

Pour améliorer la performance, au lieu de faire une discrétisation statique, il est possible d'utiliser des dimensions de cases dynamiques en fonction de la croyance *a posteriori*. Les zones de l'espace d'état de faible probabilité sont représentées avec des cases d'autant plus grandes que la zone où $bel(x_t)$ varie fortement et est plus significative. Ce principe revient un peu à celui du filtre particulaire que nous allons aborder dans la section 1.3.4.3.

Le calcul d'un estimé de $\hat{x}_t$ à partir de $bel(x_t)$ discret n'est pas forcément trivial :

— Un simple maximum de la croyance $\hat{x}_t = \max_k(bel(x_{k,t}))$ peut poser des problèmes en cas de multi-modalité. Il peut se produire des sauts entre les différents maximums, qui ne sont pas acceptables si le $\hat{x}_t$ intervient ensuite dans une boucle de contrôle.

— Une moyenne pondérée de la croyance $\hat{x}_t = \sum_k bel(x_{k,t}) x_k$ peut également amener à un résultat erroné en cas de multi-modalité. Imaginons une croyance avec deux maximums dus par exemple à une symétrie de l'environnement. Le résultat de la moyenne aboutira un peu comme le filtre de Kalman figure 1.18 entre les deux maximums.

— Levinson et Thrun [2010] proposent une alternative permettant « d'adoucir les sauts entre les différents maximums de vraisemblance » :

$$\hat{x}_t = \frac{\sum_k bel(x_{k,t})^\alpha x_k}{\sum_k bel(x_{k,t})^\alpha} \quad avec \quad \alpha > 1 \tag{1.29}$$

Lors de l'initialisation, si aucune approximation de l'estimation $\hat{x}_t$ n'est connue, la croyance est répartie de manière uniforme entre toutes les cases k.

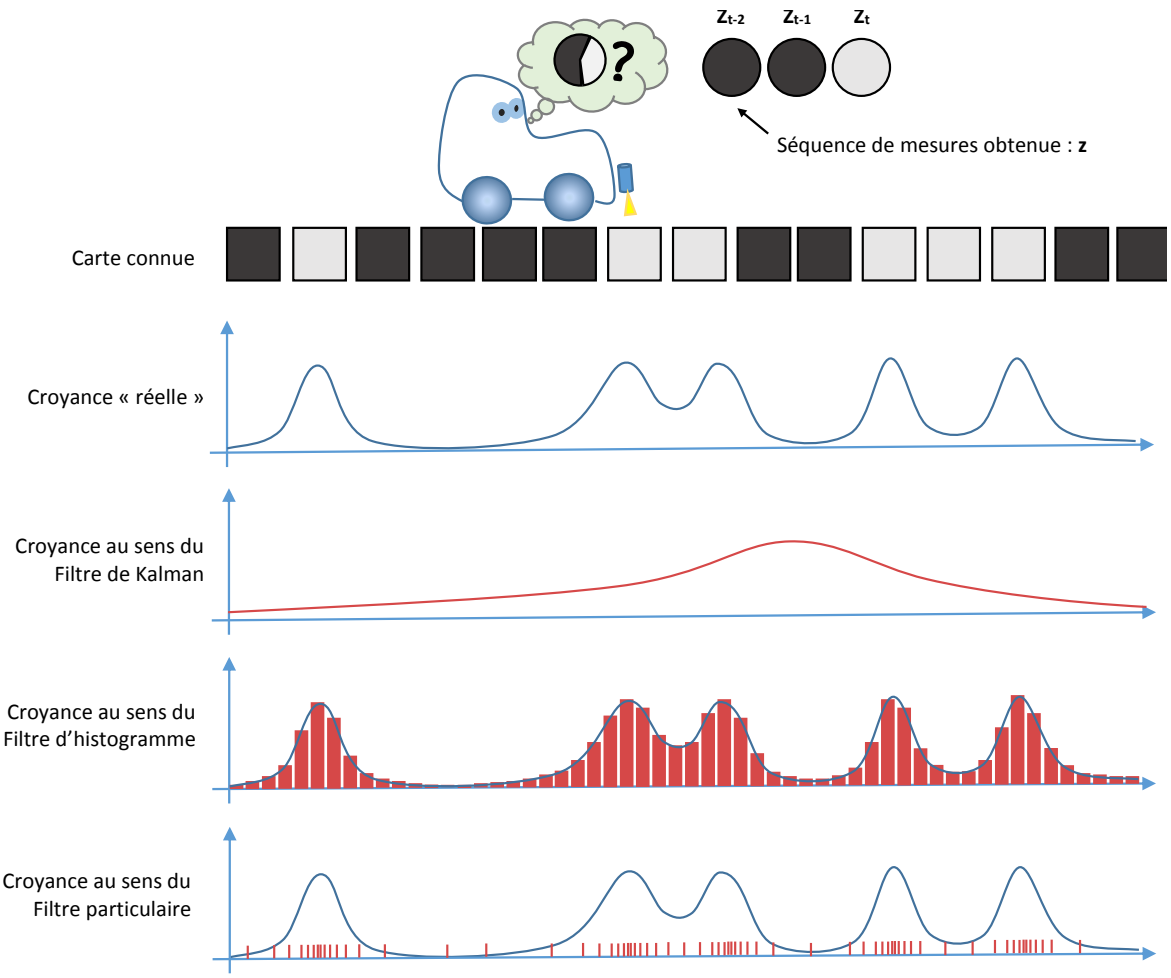


FIGURE 1.18 – Croyance au sens des différents filtres bayésiens, sur l'exemple du robot 1D de la figure 1.15 : *KF*, décrit par une gaussienne uni-modale ; *HF*, discrétisation régulière de l'espace d'état ; *PF*, discrétisation de l'espace d'état en fonction de la densité de probabilité.

1.3.4.3 Filtre particulaire

Malgré sa relative jeunesse, le filtre particulaire est rapidement devenu très populaire dans le domaine de la localisation : Dellaert et collab. [1999]. Tout comme le filtre d'histogramme, c'est un filtre non paramétrique. Il représente la croyance $bel(x_t)$ par un ensemble d'hypothèses $\mathscr{P}_t$ appelées particules $p_t^{[i]}$:

$$\mathscr{P}_t = \left\{ p_t^{[1]}, p_t^{[2]} \cdots p_t^{[m]} \cdots p_t^{[\mathrm{M}]} \right\} \tag{1.30}$$

Contrairement au filtre d'histogramme « standard » où la discrétisation de la croyance est uniforme, le filtre particulaire ne va conserver des hypothèses qu'aux endroits de forte vraisemblance (figure 1.18) ; c'est-à-dire que les particules vont se répartir dans l'espace d'état où la croyance sera la plus forte.

Chaque particule est constituée d'une représentation du vecteur d'état et d'un *poids* associé w_m.

$$p_t^{[m]} = \left\{ x_t^{[m]}, w_t^{[m]} \right\} = \left\{ \begin{bmatrix} x_{1,t}^{[m]} & x_{2,t}^{[m]} & \dots & x_{n,t}^{[m]} \end{bmatrix}^{\mathrm{T}}, w_t^{[m]} \right\} \tag{1.31}$$

En localisation, le vecteur d'état $x_t^{[m]}$ peut être vu comme la pose de la particule, et le poids $w_t^{[m]}$ comme la vraisemblance avec la mesure correspondante à la pose.

Algorithme 1.3 Filtre particulaire avec ré-échantillonnage d'importance($\mathscr{P}_{t-1}, u_t, z_t$)

1: $\overline{\mathscr{P}}_t = \mathscr{P}_t = \varnothing$
2: **for** $m = 1$ to M **do**
3: tirer $x_t^{[m]} \sim \mathrm{P}(x_t | u_t, x_{t-1}^{[m]})$
4: $w_t^{[m]} = \mathrm{P}(z_t, x_t^{[m]})$
5: $\overline{\mathscr{P}}_t = \overline{\mathscr{P}}_t + \{x_t^{[m]}, w_t^{[m]}\}$
6: **end for**
7: **for** $m = 1$ to M **do**
8: tirer i avec la probabilité $w_t^{[m]}$
9: ajouter $p_t^{[m]}$ à $\mathscr{P}_t$
10: **end for**
11: **return** $\mathscr{P}_t$

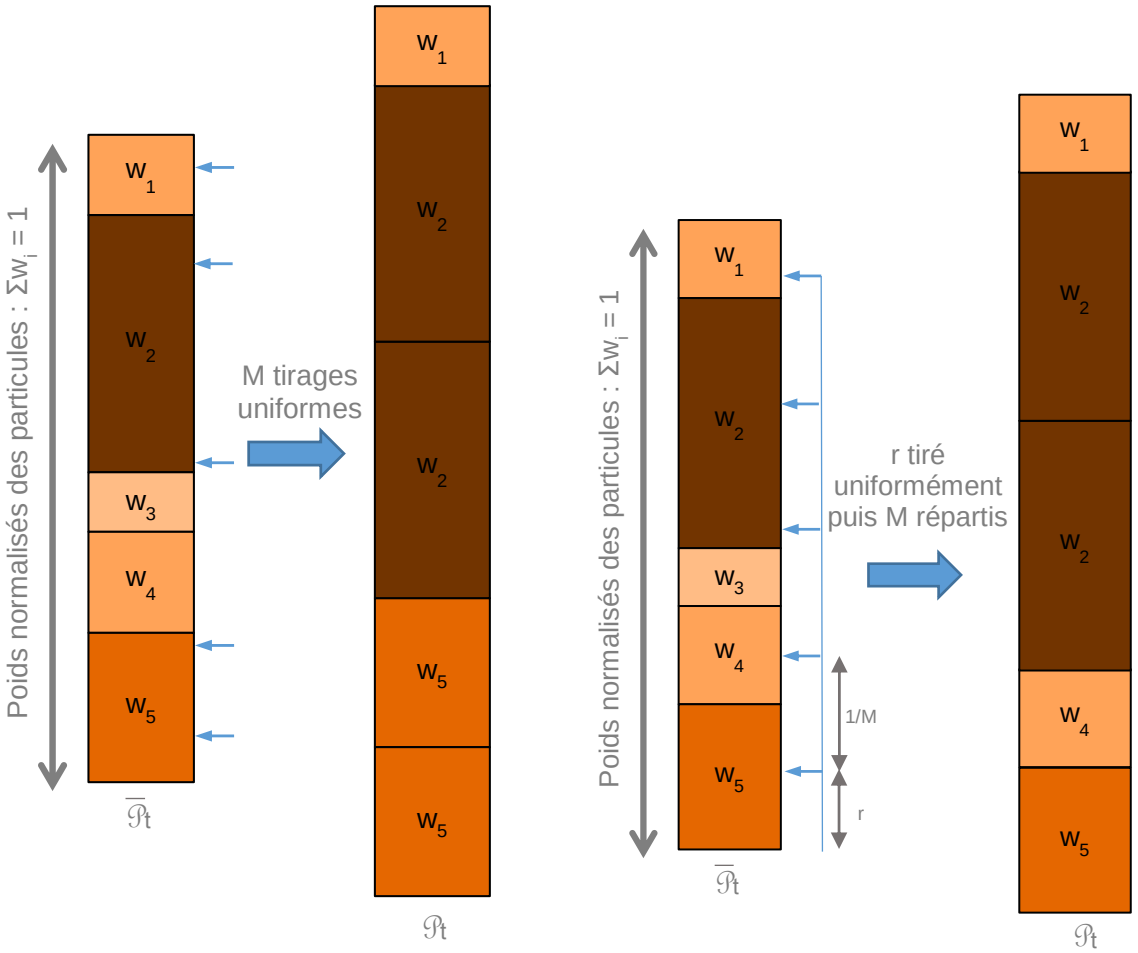
L'algorithme 1.3 du filtre particulaire se déroule de la manière suivante :

— Ligne 3, le *motion update* ou étape de prédiction, les hypothèses d'état des particules sont déplacées en fonction de la probabilité de déplacement $\mathrm{P}(x_t | u_t, x_{t-1}^{[m]})$, qui tient compte de la commande u_t.

— La ligne 4, détermine le *measurement update* ou étape de correction. Les poids $w_t^{[m]}$ ou facteurs d'importance des particules vont être attribués en fonction de la probabilité que la mesure corresponde à l'environnement $\mathrm{P}(z_t, x_t^{[m]})$.

— Les ligne 8 et 9 correspondent au ré-échantillonnage (*resampling*). C'est ici que les particules vont être re-réparties en fonction de $bel(x_t)$. Pour constituer le nouvel ensemble $\mathscr{P}_t$, les particules ayant un poids faible vont être supprimées, et les autres dupliquées pour constituer $\overline{\mathscr{P}}_t$.

Il existe plusieurs méthodes de resampling (Hol et collab. [2006]), mais la plus connue reste le resampling systématique (figure 1.19a). D'un point de vue algorithmique, les poids des différentes particules $\overline{\mathscr{P}}_t$ sont représentés sous forme d'une somme cumulée. N nombres entre [0,1[sont tirés uniformément et désigneront les particules conservées dans $\mathscr{P}_t$.



(a) Ré-échantillonnage systématique : M tirages permettent de sélectionner les particules

(b) Ré-échantillonnage « *low variance* » : r est tiré entre $[0,1[$, puis les particules sont sélectionnées toutes les $1/M$

FIGURE 1.19 – 2 types de ré-échantillonnage du filtre particulaire, exemple avec M = 5 particules. La somme des poids est normalisée avant ré-échantillonnage. Elle ne l'est plus après, d'ailleurs le poids des particules après ré-échantillonnage est généralement fixé à 1.

Représenter l'espace d'état de manière discrète avec un nombre fini de particules est une approximation de la densité de probabilité. Par exemple, la représentation d'une densité de probabilité gaussienne introduira une erreur sur la moyenne et la variance. Cette erreur est appelée la variance du filtre, et bien sûr le but consiste à la minimiser.

Dans la pratique, le resampling pose parfois problème, typiquement dans le cas d'un robot immobile, $x_t = x_{t-1}$. Le filtre particulaire de base supprimera les particules ayant les poids les plus faibles, mais sans recréer de nouvelle particule. Cela finira par tendre vers une situation ne comportant plus que quelques particules toutes identiques, et donc à augmenter la variance du filtre. La variété de la population des particules va diminuer au point de ne conserver que les quelques hypothèses les plus probables. Dans ce cas, la densité de probabilité est très mal représentée. Il existe plusieurs solutions pour pallier ce problème, en voici trois :

— Réduire la fréquence des resampling et actualiser les poids en les multipliant avec le précédent : $w_t^{[m]} = \mathrm{P}(z_t|x_t^{[m]})\, w_{t-1}^{[m]}$. Ou même ne pas faire de resampling si le robot ne bouge pas, mais faut-il encore le savoir.

— Utiliser le *low variance resampling* (figure 1.19b) qui favorise la diversité des particules. Seul le décalage r est tiré au hasard, puis les M particules sont sélectionnées uniformément.

— Ajouter du bruit, généralement à l'étape de *motion update*, de manière à « maintenir un peu de mouvement » dans les particules.

Le calcul d'un estimé de $\widehat{x_t}$ à partir de $bel(x_t)$ représenté par des particules pose sensiblement le même problème que pour le filtre d'histogramme (section 1.3.4.2) ; mais généralement une moyenne pondérée est utilisée :

$$\widehat{x_t} = \sum_{m=1}^{N} w_t^{[m]} x_t^{[m]} \tag{1.32}$$

La matrice de covariance, indicateur de qualité de l'estimation de chaque dimension, pourra être obtenue de cette manière :

$$\widehat{Q_t} = \sum_{m=1}^{M} w_t^{[m]} \left(x_t^{[m]} - \widehat{x_t}\right)^T \left(x_t^{[m]} - \widehat{x_t}\right) \tag{1.33}$$

Le nombre de particules M est essentiel au bon déroulement du processus de localisation. Si le nombre de particules est faible, l'espace d'état est insuffisamment couvert, et il y a un risque de se trouver dans un cas de *particle deprivation*. Il n'y a pas assez de particules pour couvrir toutes les régions de forte vraisemblance. Une mauvaise séquence de tirage aléatoire lors du resampling peut alors amener à éliminer les particules proches de la position réelle, et donc à une divergence du filtre. Une solution courante consiste à ajouter aléatoirement une partie des particules lors de la phase de resampling.

Les ressources en temps de calcul sont directement liées au nombre de particules. Si aucune connaissance *a priori* n'est envisagée avant le processus de localisation, il faudra des milliers de particules pour couvrir correctement l'espace d'état et éviter une convergence dans un minimum local (à titre d'exemple [Thrun et collab., 2005, p. 255] : 100000 particules dans un environnement 2.5D de $54m \times 18m$). Afin de rendre possibles les calculs en temps réel, différentes méthodes existent pour pallier ce nombre excessif d'hypothèses à tester, en voici deux :

— le *KDL-resample* (Kullback-Leibler Divergence)(Fox [2003]) qui permet de choisir dynamiquement le nombre de particules M en fonction de la vraisemblance.

— le filtre particulaire Rao-Blackwellisé (*RBPF*)(Casella et Robert [1996]) : pour certains modèles appelés modèles conditionnellement linéaires gaussiens, il est possible d'estimer la partie linéaire du vecteur d'état avec un filtre de Kalman, et la partie non linéaire est estimée avec un filtre particulaire. Typiquement, il est utilisé dans l'algorithme du FastSLAM (Montemerlo et collab. [2002]). Les positions des amers sont estimées par des filtres de Kalman et la pose du robot par un filtre particulaire.

Toutes ces méthodes de filtrage nécessitent un paramétrage dans leur mise au point. Dans ce cadre, une solution efficace de développement consiste à créer un jeu de données de référence à partir des capteurs réels. Ce point sera abordé dans le chapitre 2.

Mais tout d'abord, nous présentons ci-dessous quelques clés pour comprendre les difficultés d'une localisation globale sans connaissance *a priori*.

1.4 Modèle en deux couches

La localisation ayant une précision suffisante pour assurer le contrôle d'un mobile est malheureusement liée à une forte complexité calculatoire. Pour atteindre une précision centimétrique, la fonction de vraisemblance doit évaluer de nombreux impacts LiDAR. Si l'espace de recherche est trop important, il devient tout simplement illusoire d'espérer résoudre le problème en temps réel. De plus, si le nombre de dimensions du vecteur d'état à estimer est important, par exemple 6 au lieu de 3, la complexité du problème d'un point de vue algorithmique « explose ».

Concrètement, les exemples d'applications entrevus à la section 1.2.5.3 utilisant une carte vaste nécessitent tous un point d'initialisation relativement proche de la position réelle. Par exemple Levinson et Thrun [2010] utilisent le GPS pour centrer un filtre d'histogramme local qui travaille avec des cellules de 15cm de côté. La surface de l'histogramme semble couvrir la dimension de l'erreur GPS, soit quelques mètres (L'information n'est pas donnée directement, mais dans le papier Levinson et collab. [2011], la figure 4 montre une vue 3D de la croyance après correction, qui semble faire approximativement la dimension d'un véhicule). Kümmerle et collab. [2009] utilisent également le GPS avant l'entrée dans le parking, qui leur sert alors de point d'initialisation. Une fois hors couverture GPS, une centrale inertielle fusionne des données odométriques pour alimenter le *motion update*. Pour répartir leurs particules à l'initialisation, Fallon et collab. [2012] supposent que la distribution initiale de la croyance est connue. Ils proposent également de réduire le nombre de dimensions à estimer dans le vecteur d'état, en déterminant le plan du sol dans le nuage de points. Sous condition que le sol soit plan (environnement intérieur), cela permet d'estimer la hauteur, le roulis et le tangage (ces deux dernières dimensions peuvent également être estimées à l'aide d'une centrale inertielle d'attitude « bas coût »).

C'est dans ce cadre que nous avons proposé dans le papier Merriaux et collab. [2015] un modèle de localisation à deux couches (figure 1.20) :

La couche 2 fournit une localisation d'une précision permettant d'assurer le contrôle, mais concrètement elle a besoin d'une initialisation relativement précise pour réduire les temps de recherche sur un espace vaste. Elle s'apparente donc plus à une couche de localisation locale (*Tracking*) précise.

La couche 1 assure une localisation de haut niveau sur des données fortement compressées, et fournit à la *couche 2* une initialisation par exemple 2D permettant de réduire rapidement l'espace de recherche.

La fonction de vraisemblance étant évaluée pour chaque hypothèse de pose, si l'on veut pouvoir explorer de vastes espaces, son coût de calcul doit être le plus léger possible. Cela entraîne une représentation de l'environnement fortement compressée. Typiquement, si l'information capteur peut être ramenée à un descripteur suffisamment discriminant d'une taille donnée, la fonction de vraisemblance consistera en un calcul de distance avec l'ensemble des descripteurs de la carte. Si plusieurs milliers (voire des millions) de descripteurs sont contenus dans la carte, ce dernier point peut tout de même être problématique. Il ne faut alors plus organiser les données de manière métrique, mais sous forme d'un arbre dans l'espace des descripteurs, de manière à pouvoir identifier efficacement les zones de l'espace géométrique méritant d'être évaluées par la couche 2.

1.4.1 Quelques candidats pour la couche 1

Dans l'ensemble, les méthodes de localisation topologique sur carte hybride, c'est-à-dire pouvant fournir des coordonnées métriques, font d'excellentes candidates pour la

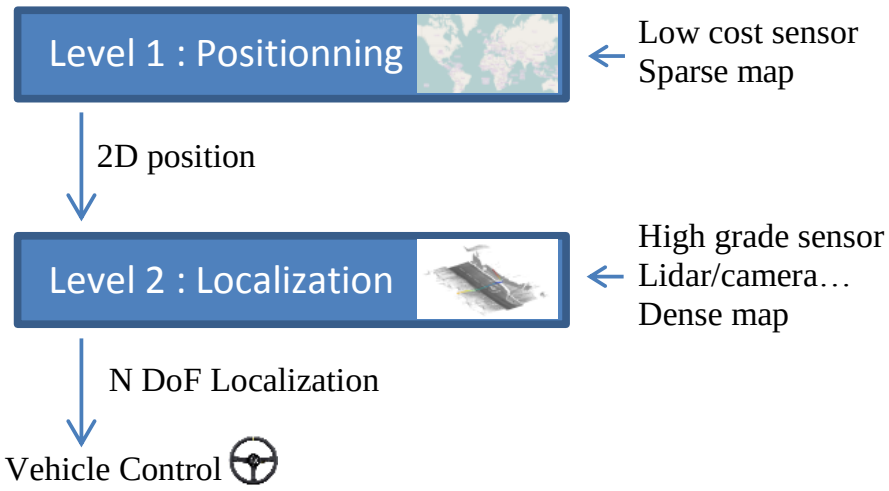


FIGURE 1.20 – Modèle de localisation en deux couches : une première couche fournit une localisation grossière 2D peu coûteuse en temps de calcul, ce qui permet l'initialisation d'une deuxième couche de localisation assurant le contrôle.

couche 1.

En intérieur, les solutions de localisation basées sur la puissance de réception (*RSSI* : Received Signal Strength Indication) de liaisons hautes fréquences peuvent très bien convenir. Les modèles de transmission étant mal maîtrisés, elles n'offrent pas une grande précision, mais généralement, elles sont suffisantes pour savoir dans quelle pièce se trouve le mobile. Il existe de nombreux travaux sur le *Bluetooth* (Sugano et collab. [2006]) et sur le *Wifi* (Martin et collab. [2010]). Un exemple de solution intérieure basée vision serait Ulrich et Nourbakhsh [2000], mais il en existe de nombreuses autres. Ils déterminent des descripteurs sous forme d'histogrammes invariants dans une image omnidirectionnelle. Basée LiDAR en intérieur, Morris et collab. [2005] proposent un descripteur des intersections de galeries de mines.

En extérieur, la solution la plus évidente est bien sûr le GPS. Mais pour les différents inconvénients déjà évoqués en début de ce chapitre (section 1.2.1), il ne peut pas convenir à toutes les situations. Bonnifait et collab. [2001] proposent de fusionner l'information avec l'odométrie du véhicule pour améliorer sa précision et sa fiabilité. Brubaker et collab. [2013] utilisent l'odométrie visuelle pour se localiser sur une carte *OpenStreetMap*. C'est dans ce cadre que nous avons proposé une localisation couche 1 sur graphe sans *GPS* avec comme sources de données les capteurs *ABS* et *ESP* d'un véhicule routier. Ces travaux ont fait l'objet de 3 publications (Merriaux et collab. [2014], Merriaux et collab. [2015] et Dupuis et collab. [2015]), et seront présentés en détail dans le chapitre 4.

Les méthodes ensemblistes peuvent également présenter d'excellents candidats pour la couche 1. Effectivement, s'il faut subdiviser l'espace de manière très fine et donc coûteuse en temps de calcul pour obtenir une localisation précise, ces méthodes sont rapides pour éliminer des sous-ensembles de l'espace, où il est certain que le mobile ne se trouve pas. Desrochers et collab. [2015] proposent une solution basée LiDAR 3D dans ce sens. Une première approximation est réalisée par une méthode ensembliste, puis sur la zone mise en évidence, ils affinent la position par une méthode probabiliste à l'aide d'un filtre particulaire.

Conclusion

Nous avons vu les bases de la localisation en robotique mobile, les capteurs et les méthodes associées. La suite du document sera consacrée à la présentation des travaux réalisés dans cette thèse.

Le chapitre 2 présentera les moyens d'essais ainsi que les méthodes de développement mis en œuvre pour évaluer les solutions proposées dans cette thèse. Puis nous aborderons la localisation couche 2, six degrés de liberté adaptée aux milieux pétro-chimiques dans le chapitre 3, pour enfin terminer par une localisation couche 1 sur graphe *OpenStreetMap* dans le chapitre 4.

Références

Alsayed, Z., G. Bresson, F. Nashashibi et A. Verroust-Blondet. 2015, «Pml-slam : a solution for localization in large-scale urban environments», dans *PPNIV-IROS 2015*. 25

Anirudh Viswanathan, B. R. P. et D. Huber. 2014, «Vision based robot localization by ground to satellite matching in gps-denied situations», dans *IROS*. 14

Arsénio, A. et M. I. Ribeiro. 1998, «Absolute localization of mobile robots using natural landmarks», dans *In Proceedings of the fth IEEE International Conference on Electronics, Circuits and Systems. IEEE*, Citeseer. 25

Bay, H., T. Tuytelaars et L. Van Gool. 2006, «Surf : Speeded up robust features», dans *Computer Vision–ECCV 2006*, Springer, p. 404–417. 16

Behzadian, B., P. Agarwal, W. Burgard et G. D. Tipaldi. 2015, «Monte carlo localization in hand-drawn maps», *arXiv preprint arXiv :1504.00522*. 26

Besl, P. J. et N. D. McKay. 1992, «Method for registration of 3-d shapes», dans *Robotics-DL tentative*, International Society for Optics and Photonics, p. 586–606. 23

Biber, P. et W. StraBer. 2003, «The normal distributions transform : A new approach to laser scan matching», dans *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 3, IEEE, p. 2743–2748. 23

Bonin-Font, F., A. Ortiz et G. Oliver. 2008, «Visual navigation for mobile robots : A survey», *Journal of intelligent and robotic systems*, vol. 53, nº 3, p. 263–296. 16

Bonnifait, P., P. Bouron, P. Crubille et D. Meizel. 2001, «Data fusion of four abs sensors and gps for an enhanced localization of car-like vehicles», dans *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 2, IEEE, p. 1597–1602. 41

Bosse, M. et R. Zlot. 2009, «Keypoint design and evaluation for place recognition in 2d lidar maps», *Robotics and Autonomous Systems*, vol. 57, nº 12, p. 1211–1224. 23

Bosse, M., R. Zlot et P. Flick. 2012, «Zebedee : Design of a spring-mounted 3-d range sensor with application to mobile mapping», *Robotics, IEEE Transactions on*, vol. 28, nº 5, p. 1104–1119. 18

Brubaker, M. A., A. Geiger et R. Urtasun. 2013, «Lost ! leveraging the crowd for probabilistic visual self-localization», *Computer*. 41

Caron, G., A. Dame et E. Marchand. 2014, «Direct model based visual tracking and pose estimation using mutual information», *Image and Vision Computing*, vol. 32, n° 1, p. 54–63. 14

Casella, G. et C. P. Robert. 1996, «Rao-blackwellisation of sampling schemes», *Biometrika*, vol. 83, n° 1, p. 81–94. 39

Cole, D. M. et P. M. Newman. 2006, «Using laser range data for 3d slam in outdoor environments», dans *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, IEEE, p. 1556–1563. 7

Dellaert, F., D. Fox, W. Burgard et S. Thrun. 1999, «Monte carlo localization for mobile robots», dans *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 2, IEEE, p. 1322–1328. 37

Desrochers, B., S. Lacroix et L. Jaulin. 2015, «Set-membership approach to the kidnapped robot problem», . 41

Drouilly, R. 2015, *Hybrid metric topological and semantic mapping for navigation in large scale environments*, Theses, Université Nice Sophia Antipolis. 6, 8

Duda, R. O. et P. E. Hart. 1972, «Use of the hough transformation to detect lines and curves in pictures», *Communications of the ACM*, vol. 15, n° 1, p. 11–15. 22

Dupuis, Y., P. Merriaux, P. Vasseur et X. Savatier. 2015, «Vehicle positioning in road networks without gps», dans *IEEE Intelligent Transportation Systems Society Conference (ITSC)*. 41

El Hamzaoui, O. 2012, *Localisation et cartographie simultanées pour un robot mobile équipé d'un laser à balayage : CoreSLAM*, thèse de doctorat, Ecole Nationale Supérieure des Mines de Paris. 25

Elfes, A. 1987, «Sonar-based real-world mapping and navigation», *Robotics and Automation, IEEE Journal of*, vol. 3, n° 3, p. 249–265. 7, 14

Fallon, M. F., H. Johannsson et J. J. Leonard. 2012, «Efficient scene simulation for robust monte carlo localization using an rgb-d camera», dans *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, IEEE, p. 1663–1670. 7, 23, 40

Fischler, M. et R. Bolles. 1981, «Random sample consensus : A paradigm for model fitting with applications to image analysis and automated cartography», dans *Communications of the ACM*, vol. 24, p. 381–395. 15, 22

Fox, D. 2003, «Adapting the sample size in particle filters through kld-sampling», *The international Journal of robotics research*, vol. 22, n° 12, p. 985–1003. 39

Frassl, M., M. Angermann, M. Lichtenstern, P. Robertson, B. J. Julian et M. Doniec. 2013, «Magnetic maps of indoor environments for precise localization of legged and non-legged locomotion», dans *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, IEEE, p. 913–920. 9

Fraundorfer, F. et D. Scaramuzza. 2012, «Visual odometry : Part ii : Matching, robustness, optimization, and applications», *Robotics & Automation Magazine, IEEE*, vol. 19, n° 2, p. 78–90. 14

Guyonneau, R. 2013, *Méthodes ensemblistes pour la localisation en robotique mobile*, thèse de doctorat, Université d'Angers. 27

Harris, C. et M. Stephens. 1988, «A combined corner and edge detector.», dans *Alvey vision conference*, vol. 15, Citeseer, p. 50. 16

Herbert, M., C. Caillas, E. Krotkov, I. S. Kweon et T. Kanade. 1989, «Terrain mapping for a roving planetary explorer», dans *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*, IEEE, p. 997–1002. 7, 27

Hol, J. D., T. B. Schon et F. Gustafsson. 2006, «On resampling algorithms for particle filters», dans *Nonlinear Statistical Signal Processing Workshop, 2006 IEEE*, IEEE, p. 79–82. 37

Jagbrant, G., J. Underwood, J. Nieto et S. Sukkarieh. 2013, «Lidar based localisation in almond orchards», . 26

Jaud, M., R. Rouveure, P. Faure et M.-O. Monod. 2013, «Methods for fmcw radar map georeferencing», *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 84, p. 33–42. 14

Julier, S. J. et J. K. Uhlmann. 1997, «New extension of the kalman filter to nonlinear systems», dans *AeroSense'97*, International Society for Optics and Photonics, p. 182–193. 34

Kalman, R. 1960, «A new approach to linear filtering and prediction problems», *Transactions of the ASME - Journal of Basic Engineering*, vol. 82, n° Series D, p. 35–45. 33

Koch, A. et A. Zell. 2014, «Mapping of passive uhf rfid tags with a mobile robot using outlier detection and negative information», dans *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, IEEE, p. 1619–1624. 9

Kuemmerle, R., B. Steder, C. Dornhege, A. Kleiner, G. Grisetti et W. Burgard. 2011, «Large scale graph-based slam using aerial images as prior information», *Autonomous Robots*, vol. 30, n° 1, p. 25–39. 26

Kumar, P., C. P. McElhinney, P. Lewis et T. McCarthy. 2014, «Automated road markings extraction from mobile laser scanning data», *International Journal of Applied Earth Observation and Geoinformation*, vol. 32, p. 125–137. 18

Kümmerle, R., D. Hähnel, D. Dolgov, S. Thrun et W. Burgard. 2009, «Autonomous driving in a multi-level parking structure», dans *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, IEEE, p. 3395–3400. 26, 40

Levinson, J., J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt et collab.. 2011, «Towards fully autonomous driving : Systems and algorithms», dans *Intelligent Vehicles Symposium (IV), 2011 IEEE*, IEEE, p. 163–168. 40

Levinson, J., M. Montemerlo et S. Thrun. 2007, «Map-based precision vehicle localization in urban environments.», dans *Robotics : Science and Systems*, Citeseer. 26

Levinson, J. et S. Thrun. 2010, «Robust vehicle localization in urban environments using probabilistic maps», dans *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, IEEE, p. 4372–4378. 23, 24, 26, 35, 40

Lim, H. et Y. S. Lee. 2009, «Real-time single camera slam using fiducial markers», dans *ICCAS-SICE, 2009*, IEEE, p. 177–182. 14

Lisien, B., D. Morales, D. Silver, G. Kantor, I. Rekleitis et H. Choset. 2005, «The hierarchical atlas», *Robotics, IEEE Transactions on*, vol. 21, nº 3, p. 473–481. 8

Loevsky, I. et I. Shimshoni. 2010, «Reliable and efficient landmark-based localization for mobile robots», *Robotics and Autonomous Systems*, vol. 58, nº 5, p. 520–528. 21

Lowe, D. G. 2004, «Distinctive image features from scale-invariant keypoints», *International journal of computer vision*, vol. 60, nº 2, p. 91–110. 16

Magnusson, M., A. Nüchter, C. Lörken, A. J. Lilienthal et J. Hertzberg. 2009, «Evaluation of 3d registration reliability and speed-a comparison of icp and ndt», dans *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, IEEE, p. 3907–3912. 23

Martin, E., O. Vinyals, G. Friedland et R. Bajcsy. 2010, «Precise indoor localization using smart phones», dans *Proceedings of the international conference on Multimedia*, ACM, p. 787–790. 41

McManus, C., W. Churchill, W. Maddern, A. D. Stewart et P. Newman. 2014, «Shady dealings : Robust, long-term visual localisation using illumination invariance», dans *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. 16

Meilland, M. 2012, *Cartographie RGB-D dense pour la localisation visuelle temps-reel et la navigation autonome.*, thèse de doctorat, Ecole des mines de paris. 15

Meilland, M., A. I. Comport et P. Rives. 2011, «Dense visual mapping of large scale environments for real-time localisation», dans *IEEE International Conference on Intelligent Robots and Systems*, p. 4242 –4248. 8

Merriaux, P., Y. Dupuis, P. Vasseur et X. Savatier. 2014, «Wheel odometry-based car localization and tracking on vectorial map», dans *ITSC*, édité par ITSC. 41

Merriaux, P., Y. Dupuis, P. Vasseur et X. Savatier. 2015, «Fast and robust vehicle positioning on graph-based representation of drivable maps», dans *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, IEEE, p. 2787–2793. 40, 41

Montemerlo, M., S. Thrun, D. Koller, B. Wegbreit et collab.. 2002, «Fastslam : A factored solution to the simultaneous localization and mapping problem», dans *AAAI/IAAI*, p. 593–598. 39

Moravec, H. 1996, «Robot spatial perceptionby stereoscopic vision and 3d evidence grids», *Perception,(September)*. 7

Morris, A., D. Silver, D. Ferguson et S. Thayer. 2005, «Towards topological exploration of abandoned mines», dans *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, IEEE, p. 2117–2123. 41

Nuchter, A., H. Surmann, K. Lingemann, J. Hertzberg et S. Thrun. 2004, «6d slam with an application in autonomous mine mapping», dans *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, vol. 2, ISSN 1050-4729, p. 1998–2003 Vol.2, doi: 10.1109/ROBOT.2004.1308117. 26

Paluri, M., A. Garg et H. Christensen. 2010, «Autonomous localization and navigation using 2d laser scanners», . 25

Pfaff, P., R. Kümmerle, D. Joho, C. Stachniss, R. Triebel et W. Burgard. 2007, «Navigation in combined outdoor and indoor environments using multi-level surface maps», *WS on Safe Navigation in Open and Dynamic Environments, IROS*, vol. 7. 7, 27

Pinto, A. M. G., A. P. Moreira et P. Costa. 2012, «Indoor localization system based on artificial landmarks and monocular vision.», *Telkomnika*, vol. 10, nº 4. 14

Pomerleau, F., F. Colas et R. Siegwart. 2015, «A review of point cloud registration algorithms for mobile robotics», *Foundations and Trends in Robotics (FnTROB)*, vol. 4, nº 1, p. 1–104. 23

Reinke, C. et P. Beinschob. 2013, «Strategies for contour-based self-localization in large-scale modern warehouses», dans *Intelligent Computer Communication and Processing (ICCP), 2013 IEEE International Conference on*, IEEE, p. 223–227. 25

Reymann, C. et S. Lacroix. 2015, «Improving lidar point cloud classification using intensities and multiple echoes», dans *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2015)*, Hamburg, Germany, p. 7p. 18

Roberge, J.-P. 2013, *Conception et intégration d'un capteur LIDAR 3D pour la navigation autonome des robots mobiles en terrain inconnu*, mémoire de maîtrise, École Polytechnique de Montréal. 18

Ronzoni, D., R. Olmi, C. Secchi et C. Fantuzzi. 2011, «Agv global localization using indistinguishable artificial landmarks», dans *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, IEEE, p. 287–292. 21, 22

Royer, E. 2006, *Cartographie 3D et localisation par vision monoculaire pour la navignation autonome d'un robot mobile*, thèse de doctorat, Université Blaise Pascal-Clermont-Ferrand II. 7, 14, 15

Ryde, J. et H. Hu. 2008, «3d laser range scanner with hemispherical field of view for robot navigation», dans *Advanced Intelligent Mechatronics, 2008. AIM 2008. IEEE/ASME International Conference on*, IEEE, p. 891–896. 18

Scaramuzza, D. et F. Fraundorfer. 2011, «Visual odometry [tutorial]», *Robotics & Automation Magazine, IEEE*, vol. 18, nº 4, p. 80–92. 14

Sugano, M., T. Kawazoe, Y. Ohta et M. Murata. 2006, «Indoor localization system using rssi measurement of wireless sensor network based on zigbee standard», *Target*, vol. 538, p. 050. 41

Takahashi, G., H. Takeda et Y. Shimano. 2014, «Automatic drawing for traffic marking with mms lidar intensity», *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 1, p. 363–370. 18

Thrun, S. 1997, «Learning metric-topological maps for indoor mobile robot navigation», *Artificial Intelligence*, vol. 99, nº 1, p. 21–71. 8

Thrun, S. 2001, «A probabilistic on-line mapping algorithm for teams of mobile robots», *The International Journal of Robotics Research*, vol. 20, nº 5, p. 335–363. 24

Thrun, S., W. Burgard, D. Fox et collab.. 2005, *Probabilistic robotics*, vol. 1, MIT press Cambridge. 24, 27, 39

Tipaldi, G. D. et K. O. Arras. 2010, «Flirt-interest regions for 2d range data», dans *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, IEEE, p. 3616–3622. 22

Tom Krajnik, O. M. M. T. D. J. E. M. H., Jaime P. Fentanes. 2014, «Long–term topological localisation for service robots in dynamic environments using spectral maps», dans *IROS*. 16

Triebel, R., P. Pfaff et W. Burgard. 2006, «Multi-level surface maps for outdoor terrain mapping and loop closing», dans *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, IEEE, p. 2276–2282. 7

Ulrich, I. et I. Nourbakhsh. 2000, «Appearance-based place recognition for topological localization», dans *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 2, Ieee, p. 1023–1029. 41

Winterhalter, W., F. Fleckenstein, B. Steder, L. Spinello et W. Burgard. 2015, «Accurate indoor localization for rgb-d smartphones and tablets given 2d floor plans.», . 26

Wu, C.-J. et W.-H. A. A. Tsai. 2009, «Location estimation for indoor autonomous vehicle navigation by omni-directional vision using circular landmarks on ceilings», *Robotics and Autonomous Systems*, vol. 57, n° 5, p. 546–555. 14

Wurm, K. M., A. Hornung, M. Bennewitz, C. Stachniss et W. Burgard. 2010, «Octomap : A probabilistic, flexible, and compact 3d map representation for robotic systems», dans *Proc. of the ICRA 2010 workshop on best practice in 3D perception and modeling for mobile manipulation*, vol. 2. 7

Yang, B., L. Fang, Q. Li et J. Li. 2012, «Automated extraction of road markings from mobile lidar point clouds», *Photogrammetric Engineering & Remote Sensing*, vol. 78, n° 4, p. 331–338. 18

Zhang, J. et S. Singh. 2014, «Loam : Lidar odometry and mapping in real-time», dans *Robotics : Science and Systems Conference (RSS)*. 18

Zindler, K., N. Geiß, K. Doll et S. Heinlein. 2014, «Real-time ego-motion estimation using lidar and a vehicle model based extended kalman filter», dans *ITSC*. 23

Chapitre 2

Méthodes expérimentales employées

« *Le fossé séparant théorie et pratique paraissait moins large en théorie qu'il ne l'était en pratique* »

Un inconnu

Sommaire

La mise au point d'algorithmes de localisation nécessite de nombreuses expérimentations afin d'évaluer chaque sous-ensemble. Dans ce cadre, il est important de mettre en place une méthode facilitant ces essais réels et simulés. Le choix entre réel et virtuel sera réalisé en fonction des limitations qu'implique chaque monde. La validation de la performance globale implique quant à elle la mise en place d'outils d'expérimentation fournissant des données de référence. Dans ce chapitre, nous allons aborder la méthodologie développée au sein de la *Plateforme de Navigation Autonome* de l'IRSEEM, ainsi que les moyens d'essais qui y sont associés.

Ces outils développés dans le cadre de mon poste de Responsable d'ingénierie ont été utilisés tout au long de ce travail de thèse. Ils seront repris dans les deux chapitres suivants qui couvrent mes principaux travaux. Deux exemples d'applications seront tout de même présentés à la fin de ce chapitre : le projet *Quasper* (section 2.3.1) réalisé en 2012-2013, et le projet *Navalis* (section 2.3.2) réalisé de manière concomitante avec le démarrage de ma thèse.

Dans ce chapitre, nous proposons deux principales contributions. La première consiste en une méthodologie de développement en adéquation avec nos moyens d'essais. Elle est utilisée pour l'ensemble de ces travaux de thèses des chapitres 3 et 4. La deuxième contribution porte sur le développement d'un coffre de toit de référence pour les expérimentations extérieures. Ce moyen d'essai est utilisé comme vérité terrain dans les travaux du chapitre 4 et de l'annexe B.

2.1 Méthodologie employée

Les systèmes robotiques deviennent de plus en plus complexes, ce qui peut rendre leur développement plutôt hasardeux sans méthodologie. Partant du constat que pour des raisons de complexité et de coût de mise en œuvre, il est rarement envisageable de développer directement sur le prototype final, nous avons structuré une méthodologie présentée figure 2.1, permettant des qualifications par étapes sans disposer du matériel. Elle reprend des approches assez classiques pour aborder des développements complexes, et se décompose en trois étapes principales :

— Simulation

— Tests à échelle réduite

— Test pleine échelle

Dans la suite de ce document, nous exposons les avantages et les limites inhérentes à chacune de ces étapes et nous présentons les moyens d'essais associés mis en place au laboratoire.

Simulation : De nos jours, les apports de la simulation ne sont plus à démontrer. Nous pouvons citer entre autres :

1. La maîtrise parfaite de l'environnement
2. L'étendue des cas testables seulement limitée par notre imagination
3. Une vérité terrain absolue
4. Une abstraction vis-à-vis des capteurs réels
5. Facilité de mise en oeuvre de scénarios de test et donc accélération des cycles de développement

Les points 1 et 2 permettent de tester des situations associées à des environnements difficiles à reproduire en réalité, comme par exemple la rencontre de deux véhicules dans un carrefour urbain par temps de brouillard.

Concernant le point 3, en simulation nous disposons à tout moment d'une vérité terrain parfaite. Le fait par exemple de connaître la position exacte d'un capteur est une aide précieuse au développement. *A contrario*, la mesure de la vérité terrain en condition réelle fait appel à des équipements onéreux. Elle nécessite des compétences spécifiques et des méthodes de calibration pour déterminer des variables inconnues. Ces différents points seront abordés dans la section 2.2 puis dans le chapitre 3.

Le point 4 peut être à la fois un avantage et un inconvénient : en simulation, nous pouvons tester plusieurs types de capteurs très facilement, par exemple évaluer des champs de vue de LiDARs ou de caméras sans disposer du matériel ou voire même sans avoir identifié de référence précise chez un fournisseur. Mais d'un autre côté, la précision des modèles utilisés peut rapidement être limitante. Par exemple, la majorité des outils de simulation ne simule que très sommairement un LiDAR. Ils nous donnent une « vue » instantanée de la distance avec l'environnement 3D pour chaque raie (méthode dite du *Zbuffer*). Ils ne tiennent compte ni de la durée d'un *scan*, qui implique une déformation des mesures dues au déplacement du véhicule, ni de la forme conique du faisceau laser. Il est à noter que rentrer dans un tel niveau de prise en compte de phénomènes physiques introduit pourtant une forte connaissance du produit, et ne peut guère s'envisager sans un partenariat avec le fabricant. Dans le cas particulier du LiDAR, des outils plus spécialisés comme *Blensor*[1] permettent de se rapprocher un peu plus des capteurs réels. Mais le plus souvent, ils génèrent des données pour une trajectoire prédéfinie, il n'est donc plus possible de les faire interagir dans une boucle de contrôle.

Et enfin, le point 5 est la conséquence des quatre points cités ci-dessus. Il est beaucoup plus rapide de travailler en simulation que de se confronter à des expérimentations réelles. Avec une bonne maîtrise des limitations des outils, la simulation permet d'aller relativement loin dans les développements, et d'explorer rapidement plusieurs voies pour en déterminer les plus prometteuses.

Tests à échelle réduite : Comme nous l'avons vu plus haut, la simulation a ses limitations. Nous disposons donc au sein du laboratoire de moyens d'expérimentation à petite échelle, nous permettant d'expérimenter des sujets orientés perception dans le cadre de la robotique et du véhicule autonome. Nous y réalisons des maquettages permettant la représentation de l'environnement à échelle réduite et la reproduction de mouvements de véhicules à l'aide d'une flotte de robots mobiles. Une structure logicielle commune, *RTMaps* (Real Time, Multisensor applications), présentée section 2.1.1.2, fait le lien entre les différents systèmes. Elle interagit aussi bien avec les outils de simulations, les moyens de référence et les robots.

Nous disposons d'une numérisation 3D de l'environnement du laboratoire qui, couplée avec les moyens de maquettage et de développement présentés précédemment, permet de mixer simulations et tests réels pour améliorer les capacités d'expérimentation avec la réalité augmentée. Par exemple, la simulation nous permettra d'émuler la caméra d'un robot réel se déplaçant dans le laboratoire afin de tester des scénarios complexes, comme l'évitement de collision, le véhicule arrivant en face étant alors simulé. Une vue du laboratoire numérisé avec un robot partiel-

1. http://www.blensor.org/

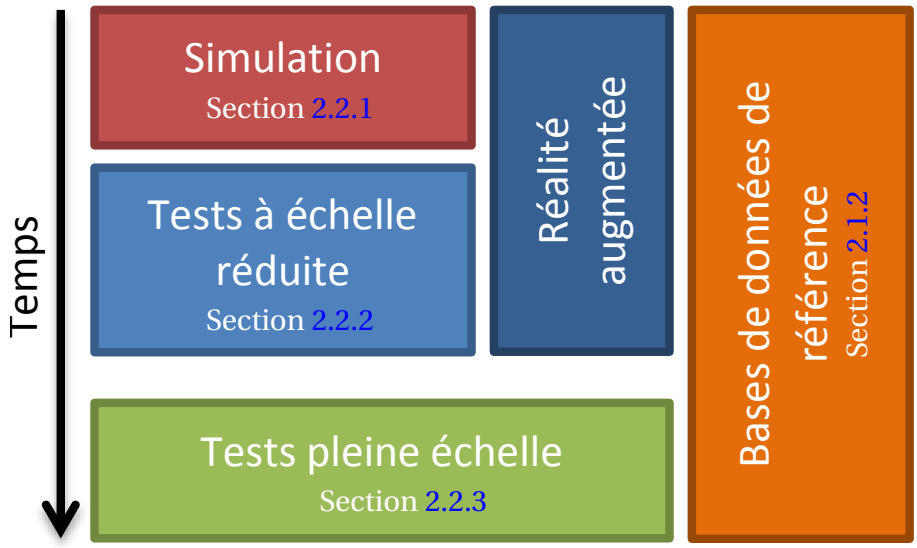


FIGURE 2.1 – Étapes de la méthode de développement utilisée

lement simulé est présentée sur la figure 2.5.

Même sans tirer parti de la réalité augmentée, il est possible de faire des essais à petite échelle représentatifs de la réalité en mettant en place une couche d'abstraction permettant de « s'affranchir » des modèles de véhicules. Il est toujours plus facile et productif de faire des tests dans un premier temps sur des robots mobiles au sein même du laboratoire que sur des véhicules routiers en louant des pistes d'essais externes. Un autre exemple : dans le cadre de la mise au point d'une calibration caméra centrale inertielle, il est plus facile de faire évoluer un robot de quelques kilogrammes dans toutes les directions plutôt qu'un véhicule réel.

Tests pleine échelle : (*échelle 1*, ou encore *grandeur réelle*) Une fois les deux étapes précédentes déroulées, il est nécessaire de pouvoir tester et évaluer la méthode proposée sur le système réel. Si l'ensemble des modèles et abstractions d'échelle a été correctement réalisé, le portage sur système réel est relativement rapide.

La différence entre échelle réduite et réelle peut être variable et dépend surtout de l'application envisagée. Il peut s'agir de l'environnement d'expérimentation que nous avons réduit comme dans le cas des essais du projet *Argos* (figure 2.8), ou bien encore le mobile lui-même comme le véhicule *Quasper* (section 2.3.1) que nous avons « émulé » par un robot *Wifibot*.

Par expérience, sur un développement, nous passerons la majorité du temps dans le laboratoire sur les expérimentations à échelle réduite.

Devant le coût humain et la complexité de certaines expérimentations réelles, il est indispensable d'enregistrer des bases de données de référence pour poursuivre le développement en parallèle des simulations et des essais échelle réduite (figure 2.1). Nous aborderons ce point à la section 2.1.2. Dans un premier temps, nous allons évoquer l'intérêt de développer autour d'une structure logicielle commune que nous dénommerons *framework* dans la suite de ce document.

2.1.1 Framework Commun

2.1.1.1 Intérêts d'un framework en robotique

Aujourd'hui, la plupart des développements robotiques sont réalisés autour d'un framework. Généralement, ces outils sont basés sur des modules multi-threads ou multi-processus, associés à des mécanismes d'échange et de synchronisation de données. Ces environnements de développement présentent plusieurs avantages, aussi bien au niveau du temps de développement que de la structuration de l'application :

Ne pas développer l'ensemble de l'application : de nombreux algorithmes sont déjà proposés dans ces environnements. Un point efficient au quotidien concerne les deux extrémités de toute chaîne de traitement qui sont déjà développées :

— les drivers d'interfaçage avec le matériel

— les outils de visualisation

De plus, il est à noter que ces deux aspects nécessitent souvent des compétences particulières pour leur développement. Le développeur va donc pouvoir concentrer son énergie sur son application ou son nouvel algorithme.

Datation des données et enregistrement / re-jeu : les applications robotiques sont très couramment multi-sources de données : caméra, IMU, LiDAR, odométrie ... Chaque donnée est horodatée (*timestamp*), ce qui permet d'assurer leur synchronisation lors des traitements. De plus, des mécanismes permettent très facilement d'enregistrer des flux de données et de la rejouer de manière synchrone. Ce dispositif est très intéressant pour continuer à développer sans le matériel sur des données enregistrées au laboratoire. Il fait réellement gagner un temps précieux dans de nombreux cas.

Capitalisation des fonctions logicielles : le fait de développer sous forme de composant permet de capitaliser facilement les fonctions développées par différentes personnes, particulièrement auprès des doctorants et des autres personnels non-permanents du laboratoire.

Facilitation du travail en équipe et de l'intégration : le cadre normalisé que l'outil propose pour les entrées/sorties facilite la décomposition en sous-fonctions unitaires. De même, si les développeurs se sont entendus sur les données à échanger, les mécanismes de synchronisation faciliteront grandement l'intégration.

Normalisation du code : le cadre du framework étant commun à tous, il est plus aisé de reprendre un développement d'un tiers au sein du laboratoire.

Adapté aux architectures multi-coeurs : le principe basé sur des composants multi-threads ou multi-processus est « naturellement adapté » aux processeurs multi-coeurs. Chaque composant s'exécutera en parallèle, sans se préoccuper d'optimisation particulière ou des difficultés de la programmation multi-threads.

Avec la complexité des systèmes en jeu en robotique, ces frameworks (ou couramment dénommés *Robotics middleware*) sont devenus « indispensables » à tout développement. Ils permettent une couche d'abstraction, afin de focaliser son énergie sur la mise au point de l'algorithme.

Il existe plusieurs solutions possibles, la plus connue d'entre elles est sans conteste *ROS*[2] (*Robot Operating System*). Elle est développée par la société *Willow Garage* depuis 2007 autour de son robot *PR2* qui est issu du laboratoire *Stanford Artificial Intelli-*

2. http://www.ros.org/

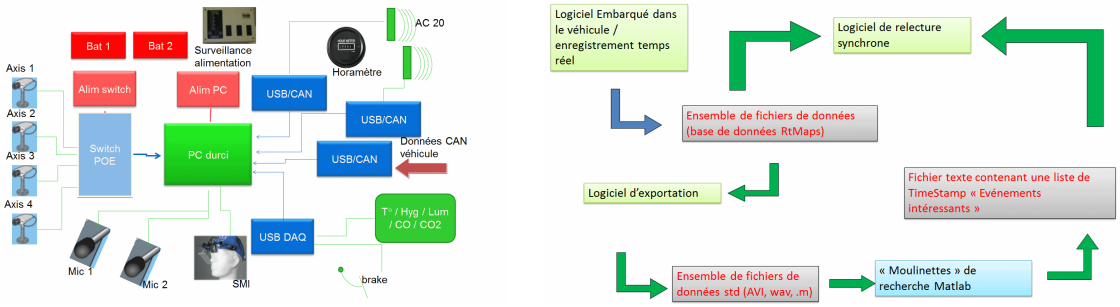
gence. Depuis, de nombreuses plateformes robotiques sont compatibles, c'est même devenu une nécessité pour un fabricant de proposer des drivers *ROS* s'il veut vendre un robot destiné à la recherche. De nombreuses implémentations d'algorithmes fonctionnant en temps réel sont également disponibles avec ce framework. Il est à noter également que la société *Willow Garage* maintient la librairie très connue de vision par ordinateur *OpenCV*[3] (*Open Computer Vision*). Comme tout outil complet et complexe, *ROS* nécessite un certain temps de prise en main.

Il existe d'autres frameworks comme *ADTF*[4] plus utilisé dans le milieu automobile ou encore *Effibox*[5] qui est une émanation du laboratoire robotique de l'*Institut Pascal*.

Pour notre part, hormis quelques développements plutôt à la marge sous *ROS*, nous utilisons principalement *RTMaps*[6] (*Real Time, Multisensor applications*) édité par la société *Intempora* qui est une émanation du laboratoire de robotique des Mines-ParisTech. Comme nous le verrons tout au long de ce chapitre, ce framework s'applique à nos trois échelles (simulation, réduite, échelle 1).

2.1.1.2 Framework *RTMaps*

Cette section n'a pas pour but de faire un cours sur les frameworks de robotique ni sur *RTMaps*, mais simplement d'aborder les concepts permettant de mettre en avant les intérêts présentés en section 2.1.1.1. Pour le lecteur intéressé, j'enseigne depuis plusieurs années autour de l'outil *RTMaps* au sein de l'*Esigelec* dans les domaines de l'instrumentation et de la robotique. Ces cours ainsi que d'autres tutoriaux sont disponibles sur le site web d'Intempora[7]. Le premier projet de l'IRSEEM avec le logiciel *RTMaps* fût l'instrumentation d'un véhicule figure 2.2) pour le *LPC* (*Laboratoire de Psychologie de la Conduite*) de l'Inrets (*Institut National de Recherche sur les Transports et leur Sécurité*)[8] en 2008.



(a) Synoptique des capteurs et de leurs connexions du véhicule instrumenté : 4 caméras, bus can, 2 radars, occulomètre...

(b) Logiciels développés autour du véhicule pour l'enregistrement et le re-jeu temps réel des données

FIGURE 2.2 – Synoptiques capteurs et logiciels du Renault Modus instrumenté pour l'Inrets en 2008 à l'aide du framework *RTMaps*

RTMaps propose une représentation et un outil graphique pour constituer les diagrammes. Les diagrammes regroupent les composants logiciels exécutés dans des threads

3. http://opencv.org/
4. https://www.elektrobit.com/products/eb-assist/adtf/
5. http://effistore.effidence.com/up/effibox.html
6. https://intempora.com/products/rtmaps.html
7. https://intempora.com/
8. Laboratoire renommé depuis *Laboratoire de psychologie des comportements et des mobilités*, et l'Inrets a fusionné en 2010 avec le *LCPC* (*Laboratoire Central des Ponts et Chaussées*) pour devenir l'*Ifsttar* (*Institut Français des Sciences et Technologies des Transports, de l'Aménagement et des Réseaux*)

séparés et les connexions de données entre les différents composants. Un exemple est présenté sur la figure 2.3. Il s'agit de l'évaluation en simulation de la localisation LiDAR présentée au chapitre 3. Le robot différentiel et le LiDAR sont simulés par les deux composants dans la zone grise. La localisation est réalisée dans la zone verte, comprenant la conversion des impacts LiDAR en points 3D et la localisation à proprement parler.

Dans ce cas, il n'y a pas vraiment d'interface capteur à gérer si ce n'est le joystick de contrôle. Par contre, nous bénéficions pleinement des outils de visualisation (zone rouge sur la figure), et le développeur peut concentrer son énergie sur les composants le concernant directement. Pour faire un essai réel, il suffira de remplacer les composants de simulation par les blocs d'acquisition des capteurs ; dans ce cas : commande/odométrie du robot, et LiDAR.

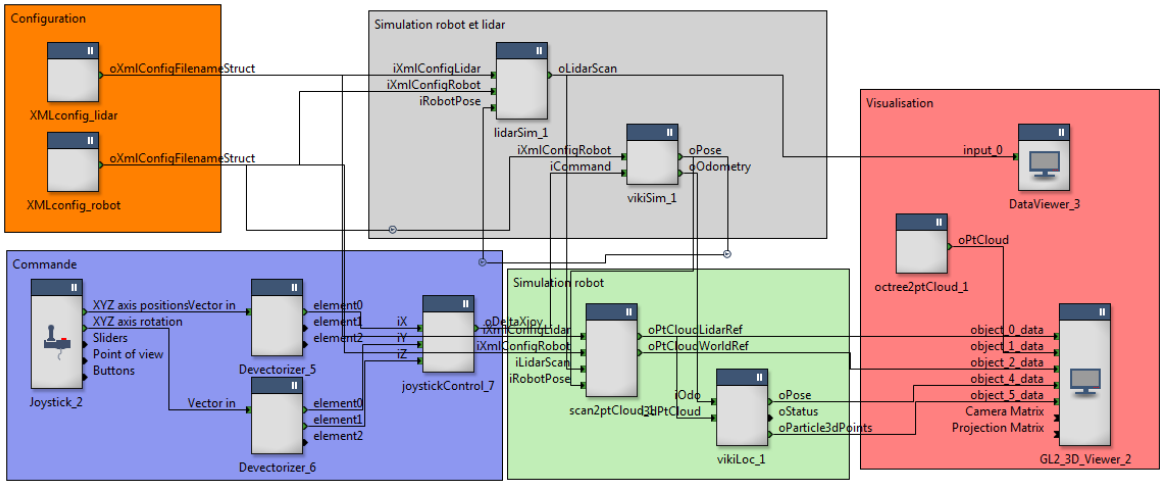


FIGURE 2.3 – Exemple de diagramme *RTMaps* pour évaluer en simulation la localisation proposée dans le chapitre *Argos*. Les blocs représentent les composants logiciels exécutés chacun dans un thread, et les fils de connexion matérialisent les échanges de données.

Cette architecture modulaire autour de composants, ainsi que la normalisation des modes d'échanges de données permettent de faciliter la décomposition de fonctions complexes en sous-parties clairement identifiées. Le travail multi-développeurs s'en trouve alors plus aisé, du moment qu'au démarrage nous nous soyons accordés sur la manière d'échanger les données. Par exemple, lors de l'automatisation du véhicule *Quasper* de l'*Ifsttar* (section 2.3.1), l'intégration et les essais des fonctions développées par l'IRSEEM et le Lemco se sont déroulés sur trois semaines.

L'aspect modulaire permet de s'adapter très rapidement à d'autres cibles. Lors du projet *Argos*, tous les développements ont démarré sur le robot *Jaguar* et ont été transférés en deux jours sur le robot *Viking* en avril 2015.

Comme nous l'avons déjà évoqué, un autre point très efficace en terme de temps de développement est l'enregistrement et le re-jeu de données. Par exemple, il est possible de réaliser en laboratoire un enregistrement de l'odométrie du robot réel et des données LiDAR. Pour cela, sur le diagramme de la figure 2.3, il suffira d'ajouter le composant d'enregistrement générique et d'y connecter les flux qui nous intéressent ; dans notre cas au moins l'odométrie du robot et les scans LiDAR. Chaque donnée va être horodatée puis enregistrée. Au re-jeu, à partir du diagramme de la figure 2.3, nous ne conservons que la partie localisation et visualisation auxquelles nous ajoutons le composant de re-jeu. A l'exécution, les flux vont être extraits de la base de données, et regénérés de manière synchrone en respectant le *TimeStamp* de chaque donnée.

Cette manière de travailler est très efficace, nous pouvons re-jouer l'expérimentation autant de fois que nécessite la mise au point de l'algorithme. De ce fait, nous pouvons travailler sur des données réelles en-dehors du laboratoire en respectant les notions de synchronisation et de temps réel.

Comme nous allons le voir dans la section suivante, cette manière de développer n'est pas seulement indispensable pour faciliter le développement mais permet également de comparer les algorithmes sur des bases de données communes. Depuis maintenant plusieurs années, de nombreuses bases de données de référence sont proposées par des laboratoires à des fins d'évaluation.

2.1.2 Bases de données de référence

Comme nous l'avons abordé plus haut, les bases de données de référence sont essentielles au moins sur deux aspects :

— la mise au point d'algorithmes, où elles permettent de ne pas dépendre des aléas des expérimentations, ainsi que de l'expertise nécessaire à leur mise en œuvre.

— la comparaison de méthodes sur des données communes, où comme leur nom l'indique, elles permettent de mettre à disposition un jeu de données indépendant pour une comparaison la plus impartiale possible.

Il existe de nombreuses bases de données de référence dans plusieurs domaines. Dans le domaine des véhicules, la plus connue est la base *KITTI Vision Benchmark Suite* développée par le Karlsruhe Institute of Technology (*KIT*). En quelques années, cette base s'est imposée comme référence pour comparer des algorithmes. Les deux articles de référence présentant ces travaux sont Geiger et collab. [2012] et Geiger et collab. [2013]. La plateforme (figure 2.4) utilisée pour l'enregistrement est équipée d'un système de position de référence qui est la fusion d'une centrale inertielle et d'un GPS RTK. Elle sert de vérité terrain pour évaluer les performances.

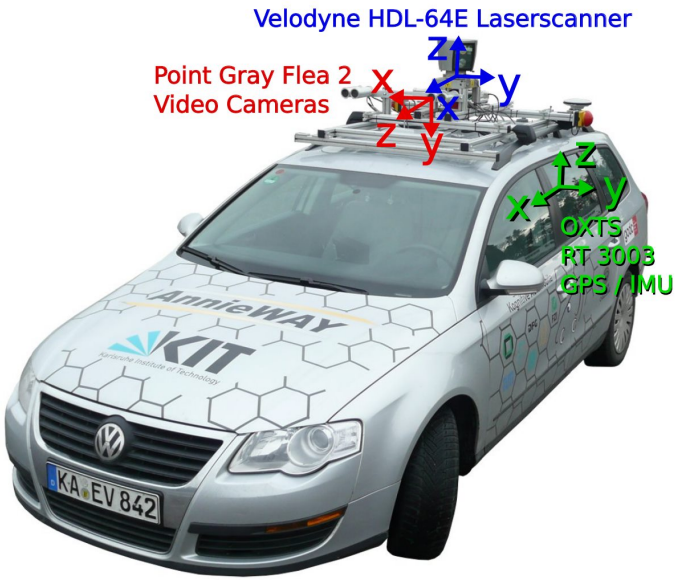


FIGURE 2.4 – Plateforme d'enregistrement des données de la base *KITTI* : VW Passat station équipée avec 4 caméras (2 couleurs et 2 noir et blanc) un LiDAR Velodyne HDL64, d'un système de positionnement de référence GPS RTK/IMU (figure issue de Geiger et collab. [2013]).

De nombreux types d'environnements différents sont disponibles : urbains denses,

périphérie de ville, voies rapides... Régulièrement des nouveaux ajouts d'enregistrements sont effectués.

La base de données est associée à un *SDK (Software development kit)* permettant d'évaluer de nombreux champs de recherche du véhicule autonome : la stéréo-vision, le flot optique, l'odométrie visuelle, la détection d'objets et leur suivi, la détection de routes, voies et marquages au sol... Chacun peut soumettre ses résultats à un serveur, et un classement en fonction des performances est proposé sur leur site [9].

Il existe bien d'autres bases de données dans le domaine du véhicule autonome avec des capteurs et environnements différents. Korrapati et collab. [2013] proposent l'*Institut Pascal Data Sets (IPDS)* enregistré sur leur campus et comprenant des images panoramiques. Ils proposent également un comparatif des principales bases de données existantes (table 2.1).

Sequence	NewCollege	Malaga	Cheddar Gorge	St Lucia Stereo	St Lucia Mul. times	KITTI	Ford	Rawseeds	IPDS
Plateforme									
Caméra omni.	**OUI**	non	non	non	non	non	**OUI**	**OUI**	**OUI**
Caméra fisheye	non	non	non	non	non	non	non	non	**OUI**
Paire stéréo	**OUI** (petite baseline)	non	**OUI** (petite baseline)	**OUI** (gde baseline)	non	**OUI** (petite baseline)	non	**OUI** (petite baseline)	**OUI** (gde baseline)
Paire à champ non recouvrant	non	non	non	non	non	non	non	non	**OUI**
Télémètres laser 2D	**OUI**	**OUI**	non	non	non	non	non	**OUI**	**OUI**
LiDAR 3D	non	**OUI**	**OUI**	non	non	**OUI**	**OUI**	non	non
Odométrie	**OUI**	non	**OUI**	non	non	non	non	**OUI**	**OUI**
IMU (bas-coût)	non	**OUI**	**OUI**	**OUI**	non	**OUI**	**OUI**	**OUI**	**OUI**
DGPS/GPS bas coût	non/non	**OUI** /**OUI**	non/**OUI**	non/non	non/non	**OUI** /non	non/**OUI**	**OUI** /non	non/**OUI**
GPS-RTK	**OUI**	non	non	**OUI**	**OUI**	non	non	non	**OUI**
INS	non	non	**OUI**	**OUI**	non	non	**OUI**	non	non
Type d'env.	Campus	Urbain	Campagne	Urbain	Urbain	Campus	Campus	Urbain	Campus
#Seq / dist. approx.	1/2 km	6/ 2+4 km [1]	1/ 32 km	1/ 10 km					6/ 18 km

TABLEAU 2.1 – Comparaison des bases de données de référence dans le domaine des véhicules autonomes (table issue de Korrapati et collab. [2013])

Un ensemble d'outils est également mis à disposition permettant d'afficher les données en fonction de la position de référence. Les travaux associés aux bases proposées dans la table 2.1 sont les suivants :

— *New College* « The new college vision and laser data set » : M. Smith et Newman [2009]

— *Malaga* « A collection of outdoor robotic datasets with centimeter-accuracy ground truth » : Gonzalez [Novembre 2009]

— *Cheddar Gorge* « The cheddar gorge data set » : R. Simpson et Revell [2011]

— *St Lucia Stereo* « Unaided stereo vision based pose estimation » : Warren et collab. [2010]

— *St Lucia multiple times* « Fab-map+ratslam : appearance-based slam for multiple times of day » : Glover et collab. [2010]

— *KITTI* « KITTI Vision Benchmark Suite » : Geiger et collab. [2012]

— *Rawseeds* : qui est un plutôt un dépôt de base de données [10]

— *IPDS* « Institut Pascal Data Sets » : Korrapati et collab. [2013]

Il existe de nombreux dépôts de données plus spécialisés dans le domaine du LiDAR. Nous pouvons citer les bases de données [11] associées à la *Point Cloud Library (PCL)*.

9. http://www.cvlibs.net/datasets/kitti/
10. http://radish.sourceforge.net/
11. http://pointclouds.org/media/

La *PCL* est une librairie de traitement de nuages de points très connue et tout comme *openCV* pour la vision ou *ROS* (section 2.1.1.1), elle est maintenue principalement par la société *Willow Garage*.

Nous pouvons également mentionner la *Robotic 3D Scan Repository*[12] de la Jacob University, qui regroupe aussi bien des données de *Kinect*, que des scans *Velodyne HDL64*.

Il existe des bases de données de référence dans tous les domaines et particulièrement un domaine qui n'est pas abordé ici, la vision par ordinateur. Les méthodes de traitement associées aux caméras sont très sujettes aux modifications d'environnement. C'est encore plus vrai si ces travaux touchent à l'apprentissage, qui par définition, nécessite beaucoup de données pour faire fonctionner le classifieur.

Je voudrais juste attirer l'attention sur le fait que la plupart du temps, afin de les rendre les plus interopérables possibles, les bases de données proposées ne sont pas associées à un framework comme *ROS* ou *RTMaps*. Nous perdons alors des notions d'implémentation fine, comme la synchronisation entre capteurs, et donc nous nous éloignons quelque peu des considérations du matériel. Le portage d'un algorithme donnant satisfaction sur la base de donnée vers le système réel risque de nécessiter quelques travaux supplémentaires pour fonctionner correctement.

Nous verrons dans l'étude des chapitres 3 et 4, que même si nous ne les avons pas forcément diffusées de manière large, nous avons développé de nombreux jeux de données de référence en intérieur comme en extérieur pour mener à bien nos projets. C'est en cela que nous allons maintenant aborder les équipements disponibles dans la plateforme de navigation autonome de l'IRSEEM.

2.2 Moyens de référence

Comme nous l'avons abordé lors de la présentation des différentes étapes de la méthode de développement résumées à la figure 2.1, il est indispensable de disposer d'une vérité terrain permettant de qualifier chaque sous-partie.

Dans cette section, nous allons aborder les moyens à disposition autour de la plateforme de navigation autonome de l'IRSEEM pour les trois échelles de développement, et permettant également de constituer des bases de référence. Cette présentation des moyens d'essai reprend la même structure que la méthodologie proposée : simulation, échelle réduite et pleine échelle.

2.2.1 Moyens de simulation

Pour la simulation, nous utilisons trois types d'outils, qui dépendent de l'ampleur du projet et surtout de l'avancement de son développement :

— Du développement rapide sous des outils de calcul numérique comme *Matlab*, permettant de tester rapidement une idée sur un point précis. Même en s'efforçant d'utiliser un maximum le calcul vectoriel, les temps de calcul deviennent rapidement trop conséquents. Par exemple, la simulation d'un LiDAR 3D de type *VLP16* (figure 1.7b) générant 5760 points dans le modèle 3D du laboratoire : sous *Matlab* la simulation d'un scan dure 380s sur un cœur de CPU (intel i7-4700MQ @ 2.4GHz), alors que le même simulateur sous *RTMaps* utilise 36ms de temps processeur (sur les 8 cœurs du CPU). Bien sûr, avec *Matlab*, les optimisations classiques ont été appliquées en déroulant un maximum de boucles, et l'environnement est stocké

12. http://kos.informatik.uni-osnabrueck.de/3Dscans/

dans une matrice 3D. De son côté, le composant *RTMaps* est multi-threadé, il y aurait donc un facteur 8 à retirer sur l'écart de performance. J'ai volontairement pris un exemple marquant du fait que cette simulation soit consommatrice de temps de calcul, car elle nécessite un lancer de rayon (*ray-tracing*) dans l'environnement pour chaque raie LiDAR.

Malgré ces contraintes, les outils de visualisation et de mise au point proposés par des outils comme *Matlab* sont très efficaces et apportent un réel avantage lors des essais préliminaires afin d'évaluer rapidement de nombreuses pistes de solution.

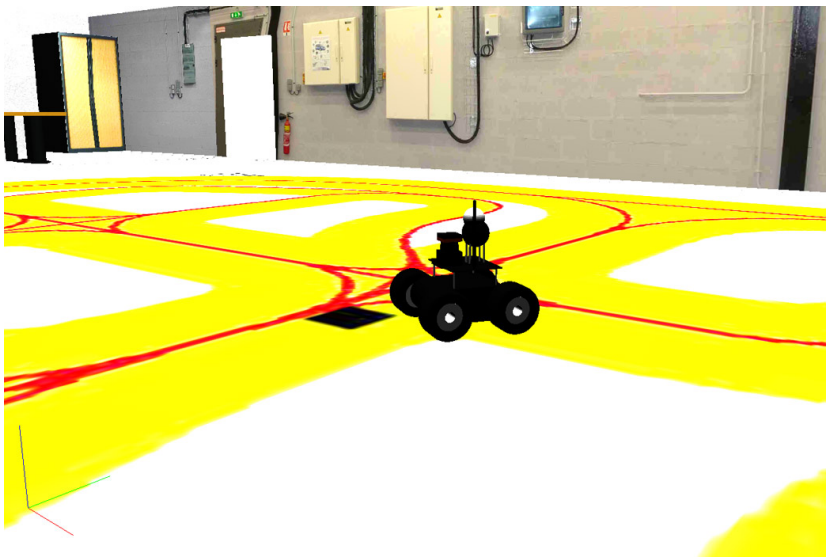


FIGURE 2.5 – Outils de simulation propriétaires basés sur *RTMaps* : simulation d'un robot de type *Wifibot* (figure 2.13a), dans l'environnement numérisé du laboratoire (figure 2.8)

— Développement propriétaire sur des outils comme *RTMaps*. La figure 2.5 présente la simulation du robot et de son LiDAR pour tester la localisation du projet Argos. Comme vu dans l'exemple précédent, le temps de calcul est sans commune mesure avec des langages interprétés. Par contre, le développement est nettement plus long et nécessite des connaissances spécifiques. L'avantage réside dans la génération des données avec des formats identiques à ceux du capteur réel. Le composant est alors intégrable directement dans notre chaîne de développement sans aucune modification ; ce qui facile l'intégration et les multiples passages de la simulation à la réalité.

— Des outils spécialisés comme *ProSivic*[13] ou *Gazebo*[14]. Ce sont des outils génériques de simulation 3D. *ProSivic* est plus orienté vers les applications véhicules routiers avec une forte composante dans la simulation réaliste de caméras, et *Gazebo* est plutôt destiné à la robotique. Ils permettent une simulation globale de la scène, de manière à pouvoir émuler le comportement de différents capteurs. Comme nous l'avons abordé succinctement dans le chapitre précédent, dans de nombreux cas la modélisation du capteur est difficile car elle dépend fortement des traitements développés par le fabricant dans l'électronique interne du capteur (par exemple les LiDARs, les radars hyper fréquence...). La simulation reste donc un outil approximatif de la réalité, il faudra procéder à des essais réels pour valider un système.

13. http://www.civitec.com/
14. http://wiki.ros.org/gazebo

Dans certains cas il existe des bancs dédiés facilitant la modélisation d'un capteur de perception. Citons l'exemple du banc *DXO* (figure2.6) qui permet l'identification de nombreux paramètres d'un système couplant imageur et objectif (étalonnage géométrique et radiométrique, rapport signal à bruit,...).

FIGURE 2.6 – Banc de calibrage de caméras *DXO*. Les paramètres intrinsèques extraits du capteur (au premier plan) nous permettent de paramétrer le simulateur *Pro-SIVIC*

Nous avons abordé une sous-partie de la robotique, avec l'aspect perception et la simulation de capteurs. Mais dans une chaîne de traitement globale (perception, planification, contrôle), il faut également être capable d'interagir avec l'environnement. Les deux simulateurs présentés proposent des moteurs physiques pour interagir avec les objets de la scène en gérant par exemple les collisions. Mais ils proposent également des modèles de véhicule, ou des modèles roues contact au sol permettant de commander une plateforme.

Ces simulateurs de haut niveau disposent d'interfaces logicielles permettant une interaction directe à travers les différents frameworks comme *ROS* ou *RTMaps*. Par exemple entre *RTMaps* et *ProSivic*, il existe des composants logiciels pour faire l'acquisition des mesures capteurs, et d'autres pour interagir avec l'environnement et commander les mobiles. Il est alors tout à fait possible de tester la chaîne complète de son application en simulation (figure 2.7).

De par leur polyvalence, ces derniers outils sont plus complexes à mettre en œuvre que des solutions propriétaires qui par définition sont très bien maîtrisées. Un deuxième désavantage, ces outils sont généralement très gourmands en ressources de calcul et nécessitent une carte graphique véloce. Mais en restant modéré sur la dimension de la scène et avec un temps de formation raisonnable, vous avez accès à des simulations nettement plus complètes et souples pour de futures évolutions.

2.2.2 Moyens pour les essais à échelle réduite

Les moyens d'essai à échelle réduite (section 2.1) sont essentiellement des moyens d'essais intérieurs. Ce point dépend forcément de l'échelle de l'application finale. Si nous parlons de robotique intérieure, le mobile sera déjà à l'échelle 1...

FIGURE 2.7 – Interaction entre le frameworks *RTMaps* et l'outils de simulation *ProSivic* lors d'un test de la chaîne contrôle complète du véhicule automatisé *Quasper* (travaux menés en partenariat avec Jean-Christophe SMAL du laboratoire *Lemco* de L'*Ifsttar*).

Ils sont regroupés au sein de la plateforme du laboratoire (figure 2.8). Elle est destinée à recevoir des essais, qui peuvent prendre la forme de maquettes de l'environnement, comme dans le cas de la structure pétrochimique factice installée durant le *challenge Argos* (figure 2.8).

Le moyen immobilier est un espace de $150m^2$ ($15m \times 10m$), avec une hauteur sous plafond de $6m$. Il est équipé d'un éclairage contrôlé pour en maîtriser les conditions lumineuses (de 0 à 10000 lux), ainsi que plusieurs moyens de référence que nous allons présenter ci-dessous.

2.2.2.1 Système de capture du mouvement *Vicon*

Le laboratoire est équipé de deux systèmes de capture du mouvement du fabricant *Vicon* (figure 2.9). Ils servent essentiellement de vérité terrain pour localiser nos plateformes mobiles :

— 20 caméras *T40S* (figure 2.9a), installées de manière fixe aux murs du laboratoire ou sur les maquettes d'essai selon le besoin.

— 5 caméras *Bonita* (figure 2.9b), beaucoup plus légères en terme d'installation et de mise en œuvre. Ce dispositif peut rapidement être installé sur des trépieds afin de couvrir une petite zone pour une expérimentation spécifique, comme l'habitacle d'un véhicule.

Ces systèmes localisent dans l'espace des marqueurs réfléchissants par triangulation entre différentes caméras. La figure 2.10 montre plusieurs dimensions de ces marqueurs dont l'usage dépend du type de mobile et de la distance aux caméras. Toutes les caméras sont synchronisées entre elles et équipées d'un anneau de leds infrarouges qui illumine la scène de manière synchrone. Chaque caméra assure les opérations de segmentation et de calcul du centroïde de chaque marqueur présent dans son champ de vue. Le réseau ne communique que les différentes positions en pixels entre les caméras et le PC de contrôle. Le PC assure les opérations de triangulation, labellisation de chaque marqueur et reconstruction d'un modèle prédéfini.

FIGURE 2.8 – Plateforme de navigation autonome en décembre 2015, équipée avec la maquette du *Challenge Argos*. Des panneaux d'éclairages installés au plafond permettent de contrôler les conditions lumineuses de l'environnement.

(a) Caméra d'acquisition *Vicon T40S*, 4 Mega pixels, 515Hz, installées à demeure dans le laboratoire.

(b) Caméra d'acquisition *Vicon Bonita 10*, 1 Mega pixels, 240Hz, installables aisément sur trépieds mobiles.

FIGURE 2.9 – Systèmes de capture du mouvement Vicon utilisés comme vérité terrain au laboratoire. Les 20 *T40* sont installées sur les murs ou dans les maquettes présentes dans le laboratoire, alors que les 5 *Bonita 10* sont destinées à être installées par exemple dans l'habitacle d'un véhicule pour suivre l'attitude de la tête du conducteur.

Lors de l'installation, ces systèmes nécessitent de maximiser les champs de vue communs entre les caméras afin d'optimiser les possibilités de triangulation. Puis, il faut procéder au calibrage de la pièce, en déplaçant un objet connu composé de cinq marqueurs (*wand*) dans tout le volume. Cette opération de calibrage détermine automatiquement les paramètres intrinsèques des capteurs et leur position relative. Il reste une dernière opération consistant à définir le zéro du référentiel ainsi que son orientation.

Il est également possible de calibrer des caméras visibles dans le même repère 3D, qui permettront une superposition de l'image visible au modèle 3D reconstruit.

Les mobiles doivent être équipés d'au moins trois marqueurs en évitant au maximum les masquages par des éléments de leur structure. Il faut ensuite définir un modèle dans le logiciel (*Nexus*[15] ou *Tracker*[16]). Ce modèle permettra d'assurer un suivi même en cas de masquage d'un ou plusieurs marqueurs.

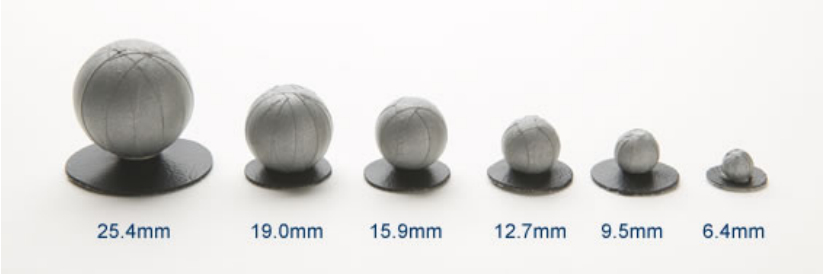


FIGURE 2.10 – Différentes dimensions de marqueurs utilisés avec le système *Vicon*

Un composant *RTMaps* a été développé pour récupérer les données issues de *Nexus* ou *Tracker* en temps réel. Un fichier XML de configuration permet de sélectionner la liste des modèles dont nous voulons faire l'acquisition, et sous quelle forme nous voulons obtenir les données (position globale, angles d'Euler, matrice de rotation, quaternions...)

Ces systèmes de capture du mouvement sont intéressants à deux titres : leur fréquence d'acquisition jusqu'à 515Hz pour les *T40S*, et leur précision. Même si la précision d'un tel dispositif dépend de la position des caméras, de la distance du mobile, de la qualité du calibrage... nous estimons l'erreur d'ordre millimétrique (typiquement 1 à 2 mm). Donc pour l'instant, leur précision est très supérieure aux autres méthodes de localisation que nous cherchons à évaluer (par exemple, la précision de la localisation LiDAR du challenge *Argos* est d'environ 2.5cm, chapitre 3).

Dans l'annexe A, nous proposons une étude complète que nous avons menée sur la précision du système Vicon en terme de répétabilité, et de précision de positionnement statique et dynamique.

2.2.2.2 LiDAR de référence Leica

Le laboratoire dispose d'un LiDAR de numérisation 3D, de type *Leica ScanStation C10* (figure 1.8a). Il nous sert principalement pour deux usages :

— Obtenir la vérité terrain pour l'évaluation d'algorithmes de reconstruction 3D (figure 2.11).

— Numérisation de nuages de points 3D, qui servent de base à la construction de cartes de références pour les algorithmes de localisation (figure 1.6).

15. http://www.vicon.com/products/software/nexus
16. http://www.vicon.com/products/software/tracker

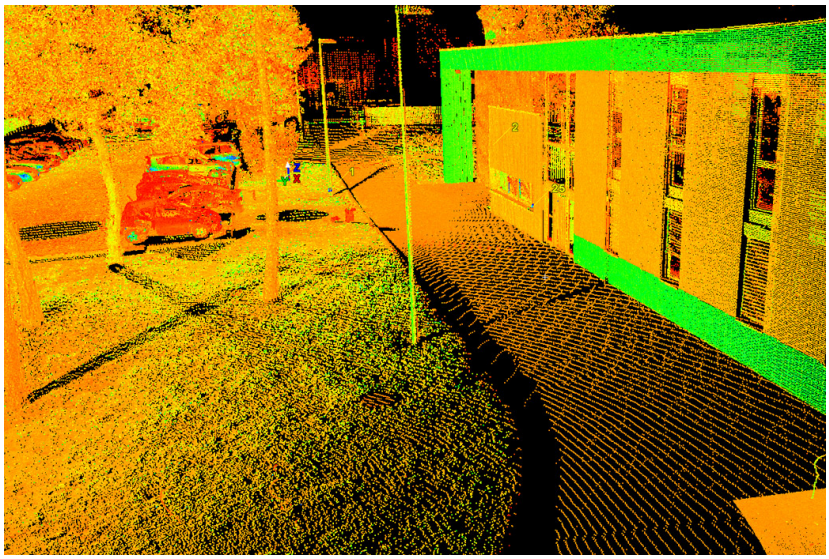


FIGURE 2.11 – Exemple de numérisation d'une façade de l'*Esigelec* obtenue avec le *Leica C10*. Les couleurs du vert au rouge correspondent à l'échelle de l'intensité de la réflectance.

Le *C10* a un champ de vue horizontal de 360˚et vertical de 270˚. Il est équipé d'une caméra, permettant de coloriser les nuages de points fournis. Le logiciel *Cyclone*[17] permet d'assembler plusieurs points de vue et de manipuler les nuages de points obtenus.

Jusque là nous avons présenté des moyens de mesures ; nous avons également des moyens de générer des mouvements que ce soit avec des robots mobiles (section 2.2.2.4) ou avec un bras robotisé que nous allons aborder ci-dessous.

2.2.2.3 Bras manipulateur

Dans le cadre du projet *Navalis* que nous présenterons brièvement dans la section 2.3.2, nous avons mis en œuvre dans le laboratoire un bras manipulateur six axes sous la forme d'un robot *Kuka KR6* (figure 2.12). Il est installé sur une base roulante, qui permet de le positionner aisément par rapport à la maquette sous expérimentation.

Sa grande précision et sa rapidité de mouvement (Table 2.2) nous permettent de faire exécuter des trajectoires de références à des capteurs comme des caméras, des LiDARs ou encore des IMU. Par exemple, toujours dans le projet *Navalis*, nous avons généré des trajectoires d'états de mer, pour représenter les mouvements de la houle que subira le LiDAR.

La version de l'armoire de contrôle du *Kuka* ne nous permet pas une communication en temps réel avec un PC. Cela nous impose deux limitations :

— Tout d'abord, il n'est donc pas possible d'obtenir la pose de l'extrémité du bras pendant l'exécution. Ce n'est pas vraiment un souci, mais implique que nous l'équipions de marqueurs *Vicon*.

— Par contre plus handicapant pour les projets d'expérimentations futures, il n'est pas possible de modifier une trajectoire en temps réel, afin par exemple d'intégrer le bras dans une boucle de contrôle. La trajectoire est forcément prédéfinie.

Même si ce système est extrêmement précis, de par les activités du laboratoire, l'essentiel des expérimentations se déroule sur robots mobiles.

17. http://www.leica-geosystems.fr/fr/Leica-Cyclone_6515.htm

FIGURE 2.12 – Bras manipulateur *Kuka* sur son support mobile, servant à l'exécution de trajectoires de références à un *LiDAR Sick* dans le cadre du projet *Navalis*. Un trièdre de marqueurs *Vicon* monté sur le LiDAR permet de contrôler la trajectoire générée.

Paramètres	**Valeurs**
Charge utile en extrémité de bras	$6kg$
Répétabilité de positionnement	$\pm 0.05mm$
Enveloppe de travail	$13.1m^3$
Point le plus éloigné atteignable	$1611mm$
Vitesse à charge max Axe 1 (sol)	$152°/s$
Axe 2	$152°/s$
Axe 3	$152°/s$
Axe 4	$250°/s$
Axe 5	$357°/s$
Axe 6 (extrémité poignet)	$660°/s$

TABLEAU 2.2 – Caractéristiques du bras manipulateur *Kuka KR6*

2.2.2.4 Flotte de robots mobiles

Comme les activités de recherche de l'équipe *IIS* (Instrumentation, Informatique et Système) portent essentiellement sur la perception embarquée et particulièrement la localisation, les robots mobiles prennent une part importante des expérimentations se déroulant dans le laboratoire. C'est donc naturellement que nous nous sommes équipés de plateformes mobiles « génériques », comme les trois *Wifibots* (figure 2.13a) ou le *Segway* (figure 2.13e). D'autres robots ont plutôt été développés dans le cadre d'un projet spécifique. C'est le cas de la chaise roulante du projet *Coalas* 2.13b) ou encore des deux robots du *Challenge Argos* : le *Jaguar* (figure 2.13c) qui nous a servi aux premiers tests et à enregistrer différentes bases de données pendant que nous développions le robot *Viking* figure 2.13d, plateforme de la compétition de Juin 2015.

L'application définit le mobile utilisé. Par exemple, les *Wifibots* ont une charge utile de quelques kilogrammes et une vitesse maximum de $1 m/s$ alors que le *Segway* peut embarquer $90 kg$ en tout terrain et $181 kg$ sur une surface lisse et tout ça jusqu'à $8 m/s$ ($29 km/h$). Le tableau 2.3 reprend l'ensemble des caractéristiques des robots du laboratoire.

Hormis le robot *Viking* qui a été conçu dans le seul but de la première compétition du *challenge Argos* de juin 2015, les autres plateformes sont génériques et les capteurs embarqués sont adaptés à l'expérimentation voulue. Cela peut aller d'une simple caméra omnidirectionnelle comme la chaise roulante du projet *Coalas*, jusqu'à embarquer l'ensemble de notre coffre de toit de référence (figure 2.16b) que nous aborderons section 2.2.3.1.

Nos plateformes embarquent tous un PC où est installé un framework « robotique » : la chaise *Coalas* utilise exclusivement *ROS*. Les *Wifibots* et le *Segway* ont *ROS* et *RTmaps* d'installés, alors que les deux robots *Argos* fonctionnent sous *RTMaps*. Cette liste semble contredire les propos de la section 2.1.1.2 comme quoi *RTMaps* est le framework déployé au sein laboratoire. Même si nous disposons de plateformes équipées de *ROS*, si nous regardons en terme de temps passé, beaucoup plus d'expérimentations se sont déroulées avec *RTMaps*, et notre « expertise » y est donc nettement plus importante.

Chaque robot sous *RTMaps* dispose d'un composant logiciel permettant son contrôle sous la forme d'un vecteur de commande $q_t = \begin{bmatrix} v & \dot{\theta} \end{bmatrix}^{\mathrm{T}}$ qui décompose la vitesse linéaire et la vitesse de rotation désirées. Ce même composant permet d'échantillonner l'odométrie et d'obtenir les mesures $u_t = \begin{bmatrix} \Delta x & \Delta\theta \end{bmatrix}^{\mathrm{T}}$, soit les déplacements linéaires et angulaires du robot pendant l'intervalle d'échantillonnage T_e. Les pendants simulés de ces composants ont également été développés. Ils disposent exactement des mêmes interfaces, ce qui rend extrêmement simple un passage entre simulation et robot et *vice versa*.

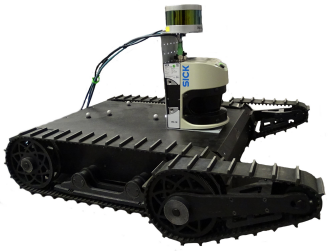
Le *Segway* et le *Viking* sont des robots avec un indice de protection permettant de fonctionner par « toutes conditions climatiques ». Mais de par leur usage en milieu souvent confiné, comme les milieux industriels ou « hypers urbains », j'ai choisi de les classer avec les moyens intérieurs. Nous allons aborder maintenant des moyens d'essais permettant d'obtenir une vérité terrain dans de vastes environnements extérieurs pour les tests à pleine échelle, à la manière de ce que nous obtenons avec le *Vicon* dans le laboratoire.

2.2.3 Moyens d'essai pleine échelle

Comme nous l'avons abordé ci-dessus, l'IRSEEM a regroupé à travers la plateforme de navigation autonome un certain nombre de moyens de référence permettant d'apporter une vérité terrain pour une expérimentation à échelle réduite. J'ai regroupé dans cette

(a) Plateforme Mobile *Wifibot*, équipée d'un LiDAR *Hokuyo UTM30LX* et d'une centrale d'attitude *VectorNav VN-100*.

(b) Chaise roulante automatisée du projet *Coalas*.



(c) Robot *Jaguar* équipé pour la comparaison localisation LiDAR mono-nappe (*LMS511*) VS multi-nappes (*VLP-16*).

(d) Robot *Viking* du challenge *Argos*, en configuration proche de la compétition de Juin 2015.

(e) Plateforme extérieure, 100kg de charge utile, *Segway RMP440*

FIGURE 2.13 – Principales plateformes mobiles de type différentielles du laboratoire.

Plateforme	**Framework**	**Charge utile** *(kg)*	**Vitesse max** *(km/h)*	**Précision odmétrique** *(points/m)*	**Protection**
WifiBot	ROS/RTMaps	3	3.6	5480	na
Chaise Coalas	ROS	130	10	2540	na
Jaguar	RTMaps	15	5.5	736	na
Viking I 2015	RTMaps	20	3.8	12788	IP67/puis ATEX
Segway	ROS /RTMaps	181	29	29680	IP67

TABLEAU 2.3 – Récapitulatif des plateformes mobiles du laboratoire

section les moyens de référence du laboratoire autour du sujet du véhicule autonome et de la mobilité extérieure. La notion échelle 1, dépend bien sûr de l'application finale.

Afin de proposer un outil facilitant la mise en œuvre des moyens de mesures externes, ils ont été regroupés dans un coffre de toit de référence.

2.2.3.1 Coffre de toit de référence

Nous avons souhaité doter le laboratoire d'un système de référence pour le domaine du véhicule autonome et de la mobilité extérieure. J'ai donc développé ce coffre de toit de référence la première année de ma thèse.

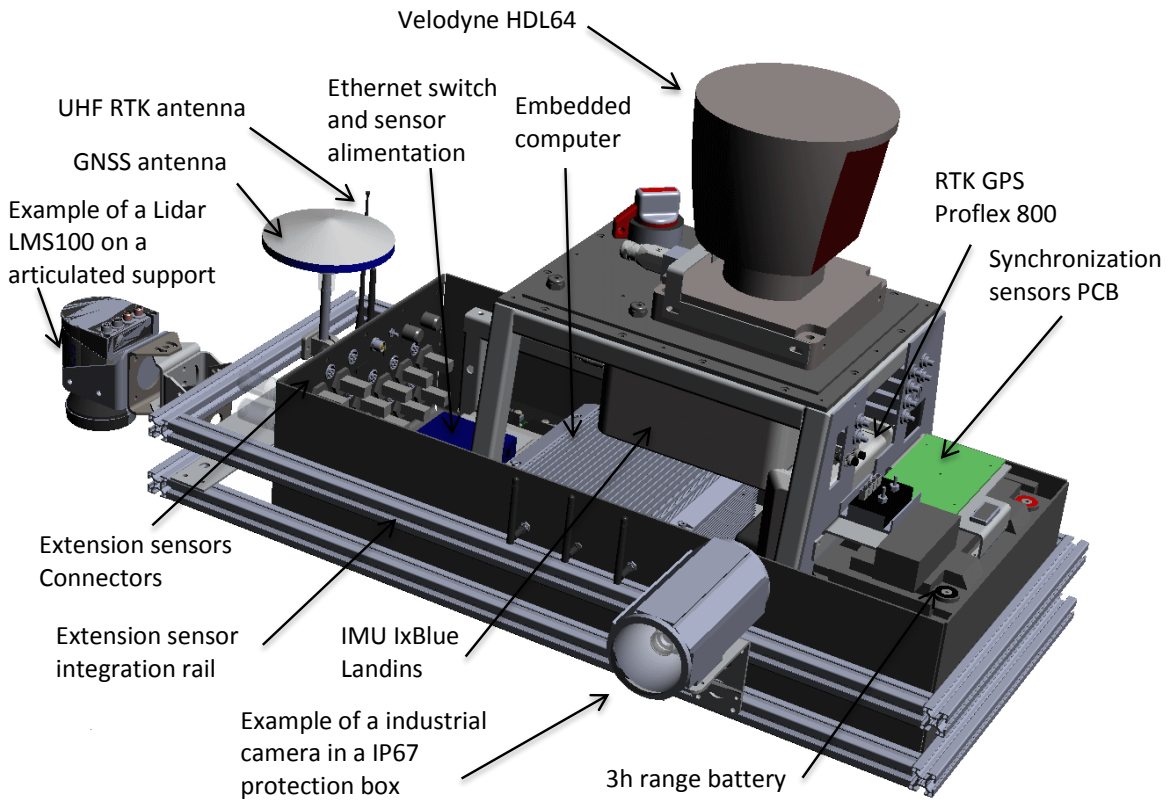


FIGURE 2.14 – Synoptique des instruments contenus dans le coffre de toit de référence développé par l'IRSEEM

Comme le montre la figure 2.14, il regroupe plusieurs instruments installés à demeure dans l'équipement lui-même :

- Une centrale inertielle basée sur une technologie de gyroscopes à *FOG* (Fibre Optic Gyroscope) : IMU Landins IxBlue
- Un GPS RTK différentiel Proflex 800
- Un Odométre Peiseler installé sur la roue arrière
- Un LiDAR multi-nappes HDL64e
- PC embarqué i7-3610, SSD 1 To

La première fonction de ce coffre de toit est de fournir une localisation de référence du véhicule porteur, et la deuxième d'obtenir une vérité terrain 3D (figure 2.15). La localisation est assurée par le triptyque : centrale inertielle *Landins*, GPS différentiel RTK et odomètre. La fusion des trois sources d'informations est assurée par la centrale inertielle qui s'appuie en interne sur un modèle de déplacement de véhicule.

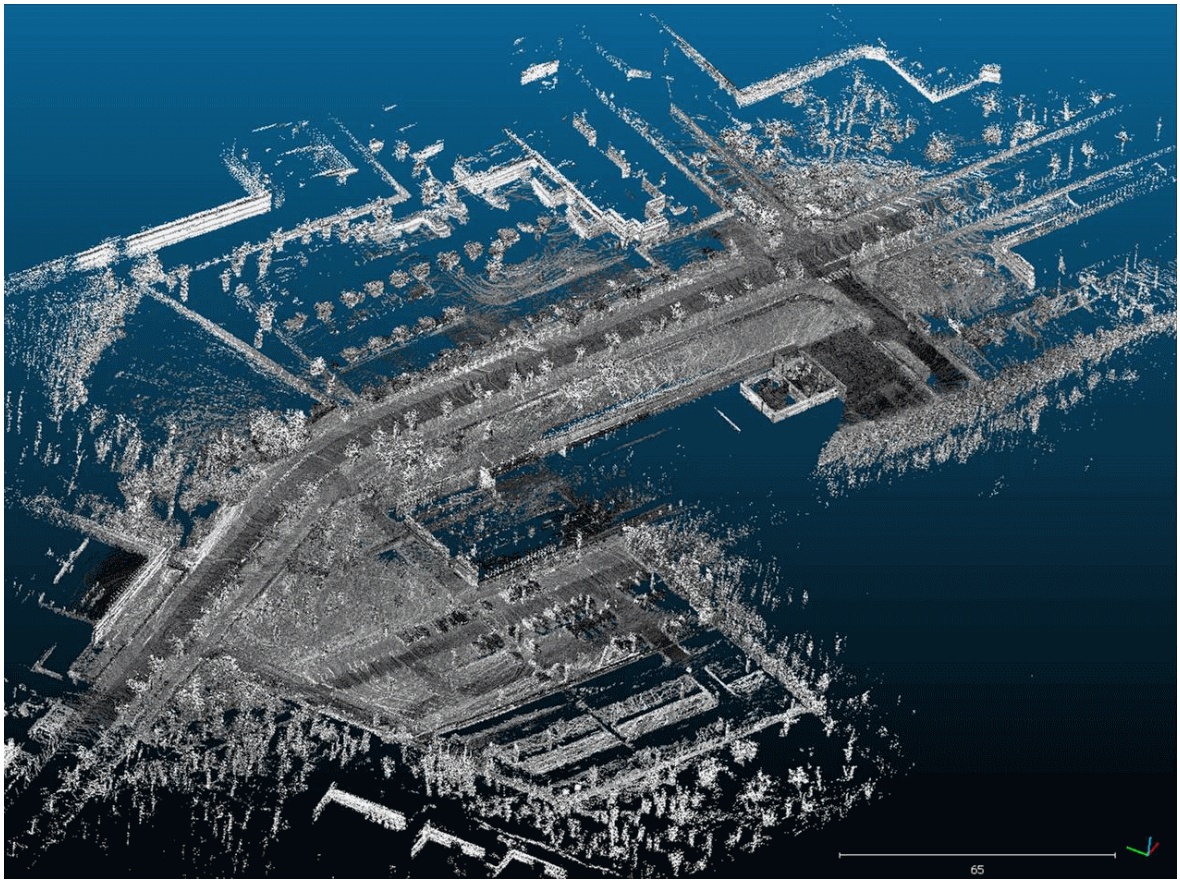


FIGURE 2.15 – Numérisation du Campus CISE de l'Esigelec obtenue avec le coffre de toit et son LiDAR *Velodyne HDL64*.

Pour le GPS RTK, nous avons installé notre propre station de référence sur le toit du laboratoire. Elle envoie les corrections toutes les secondes à l'équipement mobile, soit par une liaison radio UHF, si nous sommes dans un rayon de quelques kilomètres du laboratoire, soit par le réseau GSM avec un lien sur réseau IP quand nous sommes hors de portée. La position « exacte » de la station de référence a été déterminée en associant un enregistrement sur plusieurs jours ainsi que des données des autres stations de références environnantes [18]. Le logiciel *GNSS Solution* [19] permet une optimisation de l'ensemble afin d'estimer la meilleure position.

Il est toujours difficile d'annoncer une précision pour un tel équipement, sachant que nous ne disposons pas de moyens d'essais permettant de la mesurer. Le fabricant *Ixsea* de la centrale garantit les performances de la table 2.4. Comme dans le cas du Vicon, pour lequel il est également difficile de réaliser une mesure de performance du moyen d'essai avec objectivité, la performance annoncée est dans la plupart des cas très supérieure aux algorithmes que nous souhaitons évaluer.

Mode	**GPS RTK**	**60s sans GPS**
True Heading (°)	0.01	0.01
Roulis/Tangage (°)	0.005	0.005
Position X,Y (m)	0.02	0.1
Position Z (m)	0.05	0.07

TABLEAU 2.4 – Précision de la localisation du coffre de toit

Il est à noter que la précision absolue d'ordre centimétrique (positionnement sur le globe terrestre) ne tient qu'à la qualité de connaissance de la position de la station de référence. Dans la plupart des cas, ce n'est pas un problème très handicapant dans le sens où les méthodes à évaluer travaillent en relatif par rapport à une carte que nous avons également construite à partir de notre moyen de référence. L'évaluation de l'erreur de l'algorithme se fait dans le repère de la carte.

Nous pouvons avoir bien d'autres sources de biais que la précision de localisation du moyen de référence. La synchronisation des capteurs de l'expérimentation avec la source de localisation peut apporter beaucoup d'imprécisions. Par exemple, avec une caméra *Basler* à 20Hz, si le véhicule roule à 50km/h, il aura parcouru 69cm entre 2 images. Il est donc très important de bien soigner la synchronisation et la datation des données issues des différentes sources de données. Une synchronisation « hardware » s'impose. Pour remplir cette fonction, nous avons développé une carte électronique qui gère les signaux issus des différents capteurs (en vert sur la figure 2.14). La table 2.5 résume le moyen de synchronisation utilisé pour chaque capteur.

Le coffre de toit comporte un système de rails d'extension sur l'ensemble de son pourtour, permettant d'embarquer des capteurs spécifiques à l'expérimentation voulue. Il a été conçu pour pouvoir connecter huit caméras réseau dans des boîtiers étanches, et cinq LiDARs de type *Sick*.

Comme le précise la table 2.6, les quantités de données à enregistrer doivent être prises en compte, car elles sont rapidement très importantes.

D'un point de vue énergie, le coffre embarque une batterie lui fournissant un autonomie de 3h ; ce qui permet de l'installer par exemple sur le robot *Segway* sans se préoccuper de son alimentation (figure 2.16b). Un chargeur interne permet de maintenir l'alimenta-

18. http://rgp.ign.fr/
19. http://www.spectraprecision.com/eng/gnss-solutions.html

Capteurs	Description
Caméra Basler	exposure out
LiDAR Sick et Hokuyo	sync out (top au passage des 180˚du miroir tournant)
Velodyne	Capteur à effet Hall + aimant (dispositif similaire à Geiger et collab. [2013])
IMU Landins	Acquisition interne du trigger de la caméra comme *event externe*

TABLEAU 2.5 – Synchronisation des principaux capteurs utilisés dans le coffre de toit de référence

Capteurs	Description	Sample rate	Log format	HDD speed rate
Landins	position WGS84 and attitude	200Hz	plain text	1MBps
Basler camera	24 bits color picture	20Hz	binary raw	80MBps
Velodyne Lildar	64 lasers 360˚	15Hz	binary raw	10MBps

TABLEAU 2.6 – Débit d'enregistrement numérique des principaux capteurs utilisés dans le coffre de toit de référence

(a) Véhicule routier de type *Renault Espace IV*

(b) Véhicule électrique léger *Segway RMP440*

FIGURE 2.16 – Coffre de toit de référence installé sur deux types de véhicules compatibles du laboratoire.

tion en continu à partir d'une prise allume-cigare sur un véhicule (figure 2.16a). Par retour d'expérience, même sur un véhicule routier la batterie présente deux gros avantages. Premièrement, il est possible de préparer l'expérimentation au garage sans alimentation externe, ou sans allumer le moteur du véhicule. Et deuxièmement, si le véhicule cale, nous ne perdons pas l'énergie dans le coffre.

Le PC embarqué exécute RTMaps associé à un ensemble de diagrammes de base que nous adaptons à chaque nouvelle expérimentation.

2.2.3.2 Véhicule

En plus du robot *Segway*, nous disposons d'un véhicule *Renault Espace* pour toutes nos expérimentations sur route ouverte. Nous pouvons bien sûr y installer le coffre de toit comme le montre la figure 2.16a.

Nous avons fait le tour de la présentation des moyens d'essais de la plateforme de navigation autonome dont je suis responsable. Nous allons maintenant aborder deux exemples de projets utilisant la méthodologie proposée en début de ce chapitre.

2.3 Deux exemples utilisant la méthodologie et les outils présentés

Les deux chapitres suivants constituent la contribution principale de ma thèse et sont basés sur l'utilisation de la méthodologie et des moyens d'essais présentés dans ce chapitre 2. Afin de mettre en évidence l'intérêt de la méthodologie, nous allons aborder deux exemples à travers le projet *Quasper* et *Navalis* sur lesquels j'ai travaillé respectivement deux ans avant et au début de ma thèse.

2.3.1 Le projet Quasper

Le projet *Quasper* consistait à développer des solutions de qualification et de certification de systèmes de perception (figure 2.17). Dans ce cadre, le laboratoire *Lemco* de L'*Ifsttar* était chargé de constituer un banc de test dynamique d'*ADAS* (Advanced driver assistance systems) autour de l'automatisation d'un véhicule. Ils ont fait appel à l'IRSEEM à travers son CRT (Centre de Ressources Techniques). J'ai donc développé et mis au point ce banc d'essai en binôme avec Jean-Christophe SMAL du *Lemco*. C'est par ces travaux que nous avons commencé à formaliser les idées exposées dans ce chapitre.

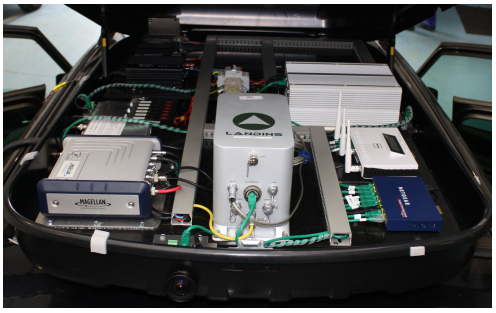
Notre développement consistait à automatiser un véhicule de série en y apportant le minimum de modifications, l'objectif étant de lui faire suivre des trajectoires plus précisément que ce que peut faire un pilote afin d'évaluer les performances de systèmes d'assistance à la conduite.

L'idée consiste à installer un robot de conduite *Stahle SAP2000*[20], destiné à une utilisation sur banc à rouleaux, au volant du véhicule pour contrôler les pédales, volant et boîte de vitesses (figure 2.17d). Les capteurs sont quant à eux installés dans un coffre de toit et un rack de contrôle trouvera place dans l'habitacle. De ce fait, l'automatisation d'un véhicule consiste en l'installation de ces trois équipements et quelques réglages de paramètres concernant sa géométrie et sa dynamique. Le véhicule utilisé pour le développement est un *Renault Espace* (figure 2.17a) mais la possibilité de transférer le dispositif sur un autre véhicule a été démontrée en louant deux semaines un *Volvo XC60* (figure 2.17d).

20. http://www.stahle.com/en/produkte.php?gr=1&prod=1

(a) Renault Espace utilisée comme base pour la validation des algorithmes échelle 1

(b) Sous-partie de carte OpenDRIVE des pistes de Satory superposée à une ortho-photo



(c) Détails des capteurs contenus dans le coffre de toit

(d) *Volvo XC60* ayant servi à démontrer la transposition sur un autre véhicule : coffre de toit, robot de conduite et rack de contrôle.

FIGURE 2.17 – Banc dynamique du projet *Quasper* développé en binôme avec Jean-Christophe SMAL du laboratoire *Lemco* de L'*Ifsttar* : automatisation d'un véhicule de série pour le test et la qualification de systèmes d'aide à la conduite par l'installation au volant d'un robot de conduite *Stahle SAP2000*, d'un coffre de toit contenant les capteurs et d'un rack de contrôle

La localisation du véhicule est fournie par la fusion d'un *GPS RTK*, d'une centrale inertielle et d'un odomètre (figure 2.17c). Le véhicule utilise le format normalisé de carte *OpenDRIVE*[21] qui permet de stocker toutes sortes d'informations routières : voies, intersections, connections, signalisations, marquages, sens de circulation, limitations de vitesse... Les géométries sont stockées sous forme d'équations de ligne, arcs de cercle et clothoïdes ((figure 2.17b)).

Dans leurs versions finales, les algorithmes permettent de sélectionner une destination en cliquant sur la carte. Le chemin le plus court est alors déterminé puis le véhicule est contrôlé pour suivre ce chemin en respectant les « cédez le passage », stops, limitations de vitesses, une accélération latérale maximale dans les courbes...

Pour mener à bien le développement et la mise au point de l'ensemble des algorithmes embarqués, j'ai appliqué la méthode explicitée dans ce chapitre 2. Nous ne disposions que d'un créneau de trois semaines sur les pistes de Satory pour finaliser la mise au point du véhicule. Il était donc inenvisageable de directement développer sur le véhicule. En parallèle de l'intégration des capteurs, de l'électronique de contrôle et du développement des drivers logiciel, le développement a suivi les étapes suivantes :

— Tout d'abord, en simulation sous deux phases :

1. Avec des outils propriétaires développés sous *RTMaps*. Ce sera d'ailleurs notre premier simulateur cinématique de véhicule (figure 2.5).
2. Des simulations nettement plus réalistes avec l'outil *Pro-Sivic*, les pistes de Satory ayant été numérisées par l'Ifsttar. L'ensemble des algorithmes développés sous *RTMaps* ont été validés conjointement avec *Pro-Sivic* (figure 2.7).

— Des essais à échelle réduite évaluant l'ensemble des algorithmes sur nos robots *Wifibots* (figure 2.13a) dans notre laboratoire. Une abstraction du modèle de véhicule est effectuée pour passer d'un *modèle Ackerman* (roues avant directrices) à un *modèle différentiel*. Un réseau de routes, intersections, signalisations... est représenté sous forme de réalité augmentée dans le laboratoire. Un aperçu de ce réseau peut être visualisé sur la figure 2.5, les voies sont en jaune et leur axe en rouge.

— Des essais à échelle réelle sur le véhicule Espace (figure 2.17a) sur la dizaine de kilomètres que comptent les pistes de Satory à des vitesses dépassant les 90km/h.

Même si « à l'époque » la méthodologie proposée n'en était qu'à ses débuts dans notre laboratoire, ce projet n'aurait pu se dérouler dans un délai aussi court sans les phases d'abstraction que représentent la simulation et les essais à petites échelles.

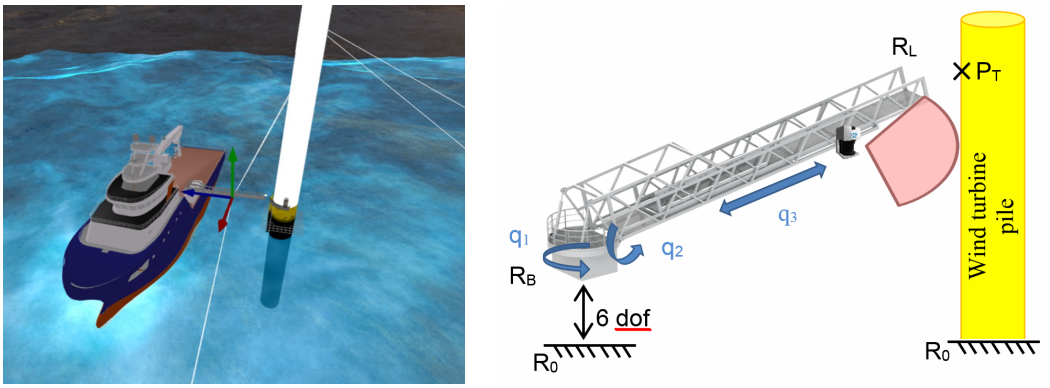
Ces travaux ont abouti à de nombreuses démonstrations. Par exemple, pendant la conférence *Vision SIA* (Société des Ingénieurs de l'Automobile) se déroulant à Versailles en 2012, nous avons roulé de nuit avec des passagers dans le véhicule sur l'ensemble de la piste.

Ces travaux ont également mené à la publication de trois papiers Belbachir et collab. [2012a], Belbachir et collab. [2012b] et Belbachir et collab. [2013] dont deux où je suis coauteur ainsi qu'un article sur IEEE Spectrum [22].

21. http://www.opendrive.org/
22. http://spectrum.ieee.org/automaton/robotics/industrial-robots/french-self-driving-car-takes-to-the-road

2.3.2 Le projet Navalis

Dans un tout autre domaine, le projet *Navalis* consiste à développer une passerelle compensée pour débarquer du personnel de maintenance sur des éoliennes offshore. La passerelle est installée sur un navire, elle doit compenser les mouvements de ce dernier, de manière à maintenir son extrémité immobile vis-à-vis du point de débarquement (figure 2.18a).



(a) Le bateau du projet *Navalis*, équipé de sa passerelle et une éolienne simulés dans *Pro-Sivic*.

(b) Les 3 actionneurs de la passerelle, le LiDAR à son extrémité et l'IMU sur le navire.

FIGURE 2.18 – Projet *Navalis*, rendu de simulation et synoptique de fonctionnement.

Vu les coûts de développement de cette passerelle d'une envergure moyenne de 10m ainsi que les coûts d'affrètement d'un navire de la classe des 70m, il est inenvisageable de travailler directement à échelle 1.

Pour ma part, je me suis attelé à la localisation de l'extrémité de la passerelle par rapport à l'éolienne, pendant que des collègues ont développé la partie modélisation et contrôle. Cette localisation est basée sur un LiDAR mono-nappe et une centrale d'attitude (figure 2.18b).

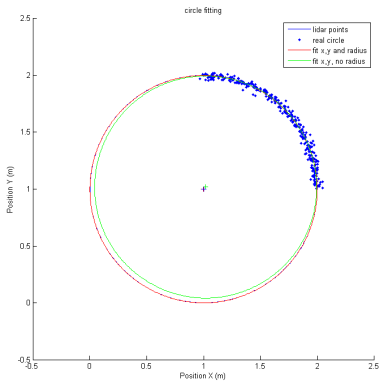
Nous avons exploité l'idée consistant à extraire le cercle formé par le tube de la pile de l'éolienne, afin d'obtenir les deux translations dans le plan. Nous obtiendrons les autres degrés de liberté avec l'IMU. Nous avons commencé par développer des premiers algorithmes sous *Matlab*, pour extraire ce cercle de la nappe LiDAR (figure 2.19a).

En parallèle, nous avons effectué une campagne d'évaluation des performances de différentes IMUs qualifiées marine sur un hexapode au bassin d'Ifremer Brest (Institut Français de Recherche pour l'Exploitation de la Mer). Par la même occasion, en partenariat avec eux, ils nous ont généré des états de mers correspondant aux caractéristiques de la coque du navire envisagé. Ce sont des trajectoires 6 degrés de liberté, représentatives du comportement du navire pour une houle donnée.

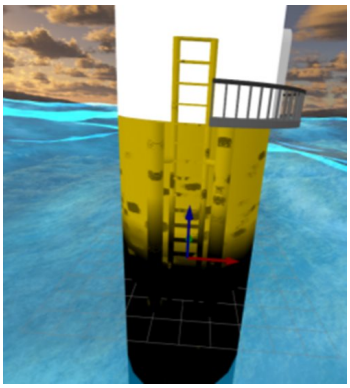
Ces deux premières étapes nous ont permis de mettre au point une chaîne de simulation complète sous Pro-Sivic (figure 2.18a et 2.19b). En animant le navire avec des mouvements issus des états de mer d'Ifremer, nous avons pu porter nos algorithmes sur *RTMaps*. Ces simulations nous ont permis de mettre en évidence des problèmes, que nous n'avions pas pu soulever dans nos représentations simplistes du monde Matlab.

Dans les phases de simulation, nous avons pu nous affranchir de la faible représentativité des modèles d'IMU simulés en utilisant directement les mesures de caractérisation acquises sur l'hexapod d'Ifremer sous forme d'une base de données.

Suite à ces simulations, différents essais à petite échelle ont été mis en œuvre pour qualifier plusieurs aspects de la simulation. Du côté de la perception, nous avons utilisé le



(a) Sous *Matlab*, simulation par un simple cercle.



(b) Sous *Pro-Sivic*, modèle 3D comprenant l'échelle, les 2 bordées de débarquement,...

(c) La maquette utilisée pour les tests à échelle réduite dans le laboratoire avec le *Kuka*

(d) Photo d'une pile d'éolienne réelle en mer du Nord

FIGURE 2.19 – Différents niveaux de représentation de l'éolienne du projet *Navalis* en fonction de l'étape de développement.

bras manipulateur *Kuka* (figure 2.12) pour appliquer les mouvements de mer à un LiDAR le tout positionné en face d'une maquette d'éolienne (figure 2.19c) dans le laboratoire.

Des essais sur une maquette complète de la passerelle à échelle réduite installée sur un hexapode devraient suivre en fin d'année. En parallèle, une expérimentation réelle sur un navire de la partie localisation sera menée en octobre 2016, avant de basculer sur les premières expérimentations de la passerelle échelle 1 sur navire (figure 2.19d).

La figure 2.19 montre l'évolution de la complexité de représentation de l'environnement en fonction des trois étapes de développements proposées : simulation (*Matlab* puis *Pro-Sivic*), maquette échelle réduite dans le laboratoire et enfin système réel. Nous pourrions faire la même analyse pour les modèles de capteurs qui prennent de mieux en mieux en compte les phénomènes physiques : du modèle simpliste sous *Matlab* dans un environnement parfait, le rejeu de bases de données de capteurs réels acquis, jusqu'à finir sur le navire avec des conditions environnementales maritimes.

Ces premiers travaux ont donné lieu à deux publications au début de ma thèse, Merriaux et collab. [2014a] et Merriaux et collab. [2014b].

Conclusion

Nous avons vu une méthode développement permettant de mettre au point des algorithmes sur des systèmes complexes en procédant par niveaux d'abstraction. D'autres laboratoires ont des équipements en partie similaires. Par exemple, avec le développement des drones ces dernières années, il existe de plus en plus de salles de vol comme à l'université d'Aix Marseille [23].

Certains chercheurs ont poussé assez loin le développement d'une partie de la chaîne. Par exemple pour améliorer les possibilités de réalité augmentée de leur laboratoire [24], Omidshafiei et collab. utilisent des vidéo-projecteurs pour afficher au sol des informations dynamiques comme la carte, l'intention de leurs robots, leurs trajectoires, leurs incertitudes de localisation, ...

Comme nous l'avons aperçu tout au long de ce chapitre, tirer parti de moyens de référence nécessite une certaine expertise technique. Malheureusement, la vérité terrain de nombreuses communications scientifiques rend leur contenu discutable.

Un des avantages majeurs de notre laboratoire consiste dans le fait que nous y avons regroupé au même endroit de nombreux moyens d'essais et les compétences techniques nécessaires aux expérimentations des domaines de la robotique ; en intérieur et en extérieur, et aussi bien mesurer des trajectoires, que les générer à travers des mobiles ou bras manipulateurs.

Ces dernières années, de nombreuses démonstrations de véhicules autonomes ont été présentées. Évidemment, il reste encore beaucoup de travail sur les algorithmes, sur l'optimisation des capteurs, des coûts... Mais peut-être qu'un des enjeux les plus importants aussi bien pour la communauté scientifique que pour les industriels sera de démontrer et prouver la fiabilité de ces systèmes. Faire des centaines de milliers de kilomètres comme la *Google car* ne démontre au final pas grand chose, tout dépend des conditions.

Comme chacun le sait, le problème de la couverture des tests est loin d'être évident. L'apport de la simulation, particulièrement en incluant une partie du matériel dans la boucle comme nos tests à échelle réduite paraissent apporter un début de solution à ce

23. http://www.ism.univ-amu.fr/spip.php?rubrique141&lang=fr
24. http://news.mit.edu/2014/system-shows-robot-intentions-1029

problème de taille en permettant de mettre le système dans des conditions limites, et de multiplier les essais. De nombreuses sociétés se lancent dans le créneau de l'automatisation des tests pour la validation d'algorithmes comme Intempora avec leur outil *IDeep*[25], dSPSACE avec *dSPACE Synect*[26] ou encore le standard *OSLC*[27] (Open Services for Lifecycle Management).

Comme le montrent Yahiaoui et collab. [2015] dans un *white paper* du groupement *ADAS* du pôle de compétitivité *Mov'eo*, au-delà de l'automatisation des tests se pose le problème de la gestion de données : un serveur où seraient stockés tous les scénarios (simulés ou réels) et proposant un moteur de recherche sur des données annotées permettant des recherches sémantiques (« trouve moi toutes les séquences en approche de rond-point par temps de pluie et avec une caméra stéréo »). La société *X-Cube* fournit un data manager[28] dans cet esprit, mais pour l'instant la plupart des industriels ont développé leurs propres solutions propriétaires. L'association *ASAM* (Association for Standardization of Automation and Measuring Systems) propose un standard *ODS*[29] (Open Data Services) qui n'est pas encore très développé.

Maintenant que nous avons posé les bases de la localisation, des méthodes et des moyens utilisés, nous allons aborder dans les deux prochains chapitres les deux contributions de ma thèse.

— Une localisation couche 2, 6 degrés de liberté temps réel basée LiDAR, utilisée dans le Challenge Argos.

— Une localisation couche 1, basée odométrie et graphe OpenStreetMap.

Références

Barrows, D. A. 2007, «Videogrammetric model deformation measurement technique for wind tunnel applications», *AIAA Paper*, vol. 1163, p. 2007.

Belbachir, A., R. Boutteau, P. Merriaux, J. Blosseville et X. Savatier. 2013, «From autonomous robotics toward autonomous car», dans *IEEE Intelligent Vehicle Symposium (IV 2013)*, Gold Coast, Australia. 74

Belbachir, A., J.-C. Smal et J.-M. Blosseville. 2012a, «A robotic platform to evalute autonomous driving systems», dans *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*, IEEE, p. 1874–1879. 74

Belbachir, A., J.-C. Smal, J.-M. Blosseville et P. Merriaux. 2012b, «Robert : a robotic platform for testing adas and autonomous car driving capabilities», dans *SIA Vision*, édité par société des ingénieurs de l'automobile, Versailles, France. 74

Diaz Novo, C., S. Alharbi, M. Fox, E. Ouellette, E. Biden, M. Tingley et V. Chester. 2014, «The impact of technical parameters such as video sensor technology, system configuration, marker size and speed on the accuracy of motion analysis systems», *Ingeniería mecánica, tecnología y desarrollo*, vol. 5, nº 1, p. 265–271.

25. https://intempora.com/products/ideep.html
26. https://www.dspace.com/fr/fra/home/products/sw/datenmanagement/synect.cfm
27. http://open-services.net/
28. www.x3-c.com/streaming-solutions/cloud-based-solutions/
29. https://wiki.asam.net/display/STANDARDS/ASAM+ODS

Geiger, A., P. Lenz, C. Stiller et R. Urtasun. 2013, «Vision meets robotics : The kitti dataset», *International Journal of Robotics Research (IJRR)*. 56, 71

Geiger, A., P. Lenz et R. Urtasun. 2012, «Are we ready for autonomous driving? the kitti vision benchmark suite», dans *Conference on Computer Vision and Pattern Recognition (CVPR)*. 56, 57

Glover, A. J., W. P. Maddern, M. J. Milford et G. F. Wyeth. 2010, «Fab-map+ ratslam : appearance-based slam for multiple times of day», dans *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, IEEE, p. 3507–3512. 57

Gonzalez, J.-L. B. . F.-A. M. . J. Novembre 2009, *A collection of outdoor robotic datasets with centimeter-accuracy ground truth : Autonomous Robots*, chap. Volume 27, Issue 4, Springer US, p. pp 327–351. 57

Korrapati, H., J. Courbon, S. Alizon et F. Marmoiton. 2013, «"the institut pascal data sets" : un jeu de données en extérieur, multicapteurs et datées avec réalité terrain, données d'étalonnage et outils logiciels», dans *Orasis, Congrés des jeunes chercheurs en vision par ordinateur*, Cluny, France. 57

M. Smith, W. C. R. P., I. Baldwin et P. Newman. 2009, «The new college vision and laser data set», *International Journal of Robotics Research : IJRR*, vol. vol 28(5), p. 595–599. 57

Manecy, A., N. Marchand, F. Ruffier et S. Viollet. 2015, «X4-mag : A low-cost open-source micro-quadrotor and its linux-based controller», *International Journal of Micro Air Vehicles*, vol. 7, n° 2, p. 89–110.

Merriaux, P., R. Boutteau, P. Vasseur et X. Savatier. 2014a, «Algorithme de positionnement d'une passerelle à mouvements compensés à partir de mesures inertielles et lidar pour les opérations de maintenance des parcs éoliens offshore», . 77

Merriaux, P., R. Boutteau, P. Vasseur et X. Savatier. 2014b, «Imu/lidar based positioning of a gangway for maintenance operations on wind farms», IEEE/RSJ International Conference on Intelligent Robots (IROS), Chicago, USA. 77

Mueggler, E., B. Huber et D. Scaramuzza. 2014, «Event-based, 6-dof pose tracking for high-speed maneuvers», dans *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, IEEE, p. 2761–2768.

Omidshafiei, S., A.-A. Agha-Mohammadi, Y. F. Chen, N. K. Ure, J. P. How, J. Vian et R. Surati. «Mar-cps : Measurable augmented reality for prototyping cyber-physical systems», . 77

R. Simpson, J. C. et J. Revell. 2011, «The cheddar gorge data set», cahier de recherche, BAE Systems (Operations) Limited, Advanced Technology Centre. 57

Warren, M., D. McKinnon, H. He et B. Upcroft. 2010, «Unaided stereo vision based pose estimation», . 57

Yahiaoui, G., N. Du Lac, L. Rafaelli, R. Katz et P. De Souza. 2015, «Validation of advanced driving assistance systems», cahier de recherche, pole-moveo. 78

Yang, P.-F., M. Sanno, G.-P. Brüggemann et J. Rittweger. 2012, «Evaluation of the performance of a motion capture system for small displacement recording and a discussion for its application potential in bone deformation in vivo measurements», *Proceedings*

of the Institution of Mechanical Engineers, Part H : Journal of Engineering in Medicine, vol. 226, n° 11, p. 838–847.

Chapitre 3

Localisation 6 DDL en milieu industriel basée LiDAR

Sommaire

3.1 Introduction et contexte

Ce chapitre présente une méthode de localisation 6 ddl basée LiDAR en environnement complexe. Cette méthode a été développée dans le cadre de la participation de notre équipe Vikings au challenge *Argos*[1] (Autonomous Robot for Gas and Oil Sites) aux côtés de quatre autres consortiums internationaux. Après une rapide présentation des enjeux du challenge Argos, nous exposerons la solution mise en oeuvre à partir de la mesure d'un LiDAR multinappes. La méthodologie et les moyens d'essais présentés dans le chapitre précédent ont été utilisés afin d'objectiver la performance et la robustesse de notre approche. Enfin nous montrons comment ces travaux peuvent être transposés au cas du véhicule autonome.

3.1.1 Projet et missions du robot

Le challenge de robotique *Argos* est organisé par la compagnie pétrolière et gazière *Total*[2] avec le soutien de l'*ANR*[3]. Démarré depuis septembre 2014, le projet est articulé autour de trois compétitions : juin 2015, avril 2016 et mars 2017.

Il a pour but de mener au développement d'un robot autonome d'inspection d'installations pétrolières et gazières, étant capable d'effectuer différentes mesures :

— Mesure de pressions sur des manomètres

— Lecture de position de vannes

— Détection et localisation de sources anormalement chaudes

— Mesure de températures

— Analyses sonores de pompe

— Inconsistance de la carte 3D

— Fuite de gaz...

Les différentes missions se déroulent à travers des modes télé-opérés ou autonomes. En terme de sécurité, le robot doit pouvoir éviter toute collision avec la structure ainsi qu'avec les obstacles dynamiques, et réagir à l'alarme sonore d'évacuation ou encore aux coupures de réseau. D'un point de vue mobilité, il doit également être capable de contourner ou de négocier les obstacles rencontrés et de franchir les escaliers. Tous ces points constituent un aperçu des 87 pages du règlement émis par Total et l'ANR.

Comme nous le verrons tout au long de ce chapitre, la méthode de développement employée est une application de celle qui a été présentée en section 2.1. Nous avons déroulé les étapes suivantes :

— Simulations Matlab ;

— Simulations sur des outils propriétaires, afin d'être compatible avec l'implémentation matérielle des capteurs et actionneurs du robot ;

— Essai à petite échelle sur prototype du robot que nous avons équipé des capteurs de localisation et d'une structure miniature représentant l'environnement. Caractérisation des performances en confrontant l'algorithme aux moyens de vérité terrain de notre laboratoire ;

1. http://argos-challenge.com/
2. http://www.total.com/
3. http://www.agence-nationale-recherche.fr//

— Constitution de bases de données référence, aussi bien du côté environnement avec les cartes, que sur des données capteurs sur les différents robots ;
— Tests pleine échelle sur le site de la compétition.

3.1.2 Navigation autonome en environnement complexe

Les installations pétrochimiques sont assez différentes de ce que nous trouvons couramment sous le terme d'environnement industriel dans la littérature. En effet, nous sommes loin d'un entrepôt ou d'une usine dans un « bâtiment carré » équipés de machines outils. Ici le milieu est très dense, les installations sont omniprésentes dans toutes les directions. Les passerelles sur lesquelles se déplacent les opérateurs et donc notre robot sont composées de plusieurs niveaux séparés par quelques marches ou des escaliers complets. Le sol n'est pas régulier, mais généralement composé par des plaques de caillebotis. La figure 3.1a présente un sous-ensemble d'un bateau de production qui illustre bien le type d'environnement de mission d'*Argos*. La figure attenante 3.1b montre le site où se déroule le challenge aux abords de l'usine de gaz de Lacq. L'*UMAD* (Unité de Mise A Disposition) est composée d'une ancienne unité de production et d'une structure de type échafaudage sur 5 niveaux. Elle sert habituellement à l'entraînement des pompiers.

(a) Exemple d'environnement visé par le challenge *Argos* : sous-partie d'un navire *FPSO* (Floating production storage and offloading) au large de l'Angola. ©Serge RUPERT Total S.A.

(b) Site d'entraînement *UMAD* (unité de mise à disposition) à Lacq, où se déroulent les épreuves du challenge *Argos*.

FIGURE 3.1 – Environnements complexes visés par le challenge *Argos* : multitudes de niveaux, structures, escaliers, installations, tuyaux...

La localisation est une des briques de base de la robotique autonome ; si nous ne savons pas où nous sommes, il est compliqué de savoir dans quelle direction nous diriger pour atteindre notre objectif. En langage plus robotique, sans localisation, pas de possibilité de *pathplanning* ni de *pathcontrol*, et par extension pas de « fonction métier » de mesure, ni de sécurité active quant aux déplacements du robot...

Afin d'effectuer les missions exigées dans le cadre du challenge, notre robot a donc besoin d'une localisation précise bien que l'environnement soit complexe.

La structure irrégulière des allées ainsi que le franchissement d'obstacles et de différents niveaux par des escaliers nécessiteront une localisation selon six degrés de liberté. La figure 3.19 donne une idée des franchissements effectués par notre robot.

Les contraintes opérationnelles, comme une utilisation de jour comme de nuit par toutes les conditions météorologiques, ainsi que le haut de niveau de sécurité requis nous ont amenés à écarter rapidement les méthodes de localisation basées vision. Nous nous concentrerons donc sur les approches basées LiDAR.

Les algorithmes devant fonctionner en temps réel et être intégralement embarqués sur le robot ; une attention particulière sera portée sur leur implémentation et leur consommation de ressources.

C'est dans ce contexte que j'ai effectué les travaux présentés dans ce chapitre. Ils ont donné lieu à la publication d'un article Merriaux et collab. [2015]. Une soumission est en cours dans *Journal of Field Robotics*.

3.2 État de l'art orienté localisation en milieu industriel

Comme entraperçu au chapitre 1, de nombreuses méthodes traitent le sujet de la localisation en milieu industriel.

3.2.1 *SLAM* versus localisation

Nous pouvons commencer par séparer deux cas : celui où les environnements sont inconnus, et celui pour lequel nous disposons d'une carte *a priori*.

— Ce premier cas est connu sous le nom de *SLAM* (Simultaneous Localization And Mapping).

  Dans sa phase de localisation, il utilise des concepts d'appariement de données communes avec la localisation que nous avons abordés dans la section 1.2.5.2. Les premiers travaux autour du *SLAM* proposent des solutions pour des déplacements plans dans des environnements simples comme des immeubles de bureaux (Thrun [1993]). Mais comme nous l'avons vu, certaines applications robotiques, plus tout terrain, nécessitent plus de degrés de liberté. Le *SLAM* 6D a été investigué entre autres par Nuchter et collab. [2004] pour l'exploration de mines abandonnées.

— Le second cas est connu sous le nom de localisation : une connaissance *a priori* de l'environnement est disponible. Sans méthode de fermeture de boucle, le *SLAM* fournit des cartes déformées par l'accumulation des erreurs de prédiction et de correction. La sécurité et la précision requises pour le pilotage du robot dans le domaine visé ne peuvent se satisfaire d'une représentation contenant une dérive. De plus, l'ensemble de ces sites industriels sont numérisés avec précision pour des raisons opérationnelles de maintenance, formation, évolutions...

Pour ces différentes raisons nous n'abordons pas dans cette étude le domaine du *SLAM*, mais celui de la localisation basée sur une connaissance *a priori* de l'environnement 3D. Cette localisation devra déterminer les 6 degrés de liberté et être adaptée aux milieux complexes avec de fortes contraintes de robustesse et d'embarquabilité.

3.2.2 Technologies en milieu industriel

Dans le cas d'environnements « simples » caractérisés par un sol plan et un milieu intérieur, il existe de nombreuses solutions pour localiser un mobile en milieu industriel. Une des premières et des plus simples consiste à suivre une ligne optique (Olivares-Mendez et collab. [2011]) ou magnétique (Taghaboni et Tanchoco [1988]). En plus de nécessiter

l'installation d'une infrastructure, ces méthodes restreignent les déplacements des mobiles à des tracés spécifiques, ce qui ne permet pas de tirer complètement parti des apports de la robotique mobile, comme par exemple le contournement d'obstacle.

D'autres solutions ont alors émergé pour se « libérer » de cette trajectoire qu'il n'était pas possible de quitter. Loevsky et Shimshoni [2010] puis Ronzoni et collab. [2011] proposent une localisation laser avec des balises connues, qu'il faut répartir dans l'usine. Les avantages et inconvénients de cette solution ont déjà été discutés dans la section 1.2.5.2. Nous pourrions ajouter que comme le LiDAR utilisé est mono-nappe, il ne sera pas possible de suivre des mouvements autres que sur plan afin de garantir une intersection entre le rayon laser et les balises.

Dans leur roadmap pour favoriser le développement des *AGV* dans l'industrie, Sabattini et collab. [2013] mentionnent le fait que l'utilisation de balises artificielles à installer est un frein à l'expansion de ces applications robotiques. Une localisation basée sur des amers naturels permettrait une bien plus grande souplesse.

3.2.3 Travaux analogues basés LiDAR

Pour pallier ces désavantages, Reinke et Beinschob [2013] proposent une localisation basée sur l'extraction d'amers naturellement présents dans les bâtiments (lignes, coins). Mais les hypothèses fortes sur les propriétés des amers rencontrés, associées au champ de vue limité des LiDARs mono-nappes, réduisent très nettement le potentiel de telles méthodes sur des sites pétrochimiques comme le montre la figure 3.1a.

La communauté s'est beaucoup intéressée ces derniers temps aux environnements simples mais dynamiques comme Tipaldi et collab. [2013] qui proposent de répondre à la localisation d'un robot en 2D sur un parking entouré de bâtiments tout en étant robuste à son taux de remplissage.

Les scènes complexes statiques sont également un sujet d'intérêt : Jagbrant et collab. [2013] et Underwood et collab. [2015] localisent des robots à vocation agricole en extérieur dans des vergers. Les scènes sont également complexes d'un point de vue microscopique, mais macroscopiquement, les rangées d'arbres bien taillés pourraient s'apparenter à des « couloirs végétaux ». Les travaux de Stoyanov et collab. [2013] se déroulent en intérieur ; ils proposent de localiser un mobile dans une laiterie avec un LiDAR HDL32 en utilisant une approche de type *NDT* (cf section 1.2.5.2). L'environnement est nettement plus complexe que des bureaux ou un parking, mais il est tout de même composé avec de grands plans formés par les murs du bâtiment et le déplacement se fait en 2D (figure 3.2). Sur ce point, les environnements d'*Argos* nous semblent plus compliqués, avec leurs multiples niveaux et la densité des équipements présents dans la scène.

3.2.4 Contraintes de l'environnement versus solutions existantes

Par rapport à la littérature, nous ne pouvons pas dire que nous sommes en *milieu intérieur*. L'environnement n'est pas constitué majoritairement de grands plans avec les murs. Nous ne nous situons pas non plus dans ce qui est communément appelé *environnement extérieur*, qui représente typiquement soit des scènes routières, soit des robots tout terrain en milieu naturel. Le fait que l'environnement soit constitué majoritairement de tubes, poteaux, et autres surfaces arrondies ne facilite pas la tâche du capteur LiDAR. Sur une surface courbe comme un tube, l'angle d'incidence de la raie à l'impact peut être nettement plus faible. Par conséquence, nous obtenons moins d'écho que sur des plans importants.

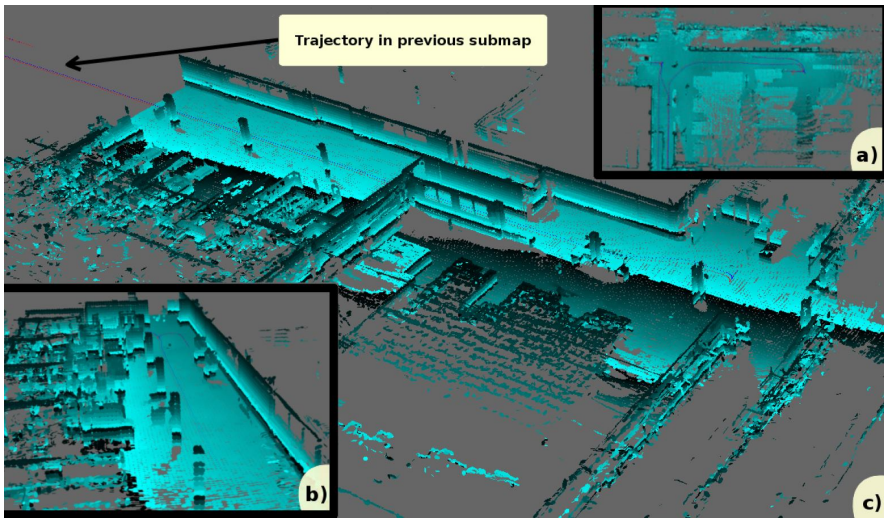


FIGURE 3.2 – Environnement complexe dans une laiterie, HDL-32 sur un AGV. La vérité terrain est obtenue par un système de LiDAR et des balises artificielles pré-existant dans l'usine. Illustration issue de Stoyanov et collab. [2013]

Un autre point que l'on peut noter dans la littérature : de nombreux travaux sur l'acquisition de données 3D se font en utilisant un LiDAR motorisé sur un axe (section 1.2.5) comme Nuchter et collab. [2004] et Nüchter et collab. [2006] ou encore Zhuang et collab. [2013] qui proposent de gérer les scènes dynamiques intérieures. L'utilisation de ce type de capteur - sauf si une correction des scans est apportée comme le font Zhang et Singh [2014] ou ce que nous proposons en annexe B - impose d'arrêter le robot le temps de l'acquisition ; ce qui est incompatible avec les notions de temps réel et opérabilité visées par les applications d'*Argos*. De plus, la plupart des méthodes d'appariement de nuage de points que nous trouvons sont basées sur l'ICP (Iterative Closest Point). La version de base de cet algorithme est lente et sensible aux variations de scène. Malgré les améliorations apportées, il est peu adapté à une utilisation temps réel dans un robot aux ressources calculatoires limitées.

Notre principale contribution consiste en l'extension du concept de champ de vraisemblance à un environnement en 3 dimensions. Nous avons également proposé une méthode originale pour son stockage en mémoire, ainsi que des solutions d'implémentation afin de respecter les contraintes d'embarquabilité. Il est intéressant de noter que notre méthode a démontré sa performance dans le milieu industriel pétrochimique. Des systèmes de localisation, fonctionnant dans des environnements de complexité similaire, sont à ce jour peu développés dans la littérature.

Nous allons maintenant aborder le développement de la méthode proposée en commençant par quelques bases sur la localisation.

3.3 Méthodologie

3.3.1 Notions sur la théorie de la localisation

Le rôle de la localisation est de déterminer la transformation entre un repère fixe global associé à la connaissance de l'environnement, et un repère mobile sur le robot (figure 3.3). Cette transformation est exprimée à travers un vecteur d'état $\mathbf{X_t}$. Nous utilisons une carte $\mathcal{M}$ comme connaissance *a priori* de l'environnement. Le capteur nous retourne des mesures $\mathbf{Z_t}$ de cet environnement. Le vecteur d'état est défini de la manière

suivante :

$$\mathbf{X_t} = \begin{bmatrix} x & y & z & \psi & \theta & \varphi \end{bmatrix}^{\mathrm{T}} \tag{3.1}$$

où :
x, y, z : Position en mètre (m)
ψ, θ, φ : Orientation en radian (rad)

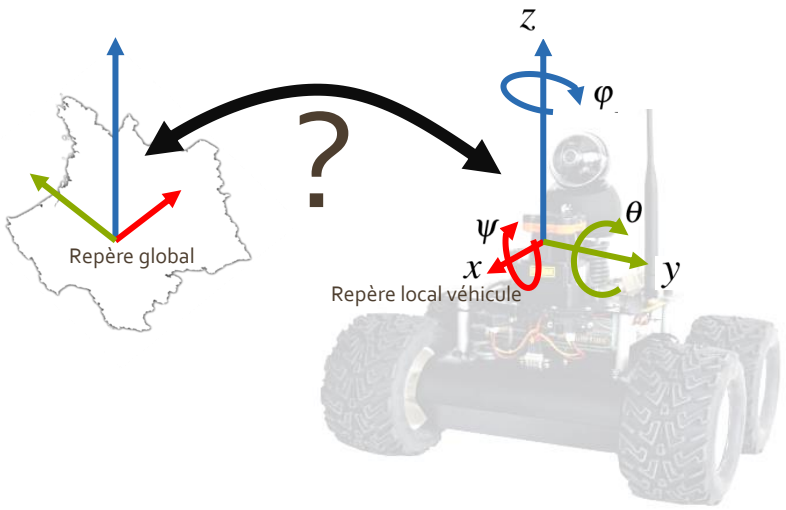


FIGURE 3.3 – Le rôle de la localisation est de déterminer la transformation entre un repère fixe global et un repère mobile sur le robot.

Les méthodes de localisation, quelles que soient leurs modalités, visent à identifier le vecteur d'état le plus probable parmi plusieurs hypothèses. La fonction de vraisemblance exprime la probabilité $\mathbb{P}(\mathbf{Z}|\mathbf{X}, \mathcal{M})$ d'obtenir une mesure $\mathbf{Z}$ pour un état donné $\mathbf{X}$ avec notre connaissance *a priori* de l'environnement $\mathcal{M}$. Elle permet de vérifier que l'hypothèse de position $\mathbf{X}$ est vraisemblable avec la mesure actuelle $\mathbf{Z}$.

Idéalement, la fonction de vraisemblance devrait être :

- Discriminante entre plusieurs hypothèses de vecteur d'état, afin de ne pouvoir conserver que les plus vraisemblables ;
- Monotone et sans maximum locaux, afin de converger vers la bonne solution ;
- Avec un gradient modéré en direction de la solution, afin de faciliter la convergence des hypothèses proches de la solution ;
- Peu coûteuse en temps de calcul, afin de tester un grand nombre d'hypothèses rapidement ;
- Indépendante entre les états, afin de pouvoir faire converger chaque dimension séparément.

Avant d'étudier la fonction de vraisemblance, nous allons aborder le méthode de filtrage retenue.

3.3.2 Méthode de filtrage retenue

Comme abordé à la section 1.3.4, il existe dans la littérature de la localisation plusieurs types de filtres bayésiens (et surtout plusieurs variantes de chacun) afin de sélectionner

les hypothèses émises :

— Les filtres de Kalman et Kalman étendu, mais les modèles d'incertitudes et surtout d'état doivent forcément être gaussiens.

— Le filtre d'histogramme. Sa complexité de calcul évolue de manière quadratique avec le nombre de dimensions. Pour des questions de temps de calcul, il est donc délicat de le porter sur 6 DDL.

— Le filtre particulaire n'est pas limité à une distribution uni-modale de probabilité du vecteur d'état. L'espace du vecteur d'état est continu contrairement au filtre d'histogramme. Le nombre de particules vs dimensions tend à être exponentiel plutôt que quadratique pour le filtre d'histogramme (Thrun et collab. [2005]).

Nous avons donc implémenté un filtre particulaire standard (Thrun et collab. [2005]), pour estimer le vecteur d'état $\hat{\mathbf{X}}_\mathbf{t}$. Son rôle est d'émettre des hypothèses de $\hat{\mathbf{X}}_\mathbf{t}$, appelées particules, et de les sélectionner à l'aide de la fonction de vraisemblance pour converger vers une solution. $\hat{\mathbf{X}}_\mathbf{t}$ est le résultat du barycentre des particules.

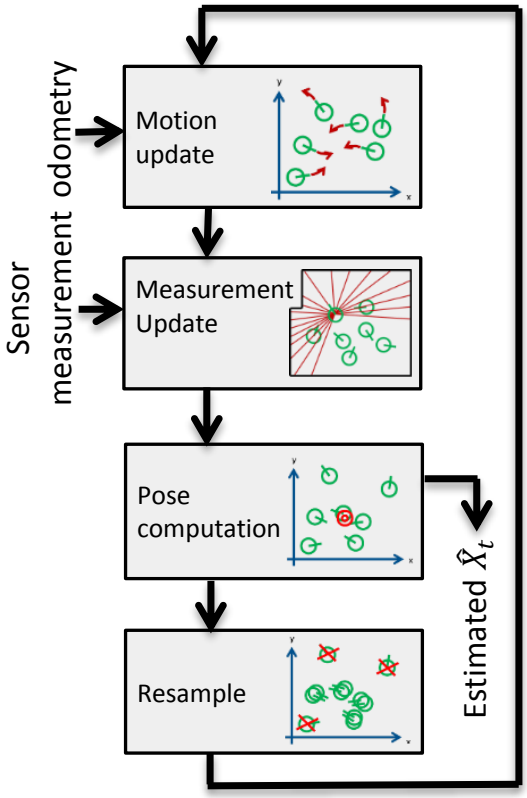


FIGURE 3.4 – Étapes principales du filtre particulaire pour la localisation

Les principales étapes du filtre particulaire illustrées sur la figure 3.4, sont les suivantes :

Prédiction/*Motion update* : nous utilisons un modèle cinématique similaire à celui de la section 1.2.2.1, mais étendu aux déplacements dans les trois dimensions. Il est utilisé pour déplacer les particules (et donc émettre de nouvelles hypothèses de X_t). Des considérations plus approfondies sur l'odométrie et ses multiples limitations sont abordées en annexe C.

Nous considérons que la mesure du mouvement du mobile est contaminée par un bruit gaussien proportionnel au déplacement qui est modélisé par une distribution

normale $\mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_p)$. Les erreurs de modélisation de la cinématique sont quant à elles modélisées par un bruit additif $\mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_a)$.

Nous considérons que ces bruits sont non corrélés entre les différentes variables du vecteur d'état du robot. Par conséquence, Σ_p et Σ_a sont des matrices diagonales définies respectivement par les vecteurs σ_p^2 et σ_a^2.

Correction/*Measurement update* : une mesure du capteur permet à la fonction de vraisemblance d'évaluer la probabilité de chaque hypothèse (le poids de chaque particule). Dans notre cas, la probabilité $p(\mathbf{Z}|\mathbf{X}, \mathcal{M})$ est déterminée en utilisant 3.5 à partir de la mesure LiDAR.

Calcul de la pose : $\hat{\mathbf{X}}_\mathbf{t}$ est le résultat du calcul du barycentre de $\mathbf{X}_\mathbf{t}$ pour chaque particule (équation 1.32).

Ré-échantillonnage/Resample : les meilleures particules sont sélectionnées en fonction de leur poids puis dupliquées en utilisant une méthode de ré-échantillonnage systématique (figure 1.19a).

Toutes les méthodes de filtrages nécessitant des ajustements pour optimiser leur fonctionnement, les différents paramètres permettant de configurer le composant RTMaps de localisation sont explicités dans l'annexe D. L'influence de plusieurs de ces paramètres est étudiée à partir de la section 3.4.3.

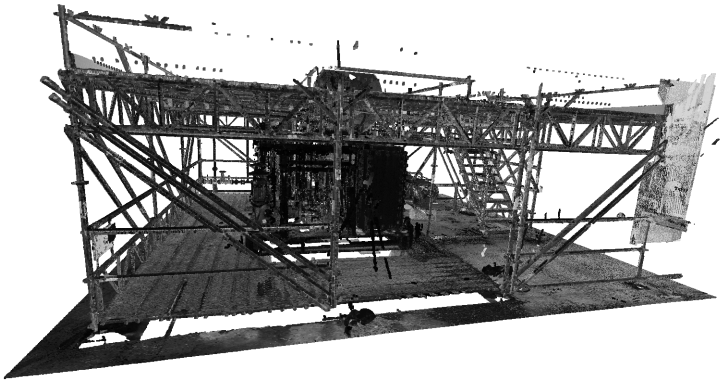
La section suivante 3.3.3 présente la fonction de vraisemblance retenue ainsi que son implémentation en proposant l'extension du concept de champ de vraisemblance (Likelihood field) en 3 dimensions.

3.3.3 Champ de vraisemblance 3D

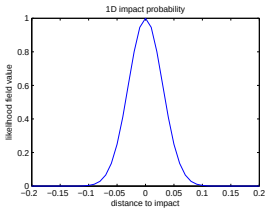
Comme étudié dans la section 1.2.5.2, il existe de nombreuses méthodes pour corréler un scan LiDAR avec une carte. Par principe, ces méthodes sont très dépendantes de la manière dont est représenté le monde dans ladite carte ; la réciproque étant également vraie. Par exemple, Levinson et Thrun [2010] utilisent l'information 3D et la réflectivité infrarouge LiDAR pour localiser un véhicule dans une carte contenant une « image de réflectivité » (figure 1.10a). D'autres méthodes, appelées « Multi-Level Surface Maps » (Pfaff et collab. [2007]), stockent dans leur carte les hauteurs de la surface du terrain qui seront comparées avec les impacts LiDAR.

Nous avons démarré ces travaux avec une approche similaire à ce que proposent Fallon et collab. [2012]. Une simulation en trois dimensions de l'environnement permet de calculer une erreur entre la mesure LiDAR et l'intersection de facettes dans la scène. C'est-à-dire que pour chaque raie LiDAR, une opération de lancer de rayon est effectuée (*ray-tracing*), ce qui permet de déterminer l'erreur de distance entre la mesure du capteur et la représentation dans la carte. L'opération est à effectuer pour chaque hypothèse de pose, la combinaison de ces erreurs sur chaque raie donnant une image de la probabilité recherchée $p(\mathbf{Z}|\mathbf{X}, \mathcal{M})$. Deux problèmes sont récurrents dans ce type d'approche :

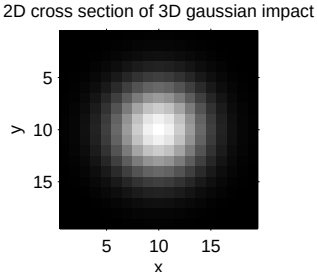
— Comme l'environnement est constitué principalement d'une multitude de tuyaux de faible diamètre, il faut un nombre très conséquent de facettes pour représenter correctement la scène. La simulation de lancer de rayon déjà gourmande par principe en temps de calcul s'en retrouve encore plus ralentie. A titre d'exemple, Fallon et collab. [2012] n'ont conservé que les plans supérieurs à $1m^2$ de leur environnement de bureau. Ils ont ensuite sous-échantillonné leur capteur *RGBD kinect* à 20×15 mesures de distance. Même malgré l'utilisation des techniques de *z-buffer*



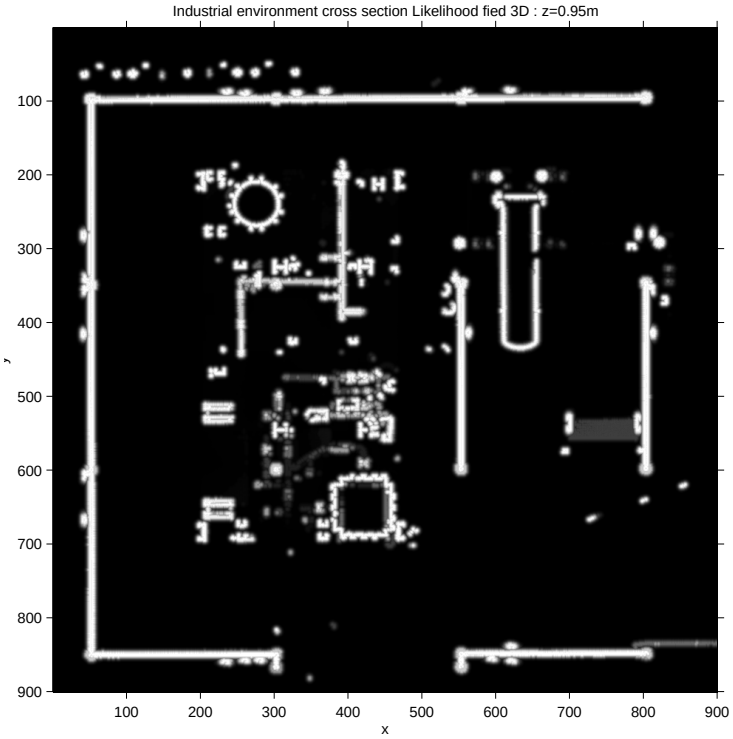
(a) Nuage de points issu de la numérisation du rez-de-chaussée du site industriel de la compétition. ©Total S.A.



(b) Likelihood 1D d'impact LiDAR



(c) Coupe 2D d'un likelihood field gaussien 3D issu d'un impact LiDAR Local



(d) Coupe du résultat du likelihood field de l'environnement complet

FIGURE 3.5 – Exemple de construction d'une carte avec le champ de vraisemblance 3D : à partir du nuage de points issu de la numérisation de l'environnement Fig 3.5a, nous utilisons la variance du LiDAR Fig 3.5b, pour calculer un champ de vraisemblance local 3D Fig 3.5c, puis nous fusionnons l'ensemble des points entre eux dans un champ de vraisemblance global Fig 3.5d.

avec l'*openGL* pour accélérer la détermination de la profondeur de la scène, la fréquence de rafraîchissement ne dépasse pas 10Hz.

— Un autre point problématique est le fait que l'environnement soit constitué essentiellement d'un grand nombre de petits objets et non de surfaces continues importantes. Nous obtenons une fonction de vraisemblance qui est très sensible à une variation de $\mathbf{X}$. Les rayons lancés vont impacter à côté de l'élément de structure voulu, et l'erreur de distance mesurée sera importante. Par exemple, un très faible écart en angle entraînera une très grande variation du résultat de vraisemblance. Par conséquent, une hypothèse proche de la solution sera rejetée du fait de son faible score. Thrun et collab. [2005] expliquent ce phénomène en deux dimensions avec un LiDAR placé dans une salle de classe au niveau des pieds de chaise. Une faible variation d'angle entraîne une forte variation de la scène perçue au sens des distances d'impact raie par raie.

Pour faire face à ces deux problèmes, nous proposons d'étendre l'utilisation du champ de vraisemblance (*likelihood field*), proposé par Thrun et collab. [2005], de deux dimensions à trois dimensions. L'environnement est discrétisé et représenté par une grille 3D de voxels, et pour chaque objet constituant la scène, nous calculons la probabilité d'impact LiDAR en utilisant une distribution normale :

$$p_{hit}(\mathbf{z}|m, \mathrm{O}^k_{\mathcal{M}}) = \frac{1}{\sigma_{map}\sqrt{2\pi}}\ \mathrm{e}^{-\frac{1}{2}\left(\frac{d}{\sigma_{map}}\right)^2} \tag{3.2}$$

Où :

m : Voxel vraisemblance

d : Distance euclidienne du voxel m au $k^{ème}$ obstacle $\mathrm{O}^k_{\mathcal{M}}$ dans $\mathcal{M}$

$\boldsymbol{\sigma}_{map}$: Incertitude de la carte

Chaque voxel m de la grille 3D de la carte constitue le champ de vraisemblance de la manière suivante :

$$p_{hit}(m) = \max_{0 \le k \le n_{obstacle}} p_{hit}(\mathbf{z}|m, \mathrm{O}^k_{\mathcal{M}}) \tag{3.3}$$

Où :

$n_{obstacle}$: nombre d'obstacles dans l'environnement.

Le champ de vraisemblance $\mathscr{L}(\mathcal{M})$, pour la carte $\mathcal{M}$, est une grille 3D qui stocke tous $p_{hit}(m)$ pour tous m résultant d'une discrétisation de $\mathcal{M}$.

Comme vu dans la section 1.2.5, la résolution angulaire du LiDAR détermine le nombre n de raies mesurées. Seul un sous-ensemble $\mathbb{L}$ de ces faisceaux sera valide et deviendra des points 3D. En fait, les objets peuvent être trop loin, leur surface trop absorbante ou bien l'angle d'incidence trop faible .

Grâce au calcul du champ de vraisemblance (Equation 3.3), la probabilité $p(\mathbf{Z}|\mathbf{X}, \mathcal{M})$ est très aisée à calculer. Dans un premier temps, nous projetons les points LiDAR $\mathbf{P}_{\mathcal{L}}$ dans le repère du robot $\mathcal{R}$ puis dans le repère de la carte $\mathcal{M}$ (Figure 3.12) :

$$\mathbf{P}_{\mathcal{M}} = \begin{bmatrix} \mathbf{R}_{\mathcal{R}\to\mathcal{M}} & \mathbf{T}_{\mathcal{R}\to\mathcal{M}} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_{\mathcal{L}\to\mathcal{R}} & \mathbf{T}_{\mathcal{L}\to\mathcal{R}} \\ 0 & 1 \end{bmatrix} \cdot \mathbf{P}_{\mathcal{L}} \tag{3.4}$$

Dans un second temps, les positions obtenues dans la carte $p_{\mathcal{M}}$ permettent de sélectionner des voxels m. Enfin, nous déterminons la vraisemblance de chaque impact $\mathbb{P}(\mathbf{Z}|\mathbf{X}, \mathcal{M})$ directement en lisant le champ de vraisemblance $\mathscr{L}(\mathcal{M})$:

$$p(\mathbf{Z}|\mathbf{X}, \mathcal{M}) = \frac{(\sum_{i=\mathbb{L}} p_{hit}(m_i|\mathrm{p}_{i,\mathcal{M}} \in m_i))^2}{n} \tag{3.5}$$

Où :
$p_{i,\mathcal{M}}$: $i^{ème}$ raie de la mesure LiDAR $\mathbf{P}_{\mathcal{M}}$
m_i : voxel de $\mathscr{L}(\mathcal{M})$ correspondant à la position 3D de $p_{i,\mathcal{M}}$

Comme les impacts LiDAR sont considérés comme indépendants, la combinaison des vraisemblances de chaque raie devrait prendre la forme d'un produit et non d'une somme. Mais avec le nombre d'impacts obtenus, nous dépassons très rapidement les capacités de codage des nombres, même sur 64 bits. Dans sa thèse, El Hamzaoui [2012] présente différents types de sommes pour pallier ce problème.

Le fait d'utiliser une grille 3D pour stocker notre carte n'est pas très favorable à une représentation compacte en mémoire de l'environnement. Pour pallier cet inconvénient et respecter nos contraintes embarquées, nous proposons l'utilisation d'un octree hybride, spécialement adapté pour le stockage de ce champ de vraisemblance 3D.

3.3.4 Stockage du champ de vraisemblance

Les octrees sont des structures récursives de type arbre qui permettent entre autres de représenter efficacement un environnement 3D. Entre chaque niveau de profondeur de nœud, l'espace représenté est divisé en 8 sous-espaces égaux. Nous parlerons de subdivision en octans. Ils sont très utilisés pour stocker des cartes sous forme de nuages de points (Wurm et collab. [2010]). Ils permettent un gain de place dû au fait que seul l'espace occupé sera codé en mémoire, et non l'intégralité du volume comme dans le cas d'un simple tableau 3D. Un autre avantage est que l'identifion des voisins, de par leur organisation géométrique de l'espace, est nettement plus efficace que dans une structure de type liste de points.

Pour stocker des données géométriques en mémoire, il faut stocker leur position et leur valeur. La position peut être décrite de plusieurs manières :

— Un tableau 3D : la structure géométrique est décrite implicitement à travers les indices du tableau. Cela ne consomme pas à proprement parler d'espace mémoire.
— Octree : la structure des liens entre les nœuds organise les coordonnées géométriques des données. Ce sont donc des pointeurs qui occupent de l'espace mémoire pour cette description.
— Liste de points : les trois coordonnées sont directement présentes en mémoire pour chaque point.

Dans le cadre de notre champ de vraisemblance, chaque probabilité d'impact $p_{hit}(m)$ est stockée sur 8 bits. Donc la plus faible valeur supérieure à 0 qui peut être codée est $1/255$. En utilisant un $\sigma_{carte} = 3cm$, notre probabilité d'un impact est égale à zéro à partir de 19cm de distance d'un obstacle (figure 3.5b). Cela veut dire qu'à proximité de tous les objets de la scène, nous aurons une vraisemblance s'étalant sur 19cm (Figure 3.6). Or si nous utilisons par exemple une résolution de carte de 1cm, nous obtenons un rayon de 19 voxels à stocker autour de chaque objet de la carte.

D'un point de vue de la mémoire occupée, un simple tableau 3D est très efficace si le taux d'occupation est très important. Un octree deviendra intéressant si les données à stocker sont relativement éparses. Effectivement sur ce dernier, en plus des données, il faudra également stocker les pointeurs utilisés pour la représentation de sa structure sous-forme d'arbre.

Comme notre champ de vraisemblance s'étale autour des objets, nous aurons une forte densité de voxels occupés autour de la matière et du vide ailleurs. Nous sommes donc dans un cas intermédiaire, et c'est pour cela que nous proposons un *octree hybride* :

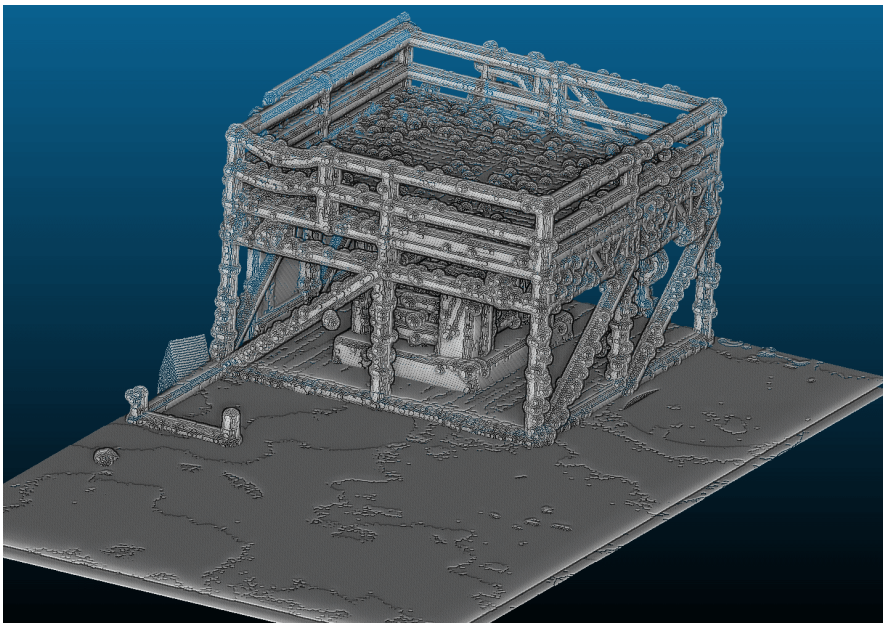


FIGURE 3.6 – Enveloppe de l'octree de la structure présente dans notre laboratoire : représentation des frontières où le champ de vraisemblance s'annule avec une représentation sur 8 bits et $\sigma_{carte} = 3cm$.

une fois que le niveau de profondeur de l'arbre de l'octree atteint une certaine résolution R_o, un tableau 3D code l'information jusqu'à la résolution finale R_a (figure 3.7).

D'un point de vue du temps d'accès, seule la lecture nous importe. En effet la construction de la carte s'effectuant hors ligne, le temps d'écriture a une moindre importance :

— Tableau 3D : accès à 3 pointeurs quelle que soit la dimension de l'environnement.

— Octree : accès à n pointeurs, n évoluant en log_2 de la dimension de l'environnement.

— Octree hybride : accès à $n - log_2(R_o)$ pointeurs + 1 calcul de d'index linéaire dans le tableau 3D, de même n évoluant en log_2 de la dimension de l'environnement.

La consommation mémoire et le temps d'accès de l'*octree hybride* dépendent donc de la dimension du tableau 3D final, et du type de champ de vraisemblance qu'il stocke.

Ses performances sont présentées dans la section 3.4.1.

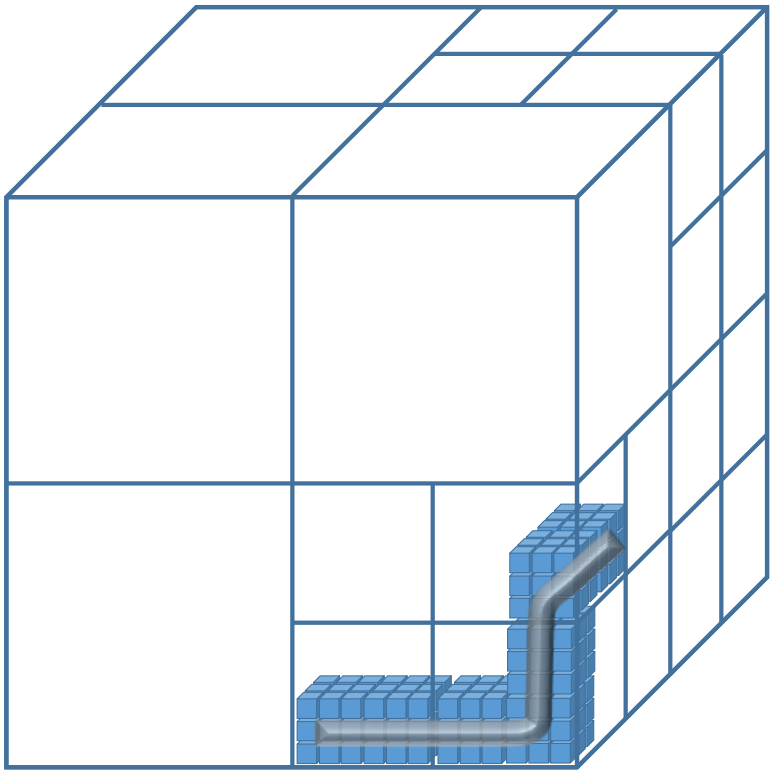


FIGURE 3.7 – *Octree hybride* optimisé pour stocker le champ de vraisemblance : afin de stocker un champ de vraisemblance comme le tuyau gris sur la figure, l'octree sera calculé jusqu'à une résolution R_o puis un tableau 3D permettra d'atteindre la résolution finale R_a. Comme le montre la table 3.1, cette méthode permet d'importants gains de place mémoire par rapport à un « octree pur » dans le cas d'objets relativement denses comme les champs de vraisemblance.

3.4 Résultats expérimentaux

Cette section présente les résultats obtenus avec cette localisation 6DDL en environnement complexe.

Nous disposons de deux sites d'évaluation :

— Notre laboratoire équipé d'une structure représentant un process pétrochimique industriel (Figure 2.8) et numérisée (Figure 2.11) avec notre LiDAR Leica C10 (Figure 1.8a). Nous avons équipé la structure du système de capture de mouvement *Vicon T40* (Figure 2.9a), afin de disposer d'une vérité terrain et de pouvoir qualifier les performances en terme de précision de l'algorithme.

— Le site de la compétition *Argos*, l'*UMAD* à Lacq (Figure 3.1b). Il n'est pas équipé de moyen de mesure de la vérité terrain en terme de localisation, mais il possède plusieurs autres avantages : il est nettement plus vaste, plus réaliste, sur cinq niveaux et en extérieur.

  Un sous-ensemble de cette structure (le premier niveau) nous a servi pour réaliser des simulations dans le but de vérifier les caractéristiques de la fonction de vraisemblance et de convergence du filtre particulaire.

Nous avons également investigué deux technologies de LiDAR afin de caractériser l'apport d'un capteur multi-nappes ML (multi layers) par rapport au classique mono-nappe SL (single layer). Les deux LiDARs testés durant ces expérimentations sont les suivants (Figure 3.8) :

— *SL* : single layer, basé sur le Sick LMS511 [4]

— *ML* : multi layers, basé sur le Velodyne VLP16 [5]

(a) Single layer LiDAR, Sick LMS511 : 190˚fov.

(b) Multiple layers LiDAR : Velodyne VLP16, 16 layer, -15˚to 15˚, 360˚fov.

FIGURE 3.8 – LiDARs utilisés pour l'évaluation de notre localisation.

Dans la suite de ce document nous dénommerons le LM511 comme SL (*Single Layer*) et le VLP16 comme ML (*Multi Layer*)

4. http://www.sick.com/instruments/EN/home/products/flow_sensors/Pages/BulkscanLMS511.aspx

5. http://velodynelidar.com/lidar/hdlproducts/vlp16.aspx

La figure 3.11 présente les robots que nous avons utilisés durant cette étude. Le Jaguar [6] est un robot commercial que nous avons acquis pendant le développement de la première version de Viking avec notre partenaire Sominex [7]. N'étant pas équipé des capteurs « métier », il nous a servi essentiellement à tester la chaîne localisation, pathplanning et pathcontrol pour des trajectoires planes.

Nous avons évalué les performances sur cinq aspects principaux :

— Stockage et exploitation de la carte ;
— Appariement des données et caractéristiques de la fonction vraisemblance ;
— Convergence de l'algorithme en simulation ;
— Localisation en simulation ;
— Localisation embarquée réelle sur les robots dans les deux sites.

3.4.1 Performance du stockage du champ de vraisemblance par l'*octree hybride*

Afin de déterminer l'utilisation mémoire de l'*octree hybride*, nous sommes partis du nuage de points de la structure contenue dans notre laboratoire. Nous n'avons pas pris en compte les murs du laboratoire pour être le plus proche possible d'une situation réaliste en extérieur avec essentiellement des installations dans la scène. Il occupe un volume de $9.8 \times 7 \times 3.8m$ soit $261m^3$ (figure 3.6).

3.4.1.1 Consommation mémoire

Le calcul du champ de vraisemblance est effectué avec une résolution d'$1cm$ et $\sigma_{carte} = 3cm$. Il donne environ 32.7 millions de points de probabilité à stocker.

Nous avons fait varier la dimension du tableau final de l'*octree hybride* selon les puissances de 2, de 1 à 32 voxels. Pour la dimension 1, nous sommes dans le cas d'un octree standard. Chaque donnée est stockée toute seule dans sa feuille. Plus la dimension augmente, plus nous nous rapprochons d'un « simple tableau 3D ». Une comparaison est également effectuée avec un classique tableau 3D. Comme déjà évoqué dans la section 3.3.4, l'espace occupé en mémoire par l'*octree hybride* dépend de la structure de la matière dans l'environnement et des caractéristiques du champ de vraisemblance (résolution spatiale et σ_{carte}).

Les résultats sont présentés dans le tableau 3.1. Le taux de remplissage correspond au rapport du nombre de tableaux présents dans l'*octree hybride* par le nombre de tableaux stockables dans le volume couvert par la structure. Nous constatons un minimum d'espace occupé en mémoire pour une dimension de tableau entre 4 et 8 voxels (figure 3.9). Si nous diminuons la dimension du tableau final, nous stockons beaucoup de pointeurs (phénomène encore accentué par la compilation en 64 bits au lieu de 32). Les données du champ de vraisemblance ne sont pas assez éparses. Au contraire, si nous augmentons la dimension du tableau final, un bonne partie des tableaux se retrouve faiblement remplie. Dans ce cas, la matière n'est pas assez dense pour la dimension du tableau choisie.

Dans ce cas d'une dimension de tableau de 4 à 8 voxels, nous pouvons noter que par rapport à l'*octree hybride* « idéal » avec un tableau de final de 4 voxels, nous économisons 87.2% de l'espace mémoire utilisé par rapport à un tableau 3D classique.

6. http://jaguar.drrobot.com/specification_V4.asp
7. http://www.sominex.fr/

Dimension tableau 3D (R_o)	**Nombre de nœuds de l'arbre**	**Nombre de tableaux 3D**	**Nombre de tableaux 3D stockable**	**Taux de remplissage** (%)	**Taille Mémoire** (Mo)
1^3	5105966	32747775	259940879	12.6	381.8
2^3	728712	4377254	32492610	13.5	83.4
4^3	110909	617803	4061576	15.2	45.3
8^3	18336	92573	507697	18.2	45.9
16^3	3468	14868	63462	23.4	58.3
32^3	727	2741	7933	34.6	85.7
tableau 3D complet	–	–	–	–	247.8

TABLEAU 3.1 – Consommation mémoire de l'*octree hybride* compilé en 64 bits en fonction de la dimension du tableau 3D final : test effectué sur une sous-partie du champ de vraisemblance de la maquette de la structure installée dans notre laboratoire (figure 1.6) $9.8 \times 7 \times 3.8m$ avec une résolution de $R_a = 1cm$ comportant 32.7 millions de points.

Si nous prenons la situation réelle du concours d'avril 2016, avec son site industriel de $43.8 \times 44.5 \times 6.1m$ soit $11890m^3$ couverts (Figure 3.1b), l'*octree hybride* généré occupe 595.2Mo et contient 471.6 millions de points. Un tableau 3D équivalent occuperait 11.1Go, soit une réduction de la place occupée de 94.6%.

Pour une extension de notre méthodologie à des véhicules routiers que nous aborderons en section 3.4.6, nous avons réalisé une carte du campus de l'Esigelec issue du nuage de points de la figure 2.15. Elle mesure $398.7 \times 256.2 \times 14.7m$ avec une résolution de $5cm$ et contenant 229 millions de points. L'octree occupe une place mémoire de 403.5Mo. Un tableau 3D équivalent occuperait 11.2Go, soit encore une réduction importante de la place occupée de 96.4%.

Nous pouvons constater que plus les scènes sont importantes, plus la méthode proposée est avantageuse en terme d'espace nécessaire en mémoire.

3.4.1.2 Temps d'accès

Comme déjà évoqué, nous nous intéressons au temps d'accès de l'*octree hybride* essentiellement en lecture, puisque la phase de construction de la carte est calculée hors ligne.

Le temps d'accès théorique dans l'octree dépend de la profondeur de l'arbre. Il faut parcourir le même nombre de nœuds donc de pointeurs pour arriver sur les données stockées dans le tableau final. La profondeur de l'octree évolue d'une part avec le log_2 du rapport de la dimension de l'environnement sur sa résolution voulue pour le représenter, et d'autre part avec la dimension du tableau final utilisé (Équation 3.6) :

$$P_{octree} = \lceil log_2 \frac{\max_{\forall i} Dim_i}{R_a} \rceil - log_2 R_o \tag{3.6}$$

Où :
- R_a : résolution finale de l'octree hybride
- R_o : dimension du tableau 3D
- Dim : dimensions de l'environnement (X,Y,Z)

Nous préférons parler de temps d'accès « théorique », car avec les systèmes de mémoire

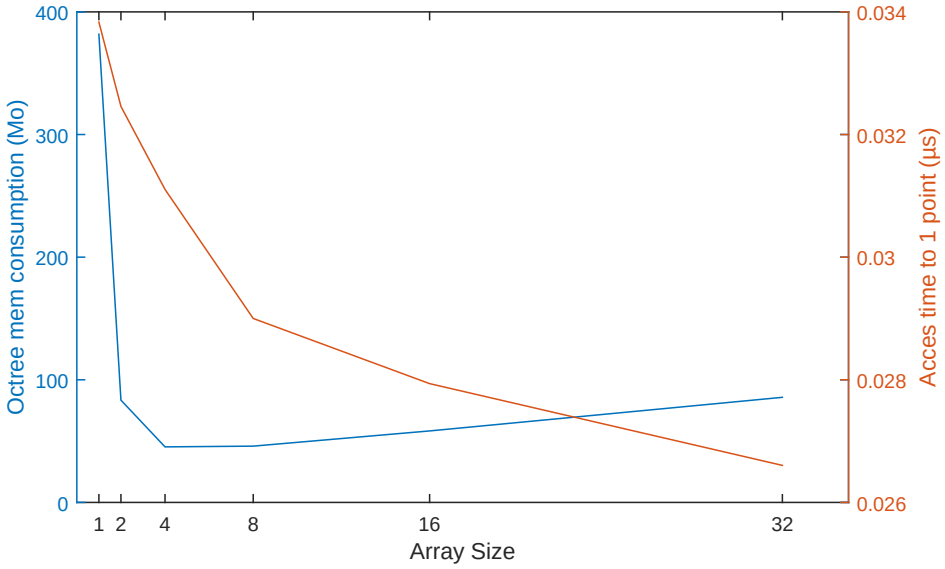


FIGURE 3.9 – Consommation mémoire et temps d'exécution de l'*octree hybride* en fonction de la dimension du tableau final. L'espace mémoire est minimal pour une dimension de tableau de 4 ou 8 éléments de côté.

cache moderne des processeurs actuels, il est très délicat de faire une prédiction de timing. Les performances peuvent être très dépendantes de l'implémentation. La mise en place d'une métrique et d'essais est la seule garante pour valider des idées d'hypothétiques optimisations.

Pour cette évaluation, nous avons utilisé le même environnement que pour la consommation mémoire 3.4.1.1. Nous mesurons le temps nécessaire à la lecture de 19.2 millions de points 3D dans l'octree. Les coordonnées de ces points sont issues d'impacts LiDAR obtenus en simulation dans la structure du laboratoire. Si nous utilisons simplement des coordonnées 3D aléatoires, il y a un biais favorable dans l'évaluation. Du fait que la scène soit plutôt creuse, nous allons majoritairement accéder à des points n'existant pas dans l'octree, c'est-à-dire des espaces vides de l'environnement. L'octree n'étant pas parcouru entièrement, le temps d'accès est nettement plus rapide. Dans le cas extrême d'un premier pointeur *NULL*, le parcours se termine alors à la première étape.

Cette mesure de temps d'accès est effectuée 25 fois, de manière à lisser d'éventuels problèmes de variance dus au système d'exploitation.

Dimension tableau 3D (R_o)	**Profondeur de l'arbre**	**Temps d'accès total** (ms)	**Temps d'accès par point** (ns)
1^3	11	64.96	33.8
2^3	10	62.32	32.4
4^3	9	59.72	31.1
8^3	8	55.68	29.0
16^3	7	53.64	27.9
32^3	6	51.08	26.6
tableau 3D complet	–	25.56	13.3

TABLEAU 3.2 – Performance en temps d'accès à l'*octree hybride* pour 19.2 millions d'impacts LiDAR en fonction de la dimension du tableau final sur 1 core d'un CPU *intel i7-4700MQ @ 2.4GHz*

Les résultats en terme de temps d'accès sont présentés dans la table 3.2 et sur la figure 3.9.

Comme prévu, plus l'octree est profond, plus le temps d'accès est long. Une comparaison par rapport à un tableau 3D simple est proposée sur la dernière ligne. Nous pouvons remarquer que si nous reprenons la configuration donnant les meilleurs résultats en terme d'occupation mémoire (dimension de tableau final entre 4 et 8), le temps d'accès est environ 2.2 fois plus long que dans un tableau 3D. Ce résultat est raisonnable au vu de la relative complexité algorithmique pour accéder à l'*octree hybride*. Le choix d'une dimension de 8^3 semble être un bon compromis d'un point de vue du temps d'accès du fait qu'il soit placé dans le coude de la figure 3.9.

Dans le cadre d'un scan LiDAR complet, en supposant que tous les impacts reviennent au capteur, le temps d'accès serait de :

— 11 µs pour les 381 points d'un LMS511 soit un choix de résolution sur l'angle d'azimut de 0.5° sur 190°.

— 167µs pour les 5760 points d'un VLP16 soit un choix de résolution sur l'angle d'azimut de 1° sur 360°.

Les performances obtenues en terme de temps d'accès et consommation sont compatibles avec l'application embarquée visée.

3.4.1.3 Outil de génération

Nous avons développé un outil en ligne de commande afin de réaliser la conversion d'un nuage de points en champ de vraisemblance puis son stockage dans un octree. Un format de fichier binaire a également été mis au point pour permettre un chargement efficace à partir d'un disque dur.

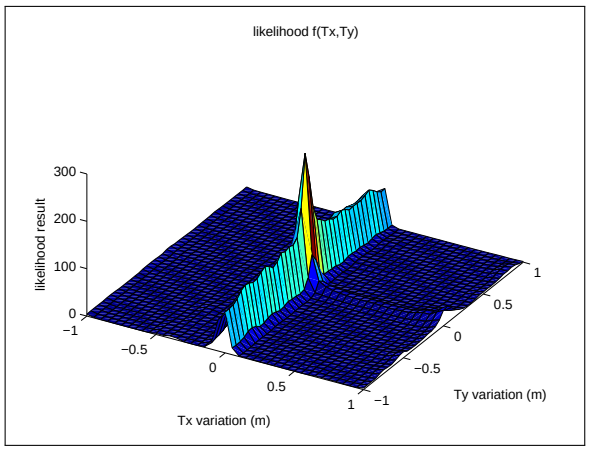
Nous avons effectué plusieurs démonstrations du robot sur des sites nouveaux comme aux journées annuelles des hydrocarbures, ou encore dans la tour Total à défense. Nous numérisons avec le LiDAR Leica la structure où nous souhaitons faire évoluer le robot, puis nous convertissons le nuage de points en champ de vraisemblance avec notre outil. Il dispose également d'une option pour extraire l'enveloppe du champ de vraisemblance (figure 3.6) très utile pour détecter des points mal filtrés.

Il nous faut environ une demie journée de travail pour mettre en place une nouvelle démonstration dans un lieu inconnu. Cela comprend : l'installation de la structure, la numérisation du site, sa conversion en carte ainsi que la création des cartes topologiques utiles au déplacement du robot (position des points de mesures, coursives, connectivité des intersections...).

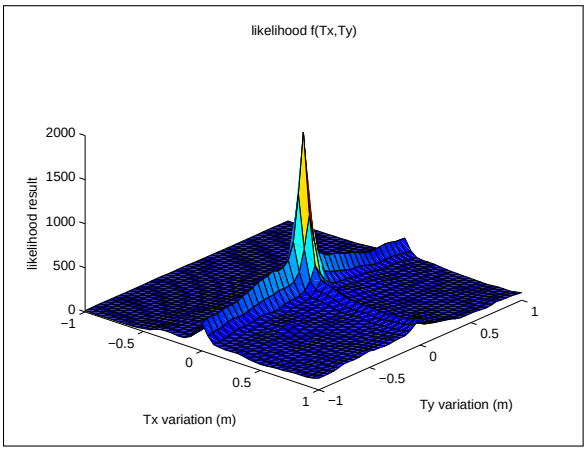
3.4.2 Caractéristiques de la fonction de vraisemblance

Les propriétés idéales de la fonction de vraisemblance que nous avons énoncées dans la section 3.3.1 ne dépendent pas que de la formule utilisée pour définir la fonction elle-même. L'environnement et le capteur jouent également un rôle important.

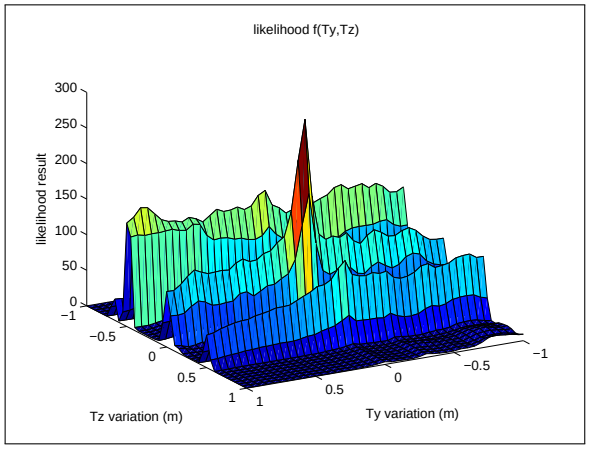
Afin de mettre ces points en évidence, nous avons utilisé le site industriel de la compétition. Nous avons simulé un scan LiDAR $S_{\mathbf{X}_{ref}}$ pour une pose $\mathbf{X}_{ref}$ donnée. Le robot se trouvait sur une passerelle du site, avec la longueur de la coursive alignée sur l'axe y de la carte. Comme il n'est pas facile de représenter des données en plus de 3 dimensions sur une figure, nous avons fait varier cette pose X_{ref} selon 2 dimensions sur un intervalle $\mathbf{I}$ (de $[-1m, 1m]$ pour les translations T_i et $[-20°, 20°]$ pour les rotations



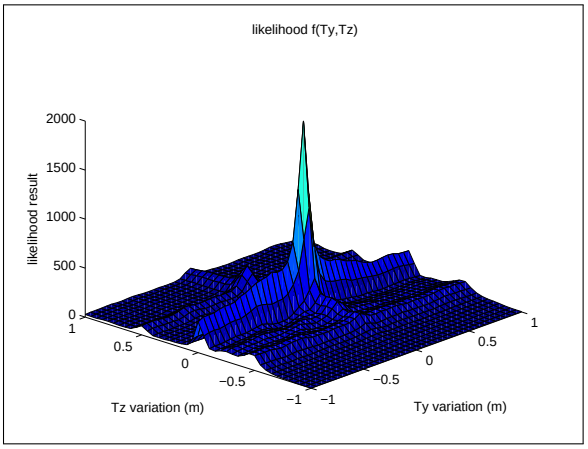
(a) SL translations horizontales



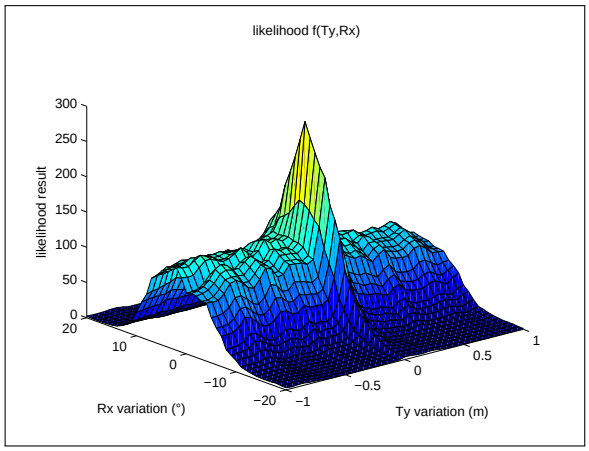
(b) ML translations horizontales



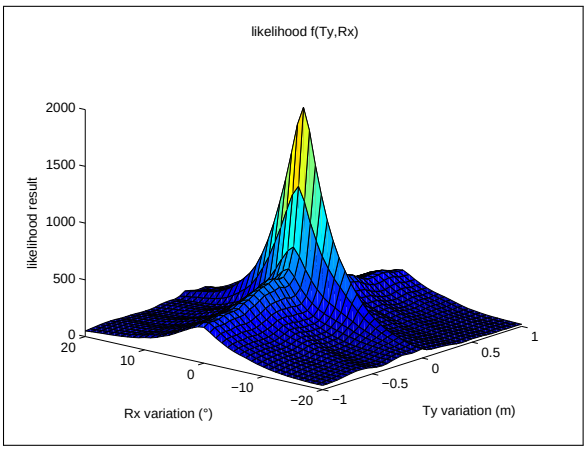
(c) SL translations verticale et horizontale



(d) ML translations verticale et horizontale



(e) SL translation et rotation



(f) ML translation et rotation

FIGURE 3.10 – Évaluation de la fonction de vraisemblance des deux LiDARs (*SL*) et *ML*). Les LiDARs sont positionnés dans un couloir du site industriel de la compétition à Lacq. L'hypothèse de pose est testée avec une variation sur deux axes.

R_i). Puis, nous avons tracé le score de vraisemblance sur l'intervalle I correspondant à $p(\mathbf{Z}|\mathbf{X},\mathcal{M}) = p(\mathbf{S_{Xref}}|\mathbf{X_{ref}}+\mathbf{i},\mathcal{M}, i \in \mathbf{I})$. La figure 3.10 montre le résultat de cette étude.

Afin de disposer d'un indicateur chiffré sur la sélectivité de la fonction de vraisemblance, nous avons également proposé une métrique : le rapport signal sur bruit (*SNR*). Il est défini comme le ratio de la valeur de la vraisemblance au point de référence $\mathbf{X}_{ref}$ sur la moyenne des vraisemblances sur l'intervalle considéré **I** :

$$\mathrm{SNR} = \frac{p(\mathbf{Z}|\mathbf{X}_{ref},\mathcal{M})}{\overline{p}(\mathbf{Z}|\mathbf{X_{ref}}+\mathbf{i},\mathcal{M}, i \in \mathbf{I})} \tag{3.7}$$

TABLEAU 3.3 – Évaluation de la robustesse de la fonction de vraisemblance avec la métrique SNR

		Pose_1		Pose_2	
		SL	ML	SL	ML
	Number of LIDAR beams per measurements	381	5760	381	5760
	Ratio of valid LIDAR beams	0.90	0.61	0.47	0.60
DoF variations	$\mathrm{T}_x, \mathrm{T}_y$	40.32	18.29	5.11	17.84
	$\mathrm{T}_x, \mathrm{T}_z$	31.69	41.66	22.55	44.10
	$\mathrm{T}_y, \mathrm{T}_z$	14.85	25.64	18.47	27.75
	$\mathrm{T}_x, \mathrm{R}_z$	36.77	14.24	3.54	14.14
	$\mathrm{T}_x, \mathrm{R}_y$	23.66	23.14	6.9	24.55
	$\mathrm{T}_x, \mathrm{R}_x$	24.75	19.19	11.26	17.79
	$\mathrm{T}_y, \mathrm{R}_y$	5.52	10.24	6.73	11.52
	$\mathrm{T}_y, \mathrm{R}_x$	6.56	11.28	11.16	10.52

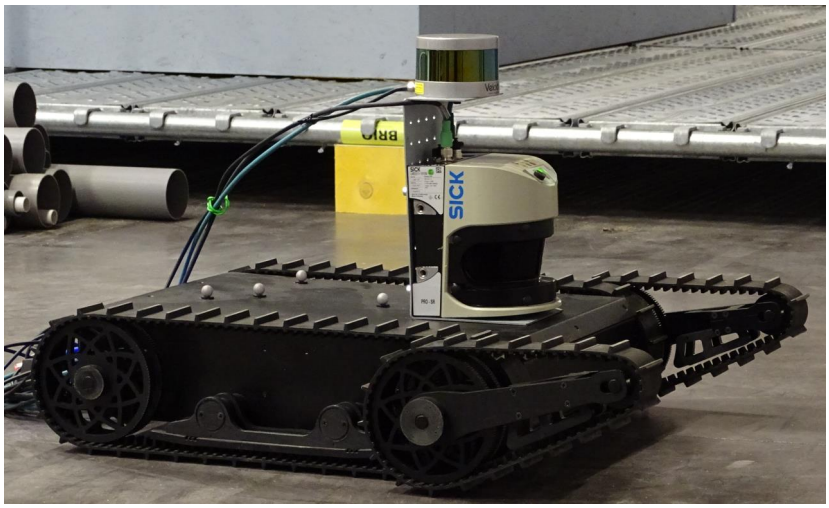
L'évaluation de la table 3.3 a été réalisée pour deux poses différentes :

— Pose_1 : le LiDAR mono-nappe coupe le rail de sécurité entourant les installations.

— $\mathrm{Pose}_2 = \mathrm{Pose}_1 + \begin{bmatrix}0 & 0 & -0.1 & 0 & 0 & 0\end{bmatrix}^{\mathrm{T}}$: sous le rail de sécurité, le LiDAR mono-nappe n'obtient des impacts que d'un côté.

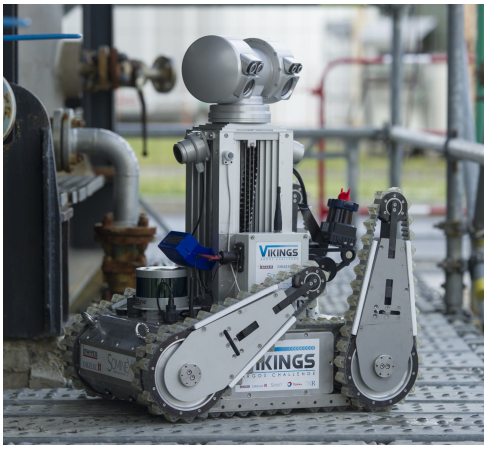
Le ratio SNR du LiDAR ML est plus stable entre les deux poses. L'écart-type entre la Pose_1 et la Pose_2 sont respectivement de 15.52 et 1.54 pour les LiDARs SL et ML. La figure 3.10 met en évidence les caractéristiques de la fonction de vraisemblance et l'apport du LiDAR ML sur le SL. Rappelons que les LiDARs sont simulés pour une pose dans un couloir du site de Lacq, avec l'axe *y* du robot aligné avec la coursive.

Dans le cas de translations purement dans le plan (figures 3.10a et 3.10a) avec le SL, nous obtenons une forte similarité dans l'alignement de la coursive. Dans ce cas de figure, le ML a une dynamique plus grande entre la pose réelle et les poses environnantes. Dans le cas de la variance de hauteur (figures 3.10c et 3.10d), le SL produit des « franges de similarité » importantes sur l'axe z alors que le ML discrimine nettement mieux. C'est tout-à-fait normal puisqu'il perçoit le sol dans ces nappes. Sur un mouvement de roulis (figures 3.10e et 3.10f), c'est toujours le même constat, le ML est plus discriminant que le SL. La fonction de vraisemblance est plus discriminante et robuste avec le LiDAR ML.



(a) Jaguar utilisé pour les expérimentations en laboratoire et la comparaison SL-ML.

(b) Viking v1.0, compétition juin 2105 VLP16 à l'arrière LMS511 à l'avant et IMU bas coût.



(c) Viking v1.2, compétition avril 2016, le LMS511 a été remplacé par un Hokuyo UTM30LX-EW motorisé sur l'axe y.

FIGURE 3.11 – Robots utilisés avec l'algorithme de localisation présenté dans cette section.

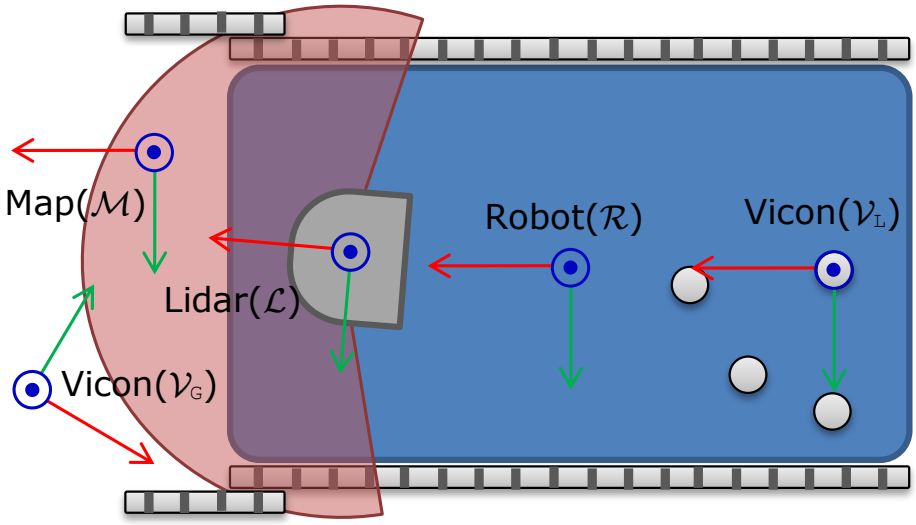


FIGURE 3.12 – Repères de référence utilisés sur les robots (voir section 3.4.5.2).

3.4.3 Etude de la convergence de l'algorithme

Pour évaluer la convergence de notre méthode, nous nous mettons dans la situation du « robot perdu » : nous initialisons le filtre particulaire sur une pose donnée X_{init} avec une erreur (ε_{pos}) par rapport à la position réelle du robot X_{truth}, puis nous observons le comportement de l'algorithme de localisation :

$$X_{init} = X_{truth} + \varepsilon_{pos} \tag{3.8}$$

Dans ce cas, le robot étant immobile, il n'y a pas d'information en provenance de l'étape de *motion update*. Seul le bruit additif assure la convergence vers la position réelle du robot X_{truth}. Nous initialisons les particules du filtre en les tirant aléatoirement selon une loi normale $\mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_i)$. Σ_i est une matrice diagonale de covariance définie comme ceci : $\sigma_i = abs(\varepsilon_{pos})$. La variance du bruit additif est donc fixée à un dixième de celle de la répartition initiale des particules. Donc les bruits proportionnels et additifs de l'étape *motion update* sont fixés à $\sigma_p = \mathbf{0}$ et $\sigma_a = \sigma_i / 10$.

Pour comparaison entre la pose réelle $\mathbf{X}_t$ et la pose estimée par le filtre particulaire $\hat{\mathbf{X}}_\mathbf{t}$, nous évaluons séparément les dimensions position et orientation du vecteur d'état (équation 3.1). Nous calculons la distance euclidienne pour comparer les positions et l'angle solide 3D pour les orientations. L'angle solide 3D Ω est défini de la manière suivante :

$$\Omega = \arccos \frac{\mathbf{X}_t \cdot \hat{\mathbf{X}}_t}{\|\mathbf{X}_t\| \cdot \|\hat{\mathbf{X}}_t\|} \tag{3.9}$$

Les résultats de cette convergence pour les deux types de LiDAR sont présentés sur la figure 3.13. Nous pouvons constater que la pente de convergence est plus raide pour le ML que le SL. L'erreur finale est également en faveur du ML.

Toujours en simulation, nous avons ensuite étudié l'influence du nombre de particules sur la précision de cette convergence pour une erreur de pose initiale :

$$\varepsilon_{pos} = \begin{bmatrix} 0.2 & -0.2 & 0.05 & 2° & -2° & 5° \end{bmatrix}^{\mathrm{T}}$$

La convergence est calculée sur 200 itérations de l'étape de *measurement update*, le résultat est défini par la moyenne et l'écart-type sur les 50 derniers échantillons. Afin de s'affranchir de la variance du processus stochastique surtout quand le nombre de particules est faible, cette évaluation a été répétée 8 fois. C'est donc la moyenne de chaque métrique qui est représentée sur la figure 3.14. Le bruit LiDAR était tiré aléatoirement selon $\mathcal{N}(0, \sigma_{lidar})$ avec $\sigma_{lidar} = 0.01m$. Nous pouvons en conclure que le nombre idéal de particules se situe entre 250 et 500.

La même étude est proposée figure 3.15, en fixant le nombre de particules à 500, et en faisant varier le bruit LiDAR. La méthodologie est en tout point semblable. La précision de notre méthode avec le multi-nappe est peu influencée par le bruit du capteur, seule la variance augmente. Par contre le mono-nappe qui n'a pas une vision « globale » de l'environnement est fortement perturbé.

La robustesse à une initialisation plus erronée a également été évaluée en altérant fortement ε_{pos}. Par exemple, avec $\varepsilon_{pos} = \begin{bmatrix} 1 & 1 & 0.1 & 5° & -5° & 5° \end{bmatrix}$, le ML converge toujours alors que le SL ne converge plus. Avec une erreur de position plus modérée mais la même erreur d'orientation, $\varepsilon_{pos} = \begin{bmatrix} 0.5 & 0.5 & 0.05 & 5° & -5° & 5° \end{bmatrix}$, le SL converge correctement pour l'orientation ou la position mais rarement les deux à la fois. Alors que pour une forte erreur de position et une erreur d'orientation plus modeste,

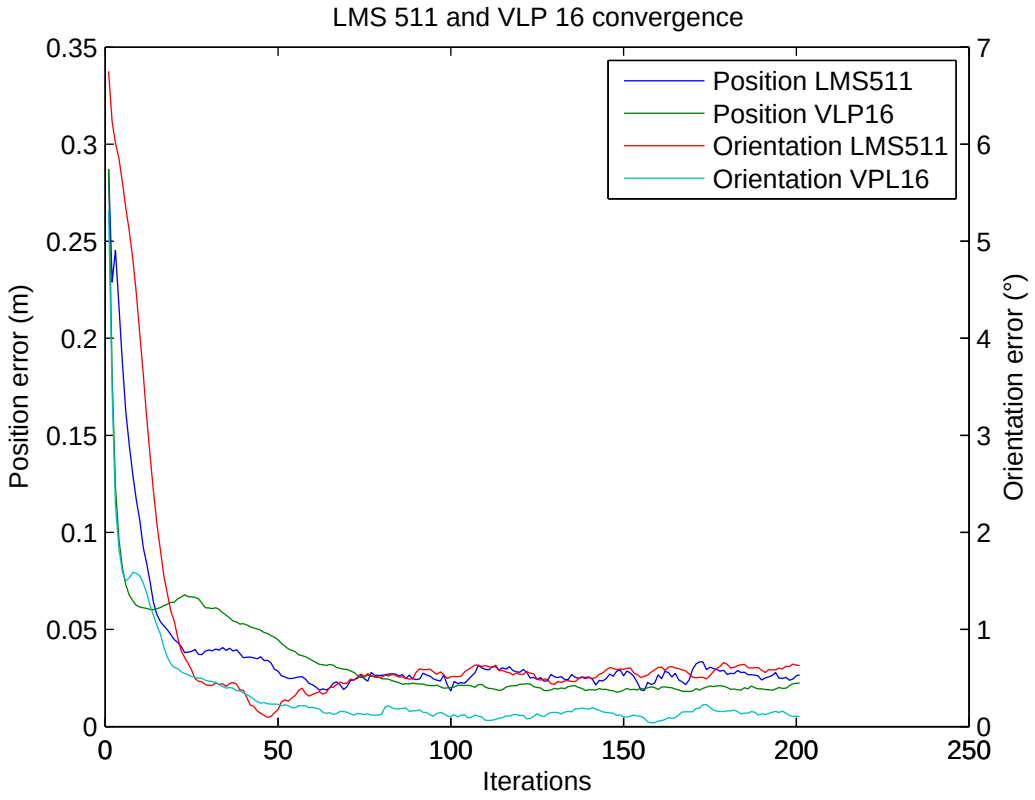
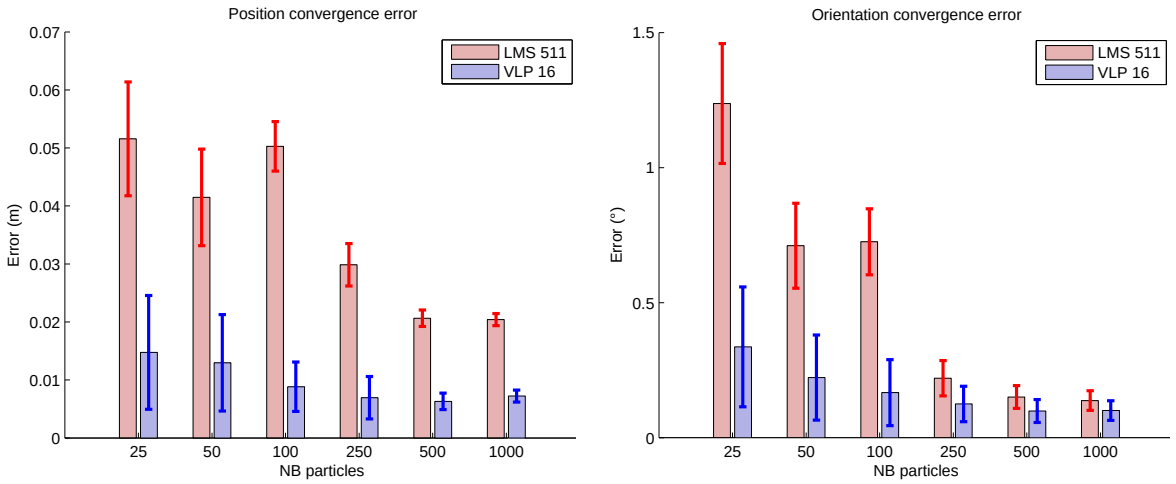


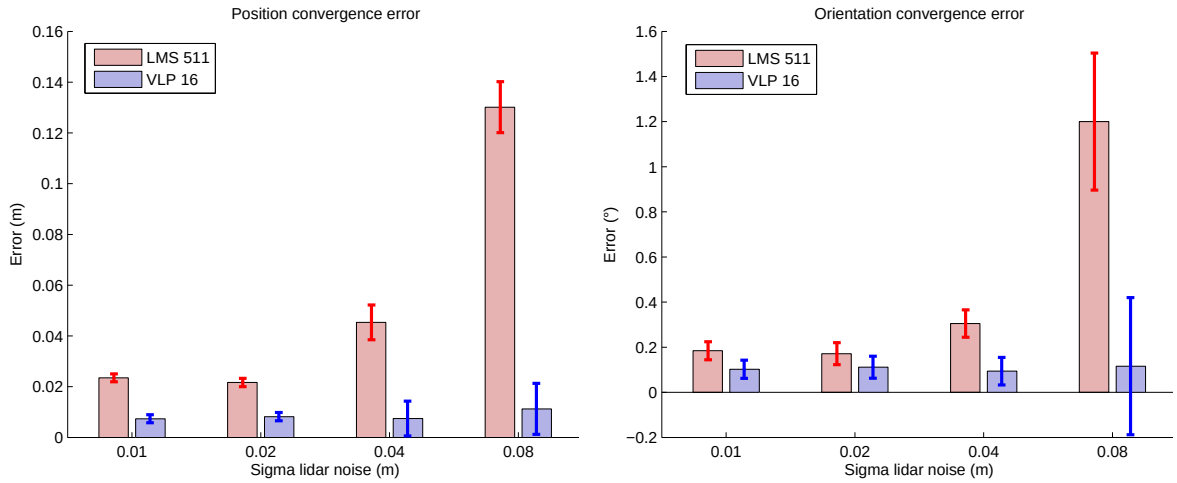
FIGURE 3.13 – Comparaison de la convergence en position et orientation avec un LiDAR mono-nappe et multi-nappes : *1000 particules*, σ_{lidar}=*1cm*



(a) Erreur de convergence en position

(b) Erreur de convergence en orientation

FIGURE 3.14 – Moyenne et écart-type de la précision de convergence en fonction du nombre de particules



(a) Erreur de convergence en position

(b) Erreur de convergence en orientation

FIGURE 3.15 – Moyenne et écart-type de la précision de convergence en fonction du bruit LiDAR σ_{lidar}.

$\varepsilon_{pos} = \begin{bmatrix} 0.5 & 0.5 & 0.05 & 1° & -1° & 5° \end{bmatrix}$, il converge systématiquement. Le LiDAR mono-nappe est beaucoup plus sensible que le LiDAR multi-nappes, par rapport aux degrés de liberté sur lesquels il ne perçoit que partiellement l'environnement : t_z, r_x and r_y.

3.4.4 Localisation en simulation

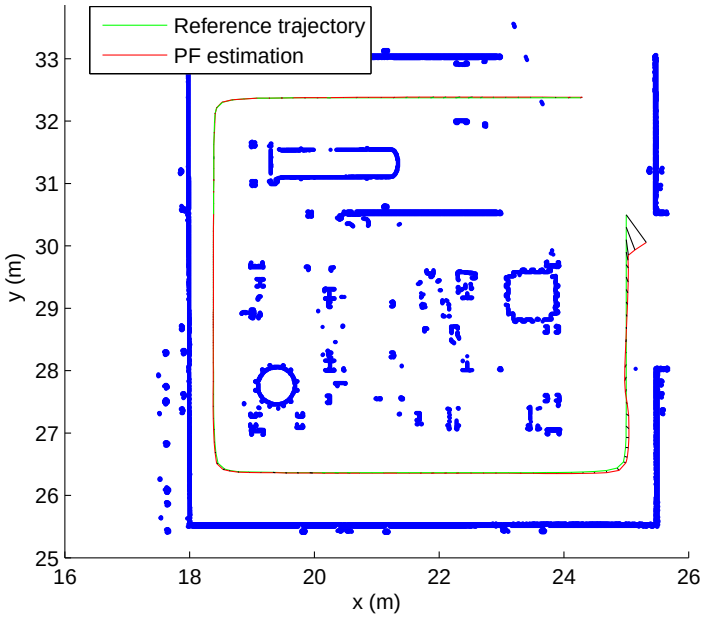
Pour tester le processus de localisation complet, nous avons généré avec le robot une trajectoire constituée de 160 poses LiDAR et de l'odométrie correspondante (figure 3.16a) sur le site de la compétition (figure 3.1 et 3.5a). Sa longueur est de 22.1m. Le bruit LiDAR est de $\sigma_{lidar} = 0.01m$. L'odométrie est bruitée proportionnellement aux déplacements effectués, 10% en linéaire et 20% en rotation (figure 3.16b). Les résultats sont obtenus avec 500 particules. La position initiale est altérée de
$\varepsilon_{pos} = \begin{bmatrix} 0.5 & -0.5 & 0.05 & 1° & -1° & 5° \end{bmatrix}$ (figure 3.16c).

Malgré une erreur initiale de position de 71cm et d'orientation de 5.1°, l'erreur de position moyenne est inférieure à 3cm et l'erreur d'orientation moyenne est inférieure au degré comme le montre la table 3.4. La performance atteinte par le LiDAR ML est supérieure à celle du LiDAR SL aussi bien pour la position que pour l'orientation.

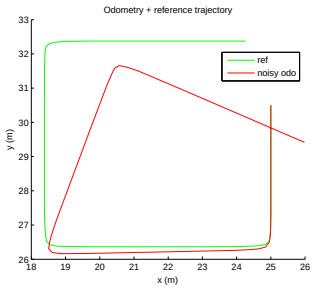
TABLEAU 3.4 – Performance de localisation en simulation sur un trajectoire de 22.1m avec 500 particules et $\sigma_{lidar} = 1cm$.

	Position error (m)		Orientation error (°)		Computation time
	mean	std	mean	std	Intel i2640M-2.8GHz (ms)
SL	0.0223	0.0145	0.45	0.81	0.9 (381 LiDAR impacts)
ML	0.0157	0.0120	0.31	0.44	2.2 (5760 LiDAR impacts)

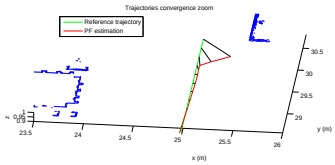
Les temps de calcul de la fonction de vraisemblance sous Matlab sont également donnés dans la table 3.4. Le temps d'exécution n'étant pas linéaire par rapport au nombre d'impacts LiDAR, nous pouvons en déduire que le temps d'accès à la mémoire du champ de vraisemblance 3D n'est pas prédominant devant l'appel des diverses fonctions Matlab.



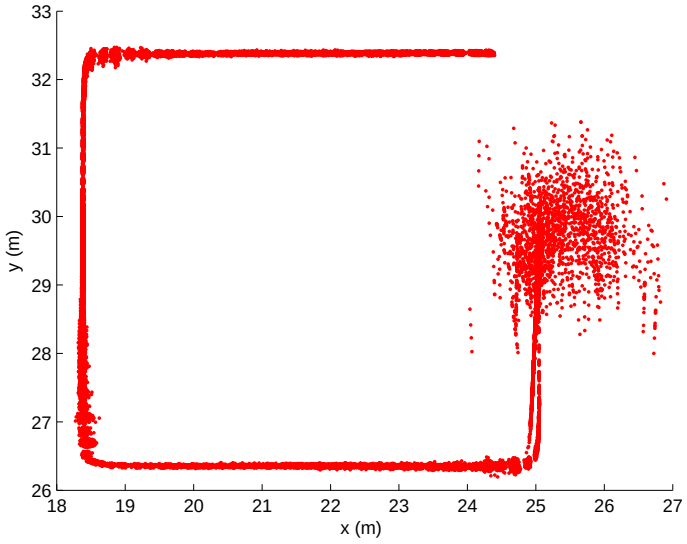
(a) Trajectoire de référence *(vert)*, trajectoire estimée par le filtre *(rouge)*, coupe de l'environnement *(bleu)*



(b) Trajectoire de référence et intégration de l'odométrie bruitée



(c) Zoom sur la convergence du filtre particulaire



(d) Intégration du nuage de particules

FIGURE 3.16 – Étude de la localisation en simulation dans l'environnement de la compétition de Lacq, à l'aide du LiDAR SL, 500 particules et avec $\sigma_{lidar} = 1cm$.

3.4.5 Evaluation des performances sur robots réels

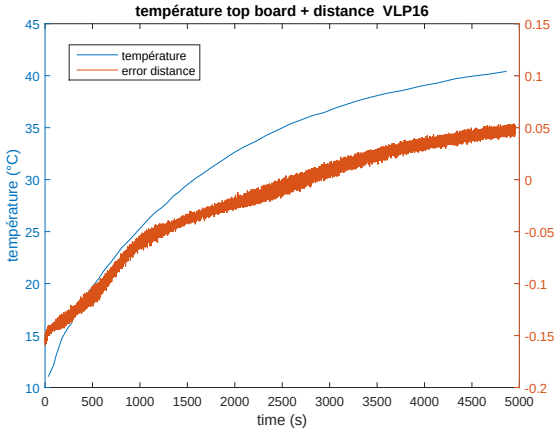
Les différentes pré-études réalisées en simulation nous ont permis de mettre rapidement en évidence certaines caractéristiques de notre méthodologie, difficiles à aborder sur des expérimentations réelles.

3.4.5.1 Contexte des expérimentations réelles

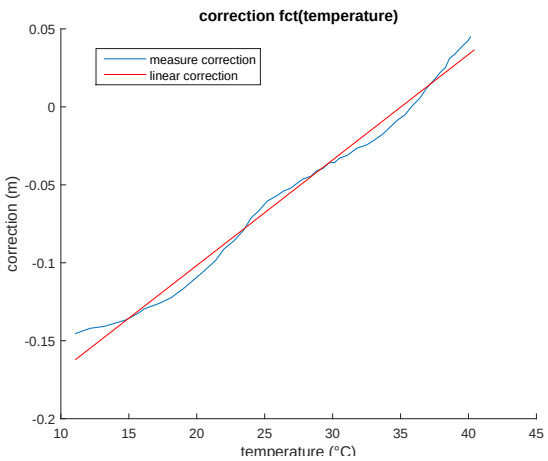
Les expérimentations sont découpées en deux parties ; tout d'abord dans notre laboratoire où nous disposons d'une vérité terrain, puis sur le site de la compétition, où nous devons nous contenter d'une évaluation « qualitative ».

Comme expliqué précédemment, nous avons démarré les expérimentations sur le robot Jaguar (figure 3.11a) avec un LiDAR mono-nappe *LMS511*, en connaissant les limitations de notre méthode de ce capteur grâce aux résultats obtenus en simulation. Puis nous avons pu installer le *VLP16* sur le Jaguar, et enfin le tester sur le Viking v1.0 avec une trajectoire 6DDL.

Comme tout développement, nous avons rencontré des difficultés face auxquelles nous avons dû nous adapter : le VLP16 n'a été disponible que trois mois avant la compétition, soit en mars 2015 bien après le démarrage nos travaux. De plus, un bug entraînant une dérive en température a nécessité une correction externe au capteur (figure 3.17). L'intégration matérielle du robot Viking 1.0 (figure 3.11b) a été terminée courant mai. La maquette représentative du process pétro-chimique intallée dans notre laboratoire (figure 2.8) était opérationnelle depuis fin mars, et équipée de la vérité terrain avec 14 caméras *Vicon T40*. Les 6 autres caméras T40 du laboratoire couvrent la zone d'entrée dans la structure.



(a) Dérive de la mesure de distance du LiDAR *VLP16* en fonction de sa montée en température.



(b) Correction linéaire en attendant que *Velodyne* corrige le bug.

FIGURE 3.17 – Problème de dérive en température du LiDAR *VLP16* et correction apportée.

Ces différents aléas de planning et techniques rencontrés lors du développement des deux versions du Viking appuient la méthodologie proposée dans la section 2.1. Quand le Viking a enfin été opérationnel, nous avions déjà testé l'ensemble de la chaîne, les simulations Matlab pour la méthode, les simulations sur les composants RTMaps propriétaires pour la charge *CPU* et les tests échelle réduite sur le Jaguar.

3.4.5.2 Éléments expérimentaux, repères et calibration

La figure 3.12 présente les différents repères utilisés pour mettre en œuvre la localisation et la vérité terrain avec le système *Vicon* :

— Le repère du robot $\mathcal{R}$ est défini comme le centre du robot. Il correspond au point d'application de la mesure odométrie. L'odométrie est échantillonnée à 25Hz. Les détails concernant l'odométrie sont abordés en annexe C.

— Le repère du LiDAR $\mathcal{L}$ est défini au centre optique du capteur. Comme expliqué dans l'équation 3.4, nous avons besoin de connaître la transformation $\mathcal{L} \rightarrow \mathcal{R}$ de manière à ce que les étapes de *motion* et *measurement update* aient le même point d'application au centre du robot. Cette transformation est directement obtenue par mesure dans le modèle CAO du robot. Le bruit additif σ_a de l'étape *motion update* du filtre particulaire permet de tenir compte d'une erreur de transformation, mais surtout d'une faible modélisation des glissements odométriques (toujours en annexe C).

— Le repère du *Vicon* local au robot $\mathcal{V}_\mathrm{L}$ est défini par les marqueurs installés sur le mobile et le modèle que nous définissons dans le logiciel de tracking *Nexus*. Afin de pouvoir comparer le résultat de la localisation avec la vérité terrain du *Vicon*, il est nécessaire de connaître la transformation $\mathcal{R} \rightarrow \mathcal{V}_\mathrm{L}$. Nous l'obtenons par une première mesure puis par minimisation.

— La localisation va nous donner une pose dans le repère de la carte $\mathcal{M}$ par rapport au repère du robot $\mathcal{R}$. Le repère de la carte est défini à la création du champ de vraisemblance avec l'outil présenté en section 3.4.1.3.

— Enfin le Vicon nous donne une vérité terrain entre le repère local Vicon sur le robot $\mathcal{V}_\mathrm{L}$ et son repère global à ses caméras $\mathcal{V}_\mathrm{G}$. $\mathcal{V}_\mathrm{G}$ est défini lors de la procédure de calibration du *Vicon* à l'aide du *wand de référence*. La vérité terrain est échantillonnée à 100Hz. Pour rappel, la précision du *Vicon* est abordée dans l'annexe A.

  La transformation entre le repère global du *Vicon* $\mathcal{V}_\mathrm{G}$ et celui de notre carte $\mathcal{M}$ est obtenue par calibration en positionnant des marqueurs Vicon sur des points de la scène dont nous connaissons les coordonnées dans la carte. Il nous faut au minimum 4 points, puis nous pouvons appliquer la méthode des moindres carrés pour identifier la transformation.

soit :

$\mathbf{P}_\mathrm{M}^n = \left\{ \begin{bmatrix} x_\mathrm{M} & y_\mathrm{M} & z_\mathrm{M} & 1 \end{bmatrix}^\mathrm{T} \right\}^n$: un ensemble de points exprimés en coordonnées homogènes dans le repère $\mathcal{M}$

$\mathbf{P}_\mathrm{V}^n = \left\{ \begin{bmatrix} x_\mathrm{V} & y_\mathrm{V} & z_\mathrm{V} & 1 \end{bmatrix}^\mathrm{T} \right\}^n$: un ensemble de points exprimés en coordonnées homogènes correspondant à $\mathbf{P}_\mathrm{M}^n$ dans le repère $\mathcal{V}_\mathrm{G}$

La transformation point par point du repère carte $\mathcal{M}$ vers le repère global Vicon $\mathcal{V}_\mathrm{G}$ peut alors s'exprimer de la manière suivante :

$$\mathbf{TrP}_\mathrm{M} = \mathbf{P}_\mathrm{V}$$

$$\begin{bmatrix} h11 & h12 & h13 & h14 \\ h21 & h22 & h23 & h24 \\ h31 & h32 & h33 & h34 \\ h41 & h42 & h43 & h44 \end{bmatrix} \mathbf{P}_\mathrm{M} = \mathbf{P}_\mathrm{V} \tag{3.10}$$

Pour résoudre le système, il nous faut l'exprimer sous la forme :

$$\mathbf{A}x = \mathbf{B} \tag{3.11}$$

$$\overbrace{\begin{bmatrix} x_{\mathrm{M}}^1 & y_{\mathrm{M}}^1 & z_{\mathrm{M}}^1 & 1 & 0 & & & & & \cdots & & & & & 0 \\ 0 & 0 & 0 & 0 & x_{\mathrm{M}}^1 & y_{\mathrm{M}}^1 & z_{\mathrm{M}}^1 & 1 & 0 & & & \cdots & & & 0 \\ 0 & & & & \cdots & & & 0 & x_{\mathrm{M}}^1 & y_{\mathrm{M}}^1 & z_{\mathrm{M}}^1 & 1 & 0 & \cdots & 0 \\ 0 & & & & & \cdots & & & & & & 0 & x_{\mathrm{M}}^1 & y_{\mathrm{M}}^1 & z_{\mathrm{M}}^1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{\mathrm{M}}^n & y_{\mathrm{M}}^n & z_{\mathrm{M}}^n & 1 & 0 & & & & & \cdots & & & & & 0 \\ 0 & 0 & 0 & 0 & x_{\mathrm{M}}^n & y_{\mathrm{M}}^n & z_{\mathrm{M}}^n & 1 & 0 & & & \cdots & & & 0 \\ 0 & & & & \cdots & & & 0 & x_{\mathrm{M}}^n & y_{\mathrm{M}}^n & z_{\mathrm{M}}^n & 1 & 0 & \cdots & 0 \\ 0 & & & & & \cdots & & & & & & 0 & x_{\mathrm{M}}^n & y_{\mathrm{M}}^n & z_{\mathrm{M}}^n & 1 \end{bmatrix}}^{16} \begin{bmatrix} h11 \\ h12 \\ h13 \\ h14 \\ h21 \\ h22 \\ h23 \\ h24 \\ h31 \\ h32 \\ h33 \\ h34 \\ h41 \\ h42 \\ h43 \\ h44 \end{bmatrix} = \begin{bmatrix} x_{\mathrm{V}}^1 \\ y_{\mathrm{V}}^1 \\ z_{\mathrm{V}}^1 \\ 1 \\ x_{\mathrm{V}}^2 \\ y_{\mathrm{V}}^2 \\ z_{\mathrm{V}}^2 \\ 1 \\ \vdots \\ x_{\mathrm{V}}^n \\ y_{\mathrm{V}}^n \\ z_{\mathrm{V}}^n \\ 1 \end{bmatrix} \tag{3.12}$$

Pour un nombre de points supérieur à 4, le système sera sur-défini. Nous pouvons donc appliquer les moindres carrés, pour identifier la meilleure transformation possible :

$$\begin{aligned} \mathbf{A}x &= \mathbf{B} \\ \mathbf{A}^{\mathrm{T}}\mathbf{A}x &= \mathbf{A}^{\mathrm{T}}\mathbf{B} \\ \left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{A}x &= \left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{B} \\ x &= \left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{B} \end{aligned} \tag{3.13}$$

(3.14)

Après estimation, la transformation **Tr** de l'équation 3.10 devrait être équivalente à :

$$\mathbf{Tr} = \begin{bmatrix} \mathbf{R} & \mathbf{T} \\ 0\cdots0 & 1 \end{bmatrix} = \begin{bmatrix} r11 & r12 & r13 & t1 \\ r21 & r22 & r23 & t2 \\ r31 & r32 & r33 & t3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \approx \begin{bmatrix} h11 & h12 & h13 & h14 \\ h21 & h22 & h23 & h24 \\ h31 & h32 & h33 & h34 \\ h41 & h42 & h43 & h44 \end{bmatrix} \tag{3.15}$$

La matrice de rotation **R** doit être orthogonale ($\det\mathbf{R} = 1$). Les moindres carrés n'assurant pas cette contrainte, il est nécessaire d'orthogonaliser la matrice obtenue *a posteriori*.

Donc en résumé, la localisation a besoin du changement de repère des points LiDAR dans le repère du robot : $\mathcal{L} \rightarrow \mathcal{R}$

Puis afin de comparer les résultats avec la vérité terrain du système Vicon, il faut exprimer les poses déterminées par l'algorithme de localisation dans le repère global du Vicon. Il y a deux transformations à réaliser :

— Du repère représentant le centre du robot à celui des marqueurs Vicon sur le robot : $\mathcal{R} \rightarrow \mathcal{V}_{\mathrm{L}}$

— Puis de la carte vers le repère global du Vicon : $\mathcal{M} \rightarrow \mathcal{V}_{\mathrm{G}}$

Sur des plateformes à chenilles comme nous l'aborderons en annexe C, la précision de l'estimation des déplacements angulaires par l'odométrie peut s'avérer extrêmement médiocre. Dans ce cadre, nous avons intégré une centrale inertielle bas coût dans les robots Viking. Elle nous permet de corriger dans une certaine mesure ces dérives. Nous l'utilisons également pour obtenir une première approximation du roulis et du tangage (r_x et r_y), qui sont au même titre que t_z, des degrés de liberté ne pouvant pas être estimés par l'odométrie.

3.4.5.3 Résultats avec vérité terrain

Nous avons donc commencé les expérimentations par des trajectoires planes avec le Jaguar et un LiDAR SL le *Sick LMS511*. Trois séquences (figure 3.18) ont été exécutées. L'algorithme détermine la pose avec 500 particules. Il est initialisé avec une erreur par rapport à la position réelle du robot. Les résultats sont présentés dans la table 3.5.

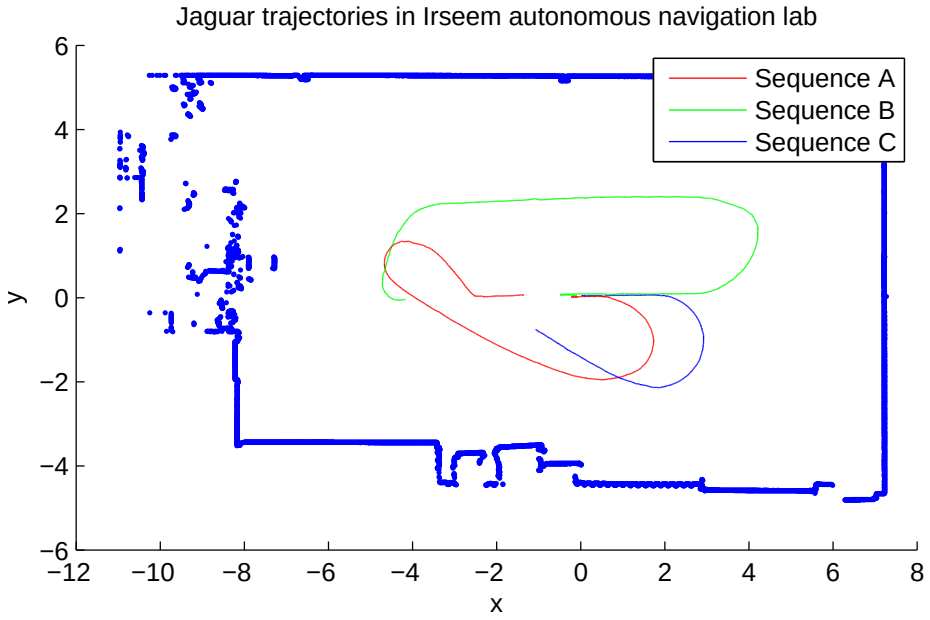


FIGURE 3.18 – Trois trajectoires dans le plan avec le Jaguar et un LiDAR *LMS511*, la vérité terrain est obtenue par le système *Vicon*

Avec le LiDAR SL *LMS511*, la localisation est calculée à la vitesse de rafraîchissement du capteur, c'est-à-dire 25Hz (trajectoire A-C). Les résultats que nous avons trouvés (table 3.5) sont plutôt légèrement meilleurs que ceux obtenus en simulation (table 3.4). Après réflexion, la trajectoire utilisée en simulation est échantillonnée en distance tous les 13cm, alors que le système réel est échantillonné à 25Hz, ce qui correspond de 3 à 7mm entre chaque scan LiDAR pour les trajectoires A-C. Pour effectuer une comparaison avec les simulations, la trajectoire A a été rééchantillonnée tous les 13cm et devient la trajectoire D. Les résultats obtenus sont alors sensiblement les mêmes qu'en simulation.

Une fois le Viking et le *VLP16* disponibles, nous avons effectué la trajectoire E. C'est une trajectoire 6DDL, dans le sens où le robot monte la marche de la structure de notre laboratoire, fait un aller-retour sur la plateforme en franchissant un obstacle puis redescend. Contrairement au Jaguar, le LiDAR du Viking est orienté vers l'arrière et 110˚de ses 360˚de son champ de vue horizontale sont masqués par le mât du robot. Le LiDAR a une fréquence de rafraîchissement de 20Hz, la localisation est donc calculée à cette cadence.

TABLEAU 3.5 – Résultats des expérimentations réelles avec vérité terrain par le *Vicon* dans notre laboratoire, 500 particules.

Sequence number and length (m)	Sampling type	Position error (m)			Orientation error (°)	
		initial	mean	std	mean	std
A : 14.97	25Hz	0.57	0.0106	0.0054	0.963	0.567
B : 17.71		0.82	0.0192	0.0128	0.804	0.389
C : 8.71		0.28	0.0178	0.0096	0.890	0.435
D : 14.97	0.13m	0.57	0.0279	0.0185	0.959	1.699
E : 22.5	20Hz	0.15	0.0236	0.0113	0.292	0.162

Les résultats sont également présentés dans la table 3.5. Nous pouvons noter que l'erreur d'orientation est nettement plus faible que pour le LiDAR SL.

En terme de ressource CPU, la localisation est calculée à 20Hz, avec 18084 points LiDAR évalués à chaque seconde et 500 particules. La fonction de vraisemblance est donc calculée environ 9 millions de fois par seconde. Le processeur embarqué est *intel i7-4600U* à 2.1GHz, le composant RTMaps utilise 16% du temps CPU qui se répartit ainsi : 3.7% pour le *motion update* et 96.3% pour le *measurement update*.

L'implémentation du code fait appel à la parallélisation. Les deux étapes *motion* et *measurement update* sont multithreadées par rapport au vecteur de particules. La consommation CPU obtenue laisse de la latitude pour les autres fonctions du robot.

Disposer d'une localisation avec un taux de rafraîchissement important, 20Hz dans notre cas, est un bon atout. Ainsi, les non linéarités du contrôle commande seront très fortement atténuées.

3.4.5.4 Résultats sur le site *UMAD* de la compétition

La méthode proposée dans ce chapitre a donc été utilisée pour le Challenge Argos sur le site de L'UMAD (figure 3.1b). Lors de la première compétition en juin 2015, les missions se déroulaient toutes au rez-de-chaussée de la plateforme. Le robot devait tout-de-même franchir la marche pour accéder aux installations. En avril 2016, les missions se déroulaient sur les deux premiers niveaux, avec franchissement des escaliers et d'obstacles positifs ou négatifs (figure 3.19). La localisation a parfaitement rempli son rôle dans toutes les situations sans défaillance majeure. Viking v1.0 et v1.2 ont remporté les deux premières compétitions du Challenge Argos.

Sur le site de Lacq, les deux robots ont parcouru un total de 7.8km et ont été correctement localisés dans les installations pendant 16h10 à une vitesse moyenne de 2km/h (figure 3.20).

FIGURE 3.19 – Exemples de trajectoires 6D exécutées par le robot Viking, *en haut* négociation des escaliers sur les installations du site de la compétition, *en bas* franchissement d'obstacles.

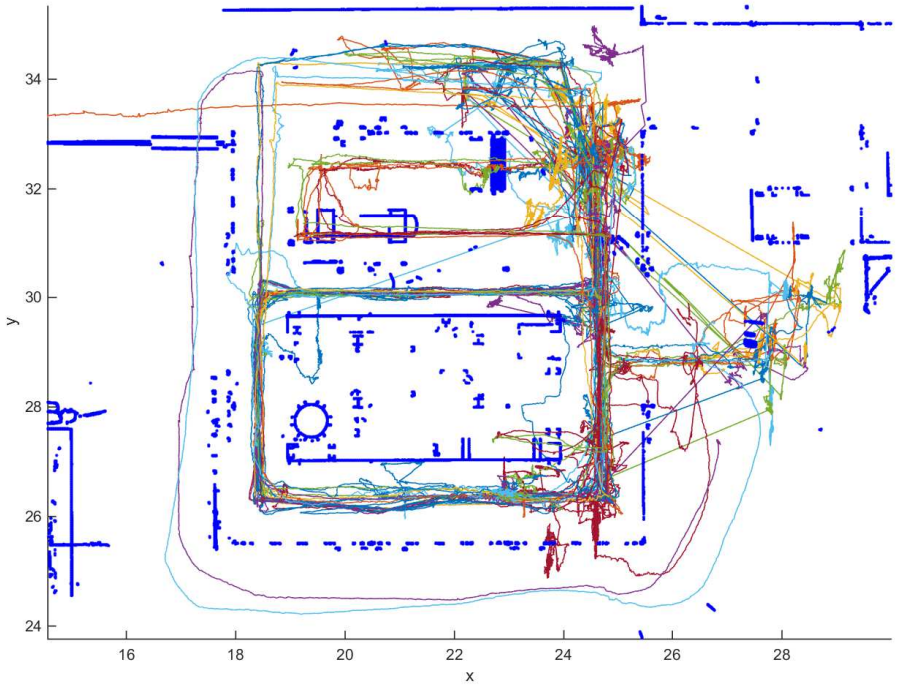


FIGURE 3.20 – Vue du dessus du cumul des trajectoires effectuées sur le site de Lacq avec les deux versions du Viking. Bleu épais, superposition de 3 coupes de la carte 3D à 0.3, 0.5 et 0.9m. En couleur : cumul des trajectoires 3D. Le robot a été correctement localisé pendant 16h10 et a parcouru 7.8km, sa vitesse moyenne est de 2km/h. Il est souvent arrêté pour faire des mesures.

3.4.6 Ouverture sur le domaine des véhicules routiers

Devant les performances de notre méthode sur nos robots, nous avons voulu évaluer les résultats sur un véhicule routier. Nous nous intéressons particulièrement à la phase d'entrée-sortie de garage et autres parkings souterrains/aériens, avec l'objectif de développer un « valet automatique ». Ces lieux comportent deux intérêts pour notre localisation : fortes pentes et un milieu entre intérieur et extérieur.

3.4.6.1 Création de la carte

Au vu des surfaces visées et de leur nombre, il n'est pas envisageable d'utiliser la même méthode de numérisation d'environnement que dans notre laboratoire. Pour rappel, nous utilisions un LiDAR 3D Leica. C'est une méthode relativement lente qui nécessite des interventions humaines, à minima pour utiliser l'instrument. Ce qui est possible dans une usine au vu des enjeux de sécurité et de maintenance ne l'est plus du tout pour numériser tous les garages personnels et les parkings par un constructeur automobile. Il est donc évident qu'il faut utiliser les LiDARs embarqués pour la création de la carte. Notons tout-de-même qu'avec des trajectoires 6DDL, l'utilisation du LiDAR embarqué, à la fois pour la localisation et l'acquisition de la carte, ne garantit pas une couverture de l'environnement similaire à un LiDAR de géomètre qui couvre pratiquement la sphère complète. En effet, si la trajectoire n'est pas la même entre les phases de numérisation et de localisation, les variations d'attitude du mobile peuvent orienter les impacts LiDAR vers des parties de l'environnement non présentes dans la carte.

Nous avons donc développé deux méthodes de numérisation :

1. Avec l'objectif de créer des cartes de référence sur des surfaces importantes : nous avons utilisé notre coffre de toit de référence (section 2.2.3.1) pour numériser l'environnement avec le LiDAR Velodyne HDL-64 ; la centrale inertielle nous fournissant alors une estimation de la pose des scans.
2. Avec l'objectif de créer des cartes de moins grande envergure : en utilisant le LiDAR embarqué et l'odométrie du véhicule pour fournir une estimation de la pose des scans. Cette méthode a été évaluée sur notre véhicule équipé de notre coffre de toit, ainsi que sur le robot Viking au rez-de-chaussée du site industriel de Lacq.

Avec la vitesse des véhicules routiers, de nouveaux problèmes apparaissent. Le déplacement du véhicule porteur n'est plus négligeable devant la vitesse de rotation du LiDAR, et la géométrie des scans est affectée. Les déformations qui étaient acceptables dans le cadre de la robotique avec des vitesses inférieures à 1m/s le sont beaucoup moins autour de 20m/s. Une caractérisation de ces déformations, ainsi que la méthode de correction mise au point sont détaillées dans l'annexe B. Ces travaux ont donné lieu à une publication Merriaux et collab. [2016].

La figure 3.21 résume les trois méthodes dont nous disposons pour obtenir un nuage de points afin de créer notre champ de vraisemblance. Dans le cas où la pose de la centrale inertielle fusionnée au GPS-RTK est disponible, la position du scan LiDAR est parfaitement connue. Une fois le scan corrigé des déformations dues au mouvement du véhicule, nous effectuons un ajustement par un algorithme de type *ICP* Nüchter [2007] en utilisant l'outil 3DTK[8]. Ce dernier ajustement permet de corriger d'éventuels problèmes de calibration entre l'IMU et le LiDAR ou encore un défaut de synchronisation d'acquisition. Ce dernier point est très critique aux vitesses envisagées. La carte obtenue, même

8. http://slam6d.sourceforge.net/

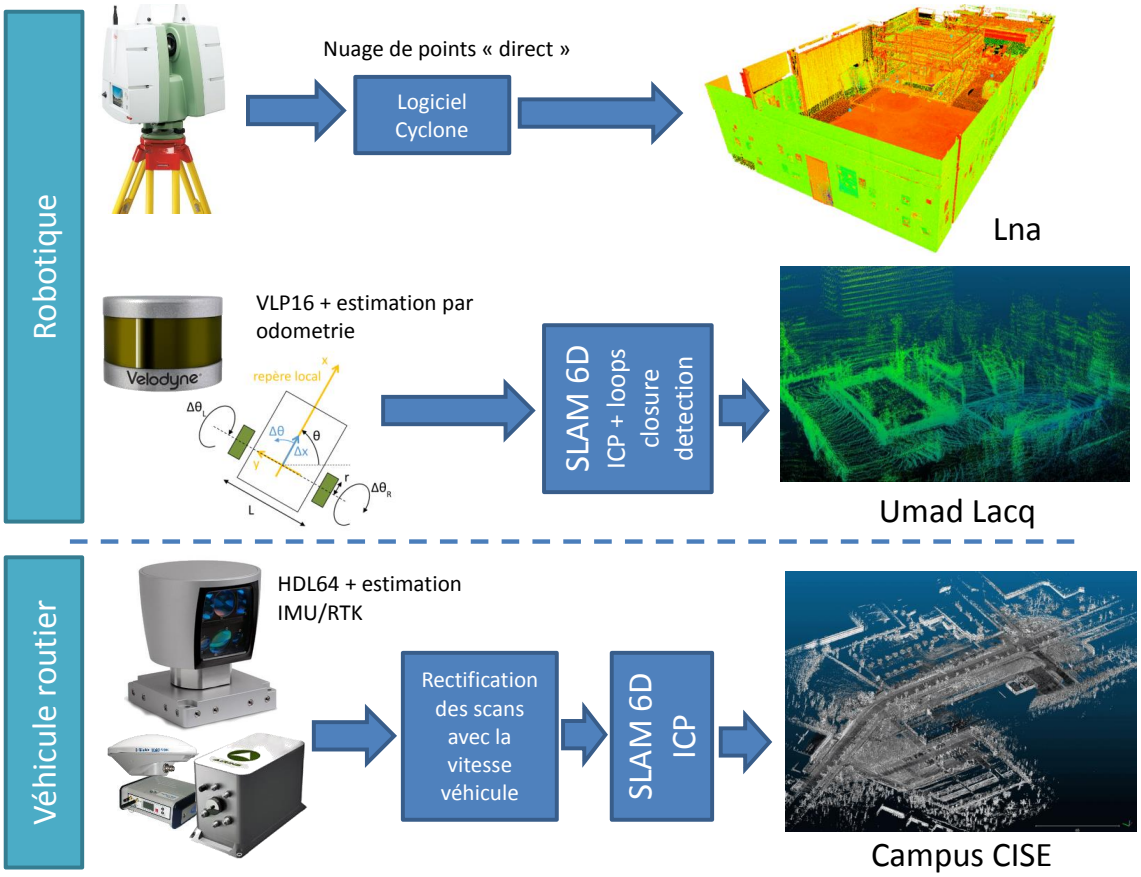


FIGURE 3.21 – Différentes méthodes de création de la carte : *en haut,* avec un LiDAR 3D de géomètre adapté aux petits volumes à grande valeur ajoutée, pour les cas où la précison et la couverture est essentielle. *En bas* avec notre coffre de référence équipé d'un moyen de localisation de référence. *Au milieu* juste avec les équipements embarqués du robot.

sur de grandes surfaces, est exempte de toute dérive grâce au système de positionnement de référence de notre coffre de toit.

Dans le cas où nous estimons la pose par intégration de l'odométrie, une dérive apparaît rapidement. Dans ce cas, nous réalisons une phase supplémentaire de recherche de fermeture du boucle Sprickerhof et collab. [2009] toujours avec l'outil 3DTK. Celle-ci permet de détecter des parties de scan communes éloignées de plusieurs poses et d'effectuer un ajustement global. Une vue du site de Lacq ainsi reconstruite est présentée sur la figure 3.21.

Une fois le nuage de points obtenu, nous utilisons le même outil que celui de la section 3.4.1.3 avec une résolution inférieure pour générer le champ de vraisemblance. Un exemple de stockage de la carte de l'ensemble de notre campus est fourni dans la section 3.4.1.1.

3.4.6.2 Premiers résultats de localisation

Nous avons constitué trois jeux de données avec le Renault Espace et le LiDAR Velodyne HDL64 du laboratoire :

1. Tour du campus dans le sens anti-horaire : constitution de la carte à partir de la méthode présentée à la section précédente 3.4.6.1.
2. Entrée en marche avant et sortie en marche arrière du laboratoire de motorisation hybride situé à l'extrémité de notre campus : validation du fonctionnement en intérieur et extérieur.
3. Tour du campus dans le sens horaire et entrée/sortie dans le bâtiment : validation de la localisation sur un scénario complet.

Pour éviter tout biais, la carte (figure B.8) est créée à partir d'un jeu de données différent de ceux qui servent à l'évaluation de la performance de localisation.

Les informations de déplacements nécessaires à l'étape de *motion update* sont issues des capteurs ABS et ESP du véhicule. Elles sont directement enregistrées à partir du bus *CAN*. A partir des capteurs ABS nous déterminons la vitesse linéaire, et grâce au gyromètre de l'ESP nous obtenons le taux de rotation. Il suffit alors d'intégrer ces deux informations par le pas de temps de l'échantillonnage, 25Hz dans notre cas.

Le nuage de points issu du LiDAR est transposé au milieu de l'essieu arrière. Ce point correspond au centre d'application de notre odométrie, soit l'équivalent du repère $\mathcal{R}$ du robot Viking (figure 3.12). Puis ils sont redressés par la méthode proposée dans l'annexe B.

Les jeux de données 2 et 3 sont calculés dans un diagramme RTMaps avec le composant de localisation du challenge Argos fonctionnant sur le Viking. Seul les paramètres de bruits additifs sont modifiés pour correspondre à l'échelle du véhicule et à la résolution de la carte. Nous obtenons les trajectoires des figures 3.22 et 3.23. Elles sont présentées superposées avec la carte ayant servi à générer le champ de vraisemblance.

A l'heure de mettre en forme ce manuscrit, nous rencontrons quelques difficultés de calibration entre le référentiel de la carte et celui de la vérité terrain obtenue grâce à notre coffre de référence. Pour l'instant, il n'est donc pas possible d'objectiver la performance de la localisation.

Sans vérité terrain, il est difficile d'avancer une quelconque conclusion sur la performance atteinte. Nous pouvons juste noter que l'algorithme ne diverge pas, que les trajectoires semblent correctes, et que le passage intérieur extérieur ne semble pas poser de problème.

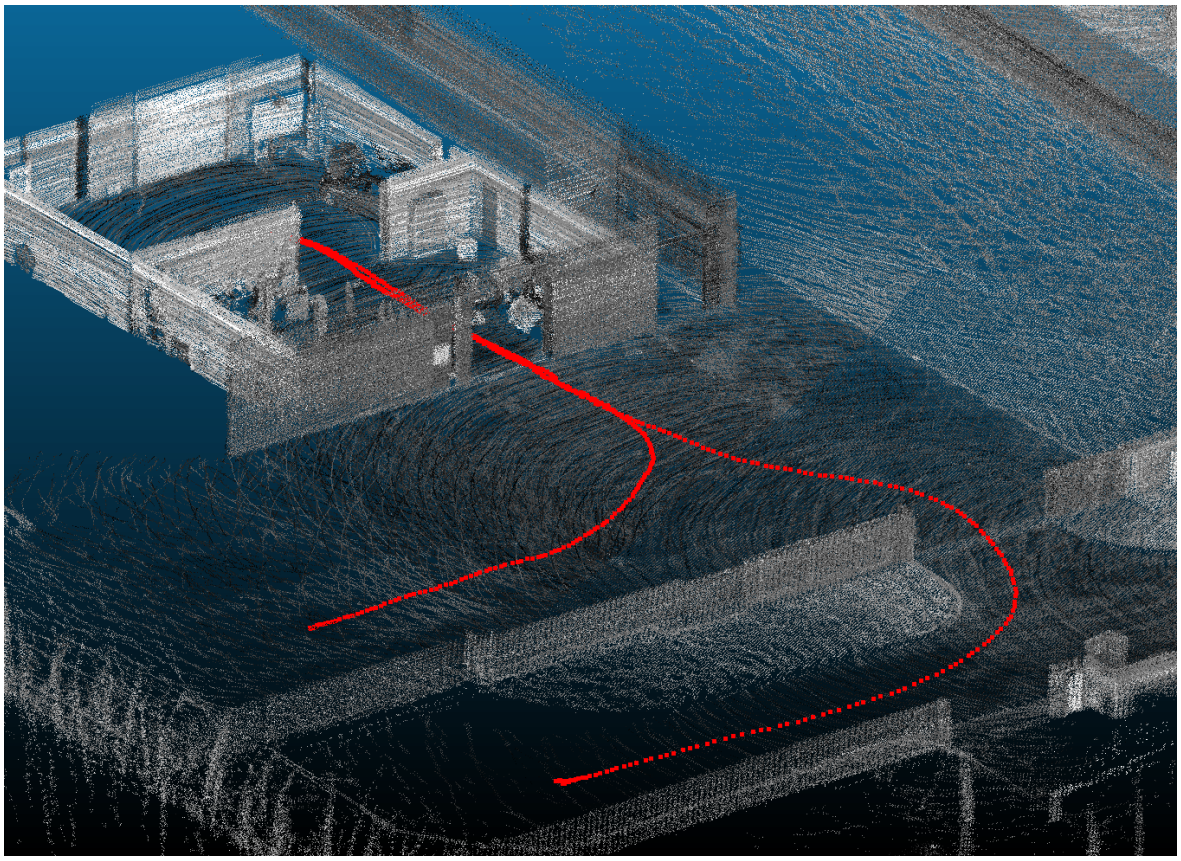


FIGURE 3.22 – Localisation véhicule - entrée/sortie du laboratoire - longueur 101.7m

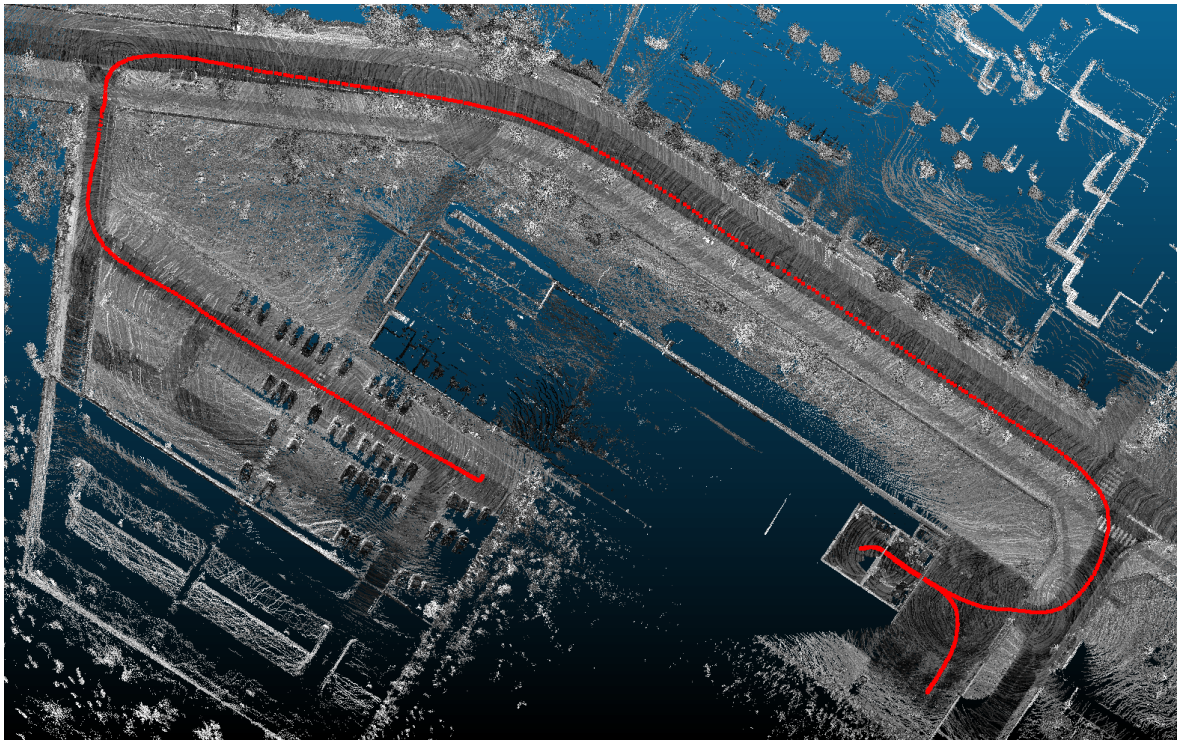


FIGURE 3.23 – Localisation véhicule - tour du campus et entrée/sortie dans le laboratoire - longueur 461.3m

Conclusion

Dans ce chapitre, nous proposons une localisation 6 degrés de liberté robuste et rapide dont l'approche est basée sur l'extension du concept de champ de vraisemblance à la 3D. Le champ de vraisemblance est stocké dans une structure de type octree optimisé. La performance a été démontrée en simulation, avec des expérimentations en laboratoire puis sur un véritable site industriel pétrochimique.

Bien que nous proposons une solution pour un stockage efficace de notre carte, l'extension de l'utilisation à des sites industriels d'ampleur toujours plus importante (figure 3.1a) posera problème. Une solution simple pourrait consister à implémenter des « tuiles d'octree hydride ». Ces différentes tuiles seraient chargées à la volée en fonction des déplacements du robot un peu à la manière des travaux de Alsayed et collab. [2015] en 2D.

Comme vu lors de la présentation du modèle de localisation à deux couches (section 1.4), la méthode proposée ici appartient typiquement à la deuxième couche. Elle est suffisamment précise pour permettre le contrôle du mobile mais il lui est impossible de répondre facilement au problème de localisation globale. En effet, nous initialisons les particules pour une liste de poses connues que nous pouvons spécifier dynamiquement. La répartition initiale des particules est de $\sigma_{init} = \begin{bmatrix} 0.1 & 0.1 & 0.0 & 0° & 0° & 5° \end{bmatrix}^{\mathrm{T}}$. Si nous avions réparti des particules avec une densité équivalente sur l'ensemble de la carte du site de la compétition, nous pourrions considérer qu'il nous faudrait environ 1.8 millions de particules. Ce nombre quelque peu démesuré augmente encore dans le cas de la scène routière proposée à la section 3.4.6. Il paraît donc complètement illusoire de penser actualiser une telle quantité d'hypothèses. De plus, les multiples similarités de l'environnement ne garantissent pas que nous puissions rapidement diminuer le nombre d'hypothèses avec des méthodes de variation dynamique de particules comme le *KDL-resample* abordée dans la section 1.3.4.3.

C'est pour ces raisons que nous avons imaginé le modèle de localisation à deux couches. Il nous faut une localisation plus globale, avec une perception de plus « haut niveau », la couche 1. Elle met en œuvre une fonction de vraisemblance encore moins coûteuse en temps de calcul, de manière à pouvoir suivre d'importantes quantités d'hypothèses. Une fois cette couche 1 convergée, la position obtenue permet d'initialiser une couche 2 semblable aux travaux présentés dans ce chapitre. C'est dans ce cadre que nous allons maintenant aborder un chapitre présentant une localisation topologique de couche 1 adaptée à l'environnement routier.

Remerciements

Cette étude est une partie du Challenge Argos, organisé par Total en partenariat avec l'Agence Nationale de la Recherche (ANR). Le nuage de points de la figure 3.5a est la propriété de Total.

Références

Alsayed, Z., G. Bresson, F. Nashashibi et A. Verroust-Blondet. 2015, «Pml-slam : a solution for localization in large-scale urban environments», dans *PPNIV-IROS 2015*. 117

El Hamzaoui, O. 2012, *Localisation et cartographie simultanées pour un robot mobile équipé d'un laser à balayage : CoreSLAM*, thèse de doctorat, Ecole Nationale Supérieure des Mines de Paris. 92

Fallon, M. F., H. Johannsson et J. J. Leonard. 2012, «Efficient scene simulation for robust monte carlo localization using an rgb-d camera», dans *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, IEEE, p. 1663–1670. 89

Jagbrant, G., J. Underwood, J. Nieto et S. Sukkarieh. 2013, «Lidar based localisation in almond orchards», . 85

Levinson, J. et S. Thrun. 2010, «Robust vehicle localization in urban environments using probabilistic maps», dans *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, IEEE, p. 4372–4378. 89

Loevsky, I. et I. Shimshoni. 2010, «Reliable and efficient landmark-based localization for mobile robots», *Robotics and Autonomous Systems*, vol. 58, n° 5, p. 520–528. 85

Merriaux, P., Y. Dupuis, R. Boutteau, P. Vasseur et X. Savatier. 2015, «Localisation robuste en milieu industriel complexe», Gretsi, Lyon, France. 84

Merriaux, P., Y. Dupuis, R. Boutteau, P. Vasseur et X. Savatier. 2016, «Correction de nuages de points lidar embarqué sur véhicule pour la reconstruction d'environnement 3d vaste», RFIA, Clermont Ferrand, France. 113

Nüchter, A. 2007, «Parallelization of scan matching for robotic 3d mapping.», dans *EMCR*. 113

Nüchter, A., K. Lingemann, J. Hertzberg et H. Surmann. 2006, «6d slam–mapping outdoor environments», dans *IEEE International Workshop on Safety, Security, and Rescue Robotics (SSRR). IEEE Press*. 86

Nuchter, A., H. Surmann, K. Lingemann, J. Hertzberg et S. Thrun. 2004, «6d slam with an application in autonomous mine mapping», dans *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, vol. 2, ISSN 1050-4729, p. 1998–2003 Vol.2, doi: 10.1109/ROBOT.2004.1308117. 84, 86

Olivares-Mendez, M. A., I. Mellado, P. Campoy, I. Mondragon et C. Martinez. 2011, «A visual agv-urban car using fuzzy control», dans *Automation, Robotics and Applications (ICARA), 2011 5th International Conference on*, IEEE, p. 145–150. 84

Pfaff, P., R. Kümmerle, D. Joho, C. Stachniss, R. Triebel et W. Burgard. 2007, «Navigation in combined outdoor and indoor environments using multi-level surface maps», *WS on Safe Navigation in Open and Dynamic Environments, IROS*, vol. 7. 89

Reinke, C. et P. Beinschob. 2013, «Strategies for contour-based self-localization in large-scale modern warehouses», dans *Intelligent Computer Communication and Processing (ICCP), 2013 IEEE International Conference on*, IEEE, p. 223–227. 85

Ronzoni, D., R. Olmi, C. Secchi et C. Fantuzzi. 2011, «Agv global localization using indistinguishable artificial landmarks», dans *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, IEEE, p. 287–292. 85

Sabattini, L., V. Digani, C. Secchi, G. Cotena, D. Ronzoni, M. Foppoli et F. Oleari. 2013, «Technological roadmap to boost the introduction of agvs in industrial applications», dans *Intelligent Computer Communication and Processing (ICCP), 2013 IEEE International Conference on*, IEEE, p. 203–208. 85

Sprickerhof, J., A. Nüchter, K. Lingemann et J. Hertzberg. 2009, «An explicit loop closing technique for 6d slam.», dans *ECMR*, p. 229–234. 115

Stoyanov, T., J. Saarinen, H. Andreasson et A. J. Lilienthal. 2013, «Normal distributions transform occupancy map fusion : Simultaneous mapping and tracking in large scale dynamic environments», dans *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, IEEE, p. 4702–4708. 85, 86

Taghaboni, F. et J. Tanchoco. 1988, «A lisp-based controller for free-ranging automated guided vehicle systems», *The International Journal Of Production Research*, vol. 26, nº 2, p. 173–188. 84

Thrun, S. 1993, «Exploration and model building in mobile robot domains», dans *Neural Networks, 1993., IEEE International Conference on*, p. 175–180 vol.1, doi: 10.1109/ICNN.1993.298552. 84

Thrun, S., W. Burgard, D. Fox et collab.. 2005, *Probabilistic robotics*, vol. 1, MIT press Cambridge. 88, 91

Tipaldi, G. D., D. Meyer-Delius et W. Burgard. 2013, «Lifelong localization in changing environments», *International Journal of Robotics Research*, vol. 32, nº 14, p. 1662–1678. 85

Underwood, J. P., G. Jagbrant, J. I. Nieto et S. Sukkarieh. 2015, «Lidar-based tree recognition and platform localization in orchards», *Journal of Field Robotics*, vol. 32, nº 8, p. 1056–1074. 85

Wurm, K. M., A. Hornung, M. Bennewitz, C. Stachniss et W. Burgard. 2010, «Octomap : A probabilistic, flexible, and compact 3d map representation for robotic systems», dans *Proc. of the ICRA 2010 workshop on best practice in 3D perception and modeling for mobile manipulation*, vol. 2. 92

Zhang, J. et S. Singh. 2014, «Loam : Lidar odometry and mapping in real-time», dans *Robotics : Science and Systems Conference (RSS)*. 86

Zhuang, Y., N. Jiang, H. Hu et F. Yan. 2013, «3-d-laser-based scene measurement and place recognition for mobile robots in dynamic indoor environments», *Instrumentation and Measurement, IEEE Transactions on*, vol. 62, nº 2, p. 438–450. 86

Chapitre 4

Localisation topologique sur un réseau routier

Sommaire

4.1 Introduction et contexte

Disposer d'une localisation de véhicule précise et robuste est un point clé dans de nombreuses applications du domaine des *transports intelligents*. Comme abordé précédemment, les méthodes de localisation locale, ayant une précision suffisante pour assurer le contrôle du véhicule, ne peuvent pas être initialisées sans une connaissance *a priori* de la position (section 1.4 et fin du chapitre 3). Pour rendre possible cette initialisation plus globale, nous avons besoin d'une méthode que nous avons dénommée « localisation de couche 1 ». Nous avons également montré dans le début du chapitre 1 que la solution évidente du GPS ne pouvait pas répondre à toutes les applications, notamment lorsque la qualité de service est essentielle, comme pour les applications de véhicules autonomes.

Dans ce chapitre, nous proposons une méthode de localisation qui utilise seulement les informations de capteurs déjà présents dans tous les véhicules actuels et une carte numérique du réseau routier. L'objectif est d'obtenir une fonction de vraisemblance très peu coûteuse à calculer permettant d'évaluer de nombreuses hypothèses.

D'autres travaux proposent une fonctionnalité équivalente, mais généralement en utilisant des capteurs supplémentaires comme une caméra, et parfois en ajoutant des informations dans la carte. Beaucoup de chercheurs se sont également attelés à l'amélioration de la précision des systèmes GPS, de la même façon, par l'ajout de capteurs. Les méthodes en mise œuvre pour résoudre ce dernier problème étant relativement proches de notre proposition, nous commencerons par les aborder dans la prochaine section.

4.2 État de l'art

Du fait des imprécisions sur le positionnement du véhicule et des simplifications géométriques contenues dans les cartes numériques, à partir d'une information GPS, il n'est pas possible de déterminer directement la position sur la carte. L'approche la plus populaire est basée sur l'utilisation de la position fournie par un GPS pour déterminer sur quel segment de la carte nous nous trouvons. Elle est connue sous le nom de *map-matching* (Quddus et collab. [2007]). La robustesse de la correction de la position du véhicule peut être améliorée en prenant en compte la trajectoire à court terme. De nombreuses méthodes de *map-matching* existent. Dans sa thèse, Velaga [2010] propose un état de l'art complet.

Une autre approche consiste à ajouter des informations dans la carte. Des représentations de carte plus complexes permettent d'atteindre la précision pour un positionnement sur une voie. Dans la représentation de notre carte que nous proposerons section 4.3.1, il y a une seule trace par route physique. Par exemple, Betaille et Toledo-Moreo [2010] définissent une trace pour chaque voie. Les deux approches dépendent des données GPS. Elles ne peuvent pas surmonter des situations où les signaux GPS sont fortement perturbés ou indisponibles pendant une longue période.

L'ajout d'autres capteurs permet d'améliorer la robustesse du positionnement du véhicule. Deux approches principales ont été investiguées par la communauté :

— La première consiste à fusionner l'information GPS avec des capteurs sur d'autres modalités.

  El Najjar et Bonnifait [2005] proposent de fusionner les mesures d'un GPS différentiel (D-GPS) avec la vitesse des roues arrières fournie par les odomètres du système ABS. Les deux informations sont fusionnées par un filtre de Kalman pour obtenir la position du véhicule ainsi qu'un taux de confiance de cette estimation. La théorie

de la croyance est alors appliquée pour déterminer le segment de la carte le plus probable et la position du véhicule.

Parra Alonso et collab. [2012] utilisent une méthode d'odométrie visuelle, un GPS et une carte numérique. Le système *map-matche* le GPS quand l'information de précision *HDOP* (Horizontal Dilution Of Position) est inférieure à 10m. Si le *HDOP* devient supérieur à ce seuil, le système commute automatiquement sur l'odométrie visuelle. Il utilise cette estimation de cap et de vitesse pour actualiser la position du véhicule.

D'autres travaux, comme ceux de Szottka [2013] et Tao et Bonnifait [2015], ajoutent les informations de marquage au sol dans la carte. Puis, ils utilisent une caméra associée à un algorithme de détection de marquage et différentes méthodes de filtrage pour localiser le véhicule sur sa voie.

— Dans la deuxième approche, l'information du GPS est complètement ignorée dans le processus de localisation. Elle est juste utilisée pour l'estimation initiale. La détermination de la position est réalisée avec d'autres capteurs. Dans ce type de travaux, la trajectoire du véhicule est *map-matchée* avec la carte. Les différentes longueurs parcourues peuvent être utilisées pour ce processus de corrélation.

Floros et collab. [2013] utilisent une méthode d'odométrie visuelle pour estimer la position du véhicule sur un graphe décrivant une carte. La position initiale est supposée inconnue dans un rayon d'1km autour de la position réelle. Une trajectoire initiale est obtenue par l'accumulation de l'odométrie visuelle sur plus de 400m. Ils utilisent ensuite la méthode du *Chamfer matching* courante en traitement d'image (Borgefors [1988] ou Liu et collab. [2010]) pour retrouver la position dans le graphe. Après cette phase d'initialisation, les hypothèses de position sont évaluées sur une trajectoire contenant les 250 dernières images de l'odométrie visuelle.

Gustafsson et collab. [2012] proposent une localisation en 2D, qui est ensuite contrainte par *map-matching* sur le réseau. Différentes sources de capteurs y sont étudiées.

Brubaker et collab. [2013] proposent une localisation basée également sur une odométrie visuelle monoculaire ou en stéréo vision comparée à une carte contenant le réseau routier de 47km. L'estimation de position initiale s'effectue sur une surface de 2km^2. Les estimations de déplacements obtenues par l'odométrie visuelle sont utilisées pour réduire l'incertitude de position jusqu'à convergence.

Après initialisation, les deux dernières méthodes présentées sont indépendantes de l'information GPS. Mais elles sont basées sur la vision, ce qui pose problème la nuit et par mauvaises conditions météorologiques. Nous proposons de nous affranchir de ces limitations en utilisant seulement des capteurs proprioceptifs déjà intégrés dans les véhicules actuels, tout en ayant une grande tolérance à la précision de la position initiale. De plus, nous nous affranchissons de l'étape de *map-matching* du fait que les hypothèses de position soient contraintes au réseau routier.

4.3 Méthodologie

Nous proposons d'utiliser les cartes *Open Street Map* (OSM) [1] comme sources de données pour la description de notre réseau. Comme leur nom l'indique, ces cartes sont libres de droit, actualisées par une communauté à la manière de l'encyclopédie communautaire *Wikipédia*. Avant d'aborder la méthodologie de localisation en elle-même, nous

1. https://www.openstreetmap.org

commencerons par expliquer comment nous définissons le véhicule sur le réseau et la constitution du graphe représentant l'environnement.

4.3.1 Vecteur d'état du véhicule et données OpenStreetMap (OSM)

Les bases de données OSM regroupent un ensemble d'informations qui ne nous sont pas forcément utiles : cadastre, délimitations de zones communes, départements, forêts, rivières, chemins de randonnée... Par conséquent, nous ne conservons dans la carte que les routes accessibles aux véhicules.

Comme annoncé plus haut, nous stockons les données OSM dans un graphe. Un exemple de sa structure est présenté sur la figure 4.1a. La géométrie du réseau y est définie par deux types d'éléments : les nœuds et les arcs. Les segments de route sont représentés par les arcs unidirectionnels, les nœuds représentent quant à eux les connexions entre les segments. En connectant plusieurs arcs sur le même nœud, nous pouvons représenter une intersection, ce qui permet de décrire la structure du réseau routier.

Au sens strict du terme, une carte OSM n'est pas seulement une carte topologique mais une carte hybride (section 1.1). En effet, elle contient à la fois une description des lieux et le moyen de les relier, mais également des coordonnées métriques qui définissent leur position. Notre méthodologie se base sur l'aspect topologique du graphe. Nous conservons les coordonnées métriques dans le seul but d'avoir un référentiel commun lors de l'évaluation par rapport à la vérité terrain.

Un nœud est défini de la manière suivante :

$$\mathrm{N}_i = \begin{bmatrix} \phi \\ \theta \end{bmatrix} \tag{4.1}$$

Où :

ϕ : latitude
θ : longitude
i : identifiant du nœud

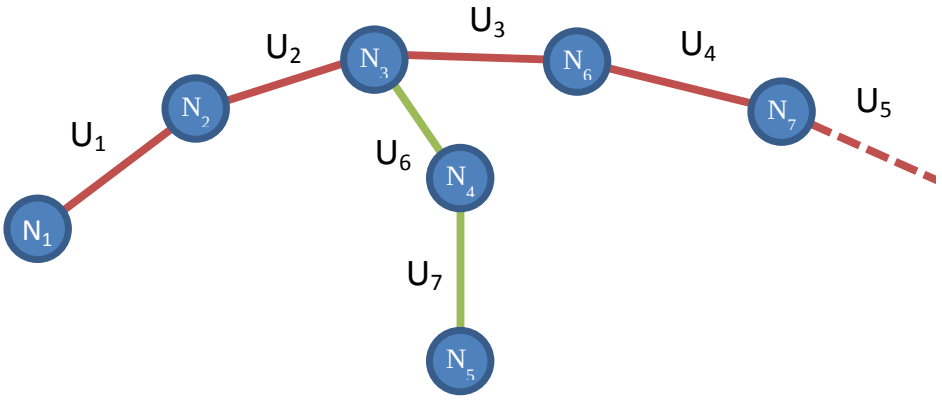
La latitude et la longitude ne sont pas exprimées en angle, mais dans un système de coordonnées cartésiennes à travers une projection Transverse Universelle de Mercator (*UTM* : Universal Transverse Mercator). Elle nous permet une comparaison plus aisée avec la vérité terrain. C'est une projection cylindrique où l'axe du cylindre croise perpendiculairement l'axe des pôles (figure 4.2a). Pour minimiser les déformations, le globe est divisé en 120 zones, la Normandie se trouve dans la zone 31U (figure 4.2b).

Les déformations engendrées par cette opération de projection ne sont pas vraiment gênantes dans notre cas. Tout d'abord, car la surface couverte par notre réseau reste relativement faible - ensuite car la projection est appliquée aux coordonnées de chaque nœud topologique ainsi qu'à la vérité terrain, qui subissent donc les mêmes déformations.

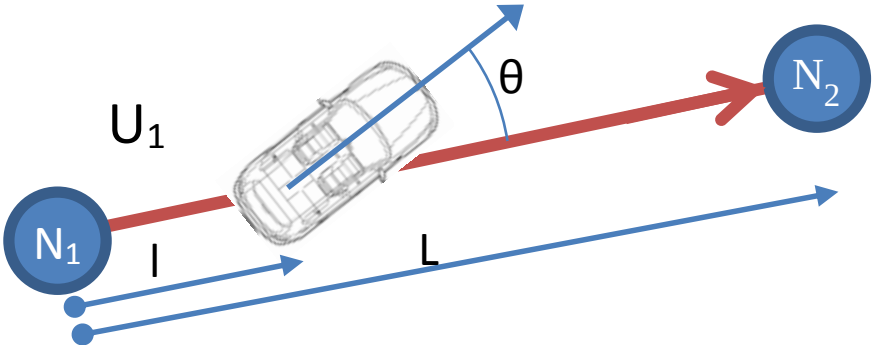
Un arc U_k connectant deux nœuds d'identifiants n_s et n_e est défini de la manière suivante :

$$\mathrm{U}_k = \begin{bmatrix} n_s \\ n_e \\ \mathrm{L} \\ \Theta \end{bmatrix} \tag{4.2}$$

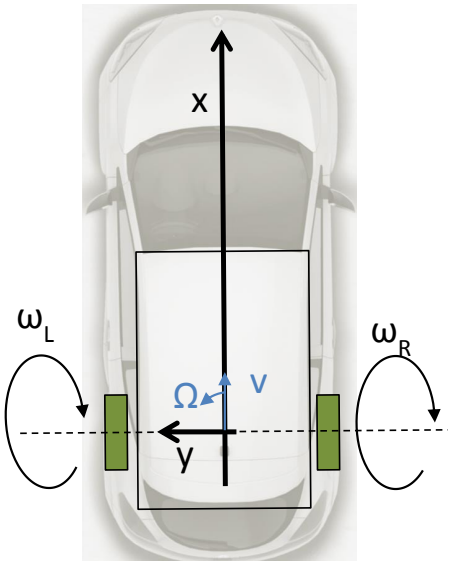
Où :



(a) Représentation du graphe OSM

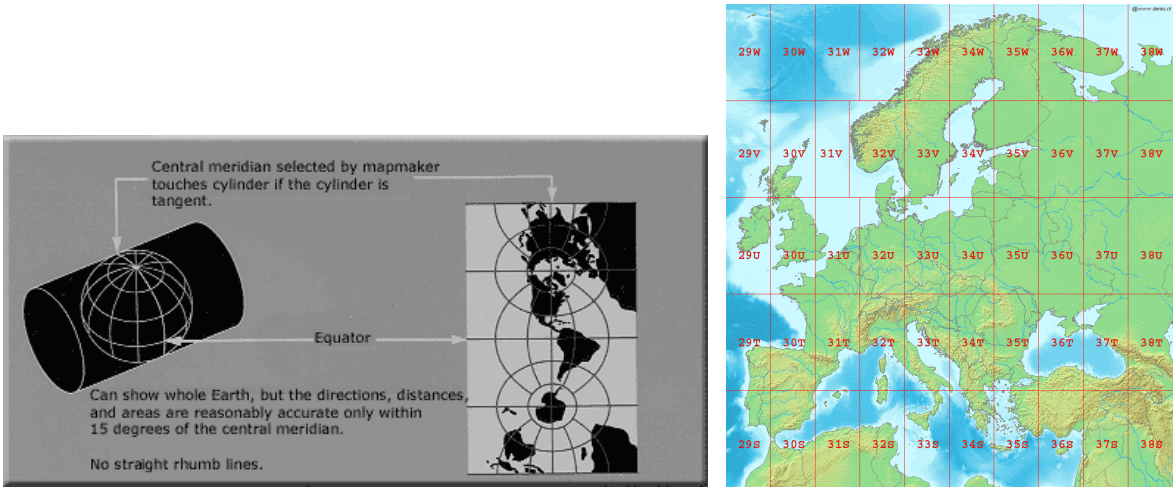


(b) Vecteur d'état du véhicule sur un segment du graphe



(c) Modèle odométrique différentiel - Differential drive kinematics (DDK)

FIGURE 4.1 – Map and vehicle data representation
OSM : OpenStreetMap



(a) Principe de la projection cylindrique (b) Zones UTM en Europe

FIGURE 4.2 – Projection Transverse Universelle de Mercator (*UTM* : projection cylindrique où l'axe du cylindre croise perpendiculairement l'axe des pôles (©*Wikipedia*)

n_s : identifiant du nœud de départ
n_e : identifiant du nœud de fin
L : longueur du segment
Θ : orientation du segment
k : identifiant de l'arc

L'orientation de l'arc Θ est définie entre $[0, \pi]$. Les identifiants des nœuds de début et de fin permettent en lien avec Θ de définir l'orientation du segment. La figure 4.1b montre les variables utilisées pour le vecteur d'état du véhicule x_t. Ce vecteur doit inclure les informations nécessaires afin de déterminer de manière unique la position du véhicule sur le réseau. Il est donc composé de la manière suivante :

$$x_t = \begin{bmatrix} e_t \\ l_t \\ d_t \\ \theta_t \end{bmatrix} \tag{4.3}$$

Où :

e_t : identifiant du segment où se trouve le véhicule
l_t : position longitudinale sur le segment orienté
d_t : direction de déplacement le long du segment orienté
θ_t : cap relatif du véhicule

d_t peut prendre deux valeurs : 1 ou -1. Si le véhicule se déplace depuis le nœud de départ vers le nœud de fin, d_t est positif. Dans le sens opposé, d_t est négatif. θ_t correspond à l'orientation relative du véhicule par rapport à ce segment.

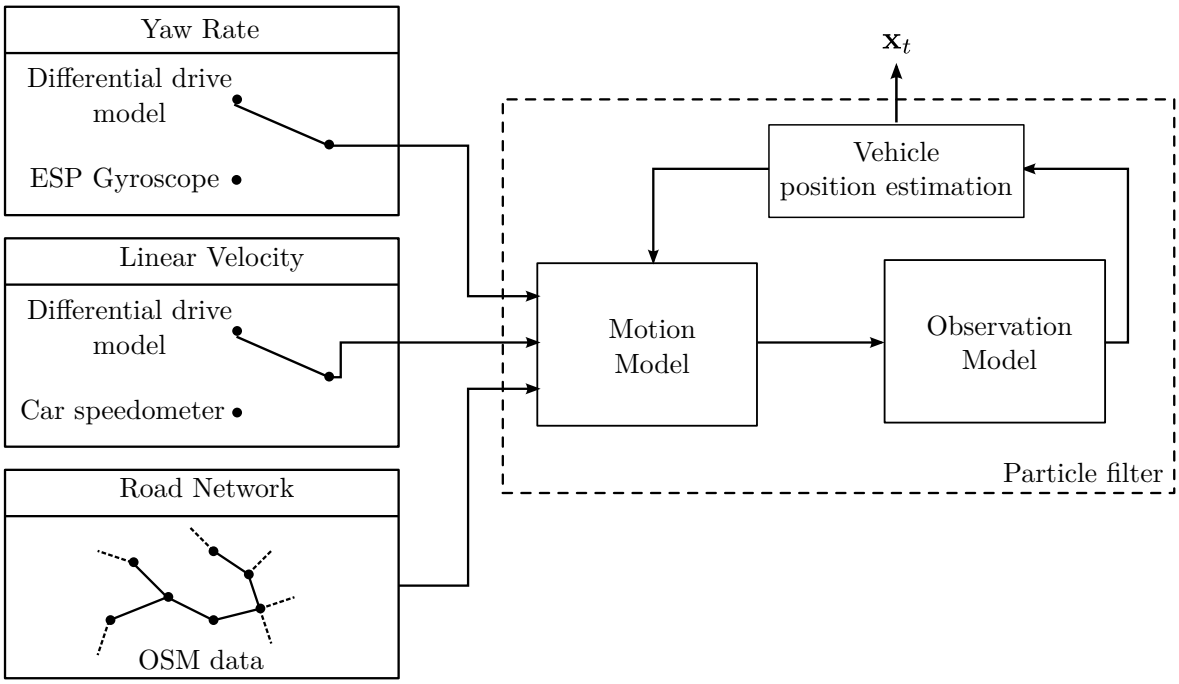


FIGURE 4.3 – Système de positionnement du véhicule

4.3.2 Filtre particulaire

Pour estimer la distribution *a posteriori* $p(x_t|z_{1:t})$, nous utilisons un filtre particulaire contenant N hypothèses. Nous ne ré-expliquerons pas ici dans le détail le principe de ce filtre bayésien, nous l'avons déjà abordé aux sections 1.3.4.3 et 3.3. Il permettra de déterminer une estimation du vecteur d'état représentant la position du véhicule sur le réseau x_t à l'instant t.

Dans cette application, le choix du filtre particulaire s'imposait de par le nombre d'hypothèses à suivre. Le filtre de Kalman étant uni-modal dans ses versions de base, il ne peut pas convenir. Le filtre d'histogramme est quant à lui limité à un nombre fini d'hypothèses.

Le filtre particulaire utilisé reprend en tous points le schéma du synoptique exposé dans le chapitre précédent sur la figure 3.4. Les modalités étant différentes, par conséquent les deux étapes majeures seront différentes : le *motion update* est actualisé par les capteurs ABS et ESP embarqués dans le véhicule, alors que le *measurement update* se contente de comparer l'orientation de la particule avec le segment courant pour déterminer son poids w_t^i.

Le jeu de particule est défini ainsi :

$$\mathscr{P}_t = \{< x_t^i, w_t^i >: i \in \{1, \ldots, \mathrm{N}\}\} \tag{4.4}$$

Les poids sont normalisés entre eux tel que $\sum_{i=1}^{\mathrm{N}} w_t^i = 1$. Les deux étapes nécessaires pour la mise à jour séquentielle de la distribution sont détaillées ci-dessous (figure 4.3) :

Le *motion update* utilise le modèle de déplacement pour faire circuler les hypothèses sur le graphe en fonction des mouvements du véhicule. Ce modèle correspond à la probabilité $p(x_t|x_{t-1})$. La propagation des particules respecte les connectivités du graphe représentant notre réseau routier. Par conséquent, les déplacements des particules s'effectuent de manière topologique. Le vecteur d'état décrivant la position du véhicule sur un segment défini dans l'équation 4.3 est utilisé comme vecteur d'état des particules x_t^i.

Sur le véhicule, nous disposons de plusieurs sources de données pour alimenter notre modèle de déplacement (figure 4.3). Les capteurs ABS nous fournissent la vitesse des quatre roues. Ils permettent de calculer un modèle odométrique différentiel (4.1c) équivalent à celui abordé dans la section 1.2.2.1 (appelé par la suite *DDK* pour, *Differential Drive Kinematic*). Nous pouvons estimer de cette manière la vitesse linéaire v et le taux de lacet du vitesse Ω.

$$v = \frac{\omega_L + \omega_R}{2} \tag{4.5}$$

$$\Omega = \frac{\omega_L - \omega_R}{T} \tag{4.6}$$

Où :

T	: voie de l'essieu arrière (m)
ω_L	: vitesse de la roue gauche (rad/s)
ω_R	: vitesse de la roue droite (rad/s)

Le taux de lacet Ω est également mesuré par le gyroscope embarqué dans le système ESP. Ce capteur dérive plutôt en fonction du temps alors que l'odométrie dérive en fonction de la distance parcourue. Nous disposons aussi d'une autre source de mesure de la vitesse linéaire, le speedomètre du véhicule, ce qui revient à faire une moyenne de la vitesse des quatre roues.

Nous avons également utilisé les mesures inertielles de notre système de référence comme sources de données. Ceci a permis de déterminer l'influence de la précision des capteurs, comme nous le verrons dans les résultats (section 4.4). Ces données seront dénommées *INS* dans la suite de ce document. L'impact de ces différentes sources de mesure du taux de lacet et de la vitesse sur la précision du positionnement sera présenté dans la section résultat 4.4.

Les mesures des capteurs entraînent le déplacement des particules. Par conséquent, les particules vont passer d'un segment à un autre. La transition entre segment apparaît lorsque la longueur du déplacement de la particule est supérieure à la longueur du segment actuel (figure 4.5). Quand plusieurs segments sont connectés à celui-ci, chacun est évalué selon une probabilité en fonction de l'écart entre l'orientation du segment successeur Θ_j et l'orientation absolue de la particule en question θ'^{i}_{t} (figure 4.5a). La probabilité de tirage s_j pour chaque segment candidat est donnée par la formule de densité de probabilité de Von Mises :

$$p(x|\mu,\kappa) = \frac{\exp\left(\kappa\cos(x-\mu)\right)}{2\pi I_0(\kappa)} \tag{4.7}$$

Où :

μ	: direction moyenne
κ	: paramètre de concentration
I_0	: version modifiée de la fonction de Bessel du premier type d'ordre 0

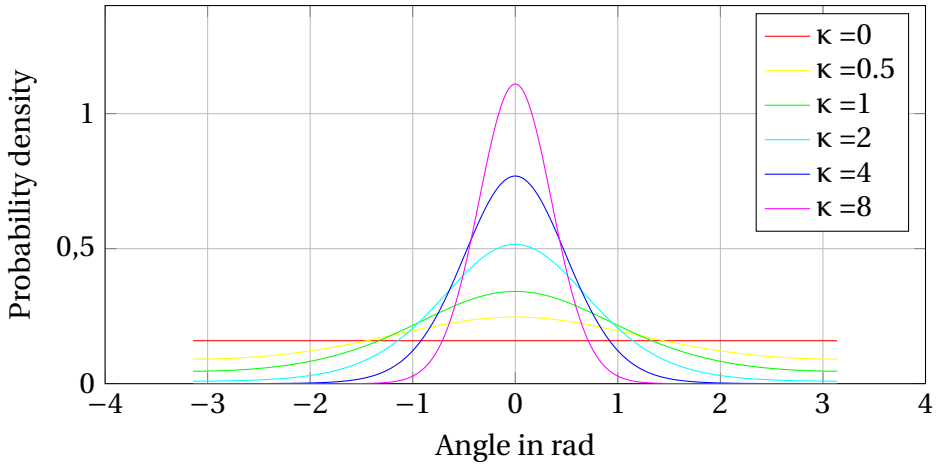


FIGURE 4.4 – Fonction de densité de probabilité de Von Mises

La distribution de Von Mises est l'équivalent d'une gaussienne circulaire. Elle est d'ailleurs également couramment appelée « distribution normale circulaire ». La figure 4.4 présente la fonction de densité de probabilité (PDF) de Von Mises sur $[-\pi,\pi]$ pour différentes valeurs du paramètre de concentration κ. κ est l'équivalent de l'écart-type σ de la gaussienne. Le dénominateur est un facteur de normalisation dépendant de κ. Nous définissons alors la probabilité qu'une particule i prenne le segment successeur j comme ceci :

$$s_j = \frac{\exp\left(\kappa \cos(\theta'^{i}_{t} - \Theta_j)\right)}{2\pi I_0(\kappa)} \tag{4.8}$$

Les poids sur les différents segments sont normalisés comme $\sum s_j = 1$. Les poids résultant définissent la fonction de masse (PMF : Probability Mass Function) depuis laquelle chaque segment candidat sera tiré aléatoirement (figure 4.5b).

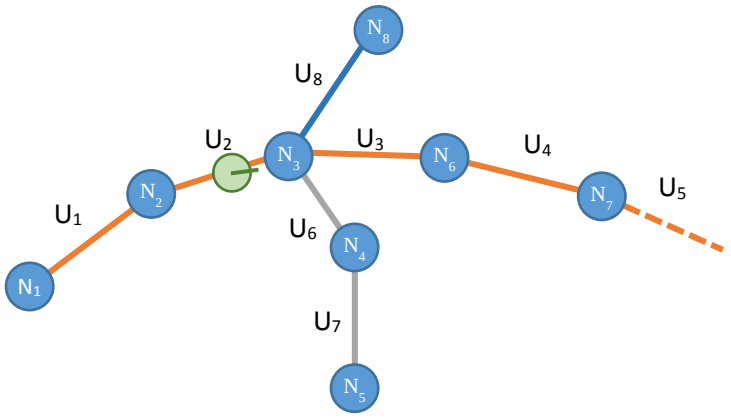
Les mesures de déplacement, c'est-à-dire la vitesse linéaire et le taux de rotation, sont considérées comme contaminées par un bruit proportionnel défini respectivement par $\mathcal{N}(0,\sigma_v)$ et $\mathcal{N}(0,\sigma_\gamma)$.

Le *measurement update* utilise le modèle d'observation pour définir la probabilité $p(z_t|x^i_t)$. Le principe de la fonction de vraisemblance est extrêmement simple. Nous comparons l'orientation de la particule avec celle du segment sur lequel elle se trouve. Le poids de la particule w^i_t est dérivé de l'équation 4.8 en utilisant le cap relatif de la particule par rapport au segment courant :

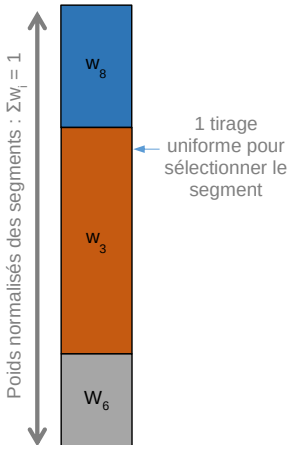
$$w^i_t = \frac{\exp\left(\kappa \cos(\theta^i_t)\right)}{2\pi I_0(\kappa)} \tag{4.9}$$

Comme dans les travaux du chapitre 3, nous utilisons un ré-échantillonnage systématique. Les particules étant déplacées par le *motion update*, au fil des intersections, celles qui étaient mal positionnées auront un angle relatif important avec le segment sur lequel elles se trouvent et donc un faible poids. Le ré-échantillonnage supprimera ces particules peu vraisemblables et dupliquera les meilleures.

Le résultat de la localisation doit être une estimation de la position du véhicule $\mathbf{x}_t$ et non la croyance représentée par l'ensemble des particules $\mathscr{P}_t$. Comme abordé en section 1.3.4.3, la moyenne pondérée des poids (barycentre) est couramment utilisée en fil-



(a) Si une particule (en vert) arrive à l'extrémité de son segment (U_2), nous évaluons son orientation par rapport à tous les candidats possibles (U_6, U_3 et U_8). Les 3 orientations relatives entre la particule et chaque segment candidat sont calculées, puis nous obtenons un poids pour chacun de ces segments grâce à la densité de probabilité de Von Mises.



(b) La somme des poids obtenus est normalisée à 1. Un tirage uniforme entre [0,1[sélectionne le candidat correspondant.

FIGURE 4.5 – Sélection des candidats successeurs dans le graphe : tirage dépendant de l'orientation relative entre la particule et les segments successeurs. Dans l'exemple ci-dessous, la particule (en vert) va se propager sur le segment U_3.

trage particulaire pour fournir une seule hypothèse de l'approximation de la distribution *a posteriori*. Dans ces travaux, le filtre particulaire fonctionne sur le graphe d'une manière topologique, et non directement dans un repère cartésien. Il n'est donc pas possible de calculer cette moyenne directement. L'estimation de la position $\mathbf{x}_t$ est calculée en coordonnées UTM. Le vecteur d'état d'une particule dans l'espace UTM est défini de la manière suivante :

$$x'^i_t = \mathrm{N}_{n_s} + l_t \begin{bmatrix} \cos(\Theta) \\ \sin(\Theta) \end{bmatrix} = \begin{bmatrix} \phi^i_t \\ \theta^i_t \end{bmatrix} \tag{4.10}$$

n_s et Θ sont respectivement le nœud de début et l'orientation de l'arc avec l'identifiant e_t où la particule x^i_t se trouve actuellement. ϕ^i_t et θ^i_t sont respectivement la latitude et la longitude de la particule x^i_t. L'estimation de la position du véhicule $\hat{\mathbf{x}}_t$ est approximativement à la position moyenne pondérée des poids des particules de $\mathcal{X}_t$ en coordonnées UTM : n_s et Θ sont respectivement le nœud de début et l'orientation de l'arc avec l'identifiant e_t où la particule x^i_t se trouve actuellement. ϕ^i_t et θ^i_t sont respectivement la latitude et la longitude de la particule x^i_t. L'estimation de la position du véhicule $\hat{\mathbf{x}}_t$ est donnée par la position moyenne pondérée des poids des particules dans $\mathcal{X}_t$ en coordonnées UTM :

$$\hat{\mathbf{x}}_t = \frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} w^i_t . x'^i_t \tag{4.11}$$

L'équation 4.11 n'est qu'une approximation de l'estimation idéale de $\mathcal{X}_t$, le résultat ne se trouve pas forcément sur un arc du graphe.

4.4 Résultats expérimentaux

Nos travaux se sont déroulés en deux phases : la première phase est sommairement résumée en introduction de cette section. La deuxième étant une évolution plus complète, ses résultats sont expliqués en détail dans la continuité de ce chapitre.

Dans nos premiers travaux sur le sujet Merriaux et collab. [2014] et Merriaux et collab. [2015], nous nous sommes attelés aux deux problèmes : les localisations globale et locale. Cette dernière est communément appelée *tracking*. Nous avons réalisé une localisation sur 17km dans un réseau routier de 100km (figure 4.6). La performance a été évaluée avec un GPS à faible coût dans les zones urbaines. La localisation globale sur l'ensemble du réseau a été démontrée, en équi-répartissant des hypothèses sur le réseau. Un aperçu des étapes de convergence est présenté sur la figure 4.7.

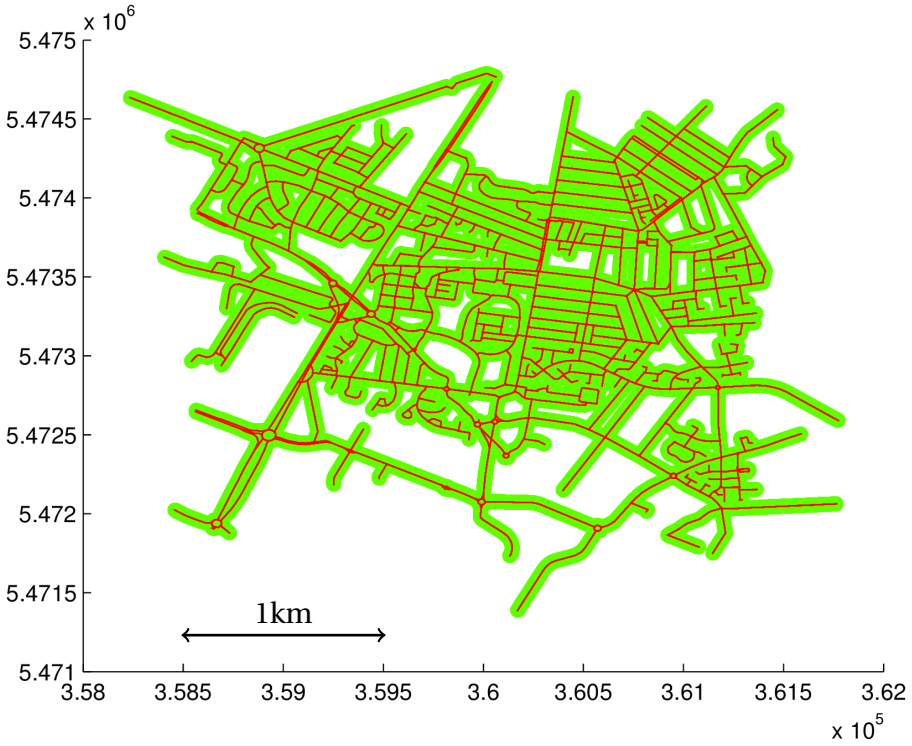
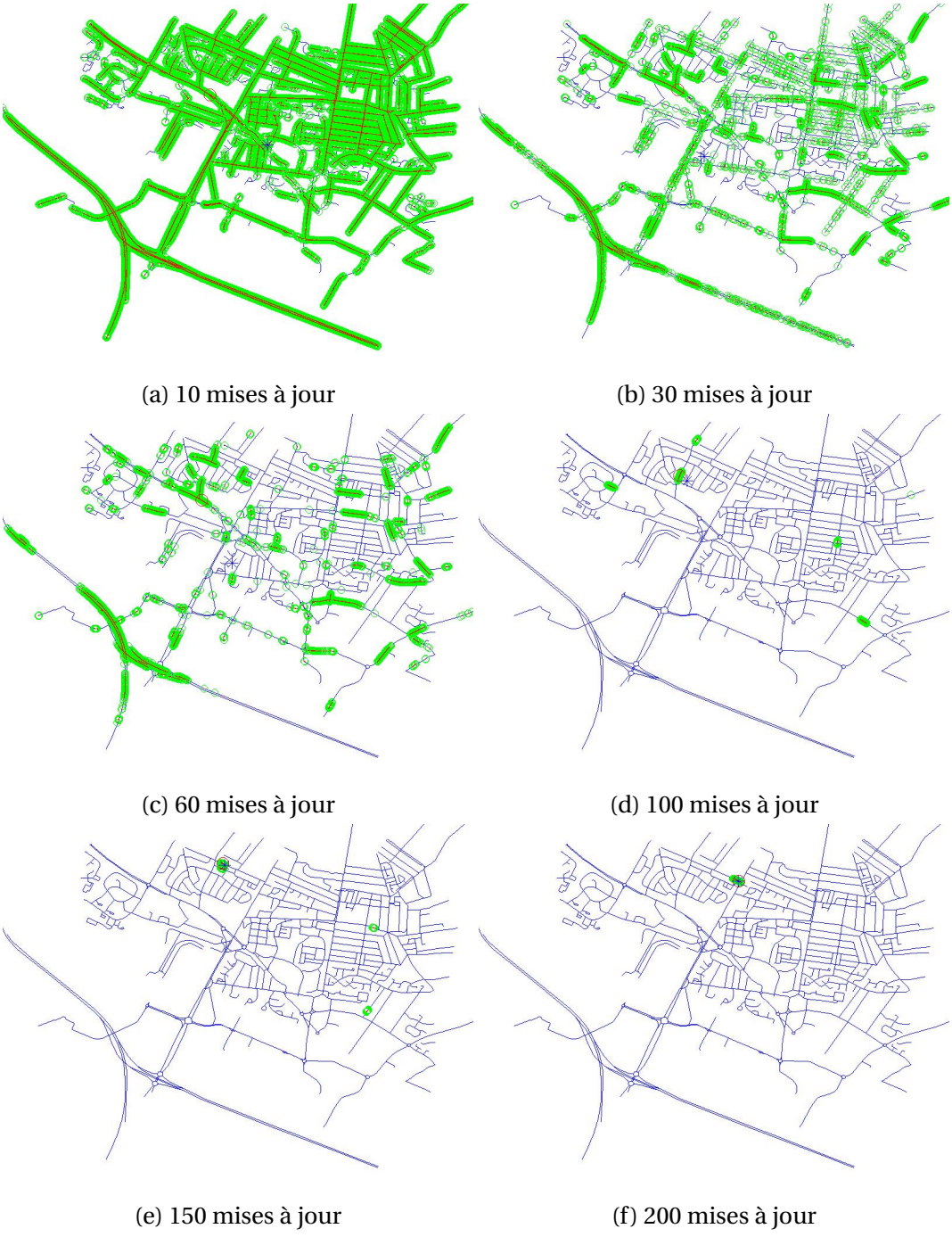


FIGURE 4.6 – Initialisation des particules sur un réseau de 100km, aucune connaissance de la position *a priori* (*rouge : réseau OSM - Vert : particules*).

Puis, nous avons voulu étendre notre méthode à un réseau de dimension plus importante comprenant 380km de routes (Dupuis et collab. [2015]). La localisation globale sans aucune connaissance *a priori* de la position n'étant plus réaliste du point de vue du temps de calcul et du taux de réussite de convergence, nous avons proposé, comme nous le rencontrons couramment dans la littérature, une initialisation avec une incertitude d'un rayon de 500m. Dans un cas concret d'utilisation, il est très rare que nous n'ayons aucune idée de la position sur une telle surface.

De manière générale, nous présentons une étude plus complète et mieux argumentée en évaluant l'impact des paramètres du modèle de déplacement et d'observation sur la performance de la localisation.

Les premiers résultats obtenus nous ont obligés à améliorer notre vérité terrain. Nous avons donc mis en œuvre notre système de positionnement centimétrique de notre coffre de toit de référence (section 2.2.3.1, figures 2.14 et 2.16a).



(a) 10 mises à jour

(b) 30 mises à jour

(c) 60 mises à jour

(d) 100 mises à jour

(e) 150 mises à jour

(f) 200 mises à jour

FIGURE 4.7 – Etapes de la convergence des particules de la localisation globale du véhicule - *Vert : Particules - Bleu : réseau OSM - Etoiles bleues : position estimée - Cercles rouges : position réelle*

4.4.1 Jeux de données

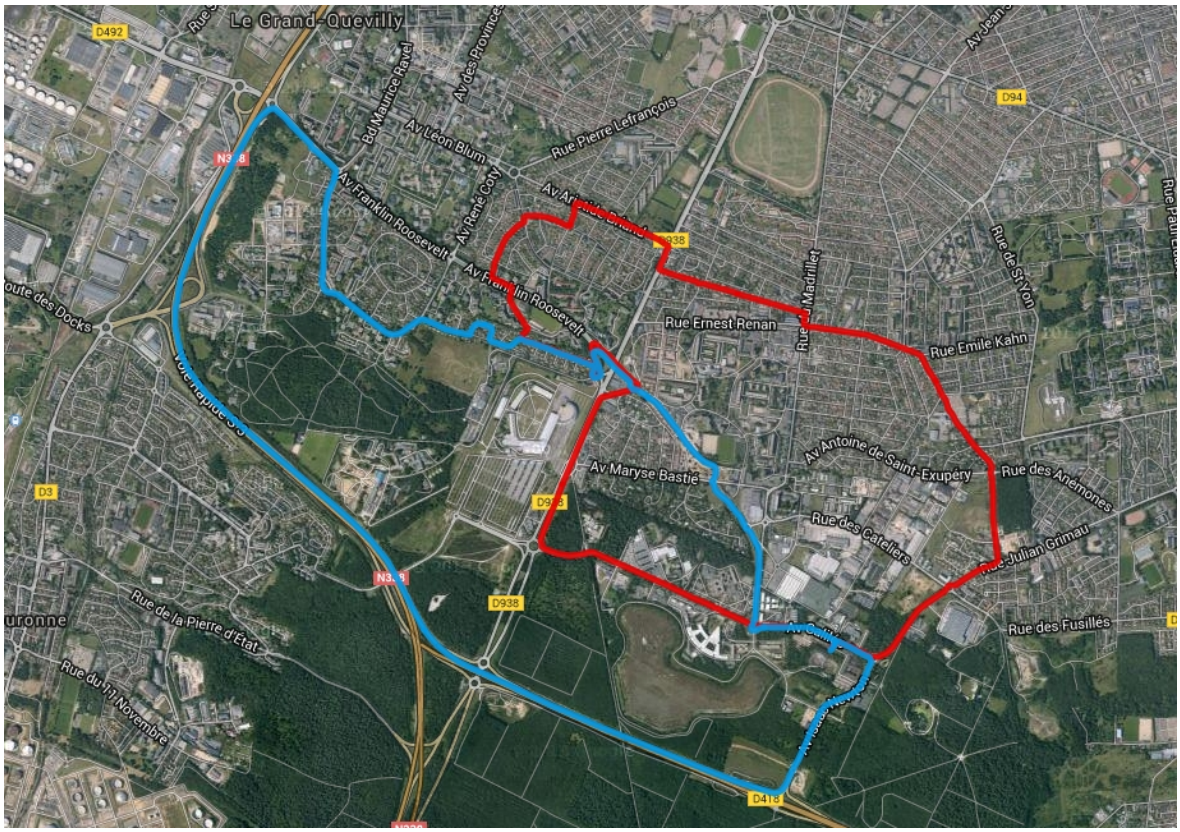


FIGURE 4.8 – Les deux circuits utilisés pour les expérimentations : *circuit 1 en rouge (8km) - Circuit 2 en bleu (11km)*

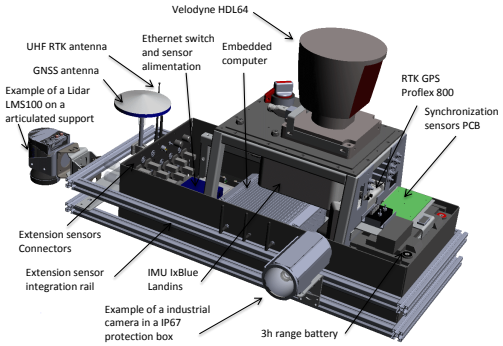
Nous avons évalué notre approche sur deux jeux de données enregistrés autour de Rouen en Normandie (figure 4.8). Le circuit 1 se déroule dans des zones résidentielles de la banlieue Sud, sa longueur est de 8.3km. Pour le circuit 2, nous avons cherché à être plus exhaustif sur les types de routes parcourues : autoroutes, zone péri-urbaines et zones résidentielles. Il mesure 10.3km.

L'ensemble des deux circuits représente une distance de 19km. Les données OSM représentent l'ensemble de la région rouennaise (figure 4.10) et contiennent 380km de réseau routier destiné aux véhicules motorisés.

Les sources de données utilisées par notre algorithme sont : vitesse véhicule,vitesse des quatre roues et taux de lacet. Elles sont directement enregistrées à partir du bus CAN du véhicule à la cadence de 25Hz. Le biais de démarrage du gyromètre fournissant le taux de rotation est soustrait à l'initialisation en supposant le véhicule statique. La voie de l'essieu arrière du véhicule a directement été obtenue dans la documentation du constructeur du véhicule. Toutes les sources d'odométrie sont ensuite intégrées pour fournir une donnée par seconde.

La vérité terrain est obtenue avec notre coffre de toit de référence, grâce à son système de positionnement basé sur la fusion d'un GPS-RTK, d'une centrale inertielle IXSEA LANDINS et d'un odométre positionné sur la roue arrière du véhicule (figure 4.9). Ce moyen d'essai a été présenté dans la section 2.2.3.1. Les données constituant la vérité terrain sont acquises à 200Hz, puis resynchronisées toutes les secondes sur les données véhicule.

(a) Coffre de toit de référence installé sur notre Renault Espace



(b) Configuration physique des capteurs

FIGURE 4.9 – Configuration du système utilisé comme vérité terrain lors de nos expérimentations

4.4.2 Contexte de l'évaluation

Comme vu lors de l'explication de la méthode (section 4.3), nous avons trois paramètres à déterminer pour optimiser la performance de l'algorithme :

— Les bruits du modèle de déplacement σ_v et σ_y

— Le paramètre de concentration κ de la distribution de Von de Mises : il détermine la probabilité de tirage du choix de segment successeur dans le modèle de déplacement, mais également dans le modèle d'observation pour déterminer le poids des particules

Nous avons évalué l'apport de capteurs de meilleures précisions « INS » pour l'étape de *motion update* de notre algorithme. Comme annoncé précédemment, nous avons utilisé les mesures de vitesse linéaire et du taux de lacet issues des capteurs inertiels du système de positionnement de référence utilisé comme vérité terrain. Nous disposons donc de trois sources d'information proprioceptives.

Pour résumer, nous disposons en terme de sources de données de mesures :

— Pour la vitesse linéaire v : DDK, speedomètre et INS

— Pour le taux de lacet Ω : DDK, gyroscope ESP et INS

L'ensemble de ces paramètres et de ces sources de données est susceptible d'influencer la performance de la localisation. Comme il n'est pas possible de faire varier tous les paramètres à la fois, nous procéderons par étapes :

— Influence du paramètre de concentration κ sur le modèle d'observation. L'étude sera menée pour les six combinaisons de sources de données possibles, et deux niveaux de bruits choisis de manière pragmatique.

— Influence des bruits de mesure σ_v et σ_y, en fonction des deux meilleures valeurs de κ et des trois meilleurs couples de sources de données identifiées précédemment.

Le circuit 2 est utilisé pour déterminer les paramètres et les sources de données en suivant le protocole présenté ci-dessus car il contient plusieurs types de routes. Une fois les meilleures configurations de paramètres identifiées, elles sont utilisées pour déterminer les performances de la localisation sur les deux circuits. Pour chaque circuit, nous évaluerons la précision finale avec la meilleure combinaison de source de données, deux valeurs de κ et trois niveaux de bruit.

La position initiale est supposée inconnue dans un rayon de 500m autour de la position réelle du véhicule (figure 4.10). Par conséquent, les particules sont d'abord réparties sur le réseau routier à l'intérieur de cette zone. Les particules sont réparties tous les 3m sur le graphe, une dans chaque direction du segment à chaque fois. Une fois cette initialisation réalisée, l'information du GPS est totalement ignorée.

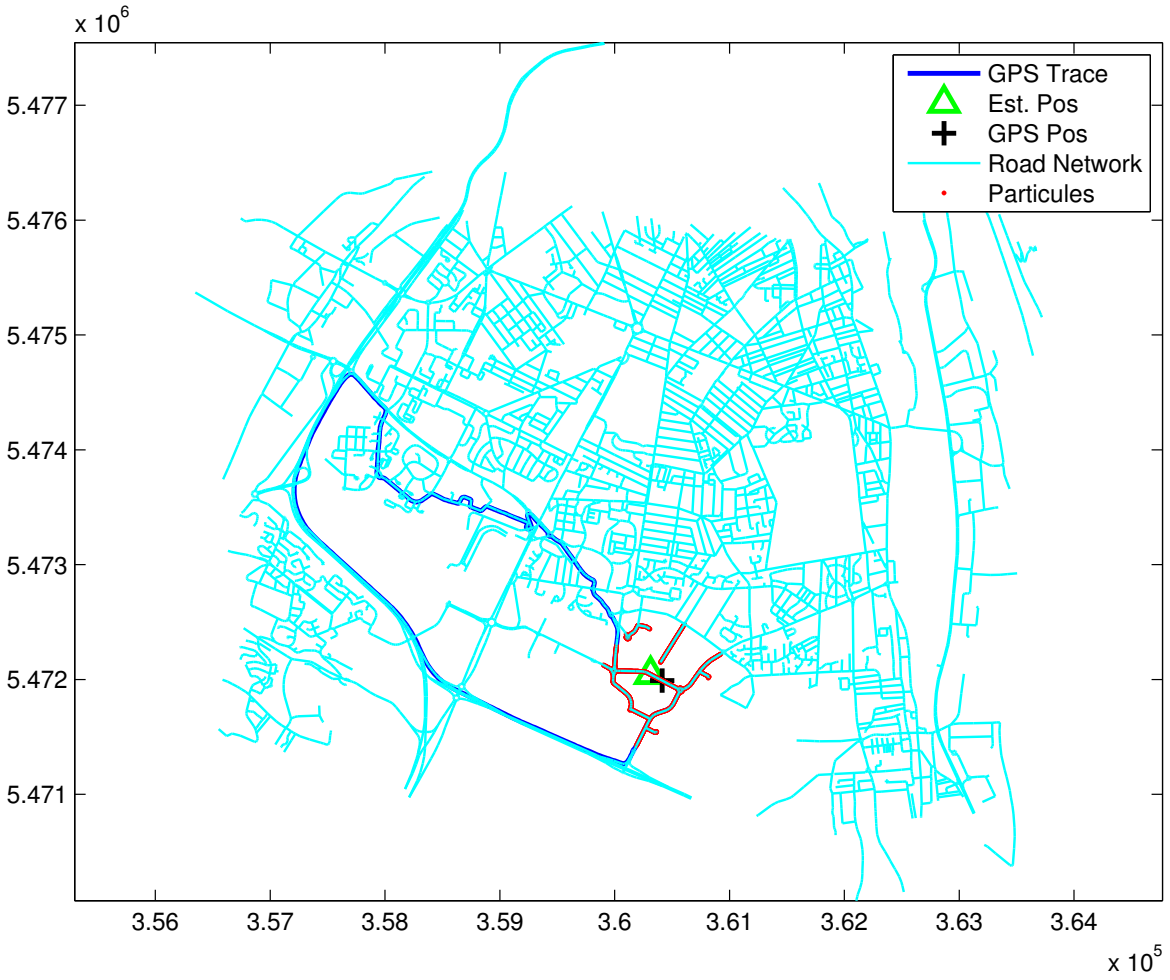


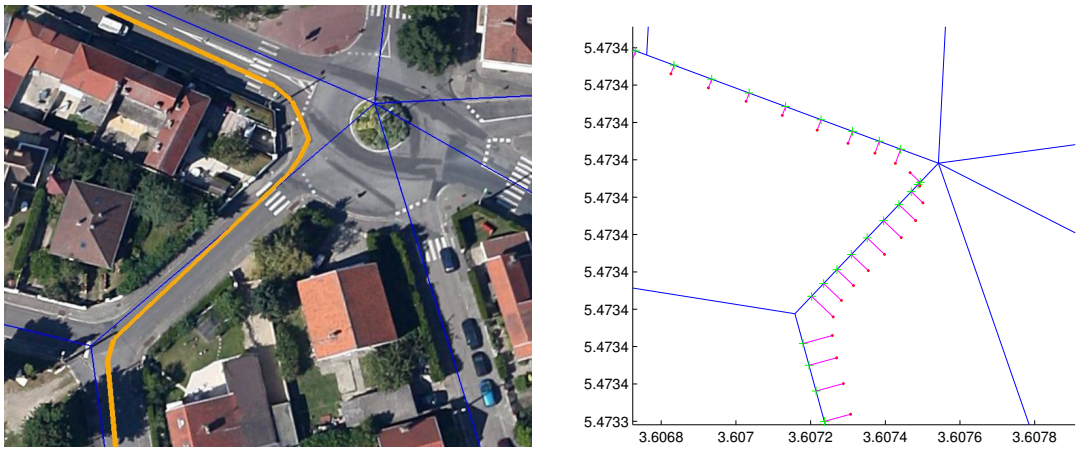
FIGURE 4.10 – Exemple d'initialisation des particules sur le réseau de 380km, avec une incertitude initiale d'1km de diamètre - Circuit 2

Les identifiants des différents segments parcourus par la véhicule sont annotés manuellement dans les données OSM. La vérité terrain de la position $\mathbf{x}_t$ est *map-matchée* sur les segments ainsi annotés avec l'algorithme « *point-to-curve* » suggéré dans les travaux de Velaga [2010] (figure 4.11b).

Au vu de la précision centimétrique de notre système de référence utilisé pour la vérité terrain, les erreurs de positionnement sont dues en très grande partie aux approximations de représentation de la carte par rapport à l'infrastructure réelle. Par exemple, les ronds-points sont souvent représentés par 6 ou 8 segments, voire d'une manière extrêmement simpliste comme le montre la figure 4.11a !

La position du véhicule estimée par notre algorithme $\hat{\mathbf{x}}_t$ est *map-matchée* sur la totalité du réseau représenté dans le graphe, soit 380km avec l'algorithme *point-to-curve*. L'erreur de position est alors définie comme :

$$e_t = ||\mathbf{x}_t - \hat{\mathbf{x}}_t|| \tag{4.12}$$



(a) Données OSM en surimpression sur une vue aérienne

(b) Trajectoire GPS-RTK *map-matchée* sur le graphe

FIGURE 4.11 – Illustration de la précision de la carte OSM
Lignes orange épais : trajectoire de la vérité terrain - Lignes bleues : carte OSM - Points rouges : points de vérité terrain - Croix vertes : Vérité terrain map-matchée

$\hat{\mathbf{x}}_t$ est calculé à chaque nouvelle mesure de déplacement du véhicule.

Afin de se prémunir de tout biais dû à l'aspect aléatoire du filtre particulaire, chaque expérimentation est répétée 10 fois. La médiane $\tilde{e}$ et la moyenne de l'erreur de position $\bar{e}$ sont utilisées comme caractéristiques de la performance du positionnement. Ces deux métriques sont calculées sur le trajet complet du véhicule. En effet, une valeur de médiane faible est un critère de qualité de convergence du positionnement. La moyenne $\bar{e}$ est altérée par les grandes erreurs comme il en existe avant convergence, lorsque seulement quelques étapes de *measurement update* ont été exécutées. Par conséquent, un faible $\bar{e}$ démontre un positionnement correct, rapide et stable.

4.4.3 Modèle d'observation - Influence du paramètre de concentration

Dans la table 4.1, nous cherchons l'influence de κ (paramètre de concentration de la loi de Von Mises) sur l'erreur de positionnement. Afin d'éviter tout biais dans nos conclusions, les essais sont réalisés pour deux niveaux de bruit de déplacement qui sont choisis par rapport à notre expérience des mesures odométriques.

Les grandes valeurs de κ tendent à réduire l'erreur de positionnement dans les deux tableaux 4.1a et 4.1b. Nous considérons donc que $\kappa = \{8, 16\}$ sont les meilleures valeurs car elles montrent une réduction significative de l'erreur médiane. Cela est vrai quand le filtre particulaire converge sur la bonne position. En fait, nous pouvons remarquer que le modèle odométrique différentiel DDK est un mauvais estimateur du taux de rotation et que les erreurs de positionnement sont significativement plus importantes lors de son utilisation.

Après investigation, nous avons identifié que la vitesse d'une des roues était sous-estimée alors que l'autre était sur-estimée. La moyenne n'étant pas affectée, la vitesse linéaire est plutôt correcte, alors que le taux de lacet est fortement perturbé. Par conséquent pour la suite de l'étude, nous avons éliminé, le taux de lacet calculé à partir des capteurs ABS.

Odometry Sources		κ					
Velocity	Yaw Rate	0.5	1	2	4	8	16
Speedometer	DDK	1605.6 (1605.8)	1177.4 (1254.1)	1212.0 (1283.8)	973.8(1193.4)	946.6 (1201.8)	1512.6(1711.1)
DDK	DDK	1623.4 (1602.5)	1191.0 (1262.1)	1294.9 (1400.6)	1077.9 (1280.6)	1020.8 (1223.6)	1037.0 1267.3)
Speedometer	Gyroscope	11.9 (56.4)	10.7 (39.7)	13.0 (27.8)	9.6 (18.6)	7.6 (13.9)	7.5 (16.6)
INS	INS	18.0 (62.2)	11.4 (39.6)	9.9 (27.1)	7.7 (20.1)	6.0 (17.6)	5.11 (45.7)
DDK	Gyroscope	14.6 (59.2)	10.8 (40.7)	9.3(27.7)	7.2 (19.8)	5.8 (15.4)	5.06 (28.9)

(a) $\sigma_v = 10\%$ and $\sigma_y = 10\%$

Odometry Sources		κ					
Velocity	Yaw Rate	0.5	1	2	4	8	16
DDK	DDK	1630.9 (1616.0)	1519.4 (1423.0)	1270.0 (1322.5)	1247.9 (1404.2)	1432.0 (1486.0)	1434.8 (1440.9)
Speedometer	DDK	1630.8 (1672.5)	1517.2 (1437.9)	1513.3 (1426.8)	1458.9 (1551.8)	1428.3 (1468.6)	1433.2 (1453.0)
Speedometer	Gyroscope	14.9 (56.1)	12.7 (38.6)	8.9 (23.0)	7.2 (14.8)	5.38 (12.37)	4.8 (30.4)
INS	INS	18.8 (61.2)	12.6 (37.6)	7.9 (21.6)	5.9 (14.1)	4.5 (12.0)	4.2 (46.9)
DDK	Gyroscope	16.4 (59.1)	11.3 (38.0)	7.3 (21.4)	5.6 (13.7)	4.3 (11.2)	4.1(22.0)

(b) $\sigma_v = 20\%$ and $\sigma_y = 20\%$

TABLEAU 4.1 – Influence du paramètre κ sur le modèle d'observation (*Erreur de localisation médiane $\tilde{e}$ en m - Erreur de localisation moyenne $\bar{e}$ en m entre parenthèses - DDK : differential drive model*)

4.4.4 Modèle de déplacement - Impact du bruit de mesure

Dans la table 4.2, nous expérimentons différents couples de bruit pour les meilleures valeurs de $\kappa = \{8, 16\}$ ainsi que 3 sources de données déterminées précédemment. Les deux principales combinaisons retenues utilisent le gyroscope de l'ESP pour estimer le taux de rotation.

Nous faisons varier les bruits de vitesse linéaire et angulaire utilisés dans notre modèle pour en déterminer l'impact sur la précision de positionnement.

Les résultats montrent clairement que, quelle que soit la combinaison de source de données retenue, de faibles bruits de vitesse linéaire donnent une mauvaise performance. Cela souligne le fait que la distance réellement parcourue est plus courte que celle représentée dans la carte. En effet, les segments présents dans le graphe de notre carte forment des angles pour représenter les intersections du réseau routier (figure 4.11). La trajectoire du véhicule coupe tous les changements de direction anguleux, ce qui entraîne une distance parcourue plus courte en réalité que sur le graphe. Un bruit important de vitesse linéaire est donc nécessaire pour compenser cette approximation de la carte.

Les trois meilleurs couples de bruit expérimentés dans la table 4.2 donnent des performances de localisation similaires pour les trois combinaisons de sources de mesure retenues.

Nous obtenons un niveau de bruit moyen à important pour la vitesse linéaire, et de faible à moyen pour le taux de lacet. La performance réalisée avec le gyroscope de l'ESP est très correcte comparée à celui de l'INS. Cela signifie que le gyroscope fournit une estimation suffisamment précise du taux de rotation pour notre algorithme. Bien que le cap du véhicule soit estimé par l'intégration d'un capteur de taux de lacet comportant une erreur non négligeable, le ré-échantillonnage systématique utilisé dans le filtre particulaire est capable de corriger cette dérive en sélectionnant les particules les plus probables. Au final, la combinaison du modèle différentiel (DDK) pour la vitesse linéaire et du gyroscope d'ESP pour estimer le taux de lacet donne des résultats légèrement meilleurs que l'utilisation des capteurs de l'INS. Dans la table 4.2, le meilleur couple de bruit $\{\sigma_v, \sigma_y\}$ semble être $\{20, 10\}$, qui nous donne le meilleur compromis pour minimiser les erreurs de position médiane et moyenne.

Comme mentionné plus tôt, le modèle différentiel combiné avec le gyroscope de l'ESP dépasse les performances obtenues avec les capteurs de l'INS. Ils dépassent également la combinaison speedomètre et gyroscope ESP. Nous retiendrons donc seulement le modèle différentiel DDK et le gyroscope ESP pour valider les performances de localisation sur le circuit 1.

4.4.5 Validation des performances

La table 4.3 compare les performances de positionnement sur les circuits 1 et 2. Nous faisons varier les deux meilleures combinaisons de sources de données, les trois meilleurs couples de bruit ainsi que les deux meilleures valeurs de κ.

Nous pouvons noter que les trois meilleures performances sont obtenues avec la combinaison DDK et Gyroscope d'ESP sur les deux circuits. Ceci confirme ce que nous avions validé sur le circuit 1. Nous pouvons également remarquer la même conclusion pour le paramètre κ. Par conséquent, $\kappa = 16$ semble être un bon choix.

La distance parcourue pour obtenir une distribution uni-modale des particules, en d'autre termes la convergence de l'algorithme, semble être indépendante de chaque circuit. Cette métrique ne peut donc pas être utilisée afin d'identifier les couples de bruits corrects. En regardant plus précisément, les deux tables révèlent que $\{\sigma_v, \sigma_y\}=\{20, 10\}$ tendent

		κ	
σ_v	σ_y	8	16
10	30	6.2 (74.8)	5.3 (78.5)
10	20	6.2 (30.3)	5.2 (54.7)
10	10	6.0 (17.6)	5.1 (45.7)
30	30	5.0 (14.4)	4.8 (34.1)
20	30	4.6 (12.1)	4.7 (85.4)
20	20	4.5 (12.0)	4.2 (46.9)
20	10	4.6 (11.7)	4.2 (11.5)
30	20	4.8 (14.1)	4.1 (13.8)
30	10	4.8 (14.1)	4.0 (13.4)

(a) Vitesse : INS - Taux de lacet : INS

		κ	
σ_v	σ_y	8	16
10	30	8.2 (51.4)	8.2 (93.2)
10	20	7.5 (14.5)	7.9 (76.1)
10	10	7.6 (13.9)	7.5 (16.6)
20	30	5.6 (12.7)	6.4 (49.7)
20	20	5.4 (12.4)	4.8 (30.4)
30	10	5.8 (15.0)	4.8 (15.0)
30	30	5.3 (14.5)	4.7 (17.5)
20	10	5.5 (12.4)	4.7 (12.3)
30	20	5.0 (14.0)	4.4 (15.0)

(b) Vitesse : Speedomètre - Taux de lacet : Gyroscope ESP

		κ	
σ_v	σ_y	8	16
10	30	5.9 (68.3)	5.1 (92.0)
10	20	5.8 (20.9)	5.1 (54.3)
10	10	5.8 (15.4)	5.1 (28.9)
30	30	4.7 (13.9)	4.7 (59.5)
20	30	4.5 (13.2)	4.4 (82.8)
20	20	4.3 (11.2)	4.1 (22.0)
30	20	4.6 (13.3)	3.9 (14.5)
20	10	4.8 (11.9)	3.9 (11.5)
30	10	4.6 (14.5)	3.8 (13.4)

(c) Vitesse : DDK - Taux de lacet : Gyroscope ESP

TABLEAU 4.2 – Influence des bruits σ_v and σ_y sur le modèle de déplacement - Circuit 2
Erreur de positionnement médiane $\tilde{e}$ (m) - DDK : differential drive model - Entre parenthèses erreur de positionnement moyenne $\bar{e}$ (m)

à être le meilleur compromis. En fait, l'erreur médiane et l'erreur moyenne sont du même ordre de grandeur pour ce choix. Ceci nous donne une performance plus stable ainsi qu'une meilleure précision par rapport aux autres couples de bruits pour les deux valeurs $\kappa = \{8, 16\}$ étudiées.

La figure 4.12 montre l'erreur de positionnement de véhicule en fonction de la distance parcourue sur les deux circuits. Les expériences sont répétées 10 fois pour la combinaison de paramètres $\{\kappa, \sigma_v, \sigma_y\}=\{16, 20, 10\}$. L'erreur présentée pour une distance de parcours donnée correspond à la moyenne de ces 10 expérimentations.

Une fois que la localisation a convergé, la position du véhicule est correctement suivie durant chaque expérimentation. Autrement dit, une fois l'algorithme convergé, il ne perd pas le véhicule jusqu'à la fin du trajet. De plus, il n'y a pas de différence significative de performances entre la combinaison proposée (DDK + gyroscope ESP) et l'utilisation des capteurs INS.

Finalement, la performance de positionnement obtenue par notre méthode est comparable à celle d'un GPS standard, en n'utilisant que des capteurs présents parmi les équipements obligatoires des véhicules et une carte libre de droit.

L'algorithme fonctionne en temps réel sous Matlab. Il consomme 84ms de temps CPU pour calculer l'actualisation du filtre particulaire toutes les secondes sur un core d'un processeur à 2.6GHz.

4.4.6 Nombre de particules minimum en localisation locale

Dans la méthode que nous proposons, le nombre de particules est variable. Lors de l'initialisation, nous répartissons des hypothèses tous les trois mètres sur le graphe dans un rayon de 500m. Le nombre de particules dépend donc de la densité du réseau routier autour de la position supposée d'initialisation.

A défaut d'avoir implémenté un algorithme faisant varier le nombre de particules (par exemple, *KDL-resample*, section 1.3.4.3), nous avons cherché à répondre à la question : « Quel nombre minimal de particules faut-il pour garantir la non divergence de l'algorithme lors d'une localisation locale (*tracking*) ? »

Si nous n'évaluons pas suffisamment d'hypothèses, il ne sera plus possible de couvrir tous les choix d'une intersection en prenant en compte le bruit de déplacement linéaire ; et l'algorithme risque de diverger. Ce seuil dépend forcément de la structure du réseau et de la trajectoire du véhicule.

Pour cela, nous avons initialisé un nombre N de particules à la position réelle du véhicule sur le réseau, puis nous avons exécuté notre méthode sur l'ensemble du circuit 1. Afin de s'affranchir de l'aspect stochastique du processus, nous avons répété chaque essai 50 fois, et calculé le pourcentage de parcours qui est localisé correctement jusqu'à la fin de la trajectoire. Les résultats sont présentés dans le tableau 4.4. Pour un nombre de particules supérieur ou égal à 200, notre méthode ne diverge plus.

La figure 4.13a présente la précision de localisation et le temps de calcul en fonction du nombre de particules. Pour réaliser cette illustration nous n'avons réalisé qu'un seul essai, les résultats sont donc fortement dépendants du tirage. C'est pourquoi l'expérimentation avec 75 particules semble meilleure que pour 100. Comme le montre la figure 4.13b en mode *tracking*, le temps de calcul nécessaire pour une actualisation du filtre particulaire est nettement plus faible que les 89ms de la méthode globale présentée à la section 4.4.5.

Velocity	Yaw rate	σ_v	σ_y	κ	D (m)	$\widetilde{e}$ ($\bar{e}$)
INS	INS	30	20	16	780	90.4 (94.8)
INS	INS	20	10	8	780	6.3 (6.2)
INS	INS	30	20	8	780	6.2 (6.1)
INS	INS	30	10	8	780	6.2 (6.0)
DDK	Gyroscope	30	20	8	780	5.6 (5.8)
DDK	Gyroscope	20	10	8	780	5.6 (5.6)
INS	INS	20	10	16	780	5.5 (5.6)
INS	INS	30	10	16	780	5.3 (5.2)
DDK	Gyroscope	30	10	8	780	5.1 (5.5)
DDK	Gyroscope	20	10	16	780	5.0 (5.1)
DDK	Gyroscope	30	20	16	780	4.9 (5.2)
DDK	Gyroscope	30	10	16	780	4.5 (5.0)

(a) Circuit 1

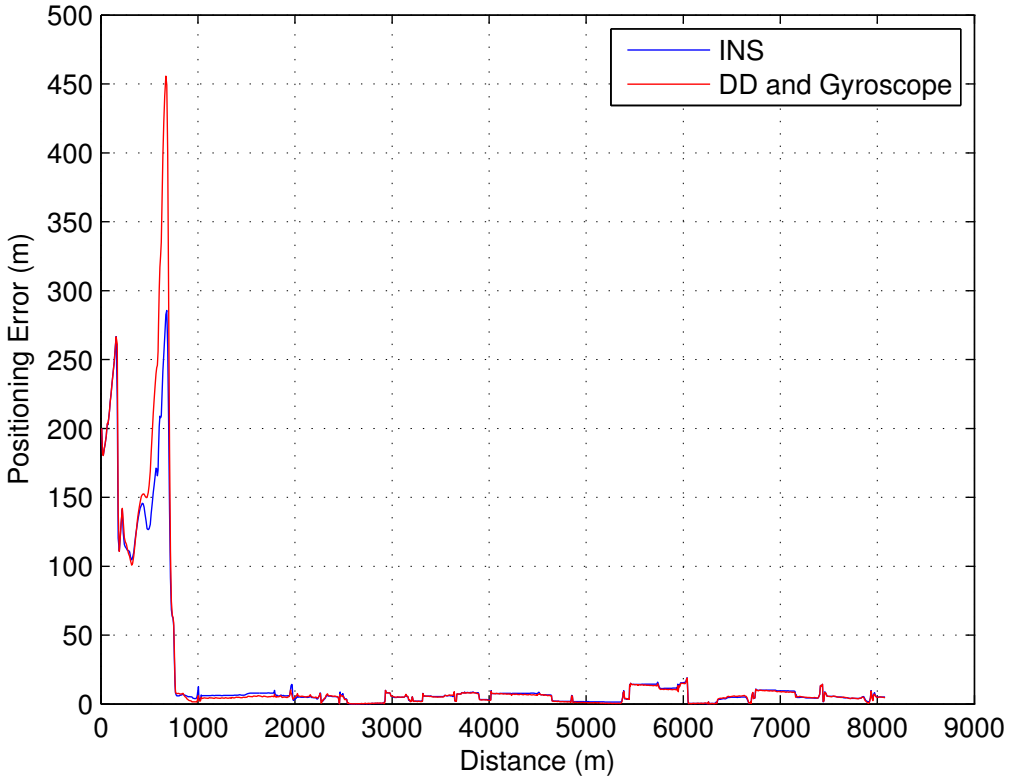
Velocity	Yaw rate	σ_v	σ_y	κ	D (m)	$\widetilde{e}$ ($\bar{e}$)
INS	INS	30	10	8	550	4.7 (8.8)
INS	INS	30	20	8	550	4.7 (8.7)
DDK	Gyroscope	20	10	8	560	4.6 (6.7)
INS	INS	20	10	8	560	4.4 (6.3)
DDK	Gyroscope	30	20	8	560	4.3 (8.1)
DDK	Gyroscope	30	10	8	550	4.2 (9.1)
INS	INS	20	10	16	550	4.2 (5.9)
INS	INS	30	20	16	560	4.0 (8.9)
INS	INS	30	10	16	550	3.9 (8.2)
DDK	Gyroscope	30	20	16	560	3.8 (9.2)
DDK	Gyroscope	20	10	16	560	3.6 (5.8)
DDK	Gyroscope	30	10	16	560	3.4 (8.3)

(b) Circuit 2

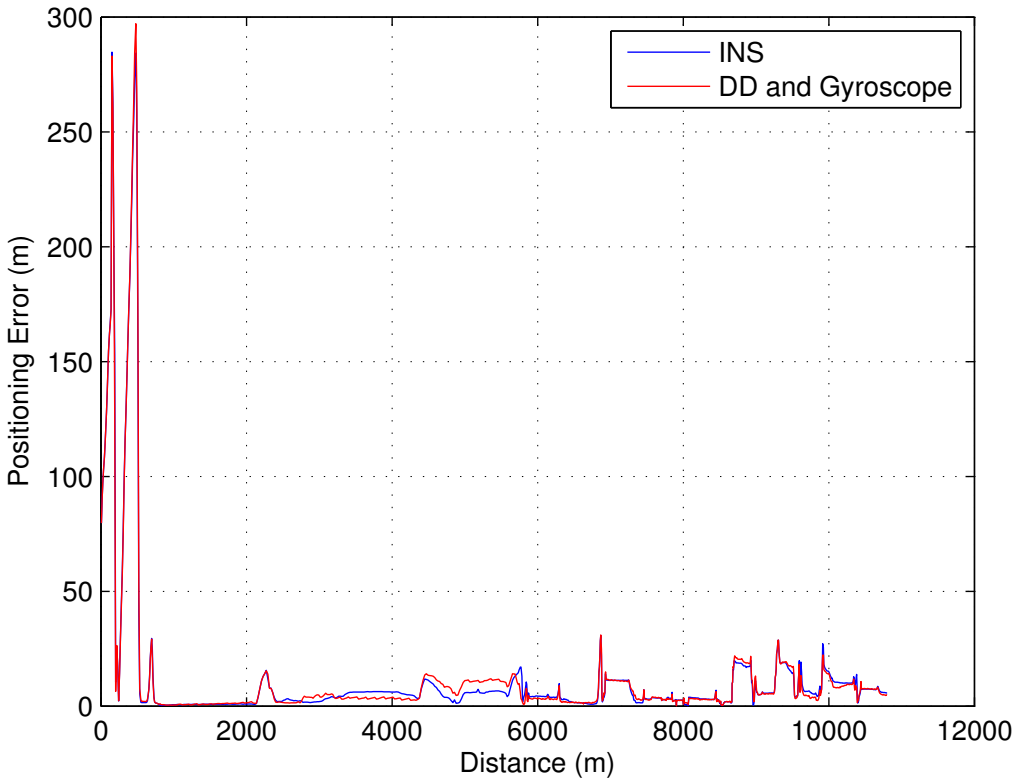
TABLEAU 4.3 – Performance de positionnement après convergence du filtre particulaire
D : distance de convergence - Erreur de positionnnement médiane $\widetilde{e}$ (m) - Erreur de positionnement moyenne $\bar{e}$ in m

Nombre de particules	25	50	75	100	200	…	1000
Ratio de positionnement correct (%)	50	80	94	96	100	…	100

TABLEAU 4.4 – Taux de convergence de l'algorithme en mode tracking - résultats obtenus sur 50 essais successifs avec le circuit 1

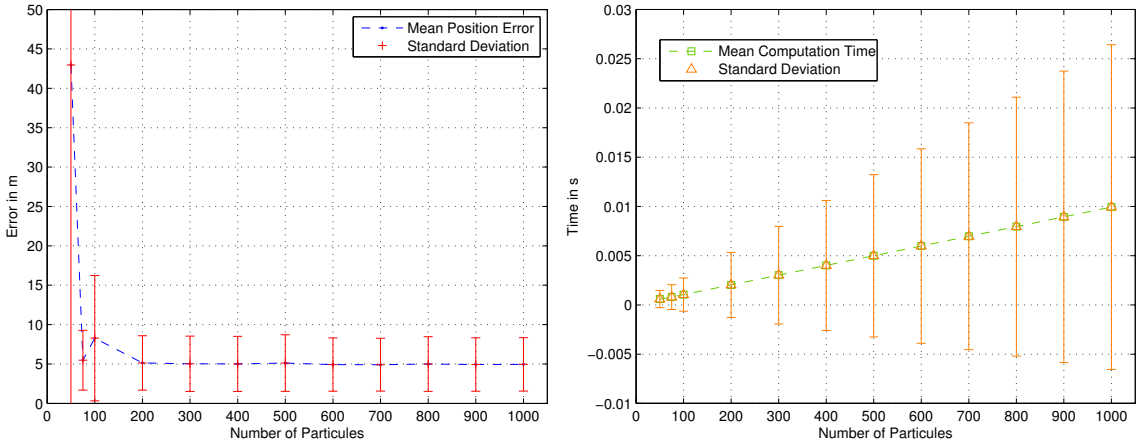


(a) Circuit 1



(b) Circuit 2

FIGURE 4.12 – Performances du positionnement du véhicule sur les circuits 1 et 2
$\kappa = 16$ $\sigma_v = 20\%$ $\sigma_y = 10\%$



(a) Précision de la position estimée en fonction du nombre de particules

(b) Temps de calcul d'une mise à jour du filtre en fonction du nombre de particules

FIGURE 4.13 – Influence du nombre de particules en mode tracking

4.5 Conclusion

Dans ces travaux, nous proposons une méthode qui permet d'obtenir la position dans un réseau routier sans utiliser d'autres capteurs que ceux existant sur les véhicules actuels. Une approche par un filtre particulaire positionne le véhicule sur une représentation en graphe du réseau. Nos expériences montrent que des capteurs de meilleurs précisions (INS) que ceux intégrés d'origine dans le véhicule n'améliorent pas la performance de notre algorithme. La précision de positionnement obtenue est du même ordre de grandeur que celle d'un GPS standard. Les véhicules sont également correctement positionnés si leur position initiale est inconnue à 500m près.

Références

Betaille, D. et R. Toledo-Moreo. 2010, «Creating Enhanced Maps for Lane-Level Vehicle Navigation», *Intelligent Transportation Systems, IEEE Transactions on*, vol. 11, nº 4, doi: 10.1109/TITS.2010.2050689, p. 786–798, ISSN 1524-9050. 122

Borgefors, G. 1988, «Hierarchical chamfer matching : A parametric edge matching algorithm», *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 10, nº 6, p. 849–865. 123

Brubaker, M. A., A. Geiger et R. Urtasun. 2013, «Lost ! leveraging the crowd for probabilistic visual self-localization», *Computer*. 123

Dupuis, Y., P. Merriaux, P. Vasseur et X. Savatier. 2015, «Vehicle positioning in road networks without gps», dans *IEEE Intelligent Transportation Systems Society Conference (ITSC)*. 131

El Najjar, M. et P. Bonnifait. 2005, «A Road-Matching Method for Precise Vehicle Localization Using Belief Theory and Kalman Filtering», *Autonomous Robots*, vol. 19, nº 2, p. 173–191. 122

Floros, G., B. van der Zander et B. Leibe. 2013, «Openstreetslam : Global vehicle localization using openstreetmaps», dans *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, IEEE, p. 1054–1059. 123

Gustafsson, F., U. Orguner, T. B. Schön, P. Skoglar et R. Karlsson. 2012, «Navigation and tracking of road-bound vehicles», *Handbook of Intelligent Vehicles*. 123

Liu, M.-Y., O. Tuzel, A. Veeraraghavan et R. Chellappa. 2010, «Fast directional chamfer matching», dans *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, IEEE, p. 1696–1703. 123

Merriaux, P., Y. Dupuis, P. Vasseur et X. Savatier. 2014, «Wheel odometry-based car localization and tracking on vectorial map», dans *ITSC*, édité par ITSC. 131

Merriaux, P., Y. Dupuis, P. Vasseur et X. Savatier. 2015, «Fast and robust vehicle positioning on graph-based representation of drivable maps», dans *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, IEEE, p. 2787–2793. 131

Parra Alonso, I., D. Fernandez Llorca, M. Gavilan, S. Alvarez Pardo, M. Garcia-Garrido, L. Vlacic et M. Sotelo. 2012, «Accurate Global Localization Using Visual Odometry and Digital Maps on Urban Environments», *Intelligent Transportation Systems, IEEE Transactions on*, vol. 13, n° 4, p. 1535–1545. 123

Quddus, M. A., W. Y. Ochieng et R. B. Noland. 2007, «Current map-matching algorithms for transport applications : State-of-the art and future research directions», *Transportation Research Part C : Emerging Technologies*, vol. 15, n° 5, p. 312 – 328, ISSN 0968-090X. 122

Szottka, I. 2013, «Particle filtering for lane-level map-matching at road bifurcations», dans *Intelligent Transportation Systems - (ITSC), 2013 16th International IEEE Conference on*, p. 154–159, doi: 10.1109/ITSC.2013.6728226. 123

Tao, Z. et P. Bonnifait. 2015, «Road invariant extended kalman filter for an enhanced estimation of gps errors using lane markings», dans *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, IEEE, p. 3119–3124. 123

Velaga, N. R. 2010, *Development of a weight-based topological map-matching algorithm and an integrity method for location-based ITS services*, thèse de doctorat, Loughborough University. 122, 135

Conclusion

Les travaux présentés dans ce mémoire ont eu comme fil conducteur la problématique de la localisation précise appliquée à la robotique mobile et à la voiture autonome. Ces travaux ont pris corps au sein de plusieurs projets de recherche auxquels j'ai participé en tant que Responsable ingénierie du laboratoire IRSEEM.

Nos premiers travaux ont été dévolus à la mise au point d'une méthode de localisation pour un robot évoluant dans un milieu industriel, par définition complexe car conjuguant les difficultés inhérentes aux environnements intérieurs (symétriques et peu discriminants) et extérieurs (étendus et perçus dans des conditions de scènes très changeantes). Cette méthode de localisation basée LiDAR, développée en marge du challenge de robotique Argos, fournit une estimation des six degrés de liberté du robot avec une précision suffisante pour en assurer le contrôle dans ses déplacements. La contrainte d'une exécution embarquée temps réel, nécessitant une localisation à une fréquence de rafraîchissement compatible avec un pilotage précis du véhicule, est un point important que nous avons tenté de mettre en avant dans ce mémoire. Notre brique de localisation étant basée sur une connaissance *a priori* de la géométrie 3D de l'environnement dans lequel va naviguer le robot, ce travail a également donné lieu à un ensemble d'outils permettant de générer ces informations à partir de données issues de la numérisation de sites. Les épreuves réalisées avec le robot Viking dans le cadre du challenge Argos ont permis de valider nos résultats dans les conditions réelles d'un site industriel. Il faut noter que le concept de challenge nous a aussi poussé à confronter nos approches avec celles d'autres équipes dans un mode d'innovation ouverte. En conclusion sur cette partie, notre succès dans les deux premières manches du challenge Argos n'aurait pas été possible sans une performance élevée de la brique de localisation du robot Viking, décrite dans le chapitre 3 du mémoire, mais surtout sans les tests que nous avons effectués avant chaque compétition et rendus possibles par les moyens mis en place dans le laboratoire décrits dans le chapitre 2 de ce manuscrit.

La méthode de localisation développée pour Argos nécessite une phase d'initialisation de la position qui peut s'avérer problématique dans de grands environnements, ce qui est le cas si l'on cherche à transposer cette approche au véhicule routier. Pour ce faire, nous avons proposé une décomposition du problème de la localisation en deux couches, la localisation métrique basée sur des mesures extéroceptives s'appuyant sur une première étape de localisation à un niveau topologique. Notre approche est partie du constat que des capteurs proprioceptifs présents dans tous les véhicules aujourd'hui, tels que l'odomètre utilisé pour la mesure de vitesse ou encore le gyromètre nécessaire au correcteur électronique de trajectoires, peuvent être exploités pour fournir cette localisation avec très peu de calcul et une performance équivalente à celle d'un récepteur GPS standard. L'avantage par rapport à un GPS est qu'elle continue de fonctionner dans les zones non couvertes par le GPS (tunnels, parkings...).

Dans le cas du développement du véhicule autonome, l'agrégation des deux méthodes de localisation constitue un système de localisation à deux couches particulièrement in-

novant ; la première permet d'initialiser la position du véhicule et la deuxième fournit une localisation avec la précision nécessaire au contrôle fin de sa trajectoire. L'évaluation dans des environnements à grande échelle (sub et inter-urbains), en s'appuyant sur les moyens de mesure de référence mis en place dans cette thèse, constitue une des perspectives de travail.

En amont, ces développements ont nécessité la mise en place de méthodes et d'outils afin d'objectiver les performances des solutions de localisation embarquée. En effet, l'étude de la performance de telles solutions nécessite une connaissance fine de la vérité terrain pendant toutes les phases de test. Nous y avons consacré une part importante de cette thèse afin de mettre en œuvre des méthodes et des moyens de référence qui constituent les principaux éléments de la plateforme *Navigation Autonome* de l'IRSEEM. Il faut souligner que la mise en place de ces moyens est un enjeu majeur dans le développement de la voiture à conduite déléguée. Dans un futur proche les constructeurs automobiles envisagent un déploiement de véhicules à haut niveau de délégation de conduite où la machine devra agir sans conducteur dans la boucle. Pour ce faire, ils devront développer des méthodes et des outils pour caractériser et qualifier la performance des capteurs et des traitements embarqués (LiDAR, RADAR, caméra…), et ce dans une très grande diversité de scénarios. Le couplage entre monde virtuel et monde réel, tel qu'introduit au chapitre 2, sera impératif pour tester tous les cas d'usage.

La localisation pour la robotique mobile autonome reste un sujet ouvert car de nombreux défis se posent dans des situations où l'environnement est complexe et dynamique. L'enjeu est aussi de trouver le capteur générique qui permettra toutes les fonctions nécessaires à la navigation autonome (localisation mais aussi détection d'obstacles, cartographie,…). Des voies nouvelles pourraient être ouvertes par des technologies encore émergentes comme les caméras temps de vol ou encore les caméras plénoptiques. Ces capteurs basés sur des matrices de pixels permettent d'obtenir directement un champ de vue large sans balayage. Ils fournissent une information de profondeur mais aussi un ensemble de longueurs d'onde, ce qui permet d'obtenir une perception de la scène sur d'autres aspects que sa seule géométrie. Là encore, se posera la question des méthodes de test et qualification pour démontrer la fiabilité des approches dans des scénarios de plus en plus complexes.

Nous constatons récemment une évolution sur la manière de concevoir des méthodes de perception. Avec les approches par apprentissage, la façon même de concevoir les algorithmes change. Nous passons d'une situation où le concepteur cherche comment résoudre le problème, à un travail consistant à comprendre le problème dans sa globalité et à sélectionner des données pour que la machine trouve une solution. Ce nouveau paradigme semble prometteur, à condition de disposer des bases de données d'apprentissage ainsi que de la vérité terrain associée permettant l'étiquetage (*labeling*) automatique. Dans ce domaine, la force de frappe des acteurs majeurs que sont les *GAFA* (Google - Apple - Facebook - Amazon) est indéniable. Néanmoins, ces nouvelles façons d'opérer avec la robotique sont suffisamment riches pour proposer de nombreux enjeux à résoudre aux scientifiques et aux industriels.

Annexe A

Étude de la précision de positionnement du système Vicon

A.1 Introduction

Dans cette annexe, nous réalisons une analyse de la précision de positionnement par le système *Vicon* que nous utilisons comme vérité terrain en intérieur (section 2.2.2.1).

Le fabricant Vicon [1] est une référence dans les systèmes de capture de trajectoire basés sur l'utilisation de marqueurs électro-optiques (au point que le nom de la marque est devenu un nom commun). Mais il existe d'autres fabricants, nous pouvons citer les dispositifs : MotionAnalysis [2], Optitrack [3], Qualisys [4], ...

Le Vicon, initialement mis au point dans le domaine biomédical pour l'analyse de la marche, a trouvé un deuxième essor très important dans l'animation info-graphique 3D. De par sa précision, il est un excellent équipement pour réaliser une vérité terrain en robotique mobile.

Sa fréquence de rafraîchissement est importante, jusqu'à plusieurs centaines de Hertz. En robotique ces systèmes sont particulièrement utiles pour la mise au point de drones multi-rotors. Il existe à travers le monde un ensemble de « salles de vol » similaire à celle décrite dans les travaux de Manecy et collab. [2015]. Que ce soit comme vérité terrain pour valider une méthode de localisation ou de reconstruction d'environnement ou directement pour obtenir la position en temps réel et directement interagir avec les boucles d'asservissement, de nombreux travaux portant sur les quadrirotors sont basés sur le système Vicon : Allen et Pavone [2016]; Ducard et Andrea [2009]; Mellinger et collab. [2012]; Mueggler et collab. [2014]...

Comme nous souhaitions utiliser ce moyen comme outil de référence pour qualifier la précision de nos méthodes de localisation, nous avons eu besoin de déterminer sa performance. Ceci nous a amené à réaliser l'étude présentée dans cette annexe.

A.2 État de l'art

Plusieurs chercheurs de la communauté se sont intéressés à la précision des systèmes de capture de mouvement.

1. http://www.vicon.com/
2. http://www.motionanalysis.com/
3. http://www.optitrack.com/
4. http://www.qualisys.com/

Manecy et collab. [2015] disposent d'une arène de vol de drone [5] équivalente en terme de volume expérimental et de nombre de caméras T40S à notre *LNA*. Ils ont démontré par l'expérience que dans le pire des cas, l'erreur de localisation d'un objet par le système de capture du mouvement n'était pas supérieure à 1.5mm. Mais ils ne disposent pas de moyen de vérité terrain autre que le système lui-même. En fait, ils ont analysé la reproductibilité de positionnement d'un ensemble de marqueurs disposés dans la scène manuellement.

Afin de valider des études de résonance aéroélastique dans une soufflerie, Barrows [2007] a qualifié des caméras Vicon MX-F40 en fixant des marqueurs sur un rail de guidage mécanique de 58cm de longueur. Par ce dispositif, il peut déplacer les marqueurs sur une dimension d'une distance parfaitement connue. En conclusion, il trouve une erreur très légèrement supérieure au millimètre, ce qui est communément admis pour ce type de système.

Yang et collab. [2012] proposent également d'évaluer la précision de positionnement en statique en utilisant une mini-machine de perçage numérique pour déplacer en trois dimensions les marqueurs. Le volume couvert est de $400 \times 300 \times 300mm$, la précision de positionnement de 20µm. Ils ont réalisé des essais avec quatre diamètres de marqueurs, ce qui ne semble pas influencer outre mesure la performance.

A des fins biomédicales, Diaz Novo et collab. [2014] nous proposent une étude plus complète comportant un aspect dynamique. Plusieurs marqueurs ont été disposés sur un support lui-même mis en rotation par un moteur électrique. La métrique mise en œuvre consiste à évaluer les longueurs des segments formés par les différents marqueurs. Les expérimentations sont menées pour deux diamètres de marqueurs, deux vitesses de rotation du moteur, et sur trois laboratoires équipés de différents systèmes de capture du mouvement : Vicon Mcam-60, Vicon T160 et Hu-m-an Canon Zr300.

Nous pouvons noter qu'à notre connaissance (table A.1), il n'existe pas d'étude en dynamique avec une vérité terrain externe pour attester de la position des marqueurs. De plus, dans les différentes travaux ci-dessus, les volumes d'expérimentation statiques sont plutôt faibles au regard des trajectoires des robots mobiles.

Sources	Volume ex.	Répétabilité	Statique	Dynamique
Manecy et collab. [2015]	$6 \times 6 \times 8m$	✖		
Barrows [2007]	soufflerie section $4.6 \times 73m$		Linéaire 0.58m	
Yang et collab. [2012]	$0.4 \times 0.3 \times 0.3m$	✖		
Diaz Novo et collab. [2014]	$2.5 \times 1 \times 1m$			Longueur de segments (vit. max 0.6m/s)
Nos travaux	$2 \times 1.5 \times 1m$	✖	✖	Vérité terrain externe (vit. max 7.6m/s)

TABLEAU A.1 – Résumé des études de précision sur les systèmes Vicon

5. www.arene-de-vol.fr/

Dans cette annexe, nous allons présenter une étude que nous avons réalisée pour qualifier la précision de notre système Vicon T40 avant d'en équiper notre laboratoire.

A.3 Méthodologie

Nous avons réalisé trois types d'études :

— Répétabilité de positionnement des marqueurs.

— Précision de positionnement statique, en utilisant un robot quatre axes à commande numérique.

— Précision de positionnement dynamique avec un dispositif motorisé de notre conception.

A.3.1 Etudes de répétabilité et de positionnement statique

Le laboratoire de CEM de l'IRSEEM dispose d'un robot 4 axes utilisé pour effectuer des expérimentations en champ proche (figure A.1). Sa résolution de positionnement est de 10µm et son volume utile de $2 \times 1.5 \times 1m$. Nous avons utilisé ce moyen d'essai pour réaliser les expérimentations de répétabilité et de précision statique. Un marqueur est fixé à l'extrémité du bras du robot dans le champ de 8 caméras T40S (figure 2.9a) puis nous relevons une liste de points.

Ces positions sont exprimées à la fois dans le repère Vicon et dans celui du robot. Afin de pouvoir comparer les résultats, nous calibrons ces deux référentiels à l'aide de la méthode des moindres carrés exposée dans la section 3.4.5.1.

A.3.2 Etude de positionnement en dynamique

Pour les tests en dynamique, nous cherchons à reproduire des vitesses au moins équivalentes à celle des robots mobiles tout en disposant d'une vérité terrain confortablement supérieure à la résolution supposée du Vicon. Nous avons conçu un « hélicoptère » présenté sur la figure A.2. Il est constitué d'un moteur électrique associé à un réducteur supportant un bras sur lequel nous avons fixé quatre marqueurs.

Afin que les inévitables jeux du réducteur ne soient pas limitants pour la précision du système, il est important que le couple de commande du moteur soit le plus stable possible. De cette manière, les jeux seront maintenus sous une contrainte de signe constant. Dans le but de garantir ce contrôle du couple, nous utilisons une carte de commande vectorielle pour piloter le moteur brushless. Cette commande a été initialement développée dans le cadre du projet NOBA [6] pour le robot de la figure C.1a.

La vérité terrain est obtenue avec un codeur 500 points en quadrature qui est installé sur l'axe du moteur. Pour garantir la synchronisation avec les données des caméras, les 3 signaux électriques sont échantillonnés directement par le boitier Vicon Giganet à 75kHz.

En utilisant les fronts montants et descendants des deux signaux et avec le rapport de réduction de 4, nous obtenons une résolution angulaire du bras de 0.045°. Ceci nous donne pour un rayon de 30cm (le cas des marqueurs A et D) une résolution de position de 0.235mm. Comme nous pouvons nous attendre à une précision du Vicon autour du millimètre, nous avons trouvé cette résolution de positionnement trop faible. Pour contourner cette limitation, nous avons interpolé les impulsions du codeur avec la fréquence

6. Projet INTERREG IIIA

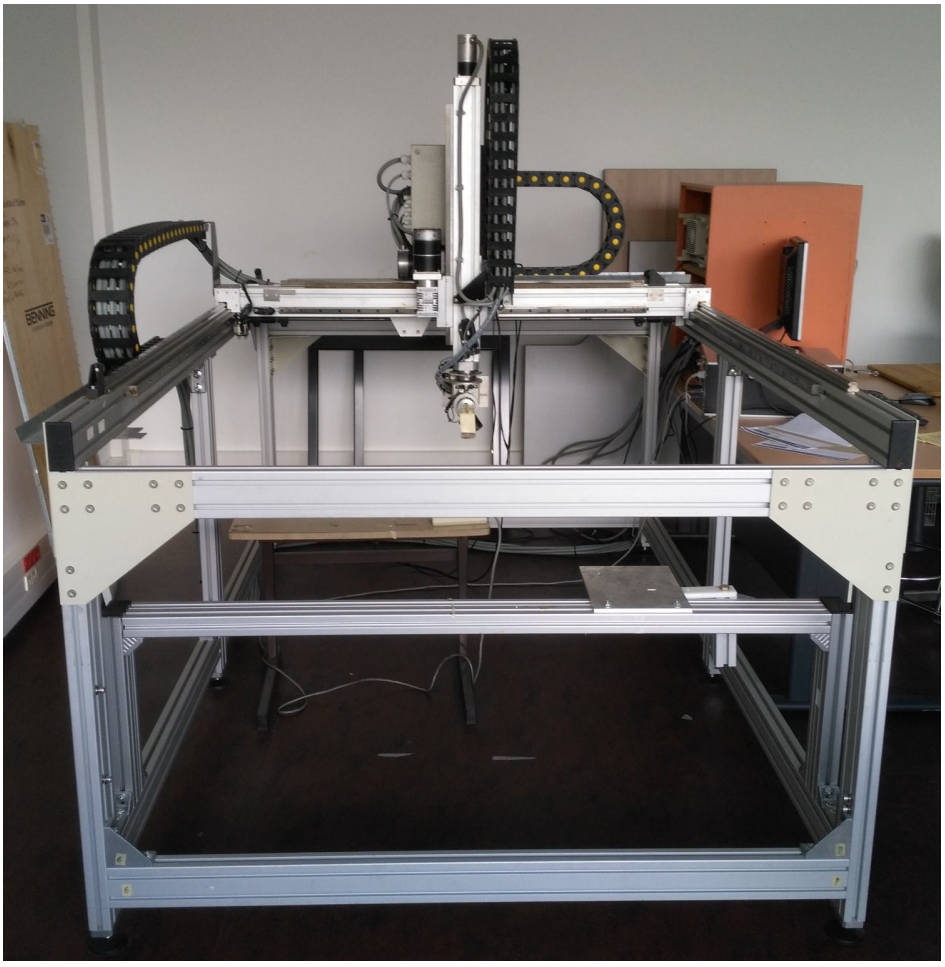


FIGURE A.1 – Robot de champ proche utilisé pour les évaluations en statique : volume utile $2 \times 1.5 \times 1 m$, résolution de positionnement 10µm

d'échantillonnage du codeur, soit 75kHz. Nous obtenons, dans ce cas, une résolution qui dépend de la vitesse de rotation du moteur. Mais à notre vitesse maximale, 1000rpm, la résolution angulaire est de 0.1257˚ ; soit toujours pour les marqueurs A et D une résolution de 0.102mm. Pour 100rpm, nous aurons une résolution dix fois supérieure autour de 10µm.

Avant de pouvoir comparer les positions des marqueurs, nous devons réaliser le changement de repère entre celui du Vicon et celui de l'axe du bras de notre hélicoptère. Une méthode de minimisation permet de déterminer le plan que forment les différents marqueurs en rotation. Une deuxième minimisation permet de d'estimer le centre du cercle formé par chaque trajectoire de marqueur. Le zéro de l'angle de rotation est ajusté arbitrairement par une calibration statique.

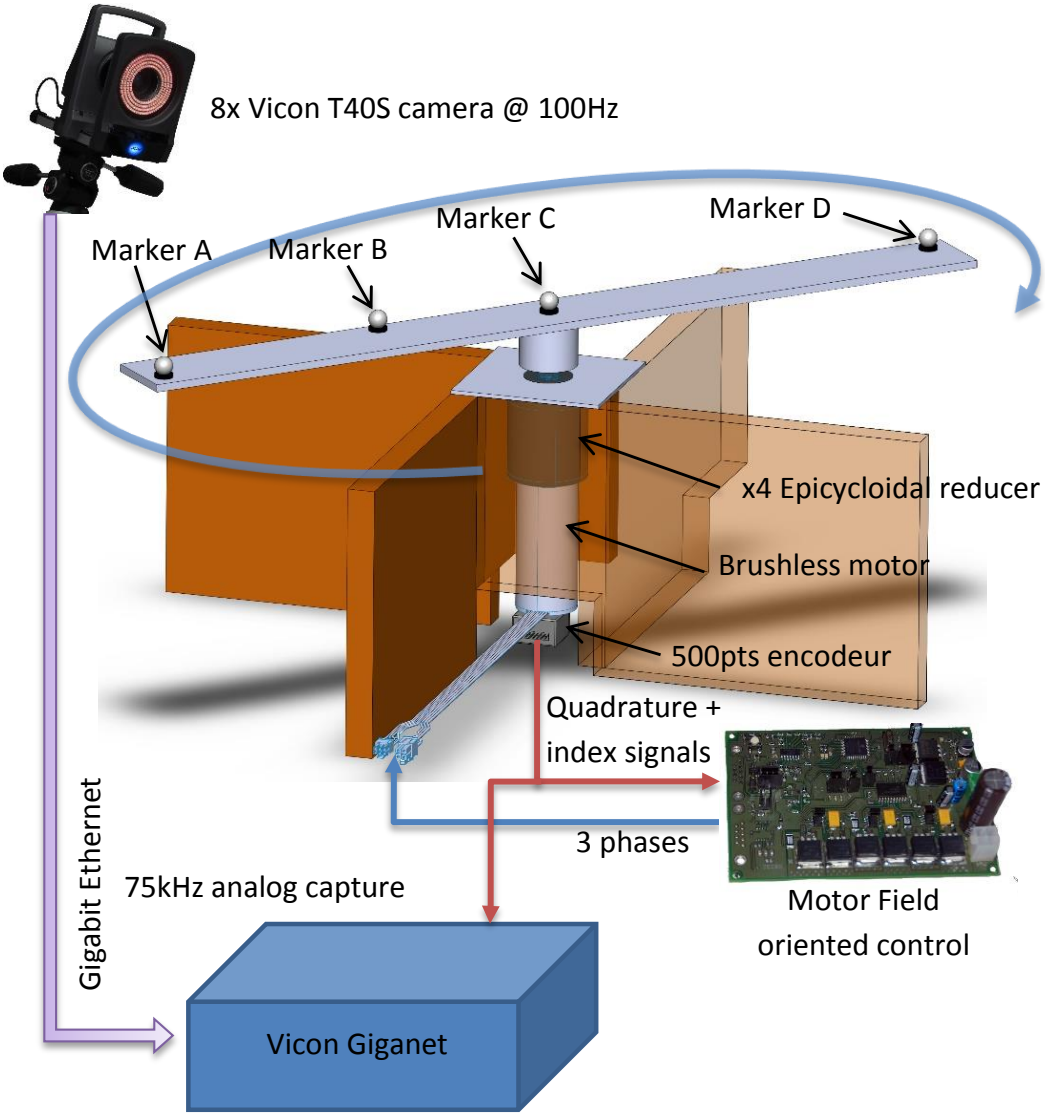


FIGURE A.2 – « Hélicoptère » utilisé pour les évaluations en dynamique.

A.4 Résultats expérimentaux

A.4.1 Répétabilité des positions

Pour déterminer la répétabilité des positions mesurées par le système, nous avons enregistré sur plusieurs minutes différents points couvrant le volume du robot. Nous calculons ensuite la position moyenne de chaque point.

La figure A.3 présente les moyennes et écarts-type des distances euclidiennes à ces positions moyennes. Hormis le cinquième point, les moyennes des erreurs de distances sont inférieures à 15µm.

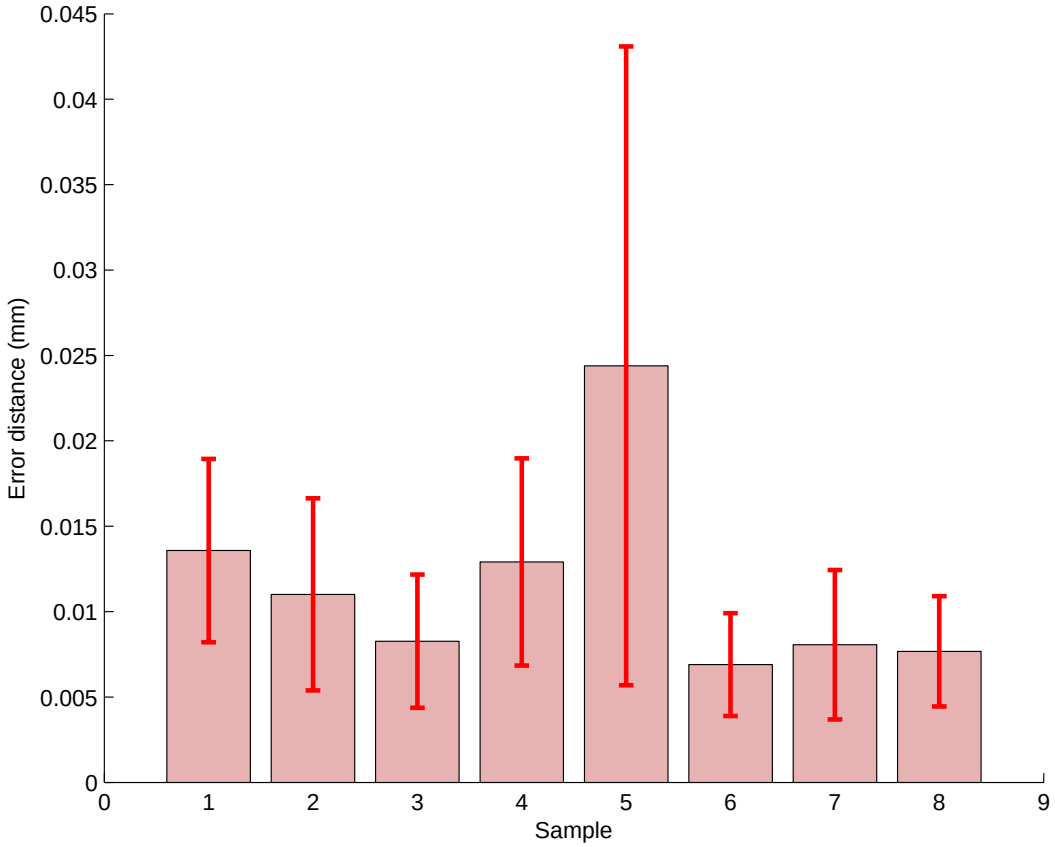


FIGURE A.3 – Erreur de répétabilité sur les 8 points utilisés pour les expérimentations statiques

La précision d'une triangulation est proportionnelle à la *baseline* entre caméras ainsi qu'à la distance avec la scène observée. Nous pouvons donc dire que la précision est proportionnelle à la dimension de l'environnement. Dans le cas de notre étude, les 8 caméras sont relativement confinées dans un volume restreint autour du robot (figures A.1 et A.6). Le volume couvert dans notre laboratoire est nettement plus important : $x \times 3.75$, $y \times 2.5$ et $z \times 2$. Ce point explique peut-être la différence de résultats avec les travaux de Manecy et collab. [2015]. Les dimensions des différents volumes expérimentaux de l'état de l'art sont résumées dans la table A.1.

A.4.2 Précision Statique

Pour l'étude de précision statique, nous comparons les poses fournies par le positionnement du robot et les mesures retournées par le Vicon. Comme expliqué section A.3.1, il est préalablement nécessaire de calibrer les deux référentiels, afin d'exprimer les mesures Vicon dans le repère robot (figure A.4).

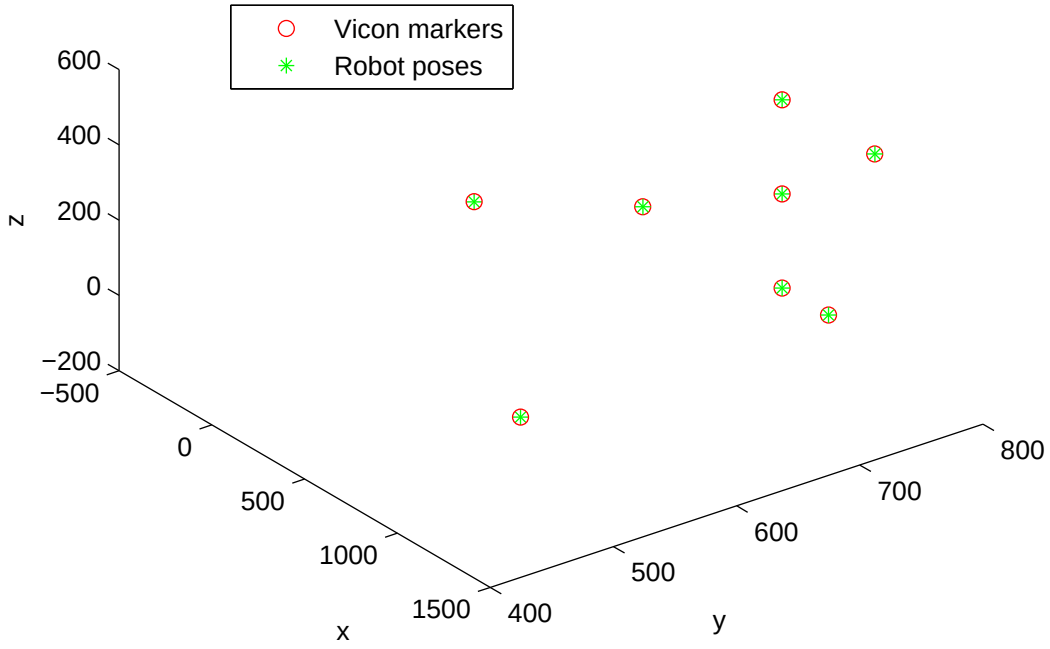


FIGURE A.4 – 8 points après calibration du repère Vicon et robot

Nous nous intéressons qu'à la position moyenne du marqueur et non au bruit de positionnement déjà étudié à la section A.4.1.

Les erreurs de distance entre le robot et les mesures Vicon sont présentées à la figure A.5. Dans la configuration de cette étude, la précision est toujours meilleure que 0.26mm.

A.4.3 Précision dynamique

L'hélicoptère (figure A.2) présenté à la section A.3.2 est utilisé pour imposer des trajectoires à 4 marqueurs tout en fournissant une vérité terrain. Les 8 caméras sont disposées de manière à couvrir convenablement la scène (figure A.6).

Nous avons réalisé 8 jeux de données en faisant varier la vitesse de rotation du moteur (table A.2). Le codeur est échantillonné à 75kHz, et nous avons utilisé deux fréquences d'acquisition pour le Vicon : 100 et 200Hz

Dataset number	Motor speed (rpm)	Usable markers	Vicon sample rate (Hz)	Encoder sample rate (kHz)
1	100	A B D	100	75
2	201	A B D	100	75
3	304	A B D	100	75
4	301	A B D	100	75
5	499	A B D	100	75
6	756	A B D	100	75
7	1000	A B D	200	75
8	1711	B	200	75

TABLEAU A.2 – Liste des jeux de données effectués sur « l'hélicoptère »

Nous avons rencontré un problème avec le jeu de données numéro 8. Avec la vitesse de

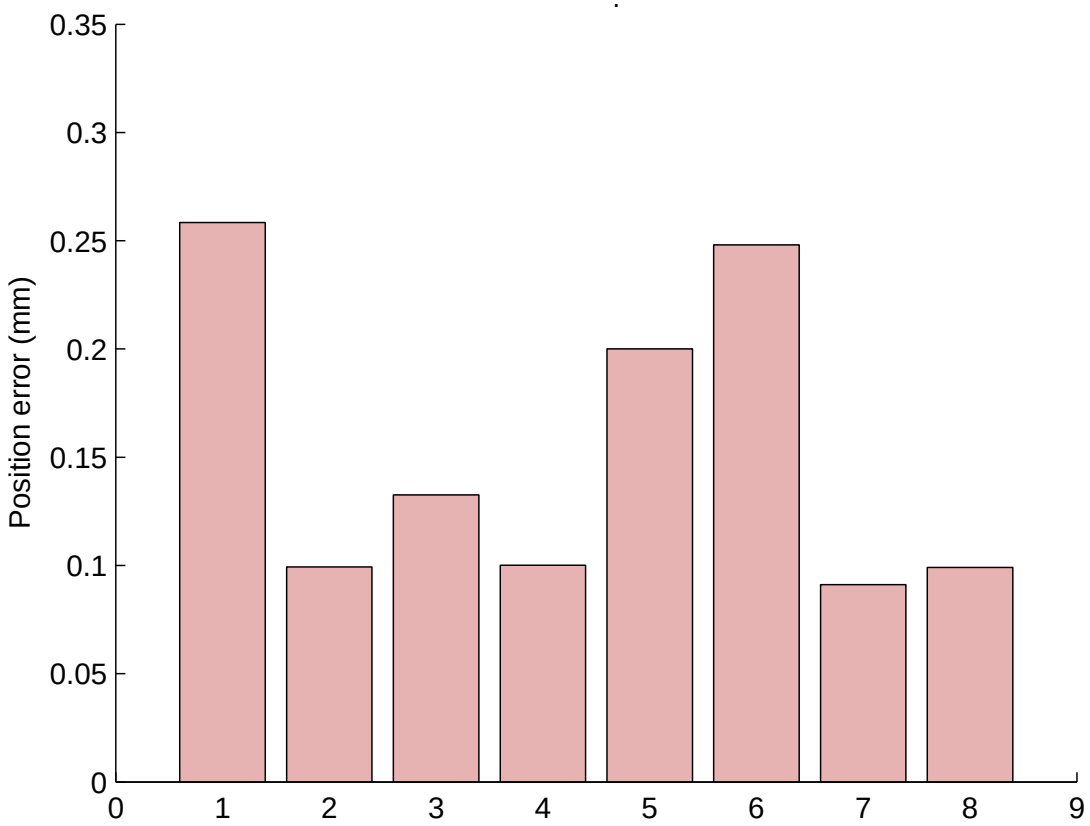


FIGURE A.5 – Evaluation du positionnement statique et de l'erreur de distance (en abscisse numéro du point).

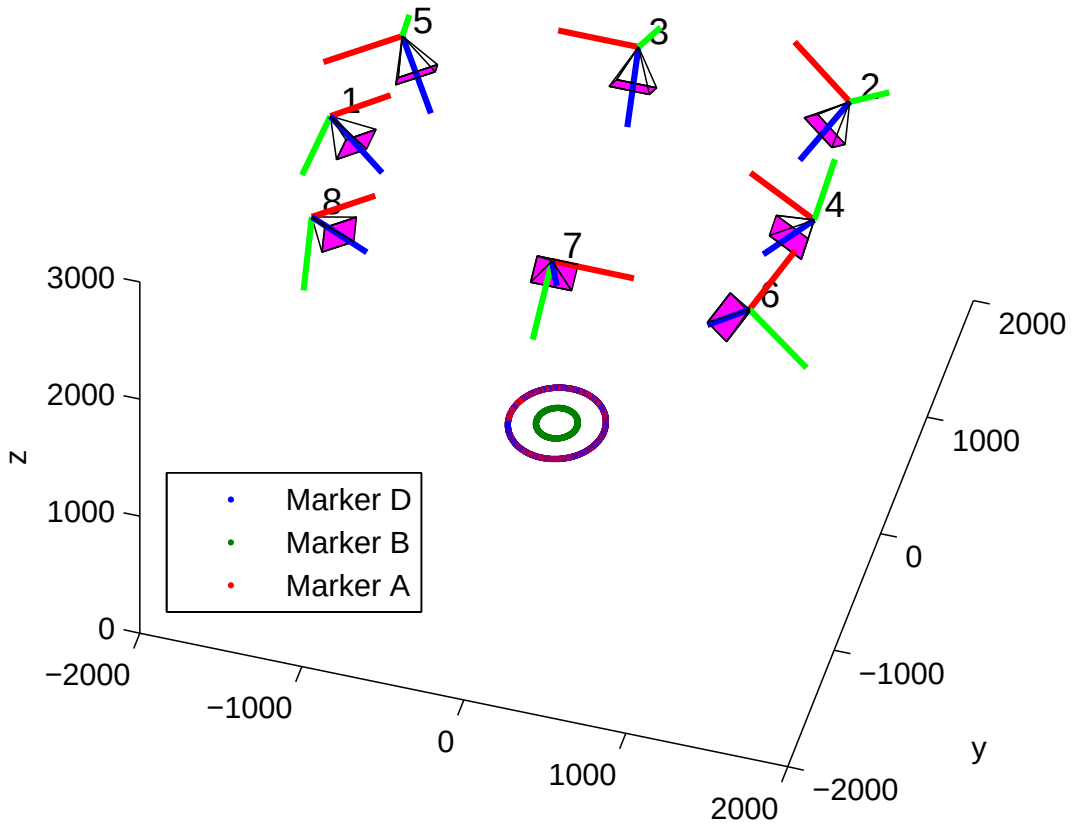


FIGURE A.6 – Dataset 1, trajectoires des marqueurs A, B et D et poses des 8 caméras Vicon T40S.

rotation importante de l'hélicoptère, les deux marqueurs extérieurs A et D se sont décollés instantanément. Pour ce dataset, seul le marqueur B est utilisable.

La figure A.7 présente la trajectoire du marqueur B et de la vérité terrain correspondante. Sa vitesse linéaire est d'environ 0.7m/s. Les échelles du graphique n'ont pas la même résolution en Z. Cela permet de voir apparaître une oscillation de la trajectoire du marqueur périodique en fonction de l'angle de rotation. Pour l'instant, nous n'avons pas identifié clairement quel phénomène pouvait amener à cette trajectoire. La mesure peut être incriminée tout aussi bien qu'un défaut mineur d'équilibrage entraînant un battement du bras. Son amplitude est faible, 0.3mm crête à crête sur l'axe Z.

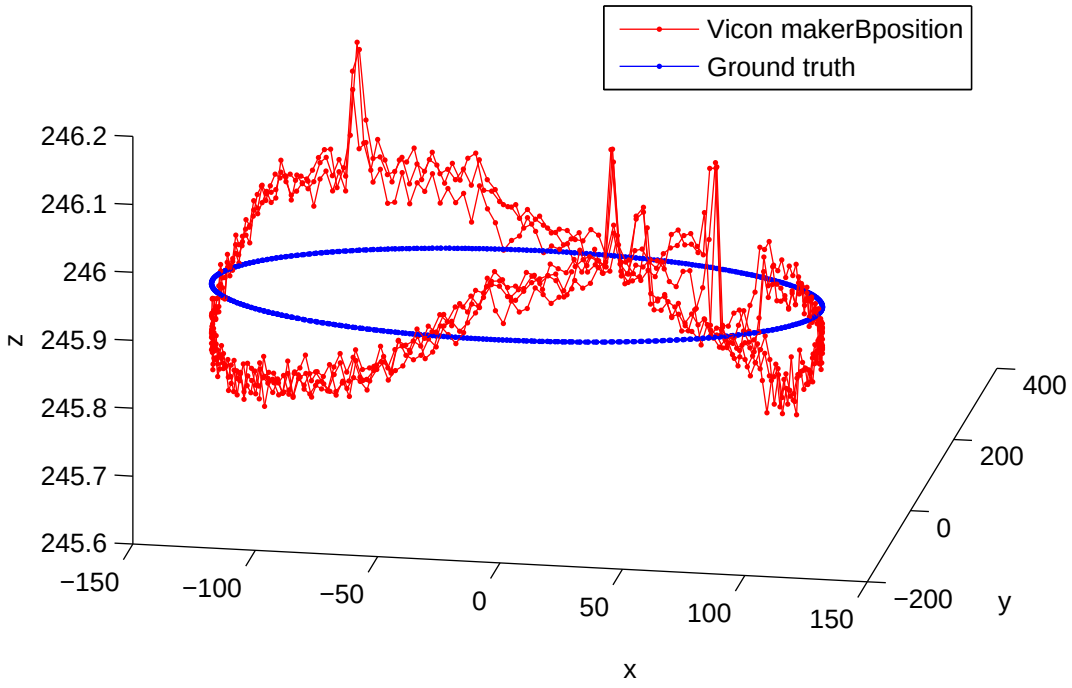


FIGURE A.7 – Trajectoire du marqueur B, dataset 1 vs vérité terrain

Les figures A.8 et A.9 présentent la distance d'erreur en dynamique, respectivement en fonction du temps et de la position angulaire du bras.

L'ensemble des résultats pour les 8 jeux de données est synthétisé dans la table A.3. Ce sont les erreurs de distances moyennes, écart-type et maximum sur l'ensemble du jeu de données pour chaque marqueur.

Hormis le marqueur D du jeu de données 2 (et des deux qui se sont décollés dans le dataset 8), l'erreur moyenne de positionnement en dynamique est contenue sous 1mm avec un écart-type inférieur à 0.6mm.

A l'heure de finir ce manuscrit, nous avons manqué de temps pour analyser finement l'ensemble les données et des métriques possibles ; comme par exemple, essayer d'identifier une corrélation entre l'erreur et la vitesse de déplacement des marqueurs qui ne semble pas se dégager à première vue.

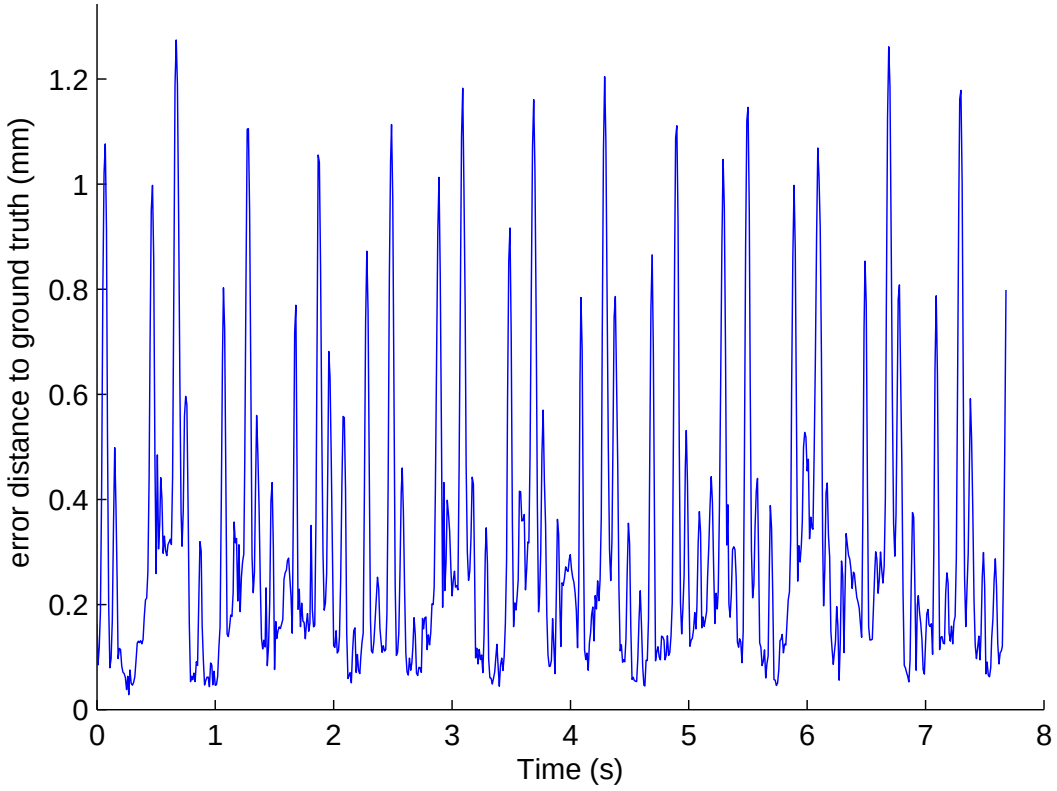


FIGURE A.8 – Erreur de position du marqueur B, dataset 1 en fonction du temps

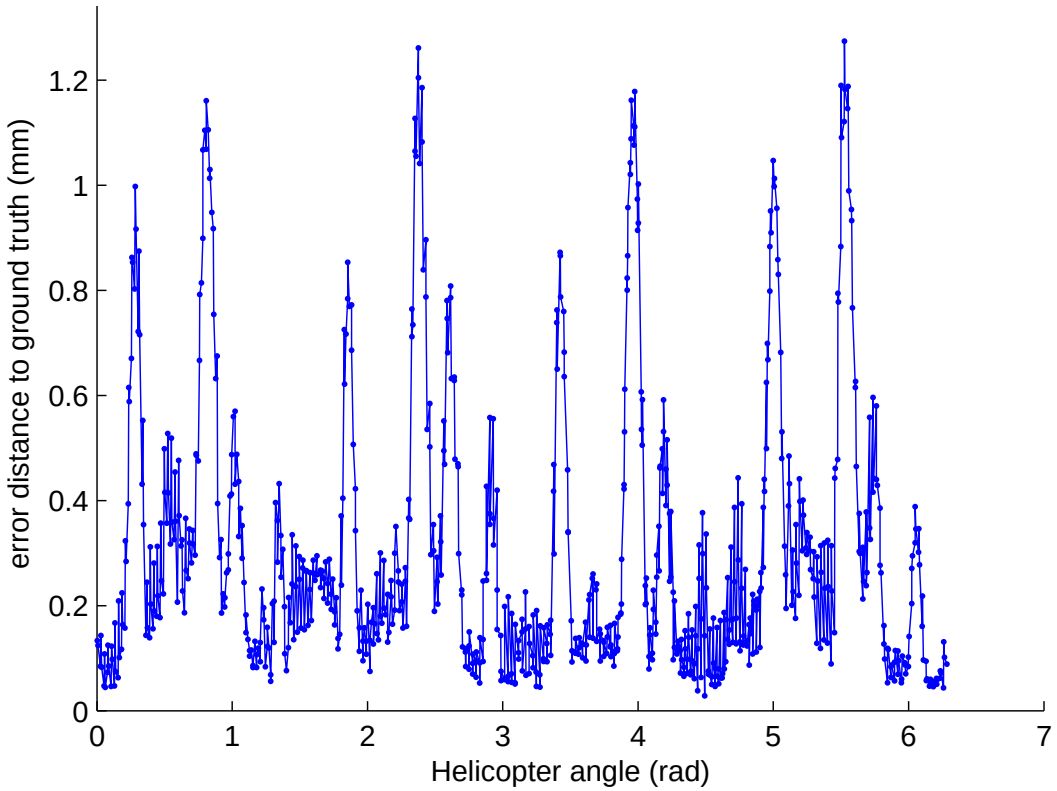


FIGURE A.9 – Erreur de position du marqueur B, dataset 1 en fonction de l'angle du bras

Dataset	Marker	Radius (mm)	Linear speed (m/s)	E mean (mm)	E std (mm)	E max (mm)
1	D	291.34145	0.75969	0.58389	0.45659	2.46132
1	B	124.31753	0.32417	0.27722	0.19096	1.02970
1	A	291.18335	0.75928	0.57178	0.46065	2.38860
2	D	291.34382	1.52970	0.50925	0.33484	1.70489
2	B	124.31848	0.65273	0.24716	0.13607	0.83483
2	A	291.18543	1.52886	0.49601	0.33335	1.76679
3	D	291.34740	2.31864	0.43318	0.29260	1.54689
3	B	124.32672	0.98943	0.22159	0.11187	0.63026
3	A	291.18954	2.31739	0.41951	0.28272	1.49675
4	D	291.34759	2.29224	0.43505	0.28801	1.51700
4	B	124.32580	0.97816	0.22421	0.10721	0.60969
4	A	291.18957	2.29100	0.42311	0.27375	1.33684
5	D	291.35390	3.80603	0.36277	0.23351	1.15113
5	B	124.34865	1.62440	0.18998	0.08958	0.53699
5	A	291.19243	3.80392	0.34661	0.22813	1.12159
6	D	291.35674	5.76989	0.35115	0.22460	1.28924
6	B	124.38350	2.46323	0.18467	0.08426	0.48978
6	A	291.19807	5.76675	0.33775	0.20917	1.04053
7	D	291.42330	7.63228	0.28310	0.16737	1.00147
7	B	124.62050	3.26377	0.15614	0.06288	0.43462
7	A	291.25648	7.62791	0.26847	0.16313	0.88642
8	D	180.08853	8.06524	229.38478	110.88958	360.17703
8	B	125.06469	5.60100	0.15262	0.05706	0.35919
8	A	315.53286	14.13109	401.62973	193.92027	631.06566

TABLEAU A.3 – Résultats bruts de l'étude dynamique avec l'hélicoptère

A.5 Conclusion et perspectives

Avec les premiers résultats de cette étude, nous avons montré que les performances du système Vicon que ce soit en terme de répétabilité de positionnement, positionnement statique ou dynamique jusqu'à des vitesses de 7.6m/s (27km/h), nous permettaient de compter sur un positionnement de l'ordre de un à deux millimètres. Au vu de la performance centimètrique atteinte par notre localisation basée LiDAR proposée dans le chapitre 3, la précision du Vicon est tout à fait compatible avec son utilisation comme vérité terrain dans le domaine de la robotique mobile.

Remerciements

Pour ces travaux, je tiens à remercier particulièrement Thomas POIRIER de la société Biometrics, qui nous a aidé à réaliser l'acquisition des données synchronisées entre la position des marqueurs et les signaux électriques issus de l'encodeur de notre moteur.

Annexe B

Redressement de scans LiDAR

Dans cette annexe, nous évaluons l'impact de différentes trajectoires de véhicule routier sur les nuages de points LiDAR embarqué. Les fréquences de balayage des LiDARs sont faibles au regard des vitesses des véhicules. Nous proposons une méthode pour pallier ce problème et des métriques de comparaison. Nos résultats montrent qu'il est nécessaire de redresser les mesures de LiDAR embarqué avant de reconstruire en 3D des environnements vastes.

Ces travaux ont donné lieu à la publication de l'article Merriaux et collab. [2016].

B.1 Introduction

Les LiDARs embarqués sur des véhicules routiers, couplés à un positionnement précis IMU et GPS RTK, sont donc couramment utilisés pour la construction de carte 3D d'environnement. Cependant afin d'atteindre une précision de reconstruction suffisante, il est nécessaire de corriger les déformations des mesures dues au déplacement du véhicule.

Ce problème est peu traité en robotique d'intérieur, car les méthodes de numérisation s'apparentent plus à une suite de déplacements et d'acquisitions statiques qu'à une acquisition en continu. Dans ce type d'applications, la vitesse est de toute façon suffisamment faible pour ne pas rencontrer d'importants problèmes de déformation de nuage de points.

La problématique est différente dans le cas d'applications sur véhicules routiers. En effet, la vitesse y est nettement plus importante qu'en robotique mobile et la précision attendue des reconstructions est supérieure aux outils de cartographie aéroportés.

Si nous pouvons supposer que ce problème est résolu - l'utilisation de LiDAR 3D pour la construction de cartes n'est pas nouvelle - il est très peu documenté dans la littérature. Or, dans toute implémentation d'un LiDAR 3D sur un véhicule lorsque le but est de construire une carte 3D précise, ce problème doit être impérativement pris en compte. Seulement quelques travaux abordent ce point. Levinson et Thrun [2014] mentionnent que chaque point LiDAR doit être corrigé en prenant en compte la pose du LiDAR à l'instant où l'écho est reçu sans donner plus de détail. Choi [2014] fait également référence à ce problème sans donner plus d'explication. De plus, nous pouvons noter que les LiDARs que nous utilisons couramment en robotique nous retournent l'ensemble des échos à la fin du scan et non à chaque instant d'échantillonnage.

Dans cette annexe, nous présentons une analyse des effets du mouvement d'un véhicule sur l'altération de mesures LiDAR et une méthode pour le recalage des acquisitions. Les expérimentations présentées mettent en évidence les effets de mouvement de translation et de rotation sur la déformation des scans et démontrent l'intérêt de mettre en

œuvre une méthode de correction.

B.2 Méthodologie

Dans le but d'évaluer un algorithme de localisation LiDAR, nous voudrions réaliser de vastes cartes de l'environnement avec un LiDAR 3D embarqué sur un véhicule. Comme évoqué plus haut, les mesures LiDAR sont déformées par le mouvement du véhicule porteur. Ces déformations sont suffisamment importantes pour altérer la précision de la carte obtenue. Il est donc nécessaire de corriger la déformation des scans LiDAR, avant de reconstruire la carte.

Il existe plusieurs solutions pour prendre en compte ce problème :

— Augmenter la fréquence d'acquisition du LiDAR. Pour une fréquence deux fois plus rapide, le véhicule aura parcouru deux fois moins de distance et les déformations seront deux fois plus faibles. C'est la solution adoptée pour la réalisation de la localisation du robot du *Challenge Argos* (chapitre 3, Merriaux et collab. [2015]), car les déplacements étaient relativement lents.

— Obtenir le déplacement du véhicule, pour corriger le nuage de points mesurés. La mesure de mouvement peut être réalisée avec un capteur annexe : odométrie, IMU, caméra Zhang et Singh [2015],... ou directement à partir de deux scans LiDAR successifs Moosmann et Stiller [2011].

Dong [2013] présente plusieurs méthodes d'interpolation de la trajectoire dans le cas où la fréquence d'acquisition de ce mouvement est faible, par exemple lorsqu'il faut attendre deux scans complets d'un LiDAR motorisé Nüchter [2007].

B.2.1 Déformation scan LiDAR

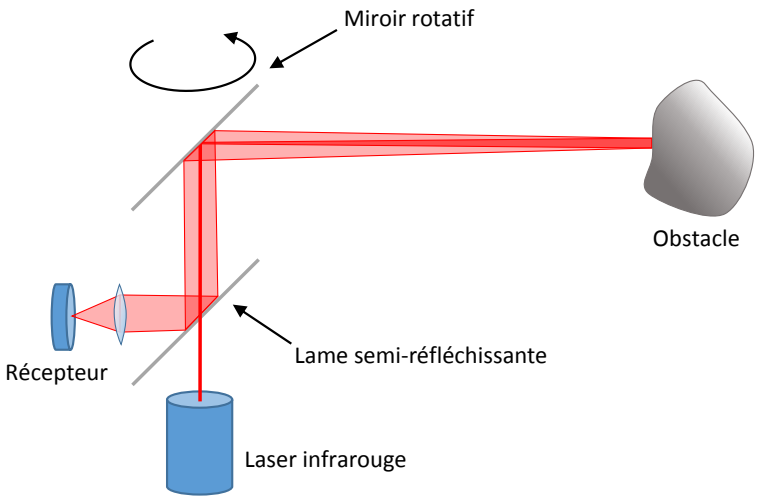


FIGURE B.1 – Synoptique simplifié du fonctionnement d'un télémètre LiDAR (*LiDAR rangefinder*)

Un LiDAR balaye l'environnement à l'aide d'un miroir déviant son faisceau laser (figure B.1). Le plus souvent, un moteur entraîne en rotation le miroir pour former un scan couvrant un certain champ de vue. La durée d'un scan T_s dépend donc de la vitesse de rotation qui peut varier de quelques hertz à quelques centaines de hertz selon les modèles.

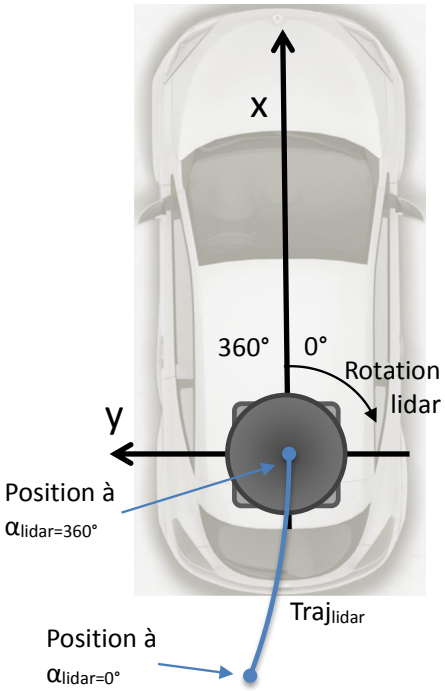
Pendant la durée T_s, si le LiDAR n'est pas immobile (figure B.2a), le scan ne sera plus centré sur un point unique. Comme le montre la figure B.2b, le mouvement du véhicule s'ajoute à la rotation du LiDAR et vient déformer la mesure obtenue.

Les déformations sont dues à un mouvement du mobile porteur et sont proportionnelles à sa vitesse et à la durée d'acquisition d'un scan T_s. Par exemple, avec un *LiDAR velodyne HDL64* à 10Hz :

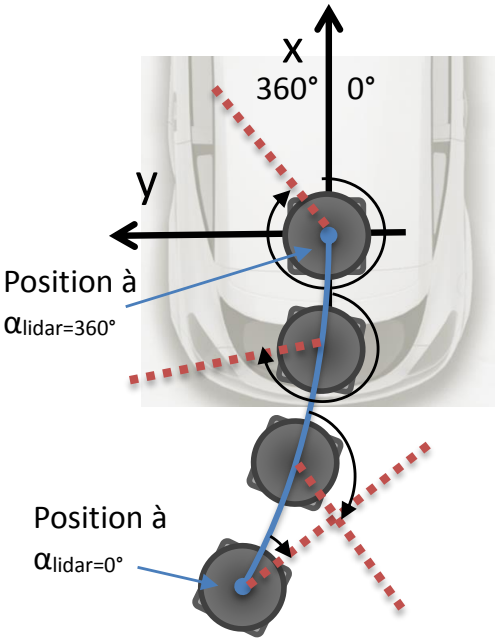
— mouvement linéaire à 50km/h : nous constatons un décalage de 1.38m entre le début et la fin du scan ;

— mouvement de rotation à 25°/s : à 50m du LiDAR, le décalage est de 2.19m.

Les nuages de points obtenus sont très déformés, au point d'être incompatibles avec une reconstruction 3D précise.



(a) Trajectoire parcourue par le LiDAR pendant la durée d'un scan.



(b) Le mouvement du véhicule s'ajoute à la rotation du LiDAR, le repère LiDAR n'est plus fixe et le scan se retrouve déformé

FIGURE B.2 – Déformation du scan LiDAR par le mouvement du véhicule.

B.2.2 Correction de la déformation

La composition du mouvement d'un véhicule routier se trouve majoritairement dans le plan. La figure B.3 nous confirme que les vitesses de rotation sont maximales sur l'axe de lacet et que les freinages et les accélérations latérales n'entraînent pas de fort taux de tangage et de roulis. Dans ce cas d'application, nous pouvons donc nous contenter d'une correction 2.5D.

Le mouvement du véhicule dans le plan peut être approximé par son odométrie. Il faut commencer par déterminer les composantes de mouvement linéaire Δx et de rotation $\Delta\theta$ par les équations B.1 et B.2.

$$\Delta x = r\frac{\Delta\theta_R + \Delta\theta_L}{2} \tag{B.1}$$

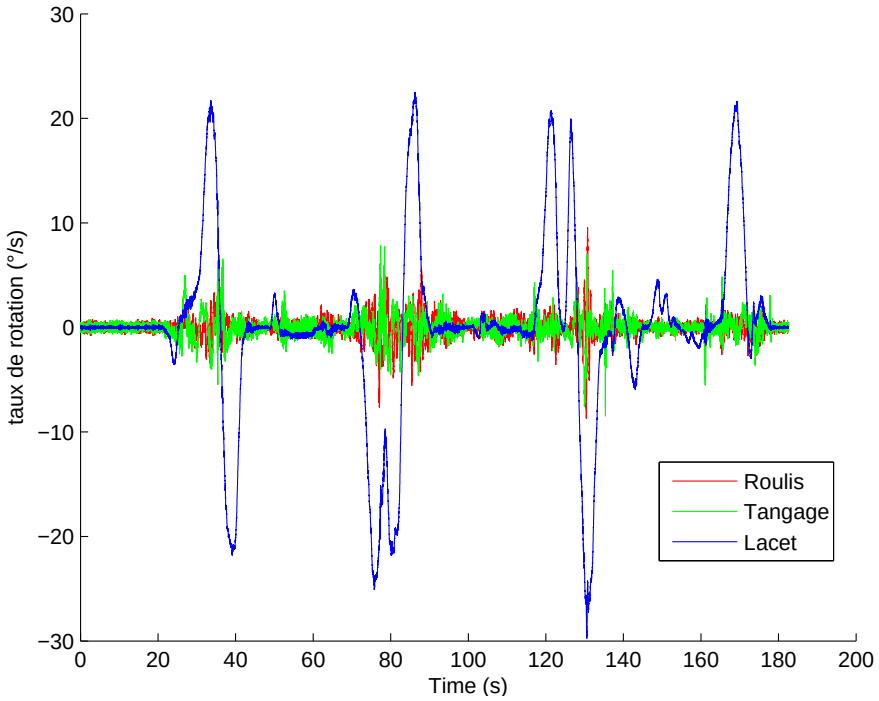


FIGURE B.3 – Taux de rotation du véhicule sur un trajet autour de notre campus : le lacet est prédominant dans les virages à basse vitesse, les 2 autres dimensions restant négligeables.

$$\Delta\theta = r\frac{\Delta\theta_R - \Delta\theta_L}{L} \tag{B.2}$$

r	:	rayon de la roue
L	:	voie du véhicule
$\Delta\theta_R$	:	angle (radian) de rotation roue droite
$\Delta\theta_R$	:	angle (radian) de rotation roue gauche
Δx	:	déplacement linéaire
$\Delta\theta$	:	rotation

Il est alors possible d'utiliser la méthode de *Runge-Kutta* pour intégrer ces déplacements locaux et déterminer la trajectoire parcourue :

$$\begin{bmatrix} X \\ Y \\ \theta \end{bmatrix}_i = \begin{bmatrix} X \\ Y \\ \theta \end{bmatrix}_{i-1} + \begin{bmatrix} \Delta x_i \cos(\theta_{i-1} + \Delta\theta_i/2) \\ \Delta x_i \sin(\theta_{i-1} + \Delta\theta_i/2) \\ \Delta\theta_i \end{bmatrix}_i \tag{B.3}$$

Dans le cas d'un LiDAR fixe, la position des échos bruts E_b en fonction des distances retournées est déterminée par une transformation du repère polaire vers un repère cartésien, avec l'équation B.4. ω représente l'élévation et α l'azimut de la raie LiDAR concernée :

$$E_b = \begin{bmatrix} X_b \\ Y_b \\ Z_b \end{bmatrix} = \begin{bmatrix} cos(\omega)cos(\alpha) \\ cos(\omega)sin(\alpha) \\ sin(\omega) \end{bmatrix} * Distance \tag{B.4}$$

Comme le LiDAR n'est pas fixe pendant T_s, nous utilisons l'équation B.3 pour déterminer une transformation le long de la trajectoire parcourue $Traj_{lidar}$; et reconstituer les échos corrigés E_c en fonction de l'évolution de l'angle d'azimut α du LiDAR.

Il est préférable d'obtenir un scan corrigé dans le repère final du LiDAR $\alpha_{lidar} = 360°$ (la fin de la trajectoire) plutôt que par rapport à la position initiale du scan $\alpha_{lidar} = 0°$

(figure B.2a). En effet, ce repère final est fixe par rapport au véhicule, et ne dépend pas de sa vitesse. Il faut donc dans un premier temps reconstituer la position du LiDAR P_α en fonction de α du miroir du LiDAR et de l'odométrie :

$$\begin{aligned} \Delta x_\alpha &= -\Delta x \frac{\alpha}{2\pi} \\ \Delta\theta_\alpha &= -\Delta\theta \frac{\alpha}{2\pi} \\ P_\alpha = \begin{bmatrix} X_\alpha \\ Y_\alpha \\ \theta_\alpha \end{bmatrix} &= \begin{bmatrix} -\Delta x_\alpha \cos(\theta_\alpha - \Delta\theta_\alpha/2) \\ -\Delta x_\alpha \sin(\theta_\alpha - \Delta\theta_\alpha/2) \\ -\Delta\theta_\alpha \end{bmatrix} \end{aligned} \tag{B.5}$$

Puis dans un second temps, nous pouvons corriger la transformation en repère cartésien (équation B.4) avec le changement de repère dû au mouvement du LiDAR. Les échos corrigés E_c deviennent :

$$\begin{aligned} E_c = \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} &= \begin{bmatrix} X_\alpha \\ Y_\alpha \\ 0 \end{bmatrix} \\ + \begin{bmatrix} cos(\theta_\alpha) & -sin(\theta_\alpha) & 0 \\ sin(\theta_\alpha) & cos(\theta_\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} &\begin{bmatrix} cos(\omega)cos(\alpha) \\ cos(\omega)sin(\alpha) \\ sin(\omega) \end{bmatrix} * Distance \end{aligned} \tag{B.6}$$

Les matrices du changement de repère sont à calculer pour chaque nouvel α, au fur et à mesure de l'acquisition du scan.

B.2.3 Évaluation des déformations

Le type de déplacement du véhicule, linéaire ou de rotation, n'influence pas de la même manière les déformations LiDAR :

— Lors d'un déplacement linéaire (figure B.4) : nous projetons les points LiDAR dans le repère de fin de la trajectoire $Traj_{lidar}$, c'est-à-dire quand $\alpha_{lidar} = 360°$. La déformation est alors minimale, alors qu'elle est maximale pour le début du scan $\alpha_{lidar} = 0°$. Par rapport au déplacement du véhicule, avec le sens de rotation du LiDAR utilisé, les scans seront beaucoup plus déformés à droite qu'à gauche.

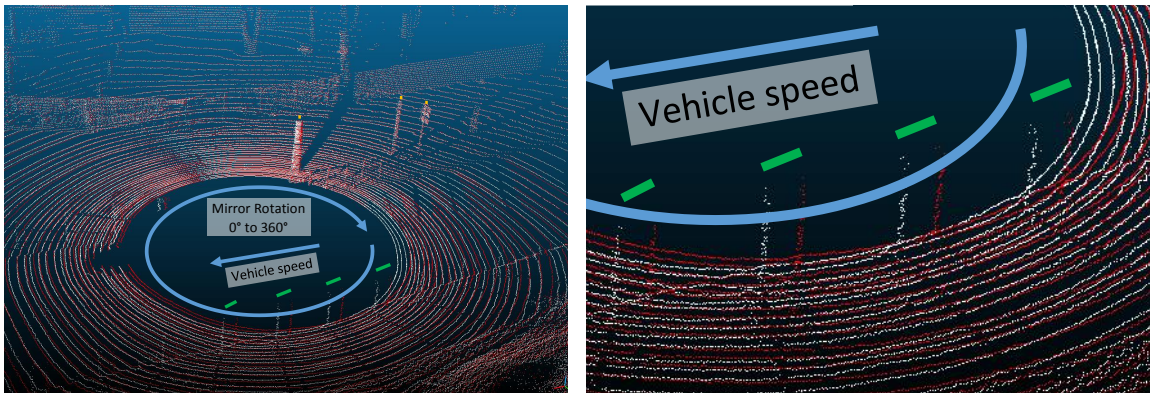
— Lors d'un déplacement en courbe : la rotation du LiDAR s'ajoute à la rotation du véhicule (figure B.5). Comme le montre la figure B.2b, selon le sens de rotation du véhicule, les raies LiDAR couvrent plus de 360˚et dans ce cas des objets de la scène seront numérisés deux fois (en début et fin de scan) ou bien moins de 360˚et une partie de la scène manquera. Évidemment, ce qui n'a pas été numérisé ne pourra pas être reconstitué, mais l'étape de correction doit corriger ce décalage angulaire.

La trajectoire d'un véhicule combinant les deux types de mouvements, les déformations sont donc entremêlées.

B.2.4 Reconstruction

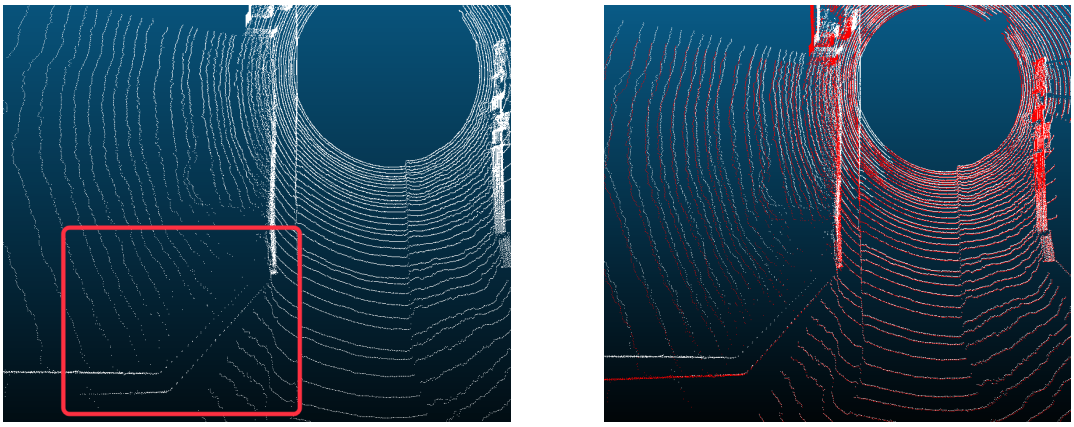
Pour évaluer la reconstruction 3D, nous utilisons l'outil 3DTK [1]. Il comporte deux phases :

1. http://slam6d.sourceforge.net/



(a) Vue générale (b) Gros plan sur le secteur angulaire $\alpha_{lidar} = 0°$

FIGURE B.4 – Déformation des scans LiDAR due à une translation du véhicule à une vitesse de 10m/s. Scan brut (blanc) et scan corrigé par l'odométrie (rouge) : les déformations sont faibles pour $\alpha_{lidar} = 360°$, les objets sont confondus ; alors que pour $\alpha_{lidar} = 0°$ les déformations sont d'environ $1m$, les poteaux au premier plan sont décalés d'1m (lignes vertes).



(a) Le véhicule et le LiDAR tournent à droite, par rapport à l'environnement. Le LiDAR parcourt plus de 360°, et la barrière encadrée en rouge apparaît deux fois

(b) Scan brut (blanc) et scan corrigé par l'odométrie (rouge), la barrière est correctement « recollée »

FIGURE B.5 – Déformation des scans LiDAR due à la rotation du véhicule à droite à la vitesse de 25°/s.

— un appariement des scans consécutifs deux à deux, basé sur une méthode optimisée Nüchter [2007] d'ICP (*Iterative Closest Point* Besl et McKay [1992]).

— une phase de recherche de fermeture de boucle Sprickerhof et collab. [2009] afin de réaliser une optimisation globale du nuage de points.

L'outil nécessite des fichiers de *points 3D* pour chaque scan, eux-mêmes associés à des fichiers de *pose*. Ces derniers servent de pose initiale à la convergence de l'algorithme d'ICP.

B.3 Expérimentation

Pour l'expérimentation, nous utilisons le coffre de toit de référence sur le Renault Espace du laboratoire (section 2.2.3.1, figures 2.14 et 2.16a). Il est équipé de différents instruments :

— IMU à *FOG* : Landins IxBlue

— Un GPS RTK différentiel Proflex 800

— Un odomètre Peiseler installé sur la roue arrière

— Un LiDAR multi-nappes HDL64e

— PC embarqué i7-3610, SSD 1 To

La précision de son positionnement est donnée dans la table B.1.

Mode	**GPS RTK**	**60s sans GPS**
True Heading (°)	0.01	0.01
Roulis/Tangage (°)	0.005	0.005
Position X,Y (m)	0.02	0.1
Position Z (m)	0.05	0.07

TABLEAU B.1 – Précision de la localisation du coffre de toit

L'odométrie linéaire Δx est obtenue directement des capteurs d'*ABS* par lecture du *bus CAN* du véhicule. Le taux de rotation $\Delta\theta$ est fourni par le gyromètre de l'*ESP*.

Un ensemble de bases de données a été enregistré dans un environnement péri-urbain couvrant les différents types de mouvements.

Le système de coordonnées géodésiques *WG84* retourné par la centrale inertielle est converti selon une projection cylindrique *Universal Transverse Mercator (UTM)*, afin de travailler dans un repère cartésien.

Nous utilisons le *framework RTMaps*[2] pour l'acquisition synchronisée des données du coffre de toit et du *bus CAN*. Un composant *RTMaps* permet en post-traitement l'extraction d'un scan 3D et d'un fichier de *pose* issu des données de la centrale inertielle.

B.4 Résultats

Comme le montre la figure B.4, pour un déplacement majoritairement linéaire, le décalage obtenu par l'algorithme par rapport à un scan correspond bien à la distance parcourue par le véhicule pendant T_s. De même pour les déplacements comprenant une

2. https://intempora.com/products/rtmaps.html

forte rotation sur la figure B.5 ; la barrière qui était vue deux fois sur la figure B.5a, est correctement recalée figure B.5b.

Cette évaluation reste qualitative. La mise en place de métriques pour objectiver la performance de la méthode n'est pas forcément évidente. Nous suggérons deux idées :

— L'appariement de scans en mouvement est perturbé par les déformations qui s'y produisent. Nous proposons donc d'utiliser l'erreur de distance point à point d'un algorithme ICP lors de l'appariement des scans successifs.

— La reconstruction 3D est moins « étalée » avec des scans correctement corrigés. Nous proposons d'échantillonner le nuage de points obtenu pour en déterminer le volume occupé.

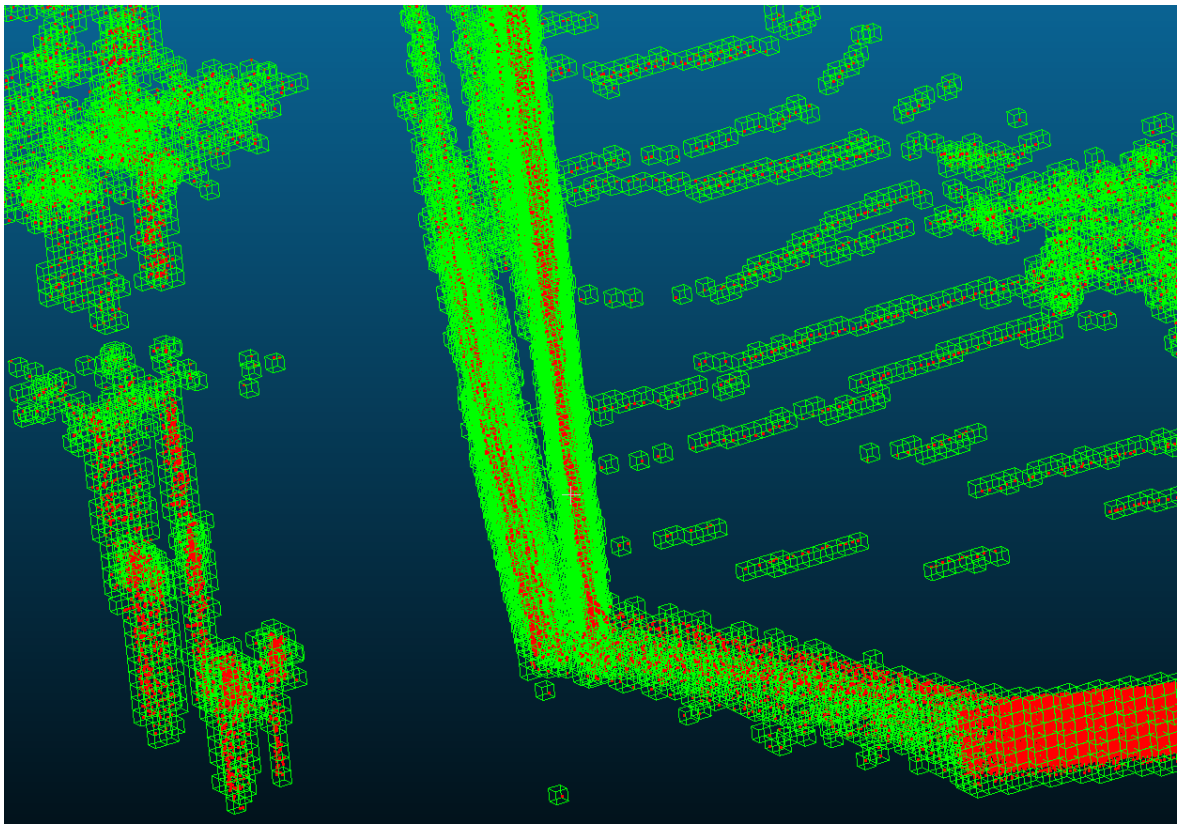
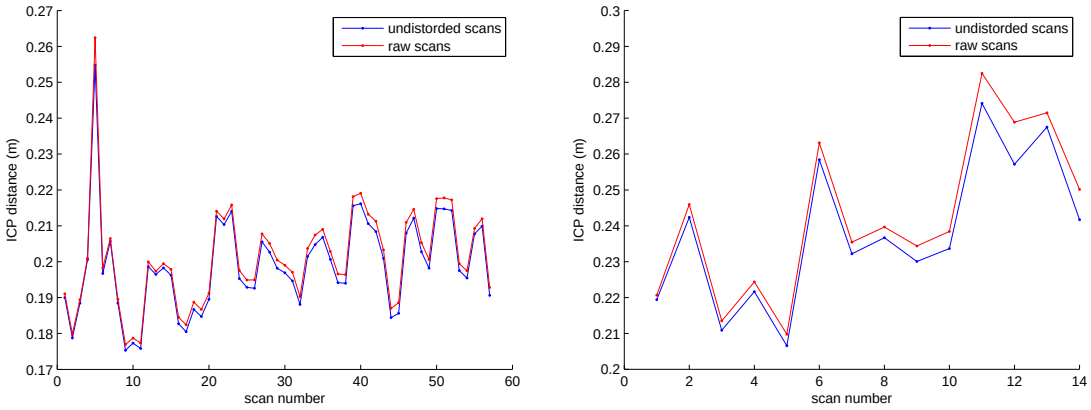


FIGURE B.6 – Reconstruction 3D du mouvement de rotation échantillonnée par un octree comportant des cases d'0.2m de côté, afin d'illustrer le principe de la table B.2. La barrière dupliquée (cf figure B.5a) dans le nuage déformé *(en rouge)* est échantillonnée par des cellules de l'octree que nous ne retrouverons pas dans la reconstruction basée sur les scans corrigés.

Ces deux idées sont développées pour les deux types de mouvements : linéaire et rotation. Un ICP est calculé entre deux scans successifs, les distances d'erreur résultantes sont présentées sur la figure B.7. Nous pouvons noter que l'écart entre les scans bruts ou corrigés n'est pas très significatif. D'un point de vue de l'appariement, la déformation n'affecte pas forcément tous les points. L'appariement de deux scans déformés contiendra toujours un grand nombre de points en commun, bien que d'un point de vue géométrique, ils ne soient pas à la position réelle de leur mesure physique ; ce qui pose problème pour la constitution d'une carte.

La différence est encore plus faible pour le déplacement linéaire (figure B.7b). Comme expliqué précédemment, deux scans successifs auront subi les mêmes déformations (cf figure B.4) et leur appariement se déroulera correctement. Si le sens de déplacement du

véhicule était inversé entre les 2 scans, les déformations entre la droite et la gauche auraient fortement nui à la distance d'ICP.



(a) 58 scans sur un déplacement linéaire à 10m/s

(b) 16 scans sur un déplacement en rotation à 25°/s

FIGURE B.7 – Résultats de la distance d'appariement entre scans consécutifs avec un algorithme ICP.

Pour chaque type de déplacement, nous avons reconstruit l'environnement 3D puis échantillonné le nuage de points obtenu à l'aide d'un octree (figure B.6) comportant des cases de 0.1m de côté. Le nombre de cases occupées nous permet de déterminer le volume occupé par la reconstruction. Les résultats sont donnés dans la table B.2. De la même manière que précédemment, seuls les points bruts très déformés et se retrouvant dans un espace vide de l'environnement, comme la barrière dupliquée des figures B.5a et B.6, viennent ponctuer les écarts avec le nuage corrigé.

TABLEAU B.2 – Approximation des reconstructions par un octree : nombre de cases de $0.1 \times 0.1 m$ occupées.

Type de mouvement	Scans bruts	Scans corrigés
Linéaire (10m/s)	1918170	1894744
Rotation (25°/s)	654488	639094

Pour valider qualitativement cette étude, nous avons procédé à une reconstruction 3D de l'ensemble de notre campus à l'aide des scans corrigés, figure B.8. Elle est constituée de 225 scans du LiDAR *Velodyne HDL64* et couvre un volume de $408 \times 275 \times 50 m$. Le nuage de points obtenu est de bonne qualité (les lignes sont bien droites, les poteaux de chaque côté de la chaussée sont alignés ...), et nous permettra de constituer notre carte de localisation. Les vues de détails de la figure B.9 mettent en évidence les erreurs de reconstruction de carte en utilisant les scans déformés directement issus des mesures brutes.

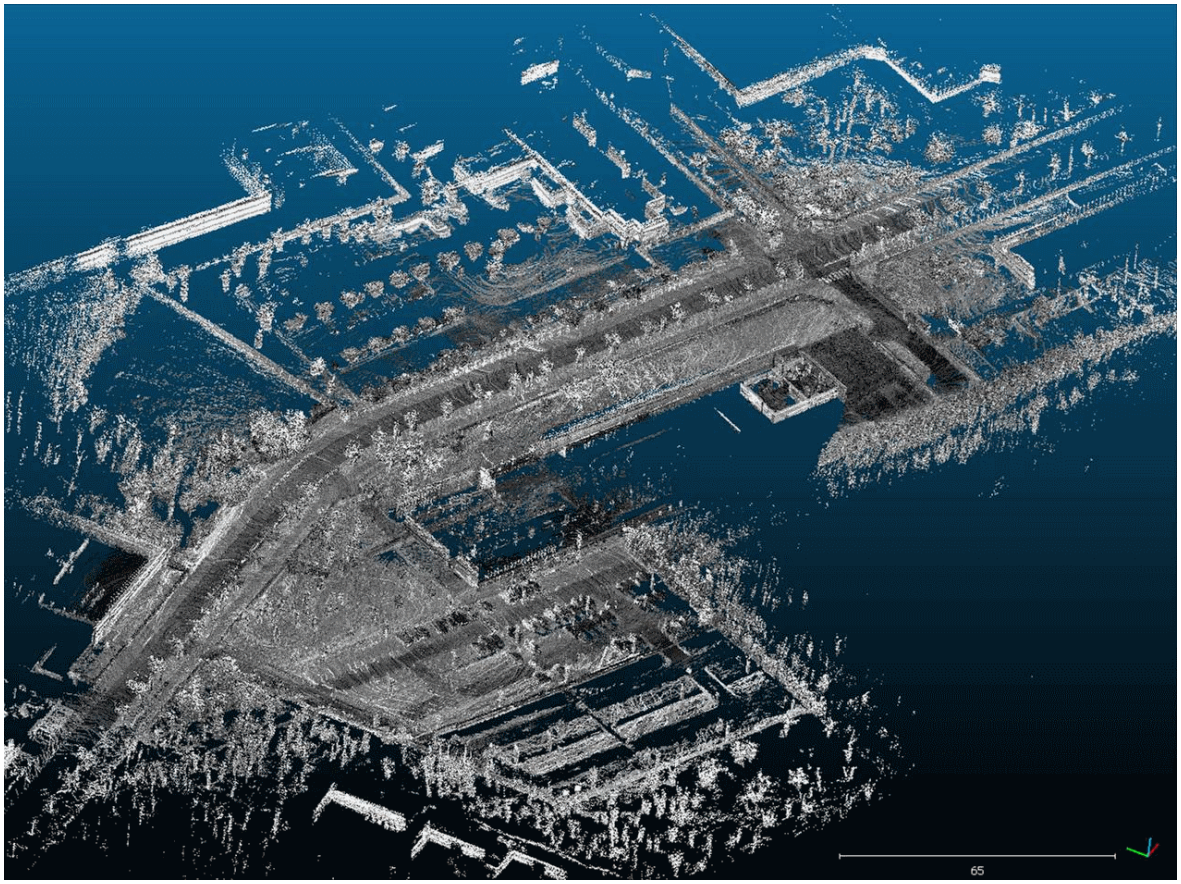
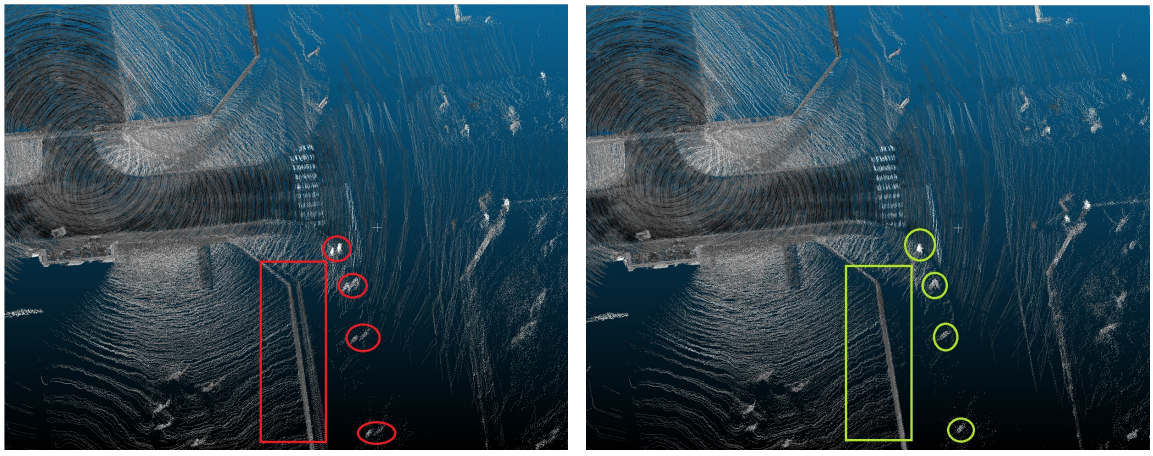


FIGURE B.8 – Première reconstruction 3D du Campus CISE de l'Esigelec obtenue avec le coffre de toit et son LiDAR *Velodyne HDL64* à partir de 225 scans. Les niveaux de gris représentent la réflectance des impacts LiDAR. Le volume reconstitué est de $408 \times 275 \times 50 m$.



(a) Reconstruction à partir des scans déformés (b) Reconstruction à partir des scans corrigés

FIGURE B.9 – Gros plan de comparaison de reconstruction de carte avec ou sans correction. Les poteaux et les arbres sont dupliqués dans la carte reconstruite à partir des scans déformés.

Conclusion et perspectives

Malgré ce que nous aurions pu penser, ce ne sont pas les mouvements de types courbes à vitesse modérée qui génèrent les plus grandes déformations. Les lignes droites rapides et les manœuvres à basse vitesse entraînent de forts taux de rotation qui sont donc plus impactants.

Bien que les métriques mises en œuvre soient perfectibles, elles permettent tout de même de mettre en évidence une meilleure géométrie des scans redressés. Afin de mieux objectiver la performance de la méthode, il faudrait numériser l'environnement statique, par exemple avec une station scanner de type *Leica C10*[3]. Cela permettrait de comparer ce nuage de points de référence avec des scans ayant subi une déformation par le mouvement du véhicule puis corrigés par notre méthode.

Nous avons réalisé une correction de déformation de scans LiDAR simple, par une estimation du mouvement en 2.5D. Le temps de calcul se limite à déterminer une transformation pour chaque angle azimut échantillonné par le LiDAR et à appliquer cette transformation à chaque écho. Les performances atteintes sont suffisantes pour reconstruire des cartes 3D d'environnements vastes.

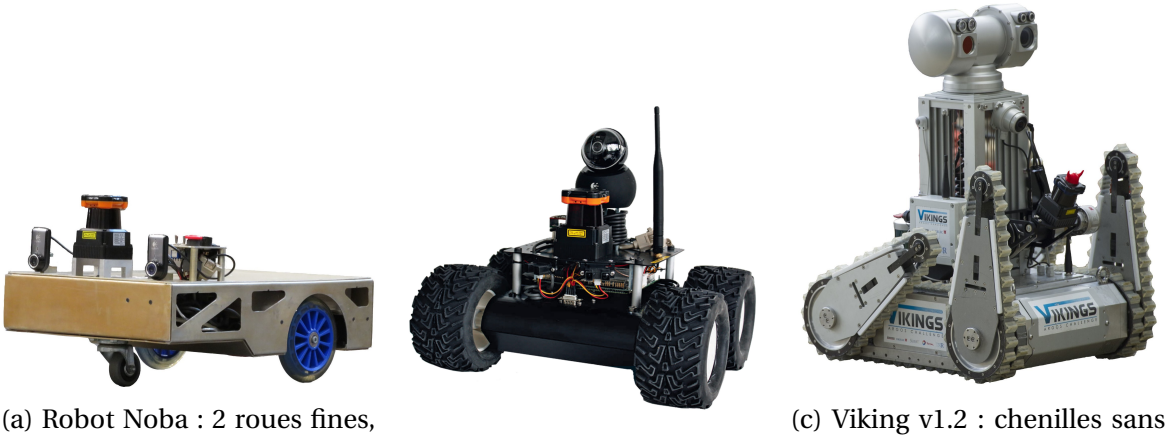
3. http://www.leica-geosystems.fr/fr/Leica-ScanStation-C10_79411.htm

Annexe C

Calibration et discussions autour de l'odométrie

C.1 Précision de l'odométrie

Comme abordé en section 1.2.2.1, la précision de l'odométrie dépend de la résolution du capteur lui-même et de la qualité du contact sol/roue. Bien que ce dernier point soit dégradé par une forte dynamique de véhicule, la structure mécanique du système de propulsion y joue un rôle prépondérant. La figure C.1 présente trois types de robots aux performances odométriques bien différentes :



(a) Robot Noba : 2 roues fines, contact au sol « ponctuel »

(b) Wifibot : 4 roues larges

(c) Viking v1.2 : chenilles sans patin d'appui

FIGURE C.1 – 3 robots, 3 qualités d'odométrie différentes

— Figure C.1a : le robot *Noba* dispose de deux roues propulsives relativement fines garantissant un bon contact au sol presque ponctuel. De plus, les contrôleurs moteurs utilisent une méthode de contrôle vectorielle. Elle permet de maîtriser finement le couple transmis, et donc de limiter les glissements. Ces performances odométriques sont donc correctes.

— Figure C.1b : le robot *Wifibot* a quant à lui quatres roues motrices. Il ne peut tourner que par un fort glissement. Du fait que les roues soient larges, le point de glissement peut se produire sur toute la largeur du pneu, et par conséquent la voie du véhicule est sujette à variation. L'estimation des rotations est d'assez mauvaise qualité.

— Le robot Viking de la figure C.1c est monté sur chenilles. C'est le moins précis des trois pour des raisons évidentes de glissement.

C.2 Calibrage et dérive

Les angles de roues ou chenilles peuvent être décomposés en déplacements linéaire et de rotation par les formules suivantes :

$$\Delta x = r\frac{\Delta\theta_R + \Delta\theta_L}{2} \tag{C.1}$$

$$\Delta\theta = r\frac{\Delta\theta_R - \Delta\theta_L}{L} \tag{C.2}$$

Où :

r	:	rayon de la roue
L	:	voie du robot
$\Delta\theta_R$	:	angle (radian) de rotation roue droite
$\Delta\theta_R$	:	angle (radian) de rotation roue gauche
Δx	:	déplacement linéaire
$\Delta\theta$	:	rotation

Il faut donc déterminer r et L par calibrage :

— r contient l'ensemble des gains de chaîne entre le codeur et la roue : gain du codeur, taux de réduction, rayon de la roue. r étant le seul paramètre de la formule C.1, nous le calibrons directement en mesurant la longueur d'une trajectoire avec notre système de référence Vicon (section 2.2.2.1). Bien sûr, l'essai doit s'effectuer dans des conditions de charge du robot similaires au cas d'utilisation envisagé.

— L représente la voie du robot et dépend de r (formule C.2). De la même manière, il est obtenu en mesurant l'angle de lacet à l'aide du Vicon avec une trajectoire circulaire.

Il est tout à fait possible de réaliser un calibrage avec des moyens de mesure nettement moins onéreux. Mais avec un système de mesure automatique comme le Vicon, il est aisé de cumuler plusieurs essais rapidement afin d'en déduire des caractéristiques de précision. Ces informations nous permettront d'ajuster la variance du bruit que nous devons fixer dans l'étape *motion update* des filtres bayésiens.

L'expérience sur des robots comme le *Wifibot* montre que la voie L n'est une constante. Elle a tendance à varier avec la vitesse de rotation. Un autre phénomène constaté par l'expérience sur les robots à chenilles *Viking* ou *Jaguar* : la voie n'est pas l'écartement entre les chenilles, mais plutôt la diagonale entre deux poulies opposées d'entraînement (figure C.2).

Cela peut s'expliquer par l'absence de patins ou de galets supports sous la chenille afin de minimiser les frottements. Sur sol plat, le contact avec le sol se fait donc sous chaque poulie.

Selon la géométrie du mobile, l'estimation odométrique peut être de piètre qualité. Le taux de rotation peut particulièrement être affecté. Dans le cas d'application exigeante sur l'estimation des mouvements, il est courant d'utiliser un gyroscope pour pallier les lacunes de modélisation du contact sol (Borenstein et Feng [1996]). Un gyroscope dérivant en fonction du temps, nous intégrons sa mesure lorsque l'odométrie nous indique un mouvement. Pour minimiser la dérive du lacet, cette intégration est stoppée si le mobile est estimé à l'arrêt.

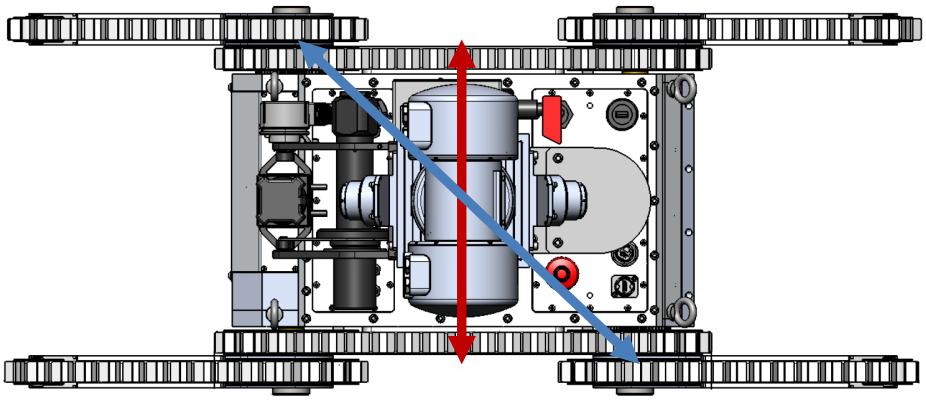
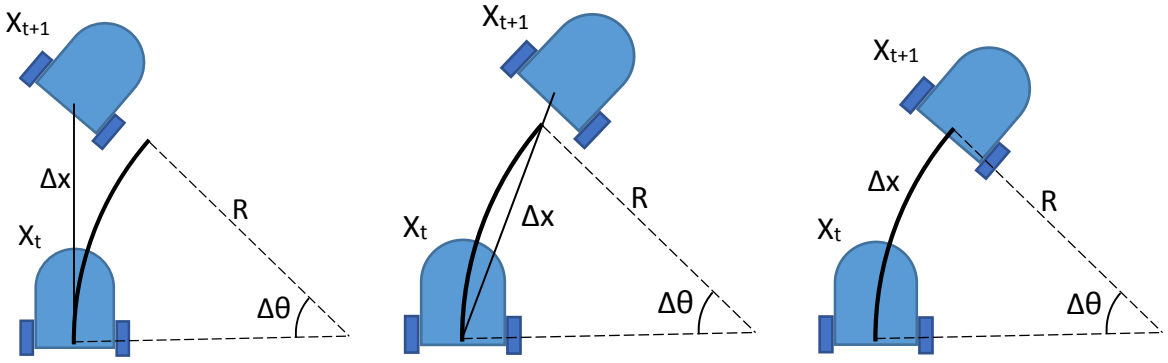


FIGURE C.2 – Différence entre la largeur du robot et la voie obtenue par calibrage sur un robot à chenilles : voie « théorique (rouge) - voie constatée par expérimentation (bleu).

C.3 Intégration de l'odométrie

Entre deux instants de X_t, un robot différentiel parcourt un arc de cercle de rayon R qui dépend du déplacement linéaire Δx et angulaire $\Delta\theta$. La figure C.3 présente plusieurs méthodes d'intégration permettant d'estimer la trajectoire parcourue dans un repère global, en fonction de l'approximation effectuée par rapport à l'arc de cercle.



(a) Euler : déplacement linéaire puis rotation

(b) Runge Kutta : demie rotation, déplacement linéaire puis demie rotation

(c) Exacte : arc de cercle

FIGURE C.3 – Différentes méthodes d'intégration de l'odométrie.

— La méthode d'*Euler* est la plus simple, mais l'estimation étant toujours à l'extérieur du cercle de rayon R, la position dérive rapidement :

$$\begin{bmatrix} X \\ Y \\ \theta \end{bmatrix}_i = \begin{bmatrix} X \\ Y \\ \theta \end{bmatrix}_{i-1} + \begin{bmatrix} \Delta x_i \cos(\theta_{i-1}) \\ \Delta x_i \sin(\theta_{i-1}) \\ \Delta\theta_i \end{bmatrix}_i \tag{C.3}$$

— La méthode « *exacte* » estime l'arc de cercle, mais nécessite deux fois plus de calculs trigonométriques que Runge Kutta :

$$\begin{bmatrix} X \\ Y \\ \theta \end{bmatrix}_i = \begin{bmatrix} X \\ Y \\ \theta \end{bmatrix}_{i-1} + \begin{bmatrix} \frac{\Delta x_i}{\Delta\theta_i}(\sin(\theta_i) - \sin(\theta_{i-1})) \\ -\frac{\Delta x_i}{\Delta\theta_i}(\cos(\theta_i) - \cos(\theta_{i-1})) \\ \Delta\theta_i \end{bmatrix}_i \tag{C.4}$$

Il y a un autre inconvénient, $\Delta x_i/\Delta\theta$ définit le rayon de courbure R de la trajectoire. En ligne droite, $\Delta\theta$ tend vers 0 et par conséquence R tend vers l'infini. L'équation C.4 est alors non définie. Il faut donc utiliser la méthode d'*Euler* ou *Runge Kutta* pour traiter ce cas.

— La méthode *Runge Kutta* est un bon compromis. La résolution du capteur étant généralement importante, l'approximation de l'arc de cercle par une droite de longueur Δx faible entraîne une dérive négligeable :

$$\begin{bmatrix} X \\ Y \\ \theta \end{bmatrix}_i = \begin{bmatrix} X \\ Y \\ \theta \end{bmatrix}_{i-1} + \begin{bmatrix} \Delta x_i \cos(\theta_{i-1} + \Delta\theta_i/2) \\ \Delta x_i \sin(\theta_{i-1} + \Delta\theta_i/2) \\ \Delta\theta_i \end{bmatrix}_i \tag{C.5}$$

Par conséquent, dans nos travaux nous avons implémenté la méthode de Runge Kutta pour estimer les déplacements.

Annexe D

Configuration de la localisation Argos

D.1 Exemple de fichier de configuration

La localisation développée pour le challenge Argos présentée dans le chapitre 3 est intégrée dans un composant RTMaps : *VikiLoc*. Ce composant utilise un fichier XML pour décrire les multiples configurations possibles :

Listing D.1 – Fichier de configuration générique du composant RTMaps *VikiLoc*

```
<?xml version="1.0" encoding="UTF-8"?>
<config nbParticles=500 NbMaxPointLidar=3600 lowVarianceResampling=0
    resampleSubsampling=1 autoPauseSubsampling=1 autoPauseDelayOdo=0></config>
<initialPosition used=2 input=0>-2.98 0.17 -0.43 0 0 0</initialPosition>
<initialPosition2>-2.98 0.17 -0.43 0 0 0</initialPosition2>
<sigmaInitialRepartition>0.1 0.1 0.01 0 0 5</sigmaInitialRepartition>
<kidnappingRobot used=1 yawSamples=8 nbMeasurementUpdate=20>D:\Documents de
    merriaux\Recherche\argos\depot\VS\lidar\config file\initParticles.xml</
    kidnappingRobot>

<IMU used=1 sigmaAngle=0.02></IMU>
<motionUpdateAddNoise>0.01 0.04 0.005 0.01 0.01 0.1</motionUpdateAddNoise>
<motionUpdatePropNoise>0.1 0.0 0.00 0.00 0.00 0.2</motionUpdatePropNoise>
<updateMeasurementLidarNoise>0.01</updateMeasurementLidarNoise>

<obstacle enable=1 distanceMax=3.0 LFmin=1 nbMax3dPoints=5000></obstacle>
<mapOctreeFile>D:\Documents de merriaux\Recherche\argos\depot\VS\lidar\config
    file\map\lna\umadLna\Lab_PF_Esc_tri5cm.xyz.octree</mapOctreeFile>
```

config configuration générale du filtre particulaire.

nbParticles fixe le nombre de particules utilisées.

nbPointLidar nombre de points LiDAR maximum à lire à chaque scan (utile pour la réservation mémoire).

lowVarianceResampling 0 utilise un ré-échantillonnage des particules de type systématique (figure 1.19a), 1 utilise un ré-échantillonnage *low variance* (figure 1.19b).

resampleSubsampling 1 le ré-échantillonnage est effectué à chaque étape *measurement update*, *n* le ré-échantillonnage est effectué à toutes les *n* étapes *measurement update*. Dans ce dernier cas, la ligne 4 de l'algorithme 1.3 sera actualisée de la manière suivante : $w_t^{[m]} = \mathrm{P}(z_t|x_t^{[m]})\, w_{t-1}^{[m]}$

autoPauseSubsampling si différent de 1, le filtre ne sera plus actualisé à chaque donnée capteur mais tous les n coups. L'idée consiste à économiser de la ressource CPU lorsque le robot ne bouge pas.

autoPauseDelayOdo délai au bout duquel si l'odométrie linéaire est nulle, le mode *autoPause* est activé.

initialPosition configuration de la position initiale du robot, similaire au vecteur d'état.

used active ou non l'initialisation de position par le fichier XML, (0 : none, 1 : une position d'initialisation, 2 : deux positions d'initialisation).

input crée une entrée sur le composant pour une initialisation dynamique, manuellement par exemple.

initialPosition2 initialisation de la deuxième position du robot, similaire au vecteur d'état.

kidnappingRobot configuration de la fonction kidnapping robot. Le fichier XML en argument décrit la liste des positions 3D qui seront évaluées au démarrage.

used active ou non le kidnapping

yawSamples nombre de subdivisions à évaluer pour la dimension R_z.

nbMeasurementUpdate nombre d'étapes de *measurement update* après lequel le nombre de particules sera réduit au paramètre *nbParticles*.

IMU configuration de l'utilisation de l'IMU pour aider à l'estimation du roulis et tangage respectivement R_x et R_y.

used active ou non l'IMU.

sigmaAngle bruit de mesure d'angle de l'IMU.

motionUpdateAddNoise sigma du bruit additif de l'étape *motion update* ($\boldsymbol{\Sigma}_a$).

motionUpdatePropNoise sigma du bruit proportionnel de l'étape *motion update* ($\boldsymbol{\Sigma}_p$).

updateMeasurementLidarNoise sigma du bruit de mesure LiDAR.

obstacle configuration de la fonction de détection d'obstacle associée à la localisation.

enable active ou non la détection d'obstacle (0 : disable, 1 : une entrée de points 3D, 2 : deux entrées de points 3D pour l'utilisation de deux LiDARs avant/arrière).

distanceMax distance maximum pour considérer le point 3D comme un obstacle.

LFmin seuil minimum du champ de vraisemblance pour considérer le point comme un obstacle.

nbMax3dPoints nombre maximum de points 3D d'obstacle que la sortie pourra renvoyer (réservation mémoire).

mapOctreeFile fichier contenant l'octree hybride au format binaire décrit à la section 3.3.4.

Références des annexes

Allen, R. et M. Pavone. 2016, «A real-time framework for kinodynamic planning with application to quadrotor obstacle avoidance», dans *AIAA Conf. on Guidance, Navigation and Control, San Diego, CA*. I

Barrows, D. A. 2007, «Videogrammetric model deformation measurement technique for wind tunnel applications», *AIAA Paper*, vol. 1163, p. 2007. II

Besl, P. J. et N. D. McKay. 1992, «Method for registration of 3-d shapes», dans *Robotics-DL tentative*, International Society for Optics and Photonics, p. 586–606. XIX

Borenstein, J. et L. Feng. 1996, «Gyrodometry : A new method for combining data from gyros and odometry in mobile robots», dans *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, vol. 1, IEEE, p. 423–428. XXV

Choi, J. 2014, «Hybrid map-based slam using a velodyne laser scanner», dans *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, ISSN 2153-0009, p. 3082–3087, doi: 10.1109/ITSC.2014.6958185. XIII

Diaz Novo, C., S. Alharbi, M. Fox, E. Ouellette, E. Biden, M. Tingley et V. Chester. 2014, «The impact of technical parameters such as video sensor technology, system configuration, marker size and speed on the accuracy of motion analysis systems», *Ingeniería mecánica, tecnología y desarrollo*, vol. 5, nº 1, p. 265–271. II

Dong, H. 2013, *Performance Improvements for Lidar-Based Visual Odometry*, thèse de doctorat, University of Toronto. XIV

Ducard, G. et R. D. Andrea. 2009, «Autonomous quadrotor flight using a vision system and accommodating frames misalignment», dans *Industrial embedded systems, 2009. SIES'09. IEEE international symposium on*, IEEE, p. 261–264. I

Levinson, J. et S. Thrun. 2014, «Unsupervised calibration for multi-beam lasers», dans *Experimental Robotics*, Springer, p. 179–193. XIII

Manecy, A., N. Marchand, F. Ruffier et S. Viollet. 2015, «X4-mag : A low-cost open-source micro-quadrotor and its linux-based controller», *International Journal of Micro Air Vehicles*, vol. 7, nº 2, p. 89–110. I, II, VI

Mellinger, D., N. Michael et V. Kumar. 2012, «Trajectory generation and control for precise aggressive maneuvers with quadrotors», *The International Journal of Robotics Research*, p. 0278364911434 236. I

Merriaux, P., Y. Dupuis, R. Boutteau, P. Vasseur et X. Savatier. 2015, «Localisation robuste en milieu industriel complexe», Gretsi, Lyon, France. XIV

Merriaux, P., Y. Dupuis, R. Boutteau, P. Vasseur et X. Savatier. 2016, «Correction de nuages de points lidar embarqué sur véhicule pour la reconstruction d'environnement 3d vaste», RFIA, Clermont Ferrand, France. XIII

Moosmann, F. et C. Stiller. 2011, «Velodyne slam», dans *Intelligent Vehicles Symposium (IV), 2011 IEEE*, IEEE, p. 393–398. XIV

Mueggler, E., B. Huber et D. Scaramuzza. 2014, «Event-based, 6-dof pose tracking for high-speed maneuvers», dans *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, IEEE, p. 2761–2768. I

Nüchter, A. 2007, «Parallelization of scan matching for robotic 3d mapping.», dans *EMCR*. XIV, XIX

Sprickerhof, J., A. Nüchter, K. Lingemann et J. Hertzberg. 2009, «An explicit loop closing technique for 6d slam.», dans *ECMR*, p. 229–234. XIX

Yang, P.-F., M. Sanno, G.-P. Brüggemann et J. Rittweger. 2012, «Evaluation of the performance of a motion capture system for small displacement recording and a discussion for its application potential in bone deformation in vivo measurements», *Proceedings of the Institution of Mechanical Engineers, Part H : Journal of Engineering in Medicine*, vol. 226, n° 11, p. 838–847. II

Zhang, J. et S. Singh. 2015, «Visual-lidar odometry and mapping : Low-drift, robust, and fast», dans *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, IEEE, p. 2174–2181. XIV

Annexe E

Liste des publications

E.1 Publications en rapport direct avec ces travaux de thèse

E.1.1 Conférences internationales

Merriaux, P., Y. Dupuis, P. Vasseur et X. Savatier. 2015, «Fast and robust vehicle positioning on graph-based representation of drivable maps», dans *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, IEEE, p. 2787–2793.

Dupuis, Y., P. Merriaux, P. Vasseur et X. Savatier. 2015, «Vehicle positioning in road networks without gps», dans *IEEE Intelligent Transportation Systems Society Conference (ITSC)*.

Merriaux, P., Y. Dupuis, P. Vasseur et X. Savatier. 2014, «Wheel odometry-based car localization and tracking on vectorial map», dans *ITSC*, édité par ITSC.

Merriaux, P., R. Boutteau, P. Vasseur et X. Savatier. 2014, «Imu/lidar based positioning of a gangway for maintenance operations on wind farms», IEEE/RSJ International Conference on Intelligent Robots (IROS), Chicago, USA.

Dupuis, Y., P. Merriaux, P. Subirats, R. Boutteau, X. Savatier et P. Vasseur. 2014, «Gps-based preliminary map estimation for autonomous vehicle mission preparation», IEEE/RSJ International Conference on Intelligent Robots (IROS), Chicago, USA.

Belbachir, A., R. Boutteau, P. Merriaux, J. Blosseville et X. Savatier. 2013, «From autonomous robotics toward autonomous car», dans *IEEE Intelligent Vehicle Symposium (IV 2013)*, Gold Coast, Australia.

Belbachir, A., J.-C. Smal, J.-M. Blosseville et P. Merriaux. 2012, «Robert : a robotic platform for testing adas and autonomous car driving capabilities», dans *SIA Vision*, édité par société des ingénieurs de l'automobile, Versailles, France.

E.1.2 Conférences nationales

Merriaux, P., Y. Dupuis, R. Boutteau, P. Vasseur et X. Savatier. 2016, «Correction de nuages de points lidar embarqué sur véhicule pour la reconstruction d'environnement 3d vaste», RFIA, Clermont Ferrand, France.

Merriaux, P., Y. Dupuis, R. Boutteau, P. Vasseur et X. Savatier. 2015, «Localisation robuste en milieu industriel complexe», Gretsi, Lyon, France.

Merriaux, P., R. Boutteau, P. Vasseur et X. Savatier. 2014a, «Algorithme de positionnement d'une passerelle à mouvements compensés à partir de mesures inertielles et lidar pour les opérations de maintenance des parcs éoliens offshore», RFIA, Rouen.

E.1.3 Revues techniques

Merriaux, P., Y. Dupuis, P. Vasseur et X. Savatier. 2015, «Méthode de localisation de véhicule sans dispositif gps», *Revue Générale des Routes et de l'Aménagement (RGRA)*

E.1.4 Communications orales

P. Merriaux, X. Savatier. juin 2016, «Localisation du robot Vikings vainqueur des deux premières étapes du challenge international Argos», dans *GDR ISIS*, Paris - Télécom ParisTech, France

P. Merriaux, X. Savatier. juin 2016, «Techniques de détection de scènes pour l'autonomie (véhicules, robots)», dans *Journée annuelle de l'AREMIF*, Paris - Télécom ParisTech, France

P. Merriaux, N. Ragot. juin 2013, «IHM pour robot autonome, Projet pédagogique», dans *GDR Robotique*, Paris, France

E.2 Autres publications

Ouerghi, S., R. Boutteau, P. Merriaux, N. Ragot, X. Savatier et P. Vasseur. 2016, «Absolute localization using visual data for autonomous vehicles», dans *International Conference on Computer Vision Theory and Applications (VISAPP)*, Rome, Italy, p. 597–603.

Seifert, L., M. L'Hermette, J. Komar, D. Orth, F. Mell, P. Merriaux, P. Grenet, Y. Caritu, R. Hérault, V. Dovgalecs et collab.. 2014, «Pattern recognition in cyclic and discrete skills performance from inertial measurement units», *Procedia Engineering*, vol. 72, p. 196–201.

Ragot, N., P. Merriaux, R. Rossi, D. Leclercq-Delapierre, X. Savatier et J. Delarue. 2014, «Le projet coalas : un exemple de formation des élèves-ingénieurs par la recherche», CETSIS, Besançon.

Merriaux, P., J. Delarue et X. Savatier. 2013, «Un environnement attractif, ludique et simple pour des travaux pratiques, c'est possible», CETSIS, Caen.

Leplomb, H. et P. Merriaux. 2016, «Success story : Autonomous navigation based on odometry - ahrs - lidar», cahier de recherche, SBG.

Guinamard, A. et P. Merriaux. 2013, «Ekinox test results», cahier de recherche, SBG.

E.3 Brevet

Gourcerol, G. et P. Merriaux. 2015, «Stimulation device for stimulating at least one part of nervous system of mammals», European patent PCT 15306344.1

Résumé

Nous constatons ces dernières années un essor important de la robotique mobile autonome avec deux grands domaines d'application : la robotique des services et le véhicule autonome. Quel que soit le domaine visé, la fonction de localisation est déterminante pour l'autonomie de ces futurs mobiles. A ce titre, nous avons consacré ces travaux de thèse à l'approfondissement des problématiques de localisation à travers deux sujets proposant des méthodes embarquées.

Dans un premier temps, avec le challenge international de robotique Argos : nous nous sommes attelés à une localisation 6 degrés de liberté basée LiDAR multi-nappes, avec une précision suffisante pour assurer le contrôle du robot. Les applications envisagées étant la surveillance des installations pétrochimiques, l'environnement rencontré est plus complexe que ceux couramment décrits dans la littérature. Nous avons mis particulièrement l'accent sur le côté embarquable, soit la consommation de ressources aussi bien en termes de mémoire que de CPU de l'algorithme. Pour cela, nous avons étendu le concept de likelihood field en 3D. L'algorithme a été évalué en laboratoire puis sur un site industriel, avec deux types de LiDAR et sur 3 robots. Nous atteignons une précision d'environ 2.5cm en utilisant 16% d'un core d'un processeur actuel.

Cette méthode de localisation n'autorise pas une initialisation sur une large zone. Dans un second temps, nous avons donc proposé une localisation topologique de véhicules routiers permettant une convergence sur plusieurs kilomètres carrés. Nous n'utilisons que des capteurs présents dans tous les véhicules (ABS et ESP) et la carte est représentée par un graphe contenant des données Open Street Map. A partir de ces informations, nous obtenons une précision inférieure à 4 m, c'est-à-dire semblable à celle d'un GPS standard.

Nos travaux portent également sur une méthodologie d'expérimentation. En effet, la mise au point d'algorithmes de localisation nécessite de nombreux ajustements. De plus, le test, la fiabilité puis la qualification de ces systèmes autonomes demeurent de véritables enjeux aussi bien pour la communauté scientifique qu'industrielle. Cette méthodologie contribue à répondre à ces challenges en fusionnant le développement des tests réalisés en simulation, à échelle réduite et pleine échelle.

Abstract

There has been a great and quick development in autonomous mobile robotics over the past few years. This growth mainly concerns autonomous vehicles and service robotics. Whatever the field of application, the localization task plays a key role in the intelligence of the mobile robots. As a result, this thesis is focused on exploring new challenges in embedding this aspect of artificial intelligence over two topics.

First, we tackled the ARGOS Challenge. We have proposed a 6 Degrees of Freedom localization method based on multi-layer LiDAR data. Results show that the localization is accurate enough to perform autonomous control of the robot. Targeted applications include Oil and Gas platform patrolling. Such environments are more challenging than regular environments found in the literature. The proposed method takes into account concerns regarding CPU and memory consumption as well as embeddability. The likelihood field concept was extended to 3D. The method was benchmarked in a lab, an industrial site with single-layer and multi-layer LiDARs and 3 different robots. Robots are being located in their environment with an average error of 2.5cm using 16% of a CPU core.

Secondly, we focused on topological localization. In fact, the approach used in the ARGOS Challenge requires a small area as initial step. Our second approach enables to find the location of a road vehicle over several square kilometers. Our method is based on sensors widely available on any modern cars (ABS and ESP). The map used is a topological representation of OpenStreet map data. Based on this information, we achieve an average error of 4m. This figure is close to GPS-based accuracy.

Finally, our works are based on a specific experimental approach. In fact, developing new localization algorithms requires fine tunings. Moreover, tests, reliability and certifications of autonomous systems are still challenging for either the scientific community or the industry actors. The proposed methodology tends to propose a solution by mixing simulation, small-scale testing and full-scale testing.